Вводная лекция

Плотников Даннил Михайлович, Закарлюка Иван Владимирович

Санкт-Петербургский государственный университет

Оглавление

Организационные вопросы

- Лекция во вторник пятой парой
- Контест в субботу 17:30 — 20:00
- Посещение свободное, в контесте можно участвовать из дома, но круче приходить на факультет
- Язык программирования — в целом любой, однако традиционно C++(и на то есть причины!)
- Все материалы лекций будут доступны в репозитории https://github.com/kamenkremen/spbu-cp-materials
- Читаем мы по методичке, ссылка в репозитории и телеге

Полезные материалы

Теория

e-maxx (http://e-maxx.ru/algo/)

- Большое количество алгоритмов
- Подробные объяснения с примерами, кодом, задачами(на других платформах)
- Проверен временем
- Иногда лежит 😳

Алгоритмика (https://algorithmica.org/ru/)

- Много алгоритмов и полезных статей, но все равно много чего нет
- Подробные объяснения с примерами, кодом
- Относительно свежий, поэтому написан более понятным языком и с нормально выглядящим сайтом

ИТМО вики (neerc.ifmo.ru/wiki/)

- Большое количество не только алгоритмов, но в целом конспектов по математике, компьютер саенсу
- Некоторые статьи написаны не очень понятно
- Некоторые статье написаны не очень правильно
- Некоторые статьи недописаны

Задачи

CSES (https://cses.fi/problemset/)

- Много базовых, хороших задач
- Задачи собраны по темам

Timus (https://acm.timus.ru/)

- Огромное количество хороших задач
- Сайт прямиком из 2000 года 🙄

Универсальные солдаты

Leetcode (https://leetcode.com/)

- Большой архив задач
- Регулярные контесты
- Больше для подготовки к собесам

acmp (https://acmp.ru/)

- 1000 изначальных задач и ещё куча с разных соревнований
- Есть несколько курсов с теорией и задачами
- Устарел не только дизайн, но и теория местами

Codeforces (https://codeforces.com/)

- Огромный, все время пополняющийся архив задач
- Регулярные рейтинговые контесты различной сложности
- Вполне живое сообщество
- Хороший курс по некоторым темам, сейчас вроде делается второй
- Группа кружка в которой будут проходить субботние контесты именно здесь(https://codeforces.com/group/RZ7bF4GcQY/)

Достойные упоминания

- AtCoder (https://atcoder.jp/);
- TopCoder (https://www.topcoder.com/);
- Usaco (https://usaco.org/);
- SortMe (https://sort-me.org/).

Теория

Анализ сложности

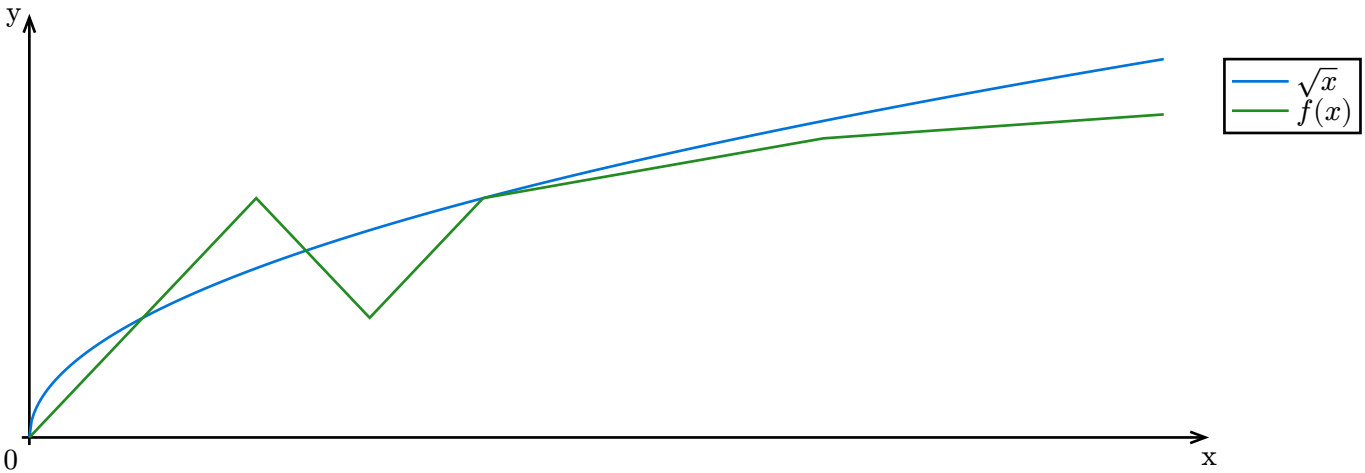
- Для анализа алгоритмов нужно научиться их сравнивать
- Самые очевидные критерии — “скорость” выполнения и используемая память. Сейчас поговорим про скорость
- Конечно, можно просто запустить алгоритм. Но тогда
  - На разных компьютерах время работы будет отличаться
  - Не всегда заранее доступны именно те данные, на которых он в реальности будет запускаться.
  - Иногда приходится оценивать алгоритмы, которые будут работать очень долго
  - Хотелось бы уметь оценить алгоритм, до того как садиться его реализовывать. Иначе как вообще придумывать новые нетривиальные алгоритмы?

- Попробуем как-нибудь приблизиться к нашим целям. Например, давайте считать количество операций которое совершает алгоритм
- Причем нужно считать количество операций в зависимости от входных данных, ведь иначе алгоритм который принимает n целых чисел и обрабатывает их суммарно за 10 операций мы можем посчитать хуже алгоритма, который обрабатывает их за n операций(ведь при $n < 10$ это правда будет так!)
- Для этого мы возьмем уже существующую вещь из математики. Она называется O-нотация

O-нотация

$$g(x) = O(f(x)) \Rightarrow \exists C > 0 = \text{const} : \exists x_1 : \forall x \geq x_1 \Rightarrow g(x) \leq Cf(x)$$

$$f(x) = O(\sqrt{x})$$



Следствия из определения:

- $O(Cf(x)) = O(f(x))$, например: $O(15x^2) = O(x^2)$
- $O(f(x) + \mathrm{C}) = O(f(x))$, например: $O(x^2 + 15) = O(x^2)$
  - При решении задач нужно не забывать о существовании этих констант. Если она будет сильно большой, то алгоритм может не проходить по времени, хотя асимптотически должно выглядит верным
- $g(x) = O(f(x)) \Rightarrow O(g(x) + f(x)) = O(f(x))$, например: $x = O(x^2) \Rightarrow$ $\Rightarrow O(x^2 + x) = O(x^2)$

Небольшое упражнение

$$f_1(n) = 2n \qquad f_1(n) = O(?)$$
$$f_2(n) = n^2 + 3n \qquad f_2(n) = O(?)$$
$$f_3(n) = \frac{n}{10^{100}} \qquad f_3(n) = O(?)$$
$$f_4(n) = \Sigma_{i=1}^{n} i \qquad f_4(n) = O(?)$$
$$f_5(n) = \frac{n}{3} \qquad f_5(n) = O(?)$$
$$f_6(n) = \log_2 n + 30 \qquad f_6(n) = O(?)$$
$$f_7(n) = n^3 + 2^n - 100 \qquad f_7(n) = O(?)$$
$$f_8(n) = 10^{10^{10^{10}}} \qquad f_8(n) = O(?)$$

Ответы

$$f_1(n) = 2n \qquad f_1(n) = O(n)$$
$$f_2(n) = n^2 + 3n \qquad f_2(n) = O(n^2)$$
$$f_3(n) = \frac{n}{10^{100}} \qquad f_3(n) = O(n)$$
$$f_4(n) = \Sigma_{i=1}^{n} i \qquad f_4(n) = O(n)$$
$$f_5(n) = \frac{n}{3} \qquad f_5(n) = O(n)$$
$$f_6(n) = \log_2 n + 30 \qquad f_6(n) = O(\log_2 n)$$
$$f_7(n) = n^3 + 2^n - 100 \qquad f_7(n) = O(n^3)$$
$$f_8(n) = 10^{10^{10^{10}}} \qquad f_8(n) = O(1)$$

Связь с алгоритмами

Самые часто встречающиеся асимптотики:

Асимптотика	Возможный вход	Пример алгоритма
$O(1)$	Любой	Формула
$O(\log n)$	Огромный	Двоичный поиск
$O(n)$	$\leq 10^8$	Поиск максимума
$O(n \log n)$	$\leq 10^6$	Сортировка
$O(n^2)$	$\leq 10^4$	Перебор пар
$O(n^3)$	$\leq 10^3$	Алгоритм Флойда-Уоршелла
$O(2^n)$	≤ 30	Перебор подмножеств
$O(n!)$	≤ 10	Перебор перестановок
$O(n^n)$	Никакой	Прям полный перебор

Упражнение побольше

Какова асимтотика этого алгоритма?

```
for j in 1..n-1
  for i in 0..n-1-j
    if a[i] > a[i+1]:
      swap(a[i], a[i + 1])
```

Какова асимтотика этого алгоритма?

```
for j in 1..n-1
  for i in 0..n-1-j
    if a[i] > a[i+1]:
      swap(a[i], a[i + 1])
```

$O(n^2)$

Ваще огромное упражнение

Какова асимптотика этого алгоритма?

```
for i = 0 to k
    C[i] = 0;
for i = 0 to n - 1
    C[A[i]] = C[A[i]] + 1;
b = 0;
for j = 0 to k + 1
    for i = 0 to C[j]
        A[b] = j;
        b = b + 1;
```

Какова асимптотика этого алгоритма?

```
for i = 0 to k
    C[i] = 0;
for i = 0 to n - 1
    C[A[i]] = C[A[i]] + 1;
b = 0;
for j = 0 to k + 1
    for i = 0 to C[j]
        A[b] = j;
        b = b + 1;
```

$$O(k + n + kn) = O(kn)$$

Мастер-теорема

Когда дело доходит до рекурсии, может быть проблематично посчитать время работы алгоритма. Для этого есть мастер-теорема для задачи размера n, которая разделяется на a задач в b раз меньшего размера с их объединением за $\Theta(n^c)$

Пусть $T(n) = \left\{ \begin{array}{c} aT(\frac{n}{b}) + \Theta(n^c) \text{ при } n > n_0 \\ \Theta(1) \text{ при } n \leq n_0 \end{array} \right\}$

Тогда

- Если $c > \log_b a$, то $T(n) = \Theta(n^c)$
- Если $c = \log_b a$, то $T(n) = \Theta(n^c \log n)$
- Если $c < \log_b$, то $T(n) = \Theta\left(n^{\log_b a}\right)$

Побитовые операции

Любое число(и не только) на самом деле хранится в виде нулей и единиц, то есть в двоичной системе счисления. Поскольку можно трактовать 0 как false а 1 как true, то к ним применимы логические операции(которые применяются к каждому биту, поэтому называются побитовые). Самые часто встречающиеся:

- И(&)
- Или(|)
- Не(~)
- Исключающее или(^), чаще xor

Так же такое представление позволяет, например, пользоваться сдвигами

- << — побитовый сдвиг влево
- >> — побитовый сдвиг вправо

Применения побитовых операий

- << — По сути, эквивалентен умножению на два
- >> — По сути, эквивалентен делению на два без остатка
- x & 1 - эквивалентно x % 2
- x & -x — взятие последнего ненулевой бита числа
- v && !(v & (v - 1)) — проверка является ли число степенью 2
- __builtin_clz(x) — посчитать количество ведущих нулей
- __builtin_ctz(x) — посчитать количество конечных нулей
- __builtin_popcount(x) — посчитать количество единиц в двоичной записи числа

Алгоритмы

Сканирующая прямая

Дан набор из n отрезков на числовой прямой. Нужно найти какую-нибудь точку, которая покрыта наибольшим количеством отрезков

Как вариант, можно перебирать все точки и считать перебором для каждой, сколько отрезков её покрывает

- Однако какова будет асимптотика такого решения?
- Можно попробовать улучшить этот вариант, например, не считать сколько отрезков покрывают точку в лоб
  - Давайте запомним в каких точках сколько отрезков начинаются и кончаются
  - Можем во время прохода по прямой поддерживать сколько сейчас отрезков покрывают точку
- А если рассматривать только точки в которых отрезок кончается/начинается?

Задача

Дан набор из n отрезков на числовой прямой. Дано q точек. Нужно для каждой точки вывести количество отрезков, которому она принадлежит

Метод двух указателей

Метод, а не алгоритм, поскольку намного менее конкретный

Дан массив чисел a, число k. Нужно найти максимальный по длине отрезок, такой, что сумма элементов на нем равна k

- Начнем искать с $l = 0, r = 0$
- Увеличиваем r пока сумма $\leq n$
- Если сумма равна n, то возможно ответ найден
- Пока сумма $> n$, увеличиваем l

К схожей задаче можно применить метод двух указателей

- если отрезок $[l, r]$ хороший, то любой вложенный в него отрезок также хороший
- или если отрезок $[l, r]$ хороший, то любой отрезок, который его содержит также хороший
- если зная ответ для $[l, r]$ можно быстро считать ответ для $[l - 1, r]$ и $[l, r + 1]$

Задача

Автобус представляет собой ряд из n мест, пронумерованных от 1 до n. Пассажиры садятся в автобус по следующим правилам:

- Если в автобусе нет занятых мест, пассажир может сесть на любое свободное место;
- Иначе пассажиру следует сесть на любое свободное место, рядом с которым есть занятое место. Другими словами, пассажир должен садиться на место с индексом $i (1 \leq i \leq n)$, только если существует хотя бы одно из мест с индексами $i - 1$ или $i + 1$, и при этом хотя бы одно из этих мест занято.

У вас есть список длины n того, как рассаживались пассажиры. Определите, правильно ли они расселись.

Префиксные суммы

Дан массив целых чисел a, и приходят запросы вида «найти сумму на отрезке с позиции l до позиции r».

- Можно отвечать в лоб, но это долго

Заведем массив, который назовем массивом префиксных сумм(далее p) и определим его так:

- $p_0 = 0$
- $p_1 = a_0$
- $p_2 = a_0 + a_1$
- ...
- $p_k = \Sigma_{i=0}^{k-1} a_i$

Посчитать его можно за $O(n)$

Теперь рассмотрим отрезок $[l, r]$:

- Сумму на отрезке $[0, r]$ мы знаем(она равна p_{r+1})
- Сумму на отрезке $[0, l-1]$ мы знаем(она равна p_l)
- Сумма на отрезке $[l, r]$ это $\Sigma[0, r] - \Sigma[0, l-1]$

Поскольку все эти значения у нас уже посчитаны, мы можем ответить на любой запрос за O(1).

Задача

Дан массив целых чисел a, и приходят запросы вида «найти xor чисел на отрезке с позиции l до позиции r».