Подсчёт вероятностей в покере

Выполнил: Ильин Кирилл 6371

Суть работы

- Суть работы заключается в подсчёте вероятностей с которыми игрок может получить комбинации при определённом сбросе карт. С помощью этой библиотеки можно будет написать советчика, который будет подсказывать с каким сбросом у игрока будут лучшие шансы.

Содержание презентации

- Хронология разработки (для понимания)
  - Полный перебор
  - Использование шаблонов
- Раскрытые шаблоны
- Вспомогательная программа
- Работа с факториалом
- Вывод о проделанной работе

Полный перебор

- Изначально был сделан перебор всех возможных комбинаций, который проверял все C[47, 5-карты пользователя] комбинаций. Предположим, что пользователь сбросил все карты и число перебора становится внушительным (1 533 939). При этом для каждой нужно проверить является ли она комбинацией, что увеличивает это число в несколько раз. Отсюда становится очевидным, что данный алгоритм очень не оптимизирован и будет долго работать.

Использование шаблонов

- Из-за неэффективности алгоритма перебора я решил не перебирать все варианты, которые могут быть у пользователя. А вместо этого смотреть под какие из комбинаций подходят те карты, которые у него на руках и может ли он дополнить их до комбинации. Реализовать это я решил через шаблоны.
- Комбинации задавались в виде шаблонов. Например, для стрит-флеша:
  - «(n, a)(n+1, a)(n+2, a)(n+3, a)(n+4, a)»
  - «(A, a)(2, a)(3, a)(4, a)(5, a)»
- Но это решение не увидело свет из-за большого числа дополнительных модулей, которые необходимо реализовать для его работы. На его место пришло менее эффективное решение, но оно всё равно намного быстрее полного перебора.

Раскрытые шаблоны

- Я принял решение заранее раскрыть шаблоны (написать все возможные заготовки, которые под них подходят) для комбинаций. Делать это руками конечно же глупо и для этого была написана дополнительная программа, о которой будет рассказано на следующем слайде.
- Благодаря заранее раскрытым шаблонам, программе достаточно проверить соответствие карт пользователя с каждой заготовкой по следующим правилам
  - Если у игрока 5 карт, то она просто сравнивает с заготовкой (при совпадении комбинация засчитывается)
  - Если у игрока меньше 5 карт, то она смотрит на наличие у пользователя карт, которых нет в комбинации (при наличии таких карт комбинация не засчитывается по очевидным причинам)
  - Если у игрока меньше 5 карт и все есть в комбинации, то проверяется, остались ли в колоде недостающие карты (если остались, то засчитывается)
- Таким образом мы высчитали количество благоприятных случаев для каждой комбинации, но нужно найти вероятность и для этого нам надо всего лишь разделить найденное количество на число всех возможных комбинаций (количество комбинаций полного перебора)
- Таким образом мы получаем вероятность выпадения комбинации. Проделав так для каждой мы получим искомую цель. Данный алгоритм проходит 74624 комбинаций, что значительно меньше, чем в полном переборе.

Вспомогательная программа для раскрытия шаблонов

- В программе буква «a» означает все масти, но одновременно одинаковые, а «*» означает все масти независимо от остальных
- Принцип работы программы:
  - Программа сортирует карты шаблона (не учитывая масть) по возрастанию и подставляет на место n и m все возможные карты (чтобы они нигде в шаблоне не дали несуществующую карту)
  - Потом она работает с мастями и заменяет все буквы «a» на масти (D, H, S, C)(Diamonds, Hearts, Spades, Clubs)
  - Таким образом каждая строка с «a» превращалась в 4 строки.
  - При раскрытии «*» сначала была допущена ошибка, которая забрала много времени на поиск, я не учёл тот момент, что нам не важен порядок.
  - Для раскрытия звёздочек выполняется поиск карт одинакового достоинства с мастью «*» (далее это называется группой).
  - Потом идёт перебор всех групп без учёта порядка, перебирая в каждой группе масти для карт (так же без учёта порядка). Пример на следующем слайде.

Пример раскрытия звёздочек

Цветами и подчёркиваниями выделены особенности перебора, облегчающие понимание

- (2, *)(2, *)(4, *)
- (2, D)(2, H)(4, D)
- (2, D)(2, H)(4, H)
- (2, D)(2, H)(4, S)
- (2, D)(2, H)(4, C)
- (2, D)(2, S)(4, D)
- (2, D)(2, S)(4, H)
- (2, D)(2, S)(4, S)
- (2, D)(2, S)(4, C)
- (2, D)(2, C)(4, D)
- (2, D)(2, C)(4, H)
- (2, D)(2, C)(4, S)
- (2, D)(2, C)(4, C)
- (2, H)(2, S)(4, D)
- (2, H)(2, S)(4, H)
- (2, H)(2, S)(4, S)
- (2, H)(2, S)(4, C)
- (2, H)(2, C)(4, D)
- (2, H)(2, C)(4, H)
- (2, H)(2, C)(4, S)
- (2, H)(2, C)(4, C)
- (2, S)(2, C)(4, D)
- (2, S)(2, C)(4, H)
- (2, S)(2, C)(4, S)
- (2, S)(2, C)(4, C)
- Выделив 2 пары (это двойки и четверки) начнём перебирать все комбинации четверки (их 4) с первой комбинацией двоек, потом со 2, потом с третьей и т.д.
- Комбинации внутри групп находятся точно так же как и находились комбинации вне групп. Для удобства понимания можно представить это числами:

```
0 1 2 3
D H S C
```

```
(01) (0) (12) (0)
(01) (1) (12) (1)
(01) (2) (12) (2)
(01) (3) (12) (3)
(02) (0) (13) (0)
(02) (1) (13) (1)
(02) (2) (13) (2)
(02) (3) (13) (3)
(03) (0) (23) (0)
(03) (1) (23) (1)
(03) (2) (23) (2)
(03) (3) (23) (3)
```

Работа с факториалом

- Для вычисления C нам понадобиться вычислить несколько факториалов и потом их делить. Это очень тяжёлые вычисления и хотелось бы это как-то облегчить
- Я понял, что факториал можно представить в виде произведения простых чисел в степенях (ведь факториал n это произведение чисел до n, а мы знаем, что любое число можно представить произведением простых чисел). Я реализовал это и при работе с факториалами я работал с простыми числами и их степенями, тем самым сведя всё к вычислению 1 факториала в самом конце.

Перевод факториала в простые числа

- Если n<0, то факториал 0; n==0, то факториал 1; n=2 или n=1, то факториал равен n
- $7!=1*2*3*4*5*6=1*2*3*2*2*5*2*3=1*2^4*3^2*5$
- Посмотрев на эту запись становится очевидным, что для разложения надо просто разложить числа [2..6] на простые множители, сделать это можно с помощью простого целочисленного деления, вычисляя количество простых чисел в числе.

Алгоритм:
Проходим по всем простым числам (в моей работе не потребуются большие числа и я просто внёс нужные в массив) и целочисленно делим факториал на это число до тех пор, пока не получится 0 (результаты деления добавляем к степени этого числа). А если число с первого раза выдало 0, то дальше идти нет смысла.

Вывод о проделанной работе

- Я выполнил поставленную задачу и сделал библиотеку, которая рассчитывает нужные вероятности эффективным алгоритмом и её можно легко использовать для программирования программы-советчика в покере.

- Спасибо за внимание! :3