t(n, input) - время работы алгоритмы при входных данных input размера n. Тогда время работы алгоритма $t(n) = max_{input} t(n, input)$.

$t(n) = O(f(n)) \Leftrightarrow \exists c > 0,\ N > 0 : \forall n > N\ t(n) \leq c \cdot f(n)$.

$t(n) = o(f(n)) \Leftrightarrow \forall c > 0, \exists N : \forall n > N\ t(n) \leq c \cdot f(n)$.

$t(n) = \Omega(f(n)) \Leftrightarrow \exists c > 0,\ \forall N,\ \exists n > N,\ input : t(n, input) \geq c \cdot f(n)$.

$t(n) = \omega(f(n)) \Leftrightarrow \forall c > 0,\ \forall N,\ \exists n > N,\ input : t(n, input) \geq c \cdot f(n)$.

$t(n) = \theta(f(n)) \Leftrightarrow t(n) = \Omega(f(n)),\ t(n) = O(f(n))$.

Алгоритм является полиномиальным, если $t(input) = O(|input|^k)$. $|input|$ - битовая длина.

Сильно полиномиальный алгоритм - $t(n) = O(Poly(n))$ - string-poly

Слабо полиномиальный алгоритм - $O(Poly(n, \log C))$ - weak-poly

Псевдо полиномиальный алгоритм - $O(Poly(n, C))$ - pseudo-poly

Алгоритмы и структуры данных, семинар 2. Решаем вместе задачи на комбинаторную и геометрическую вероятности.

31 октября 2019

1 Теоретический материал

Обратите внимание, что большинство приведённых ниже опредлений справделивы **только** для случая конечного множества элементарных исходов, хотя и сохраняют свой смысл при обобщении. Особенно будьте осторожны с определениями касающимися **случайных величин**.

1. Вероятностное пространство для случая **конечного** множества элементарных исходов это тройка $(\Omega, 2^\Omega, \mathbb{P})$, где:
   - Ω — множество элементарных исходов. Элементарные исходы обозначаются как $w \in \Omega$;
   - 2^Ω — множество событий, каждое событие является подмножеством множества элементарных исходов. Например: $A \in 2^\Omega, A \subset \Omega$.
   - $\mathbb{P}$ — вероятности событий. Вероятность является числом от 0 до 1. В рассматриваемом нами случае вероятность события определяется как сумма вероятностей составляющих его элементарных исходов: $P(A) = \sum_{w \in A} P(\{w\})$.
2. Если $P(A) = 0$, то событие A называют невозможным.
3. Если $P(A) = 1$, то событие A называют достоверным.
4. События A и B называют независимыми, если вероятность того, что событие B произойдёт не меняется от того, что произошло событие A. Формальнее, будем называть A и B независимыми $(P(A) > 0, P(B) > 0$, если $P(A \cap B) = P(A) \cdot P(B)$.
5. Если $P(A) > 0$ и $P(B) > 0$, но $P(A \cap B) = 0$, то события A и B называются несовместными.
6. События $A_1, A_2, \ldots, A_n$ называются попарно независимыми, если любые два из них являются независимыми.
7. События $A_1, A_2, \ldots, A_n$ называются независимыми в совокупности, если для любого набора из этих событий $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ верно $P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}) = P(A_{i_1}) \cdot P(A_{i_2}) \cdot \ldots \cdot P(A_{i_k})$.
8. Вероятность наступления A при условии наступления B записывается как $P(A|B)$. Если $P(B) > 0$, то справедливо $P(A|B) = \frac{P(A \cap B)}{P(B)}$.
9. События $A_1, A_2, \ldots, A_n$ называются полной группой событий, если любой элементарный исход принадлежит **ровно** одному из них.
10. Пусть события $A_1, A_2, \ldots, A_n$ образуют полную группу, тогда имеет место: $P(B) = \sum_{i=1}^{n} P(B|A_i) \cdot P(A_i)$.

2 Задания

1. Обсудите понятия попарной независимости событий и независимости в совокупности. Приведите пример трёх событий, которые являются попарно независимыми, но не являются независимыми в совокупности.

2. Обсудите понятие полной группы событий. Решите следующую задачу. Имеется три корзины. Изначально в первой корзине находится два белых шара и один чёрный, во второй три белых и два чёрных, а третья корзина пуста. Из обеих корзин случайно выбирают по два шара и кладут в третью. Затем из третьей корзины равновероятно выбирают один шар, найдите вероятность того, что этот шар будет чёрным.

3. Какова вероятность того, что при 8 бросаниях честной монеты герб выпадет в точности 5 раз?

4. На полке в случайном порядке расставлено 40 книг, среди которых находится трехтомник Пушкина. Найти вероятность того, что эти тома стоят в порядке возрастания номера слева направо, но не обязательно рядом.

5. Двое играют в русскую рулетку с шестизарядным револьвером, в который вставлены два патрона. Первый прокрутил барабан и нажал на курок — пистолет не выстрелил. Пришёл черёд второго, он может выбрать, сразу нажать на курок или сначала прокрутить барабан. Как ему выгоднее поступить, если известно, что он не хотел бы выстрела и

   (a) патроны были вставлены в две соседние ячейки;

   (b) патроны были вставлены в две противоположные ячейки;

   (c) патроны были вставлены в две случайные ячейки.

6. Задача. У Стаса есть 3 красных, 4 синих и 5 зелёных вагончиков, из которых ему хочется собрать паровозик. Сколько есть способов сделать это так, чтобы все вагончики были использованы и никакие два синих вагончика не стояли рядом? Вагончики одного цвета считаются полностью одинаковыми.

7. Обсудите понятие геометрической вероятности. Решите следующую задачу. Расстояние между пунктом А и пунктом Б составляет 100 километров. Известно, что запорожец и волга выехали в какие-то случайные моменты времени между 12:00 и 14:00, при этом волга выехала после запорожца. Скорость запорожца составляет 50 километров в час, а скорость волги — 100 километров в час. Найдите вероятность того, что волга приедет в пункт Б раньше.

8. Дана следующая реализация функции *random_shuffle*:

```
void random_shuffle(std::vector<int>& array) {
    for (size_t i = 0; i < array.size(); ++i) {
        std::swap(array[i], array[rand(0, i)]);
    }
}
```

   Верно ли, что функция равновероятно получает любую перестановку массива (при условии, что все элементы массива различны)?

9. Студент на экзамене играет с преподавателем в следующую игру. На столе имеется три перевернутых текстом вниз листочка с вопросом(только к одному из листочков студент перед экзаменом подготовился).

   Далее студент показывает пальцем на какой-то листочек, а преподаватель открывает один из двух оставшихся листочков, к которому студент гарантированно не готовился.

   Стоит ли студенту взять тот листочек, на который он показывал пальцем, или стоит взять оставшийся неоткрытый, если студент хочет максимизировать вероятность сдачи экзамена?

10. У вас имеется честная монета.

    (a) Предложите трём игрокам способ равновероятно выбрать победителя.

    (b) Покажите, что не существует способа этого сделать, используя фиксированное количество бросков монеты.

11. У сайта имеется n рекламных баннеров, причём владелец баннера i заплатил c_i рублей. Схема выбора называется честной, если вероятность p_i каждого баннера быть выбранным пропорциональна величине c_i. Считая все числа c_i целыми, а ваш генератор случайных чисел идеальным и способным равновероятно возвращать любое целое число в заданном диапазоне, предложите схему:

    (a) Честного выбора одного баннера из n.

    (b) Честного выбора k различных баннеров из n.

Алгоритмы и структуры данных.
Упражняемся в математическом ожидании.

1 Теоретический материал

Обратите внимание, что большинство приведённых ниже опредлений справделивы **только** для случая конечного множества элементарных исходов, хотя и сохраняют свой смысл при обобщении. Особенно будьте осторожны с определениями касающимися **случайных величин**.

1. Случайная величина ξ есть веществennозначная функция элементарного исхода, то есть $\xi : \Omega \to \mathbb{R}$.
2. Две случайные величины ξ и ψ независимы, если для любых $c, d \in \mathbb{R}$ верно, что события $A = \{\omega : \xi(\omega) = c\}$ и $B = \{\omega : \psi(\omega) = d\}$ независимы.
3. Математическим ожиданием случайной величины называется её средневзвешенное значение $E\xi = \sum_{w \in \Omega} \xi(w) \cdot P(w)$.
4. Линейность математического ожидания. Математической ожидание линейной комбинации случайных величин равно соответствующей линейной комбинации их матожиданий, при этом неважно, выполненяется ли условие независимости. $E(\alpha\xi + \beta\psi) = \alpha E\xi + \beta E\psi$.
5. Математическое ожидание произвдения **независимых** случайных величин равно произведению математический ожиданий, то есть $E(\xi \cdot \psi) = E\xi \cdot E\psi$.
6. Дисперсией случайной величины называется среднеквадратичное её отклонение от математического ожидания, или $D\xi = E(\xi - E\xi)^2$.
7. Для оценки вероятности отклонения неотрицательной случайной величины от математического ожидания используют неравенство Маркова:
$$P(\xi \geq a) \leq \frac{E\xi}{a}$$
8. И Чебышёва (требование неотрицательности не накладывается):
$$P(|\xi - E\xi| \geq a) \leq \frac{D\xi}{a^2}$$

2 Задания

1. В магазине имеется 10 автомобилей. Из них 5 черных, 3 красных и 2 синих. Некоторый покупатель случайным равновероятным образом выбрал 3 автомобиля. Найдите распределение (в данном случае подразумевается вероятность принять каждое конкретное значение) случайной величины «количество чёрных автомобилей среди выбранных». Найдите математическое ожидание этой случайной величины.

2. Найти математическое ожидание модуля разности двух независимых равномерно распределённых целых чисел от 1 до n. Укажите асимптотическую формулу и коэффициент при старшей степени.

3. Рассмотрим следующую игру: вы можете бросить обычный шестигранный кубик, после чего решаете оставить выпавшее на нём число, или отказаться от него и бросить заново, при этом отказаться можно не более чем два раза. Предложите стратегию игры, при которой максимизируется математическое ожидание числа, выпавшего на последнем кубике.

4. У Аркадия есть n карточек с числами, на каждой карточке написано некоторое целое число a_i(числа могут быть как положительными, так и отрицательными). Аркадий каждый вечер поступает следующим образом: он раскладывает карточки числами вниз в случайном порядке слева направо(каждая перестановка карточек реализуется с равной вероятностью) и начинает их по очереди открывать также слева направо. Во время этого процесса он складывает в уме все числа, встреченные на открытых карточках, но как только встречает карточку с отрицательным числом, он прибавляет его к ответу, заканчивает процесс игры и произносит вслух текущую сумму.

   Аркадий занимается этим уже несколько десятилетий и теперь ему стало интересно мат. ожидание числа, которое он произнесет. Придумайте алгоритм, который за время $O(n)$ определит это мат. ожидание.

5. n участников играют в игру, состоящую из n раундов. Каждому участнику i сопоставлено некоторое положительное число a_i. Каждый раунд проходит следующим образом. Из оставшегося в игре множества $\{v_1, v_2, \ldots, v_k\}$ участников выбирается случайный: участник с номером v_j выбирается с вероятностью $a_{v_j}/(a_{v_1} + a_{v_2} + \ldots + a_{v_k})$. После выбора нового участника, он считается победителем этого раунда и выходит из игры. Опишите алгоритм, который для каждого участника определяет мат. ожидание номера раунда, в котором он победит. Алгоритм должен работать за время $O(n^2)$.

6. Покажите, что умножение случайной величины на константу c умножает её математическое ожидание на c, а её дисперсию на c^2.

7. Как вы помните, математическое ожидание обладает свойством линейности, то есть математическое ожидание суммы равняется сумме математических ожиданий. Удивительно, но этим же свойством обладает и дисперсия, если конечно величины независимы. Докажем же это!

8. Скомбинируем теперь результаты предыдущих двух пунктов. Пусть у вас есть некоторый n-гранный кубик, на гранях которого написаны неизвестные вам числа a_i. Вы бросаете кубик и наблюдаете итоговое значение. Вы хотели бы оценить ожидаемое значение, которое выпадает на кубике. Для этого вы делаете m независимых бросков и берёте среднее арифметическое. Разумеется, это значение можно также трактовать как случайную величину, являющуюся средним m независимых одинаково распределённых случайных величин. Что вы можете сказать про её математическое ожидание и дисперсию?

9. Докажите неравенства Маркова и Чебышёва.

- unit test (обычные, запускаем просто тесты)
- integration (разные компоненты программы нормально живут вместе)
- prod (тестирование от самого начала до конца)

Алгоритмы и структуры данных.
Сортировки и порядковые статистики, решаем вместе.

1. Обсудите рандомизированный алгоритм поиска порядковой статистики. Используя схему доказательства асимптотики алгоритма по индукции, покажите, что рандомизированный поиск порядковой статистики работает за ожидаемое линейное время.

2. Покажите, как реализовать алгоритм сортировки вставками, чтобы время его работы составляло $O(n + inv)$, где inv — количество инверсий в массиве.

3. Пусть у вас есть массив с элементами некоторого типа, для любой пары элементов вам доступна операция сравнения. Некоторые элементы равны, но при этом всё равно отличимы по вспомогательной информации. Разрешается изменять вспомогательную информацию элементов (то есть никак не используемую при сравнении). Также имеется функция *MagicSort* которая за линейное время сортирует массив элементов данного типа. Обращение к данной функции возможно только как к чёрному ящику. Предложите, как на основе функции *MagicSort* построить устойчивый алгоритм сортировки за линейное время.

4. Оцените время работы алгоритма быстрой сортировки Хоара, в случае если в качестве *pivot* элемента выбирается:

   (a) Элемент стоящий посередине, то есть $a[(lb+rb)/2]$. Оцените время работы в худшем случае, в среднем (ожидаемое) и на случайных данных.

   (b) Медиана случайных $\sqrt{rb - lb + 1}$ элементов. Оцените время работы в худшем, в среднем (ожидаемое) и на случайных данных.

5. Даны n строк суммарной длины L. Чему будет равно математическое ожидание времени работы алгоритма быстрой сортировки, при условии что лексикографическое сравнение двух строк выполняется наивно?

6. Инверсией называется пара (i, j), такая что $i < j$ и $a_i > a_j$. Обсудите как модифицировать алгоритм сортировки слиянием, чтобы параллельно вычислять число инверсий.

7. Супер-инверсией называется тройка (i, j, k), такая что $i < j < k$ и $a_i > a_j > a_k$. Предложите алгоритм подсчёта количества супер-инверсий за время $O(n \log n)$, основанный на сортировке слиянием. Как обобщить алгоритм на поиск k-инверсий, то есть наборов $i_1 < i_2 < \ldots < l_k$, что $a_{i_1} > a_{i_2} > \ldots > a_{i_k}$.

8. Пусть некоторый алгоритм выбирает случайную пару элементов (i, j) и меняет их местами, если они нарушают условие упорядоченности. Покажите, что математическое ожидание количества действий данного алгоритма:

   (a) $O(n^4)$;

   (b) $O(n^2 \log n)$;

9. Докажите, что поиск элемента в отсортированном массиве, использующий лишь операции сравнения, в худшем случае работает не быстрее, чем за $O(\log\ n)$.

Сортировки и порядковые статистики, снова решаем вместе.

1. Рассмотрим следующий алгоритм:

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        if (x[i] > x[j])
            swap(x[i], x[j]);
```

   Верно ли, что в результате его работы значения x всегда будут упорядочены по неубыванию **или** по невозрастанию? Ответ обоснуйте.

2. Рассмотрим набор слов $s_1, s_2, \ldots, s_n$ над бинарным алфавитом. Суммарная длина слов — L. Считая, что операция поменять два слова местами выполняется за $O(1)$ (не зависит от длины), предложите алгоритм их лексикографической сортировки за время $O(L)$, использующий $O(\max |s_i|)$ (максимальная длина слова) дополнительной памяти.

3. Даны n точек, равномерно распределенных в единичном круге (с центром в начале координат). Предложите алгоритм, который за $O(n)$ в среднем сортирует их по удаленности от начала координат.

4. Докажите, что сортировка элементов, использующая только сравнения с тремя исходами (меньше, равно или больше) и некоторую внутреннюю рандомизацию, имеет ожидаемое время работы $\Omega(n \log n)$. Считайте, что последовательность бит, используемая для получения случайных чисел и определяющая поведение программы определяется заранее. Таким образом, в каждый момент времени текущее состояние программы зависит от этой заранее выбранной последовательности и результатов предыдущих сравнений. Что ломается в доказательстве, если считать, что последовательность бит может быть проинициализирована от значений элементов?

5. Перенесём во внешнюю память алгоритм сортировки подсчётом. Предложите различные варианты для следующих ситуаций. Поскольку речь идёт именно про сортировку подсчётом, то считайте, что мы хотим прочитать массив не более двух раз.

   (a) Значение U (диапазон значений) меньше M (размер оперативной памяти) и элементы не содержат дополнительной информации, то есть требуется просто отсортировать массив чисел.

   (b) Элементы содержат дополнительную информацию, которая делает их все различными, но ключи по-прежнему являются числами от 0 до $U - 1$. При каких значениях U вы можете получить сложность $O(\frac{n}{B})$?

   (c) Значение U меньше M, но элементы содержат дополнительную информацию. Какое время работы можно получить?

6. Перенесите в модель с внешней памятью поразрядную сортировку, получив сложность $O(\frac{n}{B} \cdot \frac{\log U}{\log \frac{M}{B}})$.

Алгоритмы и структуры данных.
Непростые задачи на простые структуры, решаем вместе.

1. Предложите способ реализовать во внешней памяти структуры стек, очередь и дек, чтобы время выполнения всех операций составляло $O(\frac{1}{B})$.

2. Докажите, что если у вас имеется корректная куча на минимум, затем какой-то элемент уменьшается, то после применения к этому элементу операции просеивания вверх у вас будет корректная куча.

3. Докажите, что если у вас имеется корректная куча на минимум, и условие кучи выполнено для всех вершин, кроме корня, то применение к корню операции просеивания вниз создаст корректную кучу.

4. Пусть в алгоритме дека с минимумом через три стека используется потенциальная функция $F(H) = \max(0, f(H) - s(H))$, где $f(H)$ — размер первого стека, а $s(H)$ — размер второго стека. Получается ли с такой потенциальной функцией показать амортизированную оценку выполнения всех операций $O(1)$?

5. **Напишите псевдокод** процедуры выполняющей объединение двух двусвязных закольцованных списков в один. Алгоритм получает два указателя на какой-то элемент каждого из списков (каждый из указателей также может иметь значение NIL). Гарантируется, что входные данные корректны.

6. Имеются две двоичные кучи размера n, записанные в массивах $a[i]$ и $b[j]$ соответственно. Оба массива индексируются с единицы. Процедура *joinHeaps(n, a, b)* выполняет объединение этих двух куч следующим образом:

```
void joinHeaps(int n, int* a, int* b) {
    for (int i = 1; i <= n; i++) {
        a[n + i] = b[i];
        siftUp(a, n + i);
    }
}
```

   В коде выше процедура *siftUpu(a, x)* выполняет стандартное просеивание вверх элемента x кучи, записанной в массиве a. Укажите асимптотическую зависимость от n времени выполнения процедуры *joinHeaps* и продемонстрируйте её неулучшаемость.

7. Пусть в массиве a_i записаны n элементов. Для построения k-ичной кучи выполняется проход от последнего к первому элементу массива и к каждому применяется операция $siftDown(i)$. Докажите, что во-первых после окончания данного процесса в массиве a_i будет записана корректная пирамида, а во-вторых время работы будет $O(n)$ для любого $k \leq n$.

8. Предложите струтуру данных «мультимножество с операцией доступа к k-й порядковой статистике». Параметр k **не зависит от запроса** и определяется в начале работы программы. Струтура данных должна поддерживать следующие операции: добавить элемент к множеству, вернуть k-ю порядковую статистику, удалить k-й элемент (при равенстве любой). При последовальных n операциях **гарантированная real-time** сложность вставки и удаления должна быть $O(\log n)$, сложность поиска k-й порядковой статистики должна быть $O(1)$. Дополнительно известно, что в любой момент времени структура будет содержать не более m элементов. Вам доступен один массив размера m для их хранения, разрешается использовать не более $O(1)$ дополнительной памяти.

9. В массиве, к которому имеется read-only доступ записаны $n + 1$ чисел от 1 до n. Требуется найти любое число, которое встречается хотя бы два раза за время $O(n)$, используя $O(1)$ дополнительной памяти.