Алгоритмы

Харитонцев-Беглов Сергей

3 сентября 2021 г.

Содержание

1. Асимптотики и правила игры.

1.1. Условия

1 теория в неделю, 1 практика в неделю.

- Теор. часть.
  Дедлайн: вторник, 23:59. Потом придут исправления, которые надо сдать до пятницы ($\leqslant 23:59$). Сдача после дедлайна — понижение коэффициента. Два типа задач:
  1. Обязательные, $\Sigma = 10 - 15$
  2. Дополнительные. «Overprice»

  Домашки сдавать обязательно в $\TeX$, если вы не в группе А. Олемской. Дедлайны можно переносить, если вам тяжело/заболели, то можно попросить перенести дедлайн лично для вас. Но если делать так слишком часто, то это неоч :(.
- Контест. 3 три задач:
  1. Must Have. Если не сдал — пиши-пропало.
  2. Обязательные. Сумма маст хэвов и обязательных – 10-15 баллов.
  3. Дополнительные. «Overprice»

Не стоит сначала обращать внимание на мелочи. Пытайтесь вычленить основную идеи. Уже когда поймете её, стоит пытаться найти интересные случаи.

Как понять что вы поняли алгоритм? Сесть и подумать: можете ли вы прямо сейчас сесть и написать код. Если не можете, то надо задавать вопрос. **Думайте, перед тем, как писать**.

Обучение – интерактивный процесс, старайтесь включаться, если вы переходите в режим зрителя, то становится плохо.

1.2. Асимптотика

Как выбрать процессор? У процессора есть несколько остальных характеристик (примеры в скобках): количество ядер (8 ядер), частота (3.3 GHz), набор инструкций, битность (32/64).

У нас все алгоритмы однопоточные, поэтому для нас важна только частота.

+, -, * /	1 операция
a[i]	1 операция
if	1 операция
f(..)	1 операция
if	1 операция

TODO: Схема

Время → константа + асимптотика. Асимптотика, если просто, число операций, к которому стремится при увеличении количества входа.

Есть асимптотика используемого времени и памяти. Утверждается, что Время $\geqslant$ Память, потому что на выделение памяти тоже время (причем 1 ячейка = 1 операция). Второй момент, время довольно безгранично, а память конечна.

***Определение* 1.1.** $f = \mathcal{O}(g(n))$: $\exists C > 0 : \exists N : \forall n \geqslant N : f(n) \leqslant C \cdot g(n)$

***Определение* 1.2.** $f = \Theta(g(n))$: $\exists C_1 > 0, C_2 > 0 : \exists N : \forall n \geqslant N : C_1 \cdot g(n) \leqslant f(n) \leqslant C_2 \cdot g(n)$

***Определение* 1.3.** $f = o(g(n))$: $\forall C > 0 : \exists N : \forall n \geqslant N : f(n) \leqslant C \cdot g(n)$

Свойство. $f \pm o(f) = \Theta(f)$

Доказательство. $\exists N : \forall n \geqslant N : o(f) \leqslant \frac{1}{2}f \Rightarrow \frac{1}{2}f \leqslant f + o(f) \leqslant \frac{3}{2}f$ □

***Свойство транзитивности* Θ.** $\Theta(\Theta(f)) = \Theta(f)$

Доказательство. Внешняя и внутренняя Θ зажата константами, а значит можно сказать, что константы внутренней равны внешней. □

Лемма. $\forall P$: $P(n) = \Theta(n^{deg\ P})$

Доказательство. **TODO:** Я не успел, смотри конспекты Сережи. □

1.3. Умножение. Карацуба

$735 = 7 \cdot 100 + 3 \cdot 10 + 5 = 7x^2 + 3x + 5$, если $x = 10$. Пусть $n = x^2 + x +)$, а $m = 3x + 7$. Тогда $nm = (x^2 + x + 1)(3x + 7)$. Дальше для умножения многочленов можно просто раскрыть скобочки. Если записать это в коде, то получим:

```
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            c[i + j] += a[i] * b[j]
//          x^{i+j}     x^i  * x^j
```

Данный код работает за $\mathcal{O}(n^2)$. Долго человечество не могло решить задачу быстрее. Но Анатолий Карацуба придумал быстрее:

1. $n = m = 2^k$
2. Разобьем $A(x)$ и $B(x)$ на две половины: A_1 и A_2, B_1 и B_2.
   $A(x) \cdot B(x) = (A_1(x) + x^{\frac{n}{2}} \cdot A_2(x))(B_1(x) + x^{\frac{n}{2}} \cdot B_2(x)) = A_1B_1 + x^n \cdot A_2B_2 + x^{\frac{n}{2}} \cdot (A_1B_2 + A_2B_1)$.

TODO: схемы

Запишем время работы нашего алгоритма: $T(n) = 4T(\frac{n}{2}) + n$, что равно (магия пока что) $\Theta(n^2)$. Но давайте напишем алгоритм, работающий за $T(n) = 3T(\frac{n}{2}) + n \approx n^{1.6}$.

Для этого заметим, что $A_1B_2A_2B_1 = (A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2$. Здесь сложение многочленов это операция соответственного суммирования коэффициентов перед степенями.

```
Mul(A, B)
    A -> A1 A2
    B -> B1 B2
    c = Mul(A2, B2)
    d = Mul(A1, A2)
    e = Mul(A1 + A2, B1 + B2)
    ...
```

1.4. Мастер-Теорема

Теорема 1.1 (Мастер-Теорема)**.** Пусть $T(n) = aT(\frac{n}{b}) + n^c$, где $a > 0, b > 1, c \geqslant 0$.
Тогда:

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & a > b^c. \\ \Theta(n^c), & a = b^c.\,. \\ \Theta(n^c \log n), & a < b^c \end{cases}$$

Пример. $T(n) = 2T(\frac{n}{2}) + n^2 = n^2 + 2(\frac{n}{2})^2 + 4(\frac{n}{4})^2 + \ldots = n^2(1 + \frac{1}{2} + \frac{1}{4} + \ldots) = n^2 \cdot 2 = \Theta(n^2)$

Доказательство. $T(n) = aT(\frac{n}{b}) + n^c = n^c + a(\frac{n}{b})^c + a^2(\frac{n}{b^2})^c + \ldots = n^{\cdot}(1 + \frac{a}{b^c} + (\frac{a}{b^c})^2 + \ldots)$. Тогда рассмотрим случаи:

- $a < b^c \Rightarrow \Theta(n^c)$
- $a = b^c \Rightarrow \Theta(n^c \log n)$
- $a > b^c$. Тогда под скобкой получается сумма вида $1 + x + x^2 + x^3 + \ldots$. Заметим, что сумма равна $\frac{(x^{k+1}-1)}{x-1}$, где -1 – константа, $x - 1$ – тоже. Тогда сумма равна $\Theta(x^{k+1}) = \Theta(x^k)$. Тогда получаем $\Theta(n^c \cdot (\frac{n}{b^c})^{\log_b n}) = \Theta(n^{\log_b a})$

□

Утверждение 1.2. $\forall a, b, c > 0$: $\log^a n < n^b < c^n$

Доказательство. Смотри в конспекте у Сережи. □