Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет
«Высшая школа экономики»

Отделение программной инженерии

Кафедра Управления разработкой программного обеспечения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему «Программа построения равновесных стратегий для игры Omaha»

Студент группы № 471ПИ

Ермолов Александр Юрьевич

Руководитель ВКР

Ульянов М. В., д. т. н.

Москва, 2014

Дипломная работа: 57 с., 7 рис., 5 табл., 14 источников литературы.

Этот документ является дипломным проектом бакалавра. Целью работы является написания приложения для нахождения ϵ равновесие по Нэшу для игры Omaha и Omaha Hi\Lo в модели jam-fold для двоих игроков. Для нахождения равновесия программа использует алгоритм фиктивного разыгрывания. Также в программе реализованы эффективные алгоритмы вычисления математического ожидания действий. Проект может служить основой для изучения зависимостей математического ожидания действий игрока от стратегий соперника и моделирование равновесной игры в турнирах по Omaha Hi\Lo. Практический аспект работы заключается в создании искусственного интеллекта который может оптимально играть в Omaha и Omaha Hi\Lo в модели jam-fold. Также на основе программы можно проводить анализ играемых раздач и давать оценку уровню игры человека.

Ключевые слова: теория игр, равновесие по Нэшу, редукция вычислений, искусственный интеллект, теория вероятностей.

Содержание

Введение

С появлением компьютеров широкое развитие получила тема искусственного интеллекта. Одним из направлений искусственного интеллекта были компьютерные шахматы. В 1951 году Алан Тьюринг написал алгоритм для игры в шахматы. Из-за отсутствия доступа к компьютеру этот алгоритм приходилось выполнять вручную. Человеку требовалось более получаса что бы сделать один ход согласно этому алгоритму [1]. В то же время другой известный учёный Клод Шеннон опубликовал свою первую статью о шахматном программировании. В ней он в частности высказал мысли о теоретическом существовании лучшего хода, а также о практической невозможности его найти. В дальнейшем шахматный программы совершенствовались и в 1996 году компьютер Deep Blue впервые выиграл партию у действующего чемпиона мира по шахматам в турнирных условиях. В настоящие время существует множество доступных шахматных программ уровень игры которых не уступает профессиональным игрокам. Теперь шахматистам не нужно ездить по многочисленным турнирам в поиске сильных соперников, так как можно оттачивать навыке своей игры на компьютере.

Шахматы были хорошим объектом для разработки и проверки методов искусственного интеллекта, в частности были апробированы такие направления как: методики оптимизации перебора (уход от «комбинаторного взрыва» при просчёте вариантов вперёд на несколько ходов), распознавание образов, экспертные системы, логическое программирование. А. С. Кронрод определил роль компьютерных шахмат известной фразой: "Шахматы — это дрозофила искусственного интеллекта". Но шахматы так и не смогли приблизить человека к созданию машин с человекоподобным интеллектом. Во-первых, шахматные программы просто перебирают множество вариантов возможных ходов обоих игроков используя при этом тривиальные алгоритмы усечения и простую функцию оценки. Вкупе с базами данных дебютов и эндшпилей этого достаточно что бы на современных компьютерах программа могла играть на гроссмейстерском уровне. Но данные подходы неприменимы к играм с большим количеством вариантов перебора, таких как го. При исследовании этих игр учёным приходится применять более умозрительный подход к игре. Во-вторых, шахматы являются игрой с полной информацией, а большинство решений в реальной жизни принимается в условиях неопределенности. Это ограничивает возможность применение методов компьютерных шахмат в практических задачах экономики, там где кроме нас могут быть ещё несколько агентов с неизвестными позициями.

Хорошей абстракцией для изучения подобных процессов может послужить игра покер. Покер это игра с неполной информацией, она имеет огромное количество стратегий и в неё могут играть более двух игроков. Также покер являет одной из самых популярных карточных игр в мире, ежегодно проводятся множество турниров с огромными призовыми фондами. В 2005 году призовой фонд Мировой Серии Покера насчитывал более 103 млн. долларов [2]. Это может служить дополнительной мотивацией для исследования данной игры. Благодаря увеличению мощности процессоров и объёмов оперативной памяти за последние 10 лет были достигнуты значительные результаты в изучении покера. Так для двоих игроков была полностью решена одна из разновидностей покера Rhode Island hold'em. Rhode Island hold'em является одним из самых простых играемых видов покера, но не смотря на это дерево перебора для этой игры насчитывало более 3 миллиардов узлов [3].

Наибольшее внимание исследователей сейчас сосредоточено на виде покера Texas hold'em, так как он является наиболее популярным на текущий момент. Но вычислительная сложность Texas hold'em значительно выше чем у уже решенных разновидностей покера. Согласно исследованиям Texas hold'em с фиксированным размером ставок имеет дерево перебора с 10^{18} узлов [4]. Вариант этой игры с произвольным размером ставок при дискретизации имеет ещё большую сложность. Поэтому для решения этих игр используют методы уменьшения количества вариантов. Одним из таких методов является модель jam-fold. Эта абстракция хорошо зарекомендовала себя в no limit Texas hold'em. Данная модель подразумевает, что игроки могут совершить только два действия: поставить все фишки или сбросить. Благодаря чему в разновидностях hold'em количество раундов торговли с трёх уменьшается до одного. Игра в данной абстракции является дискретной, в связи с чем в отсутствует проблема выбора размера ставок. В совокупности с уменьшение количество раундов это позволяет резко сократить размер дерева перебора. Так для каждого игрока в начале игры будет всего 169 стратегически различных состояний. Ещё одним достоинством этой абстракции можно назвать то, что при игре с большими обязательными ставками стратегии рассчитанные в данной модели могут быть применимы и в игре без ограничений этой модели. Так на основе этой модели работают программы для анализа no limit Texas hold'em: SNG Wizard, ICMTrainer, SNGSolver, Holdem Resources Calculator. Ввиду небольшой вычислительной сложности этой модели были подсчитаны равновесные по Нэшу стратегии для двух человек с помощью линейного программирования и для трёх человек с помощью алгоритма Брауна-Робинсона (фиктивного разыгрывания). Также

поиск равновесия Нэша в более ограниченной абстракции с помощью алгоритма Брауна-Робинсона используется в программах SNGSolver и Holdem Resources Calculator.

Другой, второй по популярности вид покера, Omaha hold'em сейчас малоизучен. В основном это можно объяснить двумя причинами. Во-первых, Omaha hold'em сильно уступает по популярности Texas hold'em, по данным сайта pokerscout.com в онлайн покере Texas hold'em в семь раз популярнее Omaha hold'em [5]. Во-вторых, это игра значительно сложнее, так как вначале каждому игроку раздают 4 карты, а не 2 как в Texas hold'em. Существует два основным вида Omaha hold'em: Omaha и Omaha Hi/Lo. Они отличаются лишь структурой функции выигрыша, так в обычной Omaha один призовой фонд и разыгрывается он по стандартным правилам покера. А в Omaha Hi/Lo основной призовой фонд делится на две части, одна часть призового фонда разыгрывается за лучшую комбинацию по правилам покера, а вторая за худшую комбинацию согласно "калифорнийской" системе старшинства лоу-комбинаций. Благодаря этому повышается вариативность игры, так при розыгрыше банка для двух игроков может быть 12 вариантов в зависимости от силы комбинаций, а не 3 как в обычной Omaha или Texas hold'em. Это должно способствовать увеличению средней продолжительности турнира по данной игре в модели jam-fold, так как вероятность того, что игрок проиграет все свои фишки за одну раздачу понижается. Поэтому есть основания полагать, что модель jam-fold в Omaha Hi/Lo может быть хорошо применима. Также на данный момент отсутствуют программы для анализа игры Omaha hold'em в абстракции jam-fold. Что вкупе с ростом популярности этой игры делает изучение модели jam-fold в играх Omaha hold'em очень перспективной не только с научной, но из коммерческой точки зрения. Но помимо сложности вычислений, реализация анализатора для этой игры затруднена большим количеством вариантов розданных карт. Для Omaha hold'em существует 16432 стратегически различных начальных комбинаций. Это значительно усложняет задание чужих стратегий вручную, так для анализа нашего действия будет необходимо задать поведения соперника для каждого из 16432 возможных вариантов. В Texas hold'em это проблема не стоит так остро, так как там всего 169 стратегических различных комбинаций, а также существуют несколько методов ранжирования рук для упрощенного задания стратегий. Эту проблему можно разрешить, если задавать стратегию сопернику согласно равновесию по Нэшу. Поэтому для анализатора необходимо разработать программу которая будет находить равновесные диапазоны в игре Omaha hold'em.

Цель работы

Целью работы является создание программы для нахождения ϵ равновесие по Нэшу для игры Omaha и Omaha Hi\Lo в модели jam-fold для двоих игроков. На основе ситуации в игре заданной пользователем, программа будет выводить рекомендуемое действие и математическое ожидания этого действия при игре против равновесной по Нэшу стратегии соперника. Равновесие будет находиться с помощью алгоритма фиктивного разыгрывания, с заданным пользователем числом итераций. Программа должна иметь возможность выбора правил игры Omaha или Omaha Hi\Lo.

Обзорная часть

Правила игры Omaha

Перед тем как описывать правила игры, внесём несколько определений терминов игры.

Колода карт — это полный набор прямоугольных или пластиковых листов (карт), предназначенных для карточных игр. В играх типа покер чаще всего используется колода из 52 карт. Каждая карт задаётся уникальной двойкой значений: мастью и рангом. В стандартной колоде из 52 карт входят карты 4 различных мастей и 13 различных рангов.

Дилер — это специальная позиция в покере условно обозначающая сдающего карты.

Блайнд — это ставка которую игроки на некоторых позициях обязаны сделать, до того как они получат карты. Также иногда этим терминам обозначают позиции игроков которые делают эту ставку. Блайнд делиться на большой и малый. Размер малого блайнда обычно в два раза меньше большого блайнда. Игрок позиция которого по часовой стрелки является следующей за дилером обязан перед раздачей карт поставить малый блайнд. Игрок позиция которого по часовой стрелки является следующей за малым блайндом обязан перед раздачей карт поставить большой блайнд.

Анте — принудительная ставка до начала раздачи которую обязан сделать из игроков в независимости от своей позиции. Если размер анте равняется нолю, то обязательные ставки перед игрой делают только игроки на позициях малый и большой блайнд.

Рука - набор личных карт игрока которые раздают в начале игры. Для игр типа Omaha стандартно раздают 4 карты, для Texas Holdem 2.

Раунд торговли — это период розыгрыша в покере, когда игроки должны делать ставки, пополняя тем самым банк, за который они борются. В процессе торговли игроки могу совершить следующие действия:

- Сделать ставку.
- Уравнять ставку соперника.
- Увеличить ставку. Поставить больше чем соперник.
- Отказаться от дальнейшего участия в раздаче и сбросить карты.
- В ситуации, когда игроком уже была сделана обязательная ставка или ставки не были сделаны соперниками игрок может не добавлять ставку.

Размер возможных ставок зависит от типа игры:

- Лимит — игроки могут делать только фиксированные ставки или поднимать только на сумму фиксированной ставки.
- Пот-лимит — игроки могут делать ставки от минимальной до текущего размера банка.
- Без лимита, ноу-лимит — игроки могут делать ставки от минимально до размера своего стека.

При торговли действия совершаются игроками по очерёдно по часовой стрелки. Круг заканчивается когда все соперники сделали равные ставки или сбросили карты. Если после раунда торговли в игре остался только один игрок, то он забирает себе все деньги из банка. [6]

Вскрытие — этап игры в котором определяется сила комбинации каждого игрока дошедшего до этого этапа. По правилам игры Omaha комбинация составляется из 5 карт, 2 берутся из карт игрока и 3 берутся из общих карт. После определения силы каждой комбинации банк игры делится в соответствии с силой комбинации игроков. В зависимости от типа игры это может происходить двумя основными способами:

- Omaha. Сила каждой комбинации определяется по стандартным правилам покера и весь банк достаётся игроку с сильнейшей комбинацией, в случае ничьи банк делится в равных частях между игроками с одинаковыми по силе комбинациями.
- Omaha Hi/Lo. Основной банк делится на две равные части. Первая часть банка разыгрывается по правилам Omaha. А вторая часть разыгрывается по правилам "8 или лучше", для определения силы комбинации используется "калифорнийская" система старшинства лоу-комбинаций. Банк достаётся игроку с самой слабой комбинаций, в случае ничьи банк делится в равных частях между игроками с одинаковыми по силе комбинациями.

Комбинация — по стандартным правилам покера это набор из 5 карт с помощью которого определяется победитель. Каждая комбинация имеет свою силу. Определение силы может зависит от правил. По стандартным правилам для сравнения комбинаций используют следующую классификацию в порядке убывания силы:

- Роял-флаш - старшие 5 карт одной масти.
- Стрейт-флаш - любые 5 карт одной масти ранг которых идёт по порядку.
- Каре - 4 карты одного ранга.
- Фул-хаус - 3 карты одного ранга и одна пара.

- Флаш - 5 карт одной масти.
- Стрейт - 5 карт любой масти ранг которых идёт по порядку.
- Сет - 3 карты одного ранга.
- Две пары - 2 пары карт.
- Пара - 2 карты одного ранга.
- Старшая карты - ни одна из вышеописанных комбинаций.

При совпадении типа комбинаций сначала поочерёдно сравнивается ранг карт участвующих в комбинации, более сильной является комбинация со старшими картам. В случае если ранг комбинации совпадает, то поочередно сравниваются карты из 5 не входящие в основную комбинацию, более сильной является комбинация со старшей картой. Всего по таким правилам существует 7462 разных по силе комбинаций.

По правилам "8 или лучше" используют классификацию похожую на стандартные правила, но не учитываются комбинации Стрейт, Флаш и производные из этих комбинаций. Также в отличии от стандартных правил, ранг 'A' играет роль самого младшего ранга. Все комбинации сильнее типа "Старшая карты" с максимальным рангом '8' всегда проигрывают. Сравнения комбинаций всегда проходит по карте младшего ранга, более сильной является комбинация с младшей картой [7].

Ход игры Omaha hold'em можно разделить на 5 основных этапов:

- Блайнды. Перед началом игры ставятся обязательные ставки, сначала блайнды потом анте.
- Пре-флоп. После того как обязательные ставки поставлены, каждому игроку из колоды раздаются 4 карты. Затем с игрока который сидит по часовой стрелки за большим блайндом начинается первый раунд торговли.
- Флоп. Из колоды выкладываются три общие карты. Затем идёт ещё раунд торговли.
- Тёрн. Из колоды выкладывается четвёртая общая карты и начинается третий раунд торговли.
- Ривер. Из колоды выкладывается пятая общая карта. Следует последний круг торговли. Если после этого круга торговли все ставки сравнялись, то следует вскрытие.

Обзор калькуляторов для покера

Как правило под покерным калькулятором подразумевают программу которая позволяет рассчитать математическое ожидание действия в заданной ситуации. Большинство существующих на данный момент покерных калькуляторов рассчитаны на анализ разновидности Texas hold'em. Наибольшей популярностью пользуются программы для анализа этой игры в абстракции Jam-Fold. К таким программа относятся: SNG Wizard, ICMTrainer, SNGSolver, Holdem Resources Calculator. Также существуют программы способные проводить расчеты для Texas Holdem в более сложных моделях. Одной из таких программ является Cardrunners EV.

Несмотря на большое разнообразие анализаторов Texas hold'em для Jam-Fold модели можно выявить базовый функционал типичный для такого рода программ, а также функции наличие которых меняется от программы к программе. К базовому функционалу можно отнести возможность расчёта математического ожидания для ситуаций с разным количество игроков. Также во всех программах можно настроить такая параметры как размер обязательных ставок и модель оценки стоимости фишек.

Дополнительные функции программ можно разделить на две группы. В первую группу можно отнести функции влияющие на точность расчётов и близость полученных результатов к реальности:

1) К этому функционалу относится возможность расчёта математического ожидания против всех возможных стратегий игроков. В более ранних программах как правило использовали ограниченный набор стратегий. Это помогало сильно ускорить расчёты, но минус данного подход заключался в том, что в реальности люди могли выбирать стратегии которые не входили в ограниченный набор. В частности такие ограничения есть в программах SNG Wizard, ICMTrainer, SNGSolver, и только Holdem Resources Calculator позволяет проводить расчёты в неограниченных стратегиях [8].

2) Второй функцией которую можно отнести к первой группе является возможность расчёта равновесных стратегий. Для расчёта математического ожидания в программах необходимо задать стратегии соперников. Для простоты использования все программы выставляют эти стратегии автоматически. В основном все программы задают чужие стратегии согласно равновесию по Нэшу. Из рассматриваемых программ только SNG Wizard не использует равновесные стратегии [9].

3) Возможность учёта будущих раздач для оценки стоимости фишек также можно отнести к первой группе. Данная функция позволяет оценивать ситуацию не только в разрезе

заданной раздачи, но и с учётом возможных раздач в будущем. Это функция полезно для анализа турнирного покера. Программы SNGSolver и Holdem Resources Calculator имеют режим расчёта с учётом будущих раздач.

Ко второй группе можно отнести функционал отвечающий за облегчение использования полученных расчётах в прикладных задачах. К этому относятся следующие возможности:

1) Импорт и автоматический анализ раздач. Как правило описание ситуаций вводятся пользователем вручную. Такой подход создаёт сильные неудобства для анализа большого количество ситуаций. SNG Wizard и Holdem Resources Calculator позволяют автоматический импортировать ситуации из файлов истории покерных клиентов или базы данных покерных трекеров. Помимо данные программа также могут проводить анализ ситуаций в автоматическом режиме. Эта функция позволяет находить наиболее спорные ситуации в которых пользователь чаще всего ошибается.

2) Режим тренировки. В этом режиме программа генерирует ситуация, а пользователь должен выбрать правильное действие. При этом программа запоминает ошибки пользователя и выдаёт ему подробную статистику о них после тренировки. Данный режим позволяет пользователю снизить количество ошибок в своей игре. Эта функция реализована в программах SNG Wizard и ICMTrainer.

На основе перечисленных функций можно составить следующую таблицу сравнения калькуляторов для покера:

Таблица 1. Сравнение покерных калькуляторов

Программа	Неограниченные стратегии	Расчёт равновесных диапазонов	Учёт будущих раздач	Импорт и автоанализ ситуаций	Режим тренировки
SNG Wizard	-	-	-	+	+
ICMTrainer	-	+	-	-	+
SNGSolver	-	+	+	-	-
Holdem Resources Calculator	+	+	+	+	-

Эквиляторы для игры Omaha

Эквилятор - это программа для расчёта распределения исходов в случае вскрытия. Распределение исходов вскрытия является основной частью в расчёте математического ожидания для действия AllIn. Поэтому точность расчёта этого распределения очень важна. Из-за большого размера множества элементарных событий данное распределение обычно рассчитывается с помощью метода Монте-Карло. Этот метод не даёт точное распределение и результат может незначительно отличатся от расчёта к расчёту. В более простых случаях возможен расчёт полным перебором - в этих случаях результат будет точным.

Можно выделить три основных пути использования эквилятора:

1) Расчёт распределения карт игрока против карт соперника с неизвестными общими картами. Это наиболее простая задача и как правило эквиляторы могут решить её с помощью перебора всех возможных комбинаций общих карт.

2) Расчёт распределения карт игрока против известного множества карт соперника с неизвестными общими картами. Множество элементарных событий этого типа ситуаций не позволяет рассчитать распределение с помощью перебора. Обычно результат именно этого типа задач необходим для расчёта математического ожидания действия AllIn для игрока с известными картами.

3) Расчёт известного множества карт игрока против известного множества карт соперника с неизвестными общими картами. Это наиболее сложная задача. Для её решения обычно применяется метод Монте-Карло.

Из-за простоты реализации на данный момент существуют множество различных эквиляторов для разных видов покера. Несмотря на это можно выделить только два эквилятора которые поддерживают игру Omaha и Omaha Hi/Low - Equilab - Omaha и сайт propokertools.com.

Инструмент предоставляемый сайтом propokertools.com позволяет рассчитывать распределения не только игр Omaha и Omaha Hi/Lo но так же и для других видов покера. Таких как: Texas hold'em, Ruzz, 5-Card Omaha, Stud. Ещё одно преимущество этого эквилфтора в том, что для расчёта распределения известных карт игрока против известных карт соперника он использует полный перебор [10].

Equilab - Omaha в отличии от propokertools.com позволяет проводить расчёты только для игр Omaha и Omaha Hi/Lo. Данный эквилятор проводит все расчёты методом Монте-

Карло, но в отличии от предыдущего инструмента имеет возможность настройки количества итераций для этого алгоритма. Еквилятор на сайте propokertools.com при использовании метода Монте-Карло всегда проводит только 600000 итераций.

Теоретическая часть

Модель Jam-Fold

Модель Jam-Fold подразумевает, что в раунде торговли на этапе "Пре-флоп" каждый игрок может сделать только одно из двух действий: Fold - отказаться от участия в раздаче или AllIn - сделать ставку равную собственному стеку. Благодаря такому упрощению на следующих этапах не будет проводится раунды торговли, потому что все игроки дошедшие до туда уже поставили все фишки и не могут совершить какое либо действие. Это позволяет значительно сократить дерево перебора и уменьшить количество информации необходимой игроку для принятия решения [11].

Так игру теперь можно разбить на три этапа:

- Блайнды. Перед началом игры ставятся обязательные ставки, сначала блайнды потом анте.
- Пре-флоп. После того как обязательные ставки поставлены, каждому игроку из колоды раздаются 4 карты. Затем с игрока который сидит по часовой стрелки за большим блайндом начинается первый раунд торговли.
- Вскрытие. Из колоды выкладывается пять общих карт после чего следует вскрытие.

При этом игроки могут совершать действия только на втором этапе. Так игра обладает следующими свойствами:

- Игра с нулевой суммой. Нельзя выиграть больше чем проиграли другие.
- Последовательная. Во время раунда торговли все игроки делают действия последовательно.
- С неполной информацией. При принятии решения игроки не знают полностью текущего состояния игры, так как они не видят карт соперников.
- Случайная. Платежи определяются на основе силы карт, которые раздаются случайно.
- Дискретная. Действия игроков и количество возможных исходов счётны.

Опишем игру в этой модели для двух игроков (A и B). У каждого игрока по 1500 фишек, обязательные ставки: 200 фишек для большого блайнда и 100 фишек для малого блайнда. Игрок A будет находится на позиции малого блайнда, а игрок B на позиции большого

блайнда. Для простоты иллюстрации предположим, что каждый игрок имеет 3 стратегии: сбросить в 100% случаях, играть если ему на этапе "Пре-флоп" раздадут карты которые в ходят в 50% сильнейших, играть в 100% случаях. В экстенсивной форме данная игра будет выглядеть следующим образом:

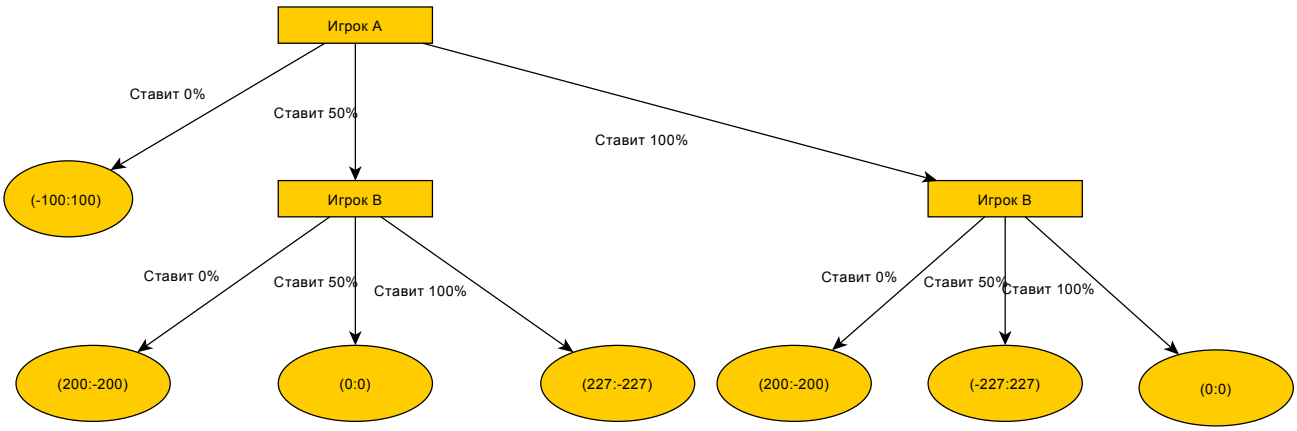

Рис.1 Пример дерева решений

По результатам игры возможно 7 состояний.

- Игрок А сбросит свои карты. В результате игрок В получит выиграет у игрока А 100 фишек (размер малого блайнда). $E_a = -100$
- Игрок А будет ставить 50% лучших рук. После этого игрок В сбросит свои карты. В результате игрок А выиграет у игрока В 200 фишек (размер большого блайнда) с вероятностью 0.5. И с вероятностью 0.5 игрок А сбросит карты и проиграет игроку В 100 фишек. $E_a = 0.5 * 200 + 0.5 * -100 = 50$.
- Игрок А будет ставить 50% лучших рук. После этого игрок В будет играть 50% своих рук. Если игрок А поставит, то с вероятностью 0.5 игрок В сбросит карты и с вероятностью 0.5 будет этап вскрытия. Математическое ожидания этапа вскрытия для каждого игрока будет равно 0, так как стратегии игроков совпадают. То есть с вероятностью 0.5 игрок А проиграет игроку В 100 фишек, с вероятностью 0.25 выиграет 200 фишек и с вероятностью 0.25 выиграет 0 фишек. $E_a = 0.5(0.5 * 200 + 0.5 * 0) + 0.5 * -100 = 0.25 * 200 + 0.5 * -100 = 0$.
- Игрок А будет ставить 50% лучших рук. После этого игрок В будет играть 100% своих рук. Если игрок А поставит, то будет вскрытие. Математическое ожидания этапа вскрытия для игрока А будет равно 227, так как 50% лучших карт имеют такое преимущество над 100% любых карт. То есть с вероятностью 0.5 игрок А

проиграет игроку B 100 фишек, с вероятностью 0.5 выиграет 227 фишек.
$E_a = 0.5 * 227 + 0.5 * -100 = 63,5$.

- Игрок A будет ставить 100% рук. После этого игрок B сбросит свои карты. В результате игрок A выиграет у игрока B 200 фишек.
- Игрок A будет ставить 100% рук. После этого игрок B будет играть 50% своих рук. То есть с вероятностью 0.5 игрок B сбросит карты и с вероятностью 0.5 будет этап вскрытия. Математическое ожидания этапа вскрытия для игрока B будет равно 227, так как 50% лучших карт имеют такое преимущество над 100% любых карт.
$E_a = 0.5 * 200 + 0.5 * -227 = 100 - 113,3 = -13,5$.
- Игрок A будет ставить 100% рук. После этого игрок B будет играть 100% своих рук. То есть всегда будет этап вскрытия. Математическое ожидания этапа вскрытия для каждого игрока будет равно 0, так как стратегии игроков совпадают.
$E_a = 1.0 * 0 = 0$

В нормальной форме данную игру можно определить как множество $G = \langle P, S, F \rangle$ где

$P = \{P_1, P_2\}$ - множество игроков,

$S = \{S_1, S_2\}$ - множество множеств чистых стратегий каждого игрока,

$S_1 = \{S_1^1, S_1^2, S_1^3\}$ - множество чистых стратегий игрока P_1,

$S_2 = \{S_2^1, S_2^2, S_2^3\}$ - множество чистых стратегий игрока P_2,

$F = \{F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9\}$ - множество функций платежа.

Таблица 2. Платёжная матрица

	S_2^1	S_2^2	S_2^3
S_1^1	(-100:100)	(-100:100)	(-100:100)
S_1^2	(50:-50)	(0:0)	(-13,5:- 13,5)
S_1^3	(200:-200)	(63,5:-63.5)	(0:0)

Стратегии

Так как игра случайная, платёжная матрица будет состоять из математических ожиданий возможных сочетаний стратегий. Стратегия в данной игре определяет поведения игрока в зависимости от карт которые ему раздадут. Всего игроку может прийти

$C_{52}^{4}=\frac{52*51*50*49}{1*2*3*4}=\frac{6497400}{24}=270275$ разных комбинаций карт. Для каждой комбинации игрок может выбрать одно из двух решений, т. е. возможно 2^{270275} различных стратегий. Но начальные 270275 комбинаций можно значительно сократить если учитывать правила определения силы комбинаций. Так как в покере все масти равноправны, до этапа "Флоп" многие из комбинаций будут иметь одинаковое распределение силы. Сила карт на данном этапе зависит от двух факторов: ранга карт и взаимного расположения мастей. Таким образом из 270275 комбинаций существует 16432 различных по силе стартовых рук которые можно разделить на 5 видов [12]:

Таблица 3. Виды комбинаций

Расположение мастей	Всего комбинаций	Различных по силе
aaaa	2,860	715
aaab	44,616	3,718
aabb	36,504	3,081
aabc	158,184	7,098
abcd	28,561	1,820

Таким образом можно сократить количество возможных стратегий с 2^{270275} до 2^{16432}. Но такое количество по-прежнему является неподъёмным для хранения и обработки, поэтому математическое ожидания результата игры для двух стратегий целесообразно вычислять по мере необходимости этих данных. Для того что бы вычислить математическое ожидание стратегии по которой играется 50% карт против стратегии по которой играется 100% карт нам потребуется перебрать

$135137*270275*C_{44}^{5}=135137*270275*1086008=39665521998271400$ вариантов.

Сложность такого перебора слишком высока, поэтому необходимо использовать более эффективные методы нахождения математического ожидания. Во-первых математическое ожидания можно найти с помощью метода Монте-Карло, но результат этого метода будет приблизительным. Во-вторых можно использоваться предрассчётные данные специального вида. Так мы можем заранее разбить все возможные комбинации начальных карт на 16432 кластера. Затем посчитать математическое ожидания для каждой пары

кластеров и записать это в отдельную таблицу. Данная таблица будет иметь $16432 * 16432 = 270010624$ элементов. Стоит также отметить, что для универсальности в таблице лучше хранить не математическое ожидание, а распределения призового фонда. Так для Omaha Hi\Lo при вскрытие возможно 9 различных исходов, поэтому если хранить количество исходов в целочисленном типе int, то под эту таблицу потребуется $270010624 * 9 * 4 = 9720382464$ байт памяти. Такая таблица может поместиться в оперативной памяти современного компьютеров под управлением 64-битной операционной системы. С помощью этой таблицы математическое ожидание можно вычислить по следующей формуле:

$$E = \sum_{a \in A} \sum_{b \in B} \left(\frac{\sum_{z \in Z} t_z^{a,b} * F_z}{\sum_{z \in Z} t_z^{a,b}} * P_{a,b} \right)$$, где

A - множество кластеров начальных комбинаций при которых первая стратегия будет играть AllIn,

B - множество кластеров начальных комбинаций при которых вторая стратегия будет играть AllIn,

Z - множество исходов при вскрытии карт у двух игроков,

F - множество выигрышей при конкретных исходах,

P - множество вероятностей начальных комбинаций двух игроков,

t - таблица предрасчётных данных.

Так не считая внутренний цикл перебора, для нахождения математического ожидания двух стратегий нам максимум потребуется перебрать $N^2 = 16432 * 16432 = 270010624$ вариантов, что в $\frac{39665521998271400}{270010624} \approx 1.5 * 10^6$ раз меньше первоначальной оценки.

Равновесие по Нэшу

В теории игр равновесием по Нэшу называется ситуация при который ни один из игроков не может увеличить свой выигрыш, в одностороннем порядке меняя своё решение. Допустим (S, H) - игра n лиц в нормальной форме, где S - набор чистых стратегий, а H - набор выигрышей. Когда каждый игрок $i \in \{1,\dots,n\}$ выбирает стратегию $x_i \in S$ в профиле стратегий $x=(x_1,\dots,x_n)$, игрок i получает выигрыш $H_i(x)$. Профиль стратегий x^* является равновесным по Нэшу, если изменения своей стратегии с x_i^* на x_i не выгодно ни одному игроку i, то есть для любого i:

$H_i(x^*) \geq H_i(x_i, x_{-i}^*)$

В 1951 году Д. Нэш в своей работе по некорпоративным играм доказал, что в любой конечной игре будет существовать хотя бы одно равновесие в чистых или смешанных (то есть при выборе чистой стратегии стохастический с фиксированной частотой) стратегиях. Так как игра Omaha в модели Jam-Fold является конечной, то в ней будет существовать хотя бы одно равновесие по Нэшу.

Существует несколько методов нахождения равновесия по Нэшу в игре. В основном они основываются на работе с матричным представлением игры. Но так как у решаемой нами игры огромное количество чистых стратегий, использовать её матричное представление проблематично. Для поиска равновесия Нэша в данной игре хорошо подойдёт метод фиктивного разыгрывания. Суть этого метода в том, что для каждого игрока находится оптимальная чистая стратегия против смешанной стратегии соперника, потом эта стратегия добавляют в смешанную стратегию игрока. В 1951 году было доказано, что в антагонистической игре для двух игроков при многократном повторении этого алгоритма смешанные стратегии игроков будут стремится к равновесию по Нэшу [13]. Таким образом с помощью этого метода за конечное число итераций возможно найти ε равновесие по Нэшу в заданной игре.

Для того что бы определить оптимальную стратегию игрока A против стратегии игрока B, нам необходимо найти множество играемых кластеров карт для которых математическое ожидание игры против стратегии игрока B будет больше ноля:

$\underset{A}{argmax}\,¿$, где

A - множество кластеров начальных комбинаций при которых стратегия игрока A будет играть AllIn,

B - множество кластеров начальных комбинаций при которых стратегия игрока B будет играть AllIn,

C - множество кластеров возможных начальных комбинаций,

Z - множество исходов при вскрытии карт у двух игроков,

F - множество выигрышей при конкретных исходах,

P - множество вероятностей начальных комбинаций двух игроков,

t - таблица предрасчётных данных.

Таким образом в самом худшем случае нам понадобится не более $N^2 = 16432 * 16432 = 270010624$ вариантов перебора.

Практические часть

Алгоритм для определения силы комбинации

В играх типа Omaha для определения силы комбинации необходимо учитывать 9 карт: 4 стартовых карты игрока и 5 общих карт. По стандартным правилам покера комбинацию составляет 5 карт. Для определения силы комбинации в игре Omaha нам необходимо перебрать все возможные варианты комбинаций 5 карт из 9. Самая сильная из этих комбинаций будет использоваться для сравнения с комбинацией соперника. При этом необходимо учитывать, что по правилам игры Omaha для построения комбинации игрок должен использовать 2 своих карты и 3 общих.

Таким образом для определения силы комбинации из 9 карт для игры Omaha нам потребуется перебрать $C_4^2 * C_5^3 = 6 * 10 = 60$ комбинаций из 5 карт. Для быстрого определения силы комбинации из 5 карт можно использовать таблицу размером $52^5 = 380204032$ ячеек. Номер каждой карты будет использоваться как индекс в пятимерном массиве. На каждую ячейку необходимо выделить 2 байта памяти, так как возможно 7462 различных по силе комбинаций. Данная таблица будет занимать $380204032 * 2 = 760408064$ байт памяти.

Для определения силы комбинации из 9 карт нам будет необходимо 60 раз обратится к этой таблице. Для ускорения данного алгоритма необходимо построить таблицу для определения силы комбинации сразу по 9 картам. Без оптимизаций такая таблица будет занимать $52^9 = 2779905883635712$ ячеек или $2779905883635712 * 2 = 5559811767271424$ байт памяти. Эффективная обработка таблиц таких размеров на данный момент невозможна. В связи с чем необходимо использовать различные техники для уменьшения размера таблицы.

Для редукции таблицы предлагается воспользоваться особенностями правил игры Omaha. Во-первых можно воспользоваться тем свойством, что изначально существует всего 16432 стратегически различных стартовых рук. На основе этого для определения силы можно использовать таблица из двух индексов. В первом будет номер стартовой руки от 0 до 16431, а во втором номер для общих карт от 0 до 380204031. Данная таблица будет по-прежнему иметь слишком большой размер, $38020403 * 16432 = 624751262096$ ячеек. Во вторых можно учесть, что при составлении комбинации необходимо всегда использовать две стартовые карты игрока. На основе этого можно разделить все стартовые руки на три большие группы:

- Руки которые не могут составить комбинацию флаш. Это будут руки типа: abcd. Их количество 1820.
- Руки которые могут составить комбинацию флаш только с помощью одной масти. Это будут руки типа: aaaa, aaab, aabc. Их количество, 715+3718+7098=11531.
- Руки которые могут составить комбинацию флаш с помощью двух мастей. Это будут руки типа: aabb. Их количество 3081.

Стартовые руки из первой группы не могут собрать комбинаций флаш, а это единственная комбинация для сбора которой важна масть. Таким образом масть никак не влияет на силу комбинаций этих стартовых рук и мы можем рассматривать карты только с учетом ранга. Вместо множества из 52 карт мы можем рассматривать множество из 13 карт для это группы стартовых рук. При этом необходимо учитывать, что каждая из 13 карт может повторятся в комбинации не более четырёх раз. С помощью перебора с фильтрацией возможных комбинаций можно определить, что таких комбинаций будет 6175. Для определения силы комбинаций для этих рук можно использовать таблицу состоящую из $1820 * 6175 = 11238500$ ячеек или $11238500 * 2 = 22477000$ байт.

Стартовые руки из второй группы могут собрать флаш только с помощью одной масти. Поэтому мы можем сократить размер множества карт с 52 до 26. 13 карт которые могут быть использованы для комбинации флаш и 13 карт незначимых мастей которые не могут быть использованы для комбинации флаш. При этом необходимо учитывать, что каждая из первых 13 карт может повторятся в комбинации не более трёх раз, а из вторых 13 карт не более одного раза. С помощью перебора с фильтрацией возможных комбинаций можно определить, что таких комбинаций будет 101608. Для определения силы комбинаций для этих рук можно использовать таблицу состоящую из $11531 * 101608 = 1171641848$ ячеек или $1171641848 * 2 = 2343283696$ байт.

Стартовые руки из третьей группы могут собрать флаш с помощью двух мастей. Поэтому мы можем сократить размер множества карт с 52 до 39. 13 карт которые могут быть использованы для комбинации флаш с помощью первой масти, 13 карт которые могут быть использованы для комбинации флаш с помощью второй масти и 13 карт незначимых мастей которые не могут быть использованы для комбинации флаш. При этом необходимо учитывать, что каждая из первых 26 карт может повторятся в комбинации не более двух раз, а из вторых 13 карт не более одного раза. С помощью перебора с фильтрацией возможных комбинаций можно определить, что таких комбинаций будет

688311. Для определения силы комбинаций для этих рук можно использовать таблицу состоящую из $3081 * 688311 = 2120686191$ ячеек или $2120686191 * 2 = 4241372382$ байт.

Для быстрого нахождения индекса в этих таблицах можно использовать вспомогательные пятимерные массивы. Для первой группы "рук" он будет иметь размер $13^5 = 371293$ ячеек или $371293 * 4 = 1485172$ байт. Для второй группы $26^5 = 11881376$ ячеек или $11881376 * 4 = 47525504$ байт. Для третьей группы $39^5 = 90224199$ ячеек или $90224199 * 4 = 360896796$ байт. Код на языке C++ для заполнения данных массивов можно найти в приложении А1.

Итого на хранения таблиц для быстрого вычисления силы комбинации из 9 карт для игры Omaha нам потребуется $22477000 + 2343283696 + 4241372382 + 1485172 + 47525504 + 360896796 = 7017040550$ байт или примерно 7 гигабайт памяти. Современные компьютеры позволяют обрабатывать такие таблицы в оперативной памяти.

Алгоритм вычислений силы руки из 9 карт для игры Omaha будет выглядеть следующим образом:

1. Смотрим 4 личных карты игрока и определяем к какой из трёх групп оно относится.
2. В зависимости от группы к которой принадлежит "рука" игрока отобразим общие карты в другое множество.
3. С помощью вспомогательных массивов найти индекс силы в таблице.
4. С помощью индекса извлечь из таблицы силу комбинации.

Рассмотрим пример. У игрока следующие карты: ♠J ♠T ♣8 ♣3. Общие карты: ♣9 ♦2 ♦7 ♠9 ♦9. Если перекодировать карты в числа от 0 до 51 до получится следующий массив $\{35,34,45,40,46,13,18,33,20\}$. Личные карты игрока будут относится к третьей группе, так как возможно собрать флаш двумя способами: мастью ♠ и мастью ♣. Затем отобразим общие карты в новое множество, масть ♠ будет первой, масть ♣ будет второй, все остальные масти будем считать третьей. Общие карты будут перекодированы $\{46 \to 20,13 \to 26,18 \to 31,33 \to 7,20 \to 33\}$. По кодам общих карт найдём в вспомогательном массиве индекс для таблицы сил комбинаций. После чего из этой таблицы извлечём искомое значение силы комбинации из 9 карт.

На языке C++ этот алгоритм можно посмотреть в приложении А2.

Алгоритм для вычисления распределения рука против руки

При заполнения каждой ячейки таблицы распределения исходов сравнения двух рук нам необходимо перебрать все возможные варианты общих карт. Таким образом нам потребуется перебрать $C_{52}^4 * C_{48}^4 * C_{44}^5 = 270275 * 194580 * 1086008 = 57113279637876000$ вариантов. Для сокращения количества вариантов перебора можно воспользоваться описанным ранее разбиением стартовых рук на кластеры. Теперь можно сократить количество переборов до $16432 * C_{48}^4 * C_{44}^5 = 16432 * 194580 * 1086008 = 3472335254868480$. Данное количество по прежнему велико для реализации. Поэтому для дальнейшей редукции вычислений будем использовать перебор каждого кластера с каждым из $16432 * 16432 = 270010624$ вариантов. Это несколько усложнит алгоритм, так как для разных сочетаний кластеров необходимо будет использовать разные правила перебора для наибольшего сокращения вычислений. Для каждого из этих вариантов будем перебирать все возможные стратегически различные сочетания карт выбранных кластеров и общих карт. Алгоритм перебора будет выглядеть следующим образом:

1. Фиксируем карты у первого кластера.
2. Перебираем все возможные сочетания карт с учётом изъятия карт первого кластера для второго кластера. Максимум возможно 24 варианта. При переборе будет идти фильтрация на не повторяемость каждой карты и на стратегическую уникальность каждого варианта.
3. Перебираем все возможные сочетания общих карт с учётом изъятия карт первого и второго кластера. При этом, в зависимости от сочетаний двух рассчитываемых кластеров возможно провести различные сокращения вычислений.
4. Умножаем полученные результаты на количество стратегически различных вариантов карт первого кластера.

Для сокращения количества вариантов для перебора сочетаний общих карт разобьем все кластера на три большие группы:

- Руки которые не могут составить комбинацию флаш. Это будут руки типа: abcd. Их количество 1820. Далее будем называть эту группу разномастные.
- Руки которые могут составить комбинацию флаш только с помощью одной масти. Это будут руки типа: aaaa, aaab, aabc. Их количество, 715+3718+7098=11531. Далее будем называть эту группу одномастные.

- Руки которые могут составить комбинацию флаш с помощью двух мастей. Это будут руки типа: aabb. Их количество 3081. Далее будем называть эту группу двухмастные.

На основе этого разбиения на группы можно помимо главного множества уникальных карт $M=\{1,\ldots,52\}$ ввести ещё три множества:

$M_0=\{1,\ldots,13\}$ - множество карт для которых масть не имеет значения. В колоде каждый элемент из этого множества должен повторяться 4 раза.

$M_1=\{1,\ldots,26\}$ - первые 13 элементов этого множества будут составлять подмножество карт для которых масть не имеет значения. В колоде каждый элемент из этого подмножества должен повторяться 3 раза. Вторые 13 элементов этого множества будут составлять подмножество карт масти 1. В колоде каждый элемент из этого подмножества должен повторяться один раз.

$M_2=\{1,\ldots,39\}$ - первые 13 элементов этого множества будут составлять подмножество карт для которых масть не имеет значения. В колоде каждый элемент из этого подмножества должен повторяться 2 раза. Вторые 13 элементов этого множества будут составлять подмножество карт масти 1. В колоде каждый элемент из этого подмножества должен повторяться один раз. Третьи 13 элементов этого множества будут составлять подмножество карт масти 2. В колоде каждый элемент из этого подмножества должен повторяться один раз.

Также введём функцию $f(C,X)$ где $C\in M=\{1,\ldots,52\}$ а X некий набор правил для отображения в множества M_0,M_1,M_2.

Благодаря сокращению числа значимых мастей мы можем сильно уменьшить количество вариантов перебора общих карт. Для этого опишем 4 процедуры перебора общих карт, в случае если личные карты каждого из игрока при сравнении отображены в одно и вышеперечисленных множеств.

Процедуру перебора для множества M_0 на языке C++ можно посмотреть в приложении А3.

Процедуру перебора для множества M_1 на языке C++ можно посмотреть в приложении А4.

Процедуры для перебора множества M_1 и M будут идти по аналогии.

Для каждого сочетания из этих групп будет своё правило перебора. Всего получится 6 таких сочетаний: разномастные - разномастные, разномастные - одномастные, разномастные - двухмастные, одномастные - одномастные, одномастные - двухмастные, двухмастные - двухмастные.

Рассмотрим правила перебора для групп разномастные - разномастные. Это одна из самых простых ситуаций. Согласно основному алгоритму зафиксируем карты первого кластера, после чего будем перебирать все возможные варианты карт второго кластера. В процессе этого перебора не будем осуществлять никаких дополнительных вычислений, а просто посчитаем количество возможных вариантов. В данной ситуации возможен лишь один стратегически значимый вариант сочетаний карт двух кластеров. Зафиксируем карты второго кластера и переберём все возможные варианты общих карты. Так как зафиксированные руки не могу собрать комбинацию флаш, зафиксированные карты мы можем отобразить в множество M_0 и перебрать все общие карты соответствующей функцией. После чего умножим полученный результат на количество вариантов полученных при переборе второго кластера.

Процедуру перебора двух кластеров данного сочетания групп на языке C++ можно посмотреть в приложении А5.

Рассмотрим правила перебора для групп разномастные - одномастные. Перед перебором зафиксируем карты карты игрока с одномастной группой. Таким образом при переборе вариантов карт разномастной группы будет $C_4^1 = 4$ стратегически различных варианта карт второго игрока. В зависимости от расположения масти которой первый игрок может собрать флаш. Для каждого из этих возможных вариантов перебираем все возможные варианты расположения мастей которыми нельзя собрать флаш и запоминаем их количество. Количество этих вариантов без фильтрации будет $3! = 6$. После этого для каждого стратегически различного варианта второго игрока зафиксируем карты и

переберём все возможные варианты общих карт. Так как первый игрок может собрать флаш лишь одной мастью а второй игрок не может собрать флаш мы можем отобразить все зафиксированные карты во множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой. Затем умножим полученные результаты на количество вариантов перебора для текущей стратегической значимой комбинации второго игрока.

Процедуру перебора двух кластеров данного сочетания групп на языке C++ можно посмотреть в приложении А6.

Рассмотрим правила перебора для групп разномастные - двухмастные. Перед перебором зафиксируем карты карты игрока с двухмастной группой. Таким образом при переборе вариантов карт разномастной группы будет $C_4^2 * 2! = 6 * 2 = 12$ стратегически различных варианта карт второго игрока. В зависимости от расположения мастей которыми первый игрок может собрать флаш. Для каждого из этих возможных вариантов перебираем все возможные варианты расположения мастей которыми нельзя собрать флаш и запоминаем их количество. Количество этих вариантов без фильтрации будет $2! = 2$. После этого для каждого стратегически различного варианта второго игрока зафиксируем карты и переберём все возможные варианты общих карт. Так как первый игрок может собрать флаш двумя мастями а второй игрок не может собрать флаш мы можем отобразить все зафиксированные карты во множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой. Затем умножим полученные результаты на количество вариантов перебора для текущей стратегической значимой комбинации второго игрока.

Процедуру перебора двух кластеров данного сочетания групп на языке C++ можно посмотреть в приложении А7.

Рассмотрим правила перебора для групп одномастные - одномастные. Это наиболее сложная ситуация. Потому что группа одномастные состоит изтрёх подгрупп: aaaa, aaab и aabc. Правила перебора для каждого сочетания из этих подгрупп будет различаться. Поэтому будем рассматривать перебор каждого из этих сочетаний отдельно.

Рассмотрим вариант перебора для подгрупп aabc - aabc. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 10 стратегически различных вариантов:

1. Когда масть которой второй игрок может собрать флаш совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет $C_4^2 = 6$. У этого сочетания будет только одна значимая масть, все карты можно будет отобразить в множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масть второй карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
4. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масти карт которые не могу собрать флаш у второго игрока не пересекаются с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
5. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

6. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масть второй карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
7. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масти карт которые не могу собрать флаш у второго игрока не пересекаются с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
8. Когда масть которой второй игрок может собрать флаш совпадает с мастью второй карты которой первый игрок не может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
9. Когда масть которой второй игрок может собрать флаш не присутствует у второй игрока и масть второй карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
10. Когда масть которой второй игрок может собрать флаш не присутствует у второй игрока и масти карт которые не могу собрать флаш у второго игрока не пересекаются с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

После перебора каждого из случаев запоминаем количество возможных вариантов карт второго игрока. Затем перебираем все возможные варианты общих карт и умножаем получившийся результат на количество вариантов перебора карт второго игрока.

Процедуру перебора двух кластеров данного сочетания групп данного сочетания подгрупп на языке C++ можно посмотреть в приложении А8.

Рассмотрим вариант перебора для подгрупп aaab - aabc. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 7 стратегически различных вариантов:

1. Когда масть которой второй игрок может собрать флаш совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет $C_4^2 = 6$. У этого сочетания будет только одна значимая масть, все карты можно будет отобразить в множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 4. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масть второй карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 4. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
4. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масти карт которые не могу собрать флаш у второго игрока не пересекаются с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 4. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

5. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
6. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масть второй карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
7. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масти карт которые не могу собрать флаш у второго игрока не пересекаются с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[24][4]=
                                {{1,2,0,0},  //Вариант 1
                                {1,3,0,0},
                                {2,1,0,0},
                                {2,3,0,0},
                                {3,1,0,0},
                                {3,2,0,0},
                                {0,1,3,3},  //Вариант 2
```

```
                {0,2,3,3},
                {0,1,2,2},
                {0,3,2,2},
                {1,0,3,3},  //Вариант 3
                {2,0,3,3},
                {1,0,2,2},
                {3,0,2,2},
                {1,2,3,3},  //Вариант 4
                {2,1,3,3},
                {1,3,2,2},
                {3,1,2,2},
                {0,2,1,1},  //Вариант 5
                {0,3,1,1},
                {2,0,1,1},  //Вариант 6
                {3,0,1,1},
                {2,3,1,1},  //Вариант 7
                {3,2,1,1}};  //Таблица для перебора мастей
    int Variants[7]={6,10,14,18,20,22,24};  //Массив вариантов перебора
    int Types[7]={1,2,2,2,2,2,2}; //Массив типов перебора общих карт для каждого варианта перебора
    int numVariants=7;  //Количество типов вариантов
```

Рассмотрим вариант перебора для подгрупп aabc - aaaa. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 4 стратегически различных варианта:

1. Когда масть которой второй игрок может собрать флаш совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет только одна значимая масть, все карты можно будет отобразить в множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

4. Когда масть которой второй игрок может собрать флаш совпадает с мастью второй карты которой первый игрок не может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[4][4]=
                                {{0,0,0,0},  //Вариант 1
                                {1,1,1,1},  //Вариант 2
                                {2,2,2,2},  //Вариант 3
                                {3,3,3,3}  //Вариант 4
        };  //Таблица для перебора мастей
        int Variants[4]={1,2,3,4};  //Массив вариантов перебора
        int Types[4]={1,2,2,2}; //Массив типов перебора общих карт для каждого варианта перебора
        int numVariants=4;  //Количество типов вариантов
```

Рассмотрим вариант перебора для подгрупп aaab - aaab. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 5 стратегически различных вариантов:

1. Когда масть которой второй игрок может собрать флаш совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 3. У этого сочетания будет только одна значимая масть, все карты можно будет отобразить в множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти,

все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

3. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока и масть первой карты которая не может собрать флаш у второго игрока не совпадает с мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 4. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
4. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока не присутствует у первого игрока. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
5. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[12][4]=
                                {{1,0,0,0},  //Вариант 1
                                {2,0,0,0},
                                {3,0,0,0},
                                {0,2,2,2},  //Вариант 2
                                {0,3,3,3},
                                {1,2,2,2},  //Вариант 3
                                {3,2,2,2},
```

```
                {1,3,3,3},
                {2,3,3,3},
                {2,1,1,1},  //Вариант 4
                {3,1,1,1},
                {0,1,1,1}  //Вариант 5
    }; //Таблица для перебора мастей
    int Variants[5]={3,5,9,11,12};  //Массив вариантов перебора
    int Types[5]={1,2,2,2,2}; //Массив типов перебора общих карт для каждого варианта перебора
    int numVariants=5;  //Количество типов вариантов
```

Рассмотрим вариант перебора для подгруппы aaab - aaaa. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 3 стратегически различных варианта:

1. Когда масть которой второй игрок может собрать флаш совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет только одна значимая масть, все карты можно будет отобразить в множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш совпадает с мастью первой карты которой первый игрок не может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[4][4]=
                                {{0,0,0,0},  //Вариант 1
                                {3,3,3,3},  //Вариант 2
                                {2,2,2,2},
                                {3,3,3,3}  //Вариант 3
        };  //Таблица для перебора мастей
        int Variants[3]={1,3,4};  //Массив вариантов перебора
        int Types[3]={1,2,2}; //Массив типов перебора общих карт для каждого варианта перебора
        int numVariants=3;  //Количество типов вариантов
```

Рассмотрим вариант перебора для подгрупп аааа - аааа. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 2 стратегически различных варианта:

1. Когда масть которой второй игрок может собрать флаш совпадает с мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет только одна значимая масть, все карты можно будет отобразить в множество M_1 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш не присутствует у первого игрока. Количество этих вариантов без фильтрации будет 3. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[4][4]=
                                {{0,0,0,0},  //Вариант 1
                                {1,1,1,1},  //Вариант 2
                                {2,2,2,2},
                                {3,3,3,3}
        };  //Таблица для перебора мастей
```

```
int Variants[2]={1,4};  //Массив вариантов перебора
int Types[2]={1,2}; //Массив типов перебора общих карт для каждого варианта перебора
int numVariants=2;  //Количество типов вариантов
```

Рассмотрим правила перебора для групп одномастные - двухмастные. Для перебора вариантов этой ситуации нам необходимо учесть, что группа одномастных состоит из трёх подгрупп: aaaa, aaab и aabc. Каждую из этих подгрупп будем перебирать с двухмастными отдельно.

Рассмотрим вариант перебора для подгрупп aabb - aabc. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 12 стратегически различных вариантов:

1. Когда масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и масть второй карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и масти карт которые не могу собрать флаш у второго игрока не пересекаются со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

4. Когда масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
5. Когда масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и масть второй карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
6. Когда масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и масти карт которые не могу собрать флаш у второго игрока не пересекаются с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
7. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок и масть второй карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
8. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок и масть второй карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый

игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.

9. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок и масть второй карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
10. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок и масть второй карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
11. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый игрок и масть второй карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
12. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый игрок и масть второй карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.

После перебора каждого из случаев запоминаем количество возможных вариантов карт второго игрока. Затем перебираем все возможные варианты общих карт и умножаем получившийся результат на количество вариантов перебора карт второго игрока.

Процедуру перебора двух кластеров данного сочетания групп данного сочетания подгрупп на языке C++ можно посмотреть в приложении А9.

Рассмотрим вариант перебора для подгрупп aabb - aaab. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 7 стратегически различных вариантов:

1. Когда масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
4. Когда масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и масть первой карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания будет две значимые масти, все карты можно будет отобразить в

множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.

5. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок и масть первой карты которая не может собрать флаш у второго игрока совпадает с первой мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
6. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок и масть первой карты которая не может собрать флаш у второго игрока совпадает со второй мастью которой может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
7. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок и масть первой карты которая не может собрать флаш у второго игрока не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[12][4]=
                                {{1,0,0,0},  //Вариант 1
                                {2,0,0,0},  //Вариант 2
                                {3,0,0,0},
                                {0,1,1,1},  //Вариант 3
                                {2,1,1,1},  //Вариант 4
                                {3,1,1,1},
                                {0,2,2,2},  //Вариант 5
                                {0,3,3,3},
                                {1,2,2,2},  //Вариант 6
                                {1,2,2,2},
                                {2,3,3,3},  //Вариант 7
                                {3,2,2,2}};
```

```
        int Variants[7]={1,3,4,6,8,10,12};  //Массив вариантов перебора
        int Types[7]={2,2,2,2,3,3,3,3}; //Массив типов перебора общих карт для каждого варианта
перебора
        int numVariants=7;  //Количество типов вариантов
```

Рассмотрим вариант перебора для подгрупп aabb - aaaa. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 3 стратегически различных варианта:

1. Когда масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы все масти.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[4][4]=
                                {{0,0,0,0},  //Вариант 1
                                {1,1,1,1},  //Вариант 2
                                {2,2,2,2},  //Вариант 3
                                {3,3,3,3},
                        };  //Таблица для перебора мастей
        int Variants[3]={1,2,4};  //Массив вариантов перебора
        int Types[3]={2,2,3}; //Массив типов перебора общих карт для каждого варианта перебора
```

```
int numVariants=3;  //Количество типов вариантов
```

Рассмотрим правила перебора для групп двухмастные - двухмастные. Зафиксируем карты первого игрока. В зависимости от расположения мастей у карт второго игрока в такой ситуации возможно 7 стратегически различных вариантов:

1. Когда первая масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и вторая масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
2. Когда первая масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и вторая масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш. Количество этих вариантов без фильтрации будет 1. У этого сочетания будет две значимые масти, все карты можно будет отобразить в множество M_2 и осуществить перебор общих карт соответствующей для этого множества процедурой.
3. Когда первая масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и вторая масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
4. Когда первая масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и вторая масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
5. Когда вторая масть которой второй игрок может собрать флаш совпадает с первой мастью которой первый игрок может собрать флаш и первая масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.

6. Когда вторая масть которой второй игрок может собрать флаш совпадает со второй мастью которой первый игрок может собрать флаш и первая масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.
7. Когда первая масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок и вторая масть которой второй игрок может собрать флаш не совпадает с мастями которыми может собрать флаш первый игрок. Количество этих вариантов без фильтрации будет 2. У этого сочетания значимы будут все масти.

После перебора этих вариантов делаем аналогичные действия что и в предыдущей ситуации.

Для процедуры на языке C++ нам нужно будет изменить значение нескольких переменных в процедуре для предыдущей ситуации:

```
char TD[12][4]=
                                {{0,0,1,1},  //Вариант 1
                                {1,1,0,0},  //Вариант 2
                                {0,0,2,2},  //Вариант 3
                                {0,0,3,3},
                                {1,1,2,2},  //Вариант 4
                                {1,1,3,3},
                                {2,2,0,0},  //Вариант 5
                                {3,3,0,0},
                                {2,2,1,1},  //Вариант 6
                                {3,3,1,1},
                                {2,2,3,3},  //Вариант 7
                                {3,3,2,2},;  //Таблица для перебора мастей
    int Variants[7]={1,2,4,6,8,10,12};  //Массив вариантов перебора
    int Types[7]={2,2,3,3,3,3,3}; //Массив типов перебора общих карт для каждого варианта перебора
    int numVariants=7;  //Количество типов вариантов
```

Расчёт таблицы

В ходе разработки программы, для эффективной работы основного алгоритма программы будет понадобилось рассчитать некоторые предрасчётные данные. Для этого была разработана отдельная программа на языке C++ в рамках которой были реализованы алгоритмы описанные в предыдущих частях.

Из-за высокой сложности реализуемых алгоритмов остро встал вопрос верификации полученных данных. Для проверки результатов был выбран эквилятор propokertools.com. Клиент-серверная архитектура данного инструмента позволила легко и быстро интегрировать программу для расчёта таблицы по средствам протокола HTTP. В результате чего любой получаемый результат мог быть автоматически проверен сторонним эквилятором. На каждом этапе верификации программ случайным образом генерировала входные данные после чего проверяла полученный результат с помощью эквилятора. Из-за ограничений со стороны web-сервера propokertools.com на каждом этапе проверки удавалось верифицировать не более 1000 наборов входных данных и результатов.

Алгоритм расчёта теоретический мог выполнятся параллельно, поэтому после реализации однопоточной версии была разработана и верифицирована многопоточная версия программы. После этого была проведена оценка эффективности расчётов и прогнозирования времени расчёта полной таблицы. Для этого было проведено тестирование на 10000 случайных входных данных. Работая в 10 потоков на процессоре Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz программа произвела расчёты для 10000 входов за 56 секунд. На основании полученного результата с помощью интерполяции было предсказано время расчёта всей таблицы:

$\frac{M}{K} * t = \frac{135005312}{10000} * 56 = 756029{,}7472$ секунд, где

$M = \frac{16432 * 16432}{2} = 135005312$ - количество уникальных ячеек в таблице,

$K = 10000$ - размер выборки для тестирования,

$t = 56$ - время обработки тестового набора.

Таким образом одному компьютеру на расчёт таблице понадобится порядка

$\frac{756029}{60*60*24} \approx 9$ суток.

Для увеличения эффективности расчётов была разработана стратегия пакетной обработки данных. Так таблица была разбита на 20 частей равного размера и сложности. Это позволило проводить вычисления на нескольких компьютерах одновременно. При добавлении нескольких компьютеров схожей мощности удалось посчитать заданную таблицу за 48 часов.

После расчётов были полученные данные были собраны в одну таблицу. Размер собранной таблицы составил 6480649344 байт. После агрегации таблицы был проведён очередной этап верификации. Теперь проверялась не работа алгоритма а результаты записанные в произвольно выбранную ячейку.

В заключении работы с таблицей была проведена редукция избыточных данных. Так в одной ячейки хранится распределение для 12 различных исходов. Для задачи нахождения математического ожидания действия AllIn в игре для двух игроков нам будет достаточно знать математическое ожидание выигрываемой части банка (далее equity). Для этого нам не обязательно хранить все 12 исходов. Достаточно хранить только два значения: equity и суммарное количество возможных исходов. Что бы минимизировать потери точности от данного перехода величину equity было решено хранить в формате double. В результате вместо 12*4=48 байт на одну ячейку таблицу удалось сократить до 1*8+1*4=12 байт на ячейку или в 4 раза. Также не стоит забывать, что расчёт equity для Omaha и Omaha Hi/Lo будет отличатся, поэтому сокращённую таблицу придётся хранить в двух экземплярах.

Расчёт равновесия по Нэшу

В рамках данной ВКР была создана программа для расчёта равновесных стратегий с помощью метода фиктивного разыгрывания. Благодаря использованию подсчитанной ранее таблицы для вычисления математического ожидания действий удалось достичь приемлемой производительности. Так расчёт одной итерации на компьютере с процессором Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz занимает в среднем 1.3 секунды.

Помимо скорости итерации метод продемонстрировал хорошую сходимость.

Для вычисления сходимости была использована следующая функция невязки

$$\varepsilon = \frac{PostEV\,1 + PostEV\,2}{2}$$, где

$PostEV\,1$ - математическое ожидание эксплотирующей стратегии для действия AllIn первого игрока против рассчитанной после текущей итерации равновесной стратегии второго игрока для действия Call,

$PostEV\,2$ - математическое ожидание эксплотирующей стратегии для действия Call второго игрока против рассчитанной после текущей итерации равновесной стратегии первого игрока для действия AllIn.

Таким образом для ситуации когда у каждого игрока по 100 фишек, обязательные ставки: 5 фишек - малый блаинд, 10 фишек - большой блаинд, после 100 итераций $\varepsilon = 0.014$. Это составляет менее 0.01% от призового фонда. Ниже приведён график иллюстрирующий процесс сходимости, по оси X количество пройденных итераций, по оси Y значение ε :

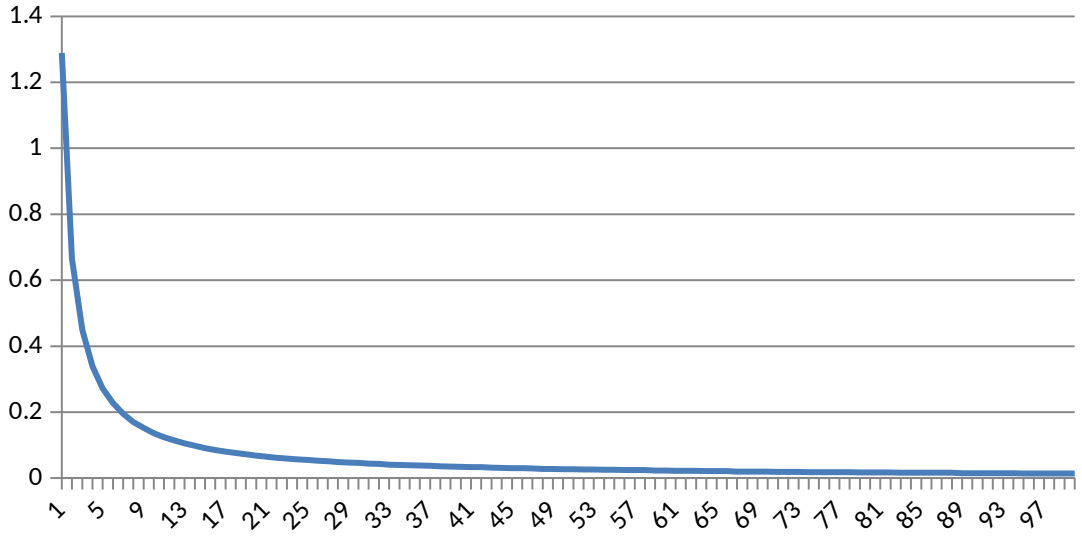

Рис.2 График сходимости

На основе низкого значения невязки после 100 итераций можно сделать вывод, что полученные стратегии являются ε - равновесными по Нэшу согласно определению, так как ни один игрок не может увеличить своё математическое ожидание изменив свою стратегию в одностороннем порядке.

Для более удобного использования программы был разработан GUI с помощью фрейморка Qt. Графический интерфейс позволяет указать следующие параметры для расчётов: тип игры, размеры обязательных ставок, количество фишек у игроков, количество итераций для расчёта.

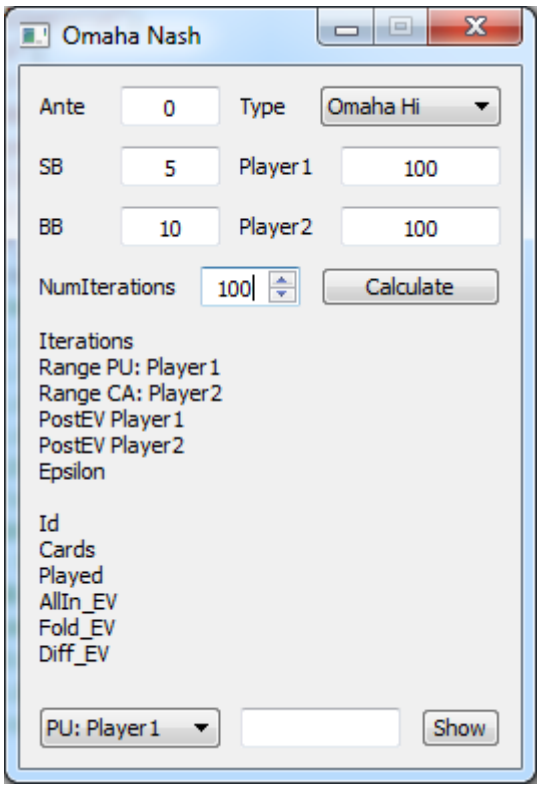

Рис.3 Окно для ввода данных

Программа может отображать получаемые данные как во время выполнения расчётов

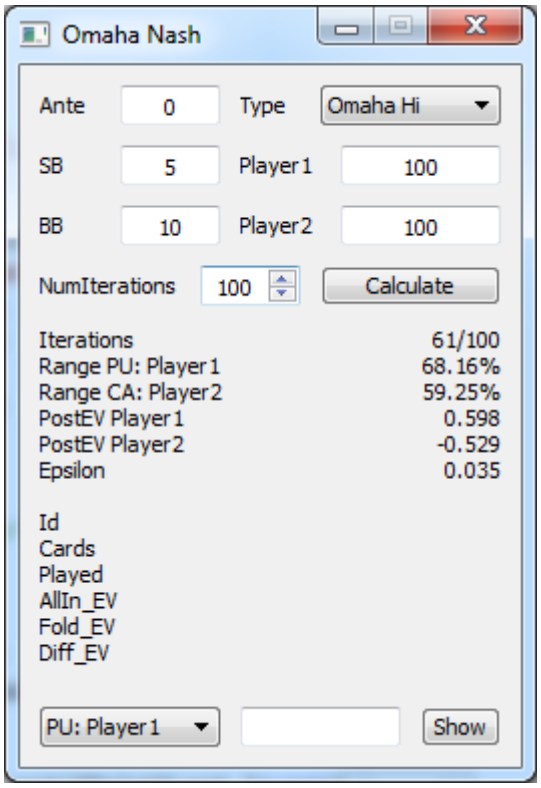

Рис.4 Интерфейс во время работы программы

так и после их завершения

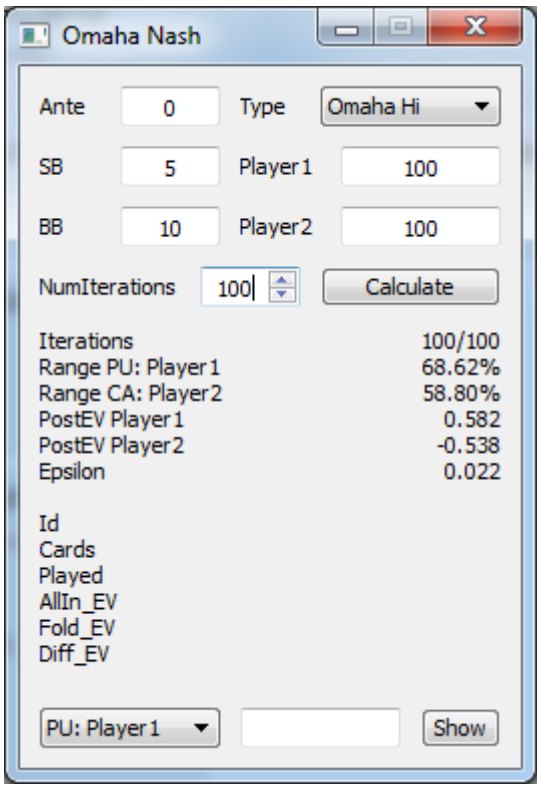

Рис.5 Отображение результатов

После расчётов также имеет возможность посмотреть математические ожидание конкретного набора карт. Выводятся математическое ожидание действия AllIn и действия Fold, а также их разница. Кроме математического ожидания напротив надписи "Played" выводится вероятность с которой нужно играть текущую карту при смешанной равновесной стратегии. Карты можно выбрать с помощью ввода номера кластера:

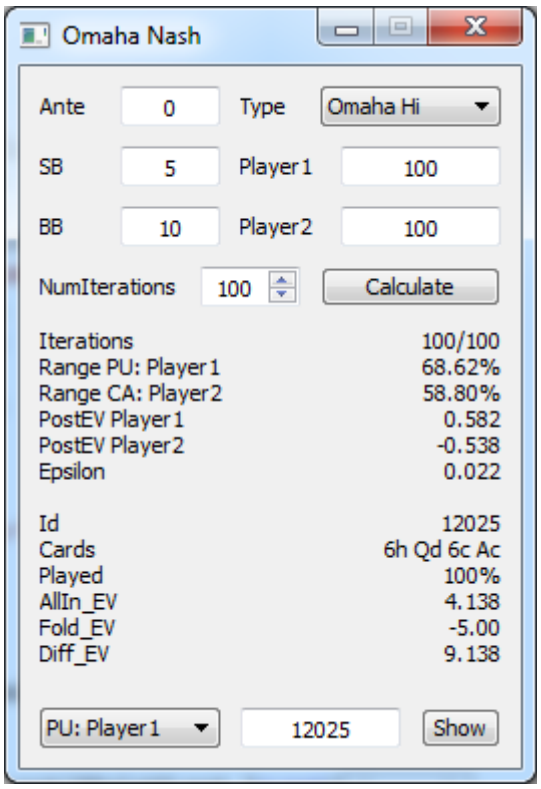

Рис.6 Выбор карты по Id

А также с помощью ввода карт в текстовой нотации:

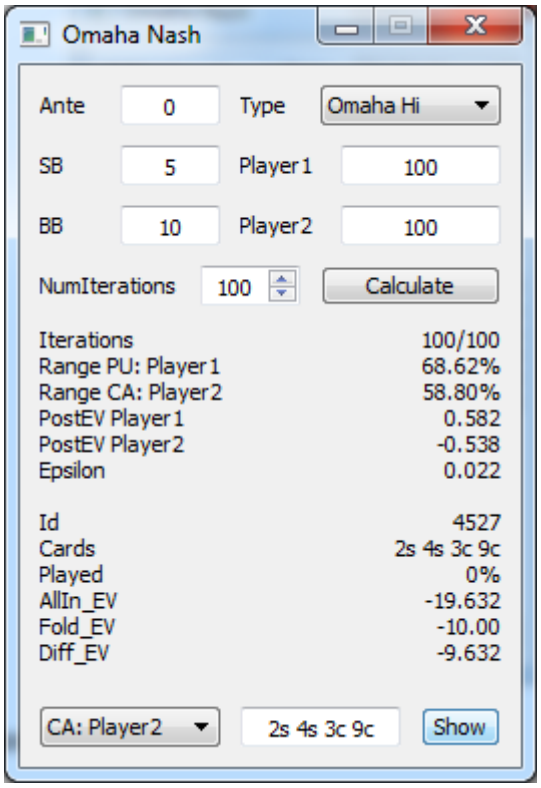

Рис.7 Выбор карты текстовой нотацией

Интерпретация полученных результатов

При изучении равновесных стратегий в модели jam-fold для двоих игроков можно заметить тот факт, что стратегии зависят только от соотношения максимально-возможного банка розыгрыша и обязательных ставок (далее эффективный стек). На основе этой особенности можно построить таблицу зависимости процента играемых в равновесии карт и эффективного стека для трёх игр: Texas hold'em, Omaha и Omaha Hi/Lo.

Таблица 4. Стратегии для действия AllIn первого игрока

Размер эффективного стека	Texas hold'em	Omaha	Omaha Hi/Lo
2	89.4	98.8	99.3
4	73.8	85.3	91.2
6	68.3	76.4	83.7
8	61.9	71.6	77.3
10	58.3	67.9	73.2
15	45.7	59.8	68.7
20	40.2	53.6	58.5

Таблица 5. Стратегии для действия Call второго игрока

Размер эффективного стека	Texas hold'em	Omaha	Omaha Hi/Lo
2	100.0	100.0	99.9
4	74.1	98.1	99.1
6	54.4	84.9	93.2
8	45.0	70.4	85.1
10	37.4	59.6	74.1
15	28.5	44.1	49.3
20	21.7	35.3	38.9

Как видно из таблиц для игр типа Omaha равновесные стратегии более "агрессивны" чем для Texas Holdem. Практический во всех ячейках значения процента играемых карт для этих игр выше аналогичных у Texas Holdem более чем на 10 пунктов. Данный эффект можно объяснить тем, что по правилам Omaha сила начальных карт игроков распределена менее дисперсионно чем у Texas Holdem.

Также можно заметить, что стратегии в Omaha Hi/Lo в среднем содержат больший процент играемых карт чем в обычной Omaha. Это можно связать с тем, что из-за дополнительного розыгрыша банка по правилам Lo при вскрытии карт будут часто возникать ситуации деления банка. Т. е. даже при слабых картах вероятность проиграть все фишки будет ниже.

Заключение

В настоящие время популярность игры Omaha растёт. Постепенно появляется всё больше регулярных турниров по этой игре на сайте Pokerstars.net. При этом стратегии для этой игры пока слабо изучены. Ощущается нехватка инструментов для её анализа и изучения. Приложения разработанное в качестве дипломного проекта позволит изучить равновесные jam-fold стратегии в игре для двух человек. Также на основе этого приложения можно проводить исследования не только отдельных раздач, но их последовательности в виде турнира. В частности есть аналогичные исследования для Texas holdem в которых оценивается зависимость вероятности победы в турнире от количество фишек у игроках [14]. На основе рассчитанных в ходе выполнения проекта таблиц можно будет эффективно вычислять математическое ожидание для игры Omaha. Это позволит анализировать стратегии для ситуаций с большим количество игроков. А в последствии создавать калькуляторы для Omaha по функционалу не уступающие текущим калькулятором для игры в Texas holdem.

Список источников

1) Alan Turing vs Alick Glennie. 1952. Turing Test.
2) WSOP 2005 [Электронный ресурс]/ Oddschecker. Режим доступа: http://poker.oddschecker.com/poker-features/wsop-world-series-of-poker/wsop-history/wsop-2005.html (Дата обращения: .2014)
3) A. Gilpin and T. Sandholm. Lossless abstraction of imperfect information games. Journal of the ACM, 54(5), 2007.
4) Billings, D.: Algorithms and assessment in computer poker. PhD thesis, Edmonton, Alta., Canada (2006)
5) Online Poker Traffic. [Электронный ресурс]/– Pokerscout © - Online Poker Traffic. Режим доступа: http://www.pokerscout.com/SiteDetail.aspx?site=PokerStars&ab=5712225(Дата обращения: .2012)
6) Покер - Правила покера - Правила техасского холдема. [Электронный ресурс]/ – PokerStars. Режим доступа: http://www.pokerstars.eu/ru/poker/games/rules/ (Дата обращения: .2014)
7) Комбинации карт в покере - Покерные комбинации по старшинству. [Электронный ресурс]/ PokerStars. Режим доступа: http://www.pokerstars.eu/ru/poker/games/rules/hand-rankings/ (Дата обращения: .2014)
8) HoldemResources - HoldemResources Calculator. [Электронный ресурс]/ /HoldemResources. Режим доступа: http://www.holdemresources.net/h/products/hrc.html (Дата обращения: .2014)
9) Introduction to the SitNGo Wizard 2 [Электронный ресурс]/ The SitNGo Wizard. Режим доступа: http://sngwiz.com/wp/home/documentation/sitngo-wizard-2-documentation/introduction-to-the-sitngo-wizard-2/ (Дата обращения: .2014)
10) ProPoketTools Online Simulator / Equity Calculator [Электронный ресурс]/ ProPokerTools. Режим доступа: http://www.propokertools.com/simulations (Дата обращения: .2014)
11) Bill Chen and Jerrod Ankenman. 2006.The Mathematics of Poker, Pittsburgh.
12) Brian Alspach (2003). "Rank Sets and Straights". Retrieved 2006-10-30.
13) Robinson, J. (1951) "An Iterative Method of Solving a Game", *Annals of Mathematics* 54, 296-301.
14) Peter Bro Miltersen and Troels Bjerre Sørensen. 2007.A Near-Optimal Strategy for a Heads-Up No-Limit Texas Hold'em Poker Tournament .