

SPb HSE, 1 курс, осень 2021/22
Практика по алгоритмам #10

Динамика

26 ноября

Собрано 22 декабря 2021 г. в 13:34

Содержание

1. Динамика	1
2. Разбор задач практики	3
3. Домашнее задание	7
3.1. Обязательная часть	7
3.2. Дополнительная часть	8

Динамика

1. Игра с камнями

Есть кучка из n камней. Двое играют в игру. Кто не может ходить, проиграл. Кто выиграет при оптимальной игре обоих?

- За ход можно брать любое число камней из $A = \{a_1, a_2, \dots, a_k\}$.
- Первый может брать a_1, a_2, \dots, a_k камней за ход, второй — b_1, \dots, b_m .

2. Профессор и железные яйца

У профессора есть k яиц и n этажное здание. Он хочет узнать максимальное x : если яйцо бросить с x -го этажа, оно не разобьётся. Не разбившиеся яйца можно переиспользовать. Минимизировать число бросков в худшем случае.

Время работы решения: $\mathcal{O}(n^2k) \rightarrow \mathcal{O}(nk \log n) \rightarrow \mathcal{O}(nk) \rightarrow \mathcal{O}(n \log n) \rightarrow o(n)$.

(*) Решите для заоблачных зданий $k, n \leq 10^9$.

3. Динамика по подотрезкам

- Произведение матриц.** Нужно посчитать произведение матриц $A_1 A_2 \dots A_n$ за минимальное число операций. Матрицы размеров $n \times k$ и $k \times t$ умножаются за nkt операций и дают матрицу размера $n \times t$.
- Свертка.** Дана строка из латинских букв длины n , нужно ее запаковать в максимально короткую, используя правило $n(S) = \underbrace{SS \dots S}_n$.

Пример: NEERCYESYESYESNEERCYESYESYES \xrightarrow{n} 2(NEERC3(YES)).

4. Динамика по дереву

Дано корневое дерево. У каждой вершины есть вес. Возможно, отрицательный. Отрезать некоторые ветки, чтобы максимизировать суммарный оставшийся вес.

5. Задачи про паросочетание

Найти максимальное по весу паросочетание (веса на рёбрах)

- на дереве за $\mathcal{O}(n)$.
- на цикле за $\mathcal{O}(n)$.
- на связном графе из n вершин и n ребер за $\mathcal{O}(n)$.

6. Рюкзак с большими весами

Рюкзак. $n \leq 1000$, $w_i \leq 10^9$, $cost_i \leq 100$.

Унести в рюкзаке размера $W \leq 10^9$ предметы максимальной стоимости.

7. Разбиения на слагаемые

- Сколько способов разбить число n на k упорядоченных слагаемых?
- Сколько способов разбить число n на k упорядоченных слагаемых, каждое не более m ?
- Сколько способов разбить число n на не более чем k неупорядоченных слагаемых?
- Сколько способов разбить число n на k неупорядоченных слагаемых?
- Сколько способов разбить число n на k неупорядоченных различных слагаемых?

8. Шаблоны

Подходит ли строка под шаблон из $*?x$ за $\mathcal{O}(nt)$.

Оптимизируйте память.

9. Пираты!

Судно атакуют пираты. Для каждого пирата известны его азимут a_i и время t_i , через которое пират приплывет и совершит непотребство.

Однако, у судна есть лазерная пушка, которой оно защищается. У пушки есть начальный азимут a и угловая скорость вращения ω . Пушка уничтожает все объекты, на которые она сейчас направлена.

Помогите судну определить порядок уничтожения пиратов за $\mathcal{O}(n^2)$, чтобы не допустить непотребства.

10. (*) Быстрая покраска забора

Дан массив длины n , который мы хотим получить. Числа в массиве от 1 до C . Получить его из массива $[0, 0, \dots, 0]$ минимальным числом операций «покраска отрезка». $\mathcal{O}(n^3C) \rightarrow \mathcal{O}(n^3)$.

11. (*) Взорвите дерево!

Найти в дереве минимальное по *суммарному весу* множество вершин, чтобы расстояние от любой вершины до одной из выбранных было не более d .

12. (*) Триангуляция многоугольника

Найдите кратчайшую триангуляцию выпуклого многоугольника.

13. (*) Переливания

Есть три стакана размеров $1 \leq A \leq B \leq C \leq N$.

Получить ровно x минимальным числом переливаний за $\mathcal{O}(N^2)$.

14. (*) Профессор и железные яйца

Решите для заоблачных зданий $k, n \leq 10^9$.

Разбор задач практики

1. Игра с камнями

- а) $f[i]$ – есть ли гарантированный способ выиграть у того, кто ходит, когда в кучке i камней.
 $f[i] = !f[i - a_1] \vee !f[i - a_2] \vee \dots \vee !f[i - a_k]$.
- б) $f[i, p]$ – есть ли гарантированный способ выиграть у игрока p , когда в кучке i камней.
 $f[i, 1] = !f[i - a_1, 2] \vee !f[i - a_2, 2] \vee \dots \vee !f[i - a_k, 2]$. Аналогично $f[i, 2]$.

2. Профессор и железные яйца

$f[n, k]$ – минимальное число бросков. После броска с этажа i в худшем случае нужно ещё $\max(f[i-1, k-1], f[n-i, k])$ бросков. Назовем $\text{pos}[n, k]$ оптимальное i для n, k .

Утверждение #1: $f[n, k] \leq f[n+1, k]$. То есть, $f[i-1, k]$ возрастает, $f[n-i, k-1]$ убывает \Rightarrow $\text{pos}[n, k] = \min i: f[i-1, k] \geq f[n-i, k-1]$, либо на 1 меньше.

Утверждение #2: $\text{pos}[n, k] \leq \text{pos}[n+1, k]$. Обоснование: если $j < \text{pos}[n, k]$, то $f[j-1, k] < f[n-j, k-1] \leq f[n+1-j, k-1] \Rightarrow j \leq \text{pos}[n+1, k]$.

\Rightarrow для каждого k можно перебирать n и $\text{pos}[n, k]$ двумя указателями. $\mathcal{O}(nk)$.

Наблюдение: ответ всегда можно найти за $\lceil \log n \rceil$ бросков \Rightarrow при $k \geq \lceil \log n \rceil$ можно отвечать сразу $\lceil \log n \rceil$ (меньше нельзя, n вариантов ответа).

3. Динамика по подотрезкам

а) Произведение матриц.

$f[l, r]$ – минимальное число действий, нужное для перемножения отрезка $A_l A_{l+1} \dots A_r$.

Пусть размер i -й матрицы – $a_i \times a_{i+1}$.

Тогда в результате умножения отрезка $[l..r]$ получится матрицы размера $a_l \times a_{r+1}$.

Перебираем, какое умножение m будет последним на отрезке $[l, r]$, получаем

$$f[l, r] = \min_m f[l, m-1] + f[m, r] + a_l \cdot a_m \cdot a_{r+1}$$

Когда мы ищем $f[l, r]$, нужно чтобы уже были найдена f на всех подотрезках.

Либо ленивая динамика, либо перебираем отрезки в порядке возрастания длины.

Для восстановления ответа храним $m[l, r]$ – оптимальное m для $[l, r]$.

Восстанавливаем рекурсивно: $\text{ans}(l, r) = (\text{ans}(l, m[l, r] - 1)) \cdot (\text{ans}(m[l, r], r))$

б) Свертка.

Главная идея – динамика по подотрезкам. Как закодировать отрезок $[i, i+len)$? Переберём все варианты, как закодировать первый символ. Отдельно? Тогда откусим его, закодируем остаток $[i+1, i+len)$. В виде «строка $[i, i+m)$, повторённая k раз»? Если так, откусим первые mk символов, закодируем их отдельно, остаток отдельно... если остаток не пуст, это корректный переход в динамике, а если остаток пуст, значит $mk = len$ и отрезок $[i, i+len)$ имеет период m . Описанное выше выражается через два перехода: $\forall m$ попробовать откусить первых m символов или разделить весь отрезок на кусочки длины m .

Собственно решение за $\mathcal{O}(n^3)$.

1. Предподсчитаем $\text{lcp}[i, j]$ за $\mathcal{O}(n^2)$.

2. Динамика $\text{ans}[i, len]$ – ответ на задачу для подстроки $[i..i+len)$.

$\text{ans}[i, len]$ мы минимизируем \Rightarrow используем функцию $\text{relax}(\&a, b) \{ a = \min(a, b); \}$.

Пары $\langle i, len \rangle$ перебираем в порядке возрастания len .

Для каждой пары изначально $ans[i, len] = len$ (не сжимаем подотрезок).

Далее в цикле по m делаем переходы двух типов:

- «откусить первые m символов» и
- «повторить подстроку длины $m \frac{len}{m}$ раз».

Проверка, что, повторив $[i, i+m)$, получим $[i, i+len)$, делается так: $lcp[i, i+m] \geq len-m$.

```

1 for m=1..len-1:
2   relax(ans[i, len], ans[i, m] + ans[i+m, len-m]) // откусили первые m символов
3   if (len % m == 0 && lcp[i, i+m] >= len-m) // m - делитель => делим на части
4     relax(ans[i, len], |len/m| + ans[i, m] + 2) // |len/m| - длина числа

```

4. Динамика по дереву

$f[v]$ – максимальная сумма для поддерева с корнем в v .

$f[v] = w[v] + \sum_x \max(0, f[x])$, где w – вес вершины, x – дети v .

5. Задачи про паросочетание

Найти максимальное по весу паросочетание (веса на рёбрах)

а) На дереве за $\mathcal{O}(n)$: считаем две величины

$f_0[v]$ – вес максимального паросочетания в поддереве v , не покрывающего v .

$f_1[v]$ – вес максимального паросочетания в поддереве v , покрывающего v .

Если нам неважно, покрывать вершину v или нет, то будем писать $f[v] = \max(f_0[v], f_1[v])$.

Пусть дети вершины v – x_1, x_2, \dots, x_k , веса рёбер (v, x_i) – w_i , тогда:

$$f_0[v] = \sum_i f[x_i]$$

$$f_1[v] = \sum_i f[x_i] + \text{ребро вниз} = \sum_i f[x_i] + \max_i (w_i - f[x_i] + f_0[x_i])$$

б) На цикле за $\mathcal{O}(n)$.

Цикл отличается от дерева только одним ребром. Давайте переберём, включаем ли мы в паросочетание первое ребро. В обоих случаях останется дерево, решим для него задачу.

Останется даже не дерево, а путь, массив, на массиве можно проще:

$$f_0[i] = f[i+1], f_1[i] = w_i + f[i+1], f_i = \max(f_0[i], f_1[i]).$$

в) На связном графе из n вершин и n ребер за $\mathcal{O}(n)$.

Что это за граф такой? Это цикл, на котором растут деревья. Выделим цикл, деревья, сперва для деревьев запустим (а), затем для цикла (б).

Более простое решение: найдем цикл, как в (б) переберем будем ли или нет включать его первое ребро в паросочетание. Остается дерево, для него умеем решать.

6. Рюкзак с большими весами

Раньше из $(i, weight, cost)$ мы $(i, weight)$ выносили в состояние, а $cost$ максимизировали.

Вместо этого выберем состояние $(i, cost)$, а $weight$ будем минимизировать:

$minWeight[i, sumCost]$, переходы, как и раньше, либо взяли i -ый предмет, либо нет.

7. Разбиения на слагаемые

а) k упорядоченных слагаемых. Если нет нулей, $\binom{n-1}{k-1}$, если можно нули, то $\binom{n+k-1}{k-1}$.

б) k упорядоченных слагаемых не более m .

Динамика: $p[n, k] = p[n-1, k-1] + p[n-2, k-1] + \dots + p[n-m, k-1]$. Можно считать за $\mathcal{O}(nkm)$, а можно делать переход за $\mathcal{O}(1)$, если посчитать частичные суммы.

с) **Не более чем k неупорядоченных слагаемых.**

Либо слагаемых меньше k , либо из каждого из них можно вычесть единичку.

$$p[n, k] = p[n, k - 1] + p[n - k, k].$$

d) **k неупорядоченных слагаемых.**

Способ #1. Посчитать динамику пункта (с), ответ $- p[n, k] - p[n, k - 1]$.

Способ #2. Либо среди слагаемых есть единица, либо можно из всех слагаемых вычесть единицу, и их останется ровно k . $P[n, k] = P[n - 1, k - 1] + P[n - k, k]$.

Способ #3. Если вычесть из каждого слагаемого 1 и убрать нули, то из разбиения n с k неупорядоченными слагаемыми мы получим разбиение $n - k$ с не более чем k неупорядоченными слагаемыми. Это биекция $\Rightarrow P[n, k] = p(n - k, k)$.

е) **k неупорядоченных различных слагаемых.**

Если среди слагаемых есть единица, то когда вычтем из всех единицу, их станет на одно меньше, иначе останется столько же. $p[n, k] = p[n - k, k - 1] + p[n - k, k]$.

8. **Шаблоны**

$f[i, j]$ – подходит ли строка $s[0, i)$ под шаблон $p[0, j)$.

Если $p[j-1]$ – буква, то $f[i, j] = f[i - 1, j - 1] \wedge p[j - 1] == s[i - 1]$.

Если $p[j-1] == '?'$, то $f[i, j] = f[i - 1, j - 1]$.

Если $p[j-1] == '*'$, то звездочка ест либо ничего, либо что-то: $f[i, j] = f[i, j - 1] \vee f[i - 1, j]$.

9. **Пираты!**

На любом отрезке последним умрет пират с одного из краев. $f_L[l, r]$ – минимальное время, чтобы убить всех на отрезке $[l, r]$ и последним убить пирата на позиции l . Аналогично $f_R[l, r]$.

Переходы: $f_L[l, r] + |a[r+1] - a[l]| \rightarrow f_R[l, r+1]$. Еще три аналогичных.

Важно следить, что мы успеваем убить последнего пирата, иначе ответ $+\infty$.

Бонус. На самом деле можно хранить только одну $f[l, r]$, где за последнего убитого пирата будет отвечать индекс l . Неоднозначности с тем, какую из половин круга выбрать, нет – ту, в которой находится начальное положение пушки.

10. (*) **Быстрая покраска забора**

Общая идея (*): отрезки, которые мы красим, или не пересекаются, или вкладываются (иначе можно первый укоротить). Если крайняя клетка уже покрашена в свой цвет, пропускаем её. Если нет, отрезок, который её покрасит по (*) можно сделать первой операцией.

а) $\mathcal{O}(n^3C)$. Нужно получить цвета $a[i]$. Динамика $f[l, r, c]$ –

минимальное число покрасок отрезка $[l, r]$, если он сейчас весь покрашен в цвет c .

$a[r] = c \Rightarrow$ не надо красить последнюю клетку, и $f[l, r, c] = f[l, r - 1, c]$.

$a[r] \neq c \Rightarrow$ первая операция – покраска какого-то суффикса в цвет $a[r] \Rightarrow$

$f[l, r, c] = \min_m (1 + f[l, m - 1, c] + f[m, r, a[r]])$. Суффикс может совпасть со всем отрезком (случай $m = l$), поэтому для отрезка $[l, r]$ надо всегда сначала считать $f[l, r, a[r]]$.

б) $\mathcal{O}(n^3)$. Вместо использования c мы будем считать, что отрезок весь покрашен в цвет $a[l]$.

$a[r] = a[l] \Rightarrow f[l, r] = f[l, r - 1]$.

$a[r] \neq a[l] \Rightarrow$ красим суффикс в цвет $a[r]$. Логично, чтобы этот суффикс начинался

$m: a[m] = a[r]$, иначе $a[m]$ все равно перекрасят $\Rightarrow f[l, r] = \min_{a[m]=a[r]} (1 + f[l, m - 1] + f[m, r])$.

11. (*) **Взорвите дерево!**

Сначала считаем динамику, учитывая расстояние только вниз. Затем считаем вторую динамику, уже учитывая пути вверх, используя предыдущую.

12. (*) **Триангуляция многоугольника**

Динамика по подотрезкам. Триангулируем многоугольник на вершинах $i, i + 1, \dots, j$ (в нём есть сторона (j, i)). Либо проводим диагональ (i, k) , либо $(j, i + 1)$.

Итого $f[i, j] = \min(w[j, i + 1] + f[i + 1, j], \min_k w[i, k] + f[i, k] + f[k, j])$.

13. (*) **Переливания**

Наивное решение за $\mathcal{O}(N^3)$: состояние – сколько налито в каждом стакане.

В реальности достижимы только состояния, где один из стаканов либо пуст, либо полон \Rightarrow ленивая версия той же динамики сработает за $\mathcal{O}(N^2)$.

14. (*) **Профессор и железные яйца**

$n[s, k]$ = max высота, для которой хватит s бросков k яиц.

$n[s, k] = n[s - 1, k - 1] + n[s - 1, k]$ (алгоритм: бросим с этажа номер $n[s - 1, k - 1]$).

Заметим, что если $k = 1$, ответ n , а если $k \geq 2$, ответ = $\mathcal{O}(\sqrt{n}) \Rightarrow$

в лоб динамика $n[s, k]$ считается за $\mathcal{O}(\sqrt{n} \cdot \log n)$.

Алгоритм (двоичные подъёмы, возведение матрицы в степень):

Заметим, что $n[s]$ выражается из $n[s - 1]$ домножением на матрицу $A: k \times k$.

- $s \leq n \Rightarrow t = \lceil \log_2 n \rceil$ считаем $A^1, A^2, A^4, A^8, \dots, A^{2^t}$. За $\mathcal{O}(k^3 \log n) = \mathcal{O}(\log^4 n)$.
- `ans=0, x=n[0]; for i=t..0 do { y=A2i*x; if y[k] < n then ans+=2i, x=y }`

Домашнее задание

3.1. Обязательная часть

1. (2) Сумма кубов

Разбить число n на сумму минимального числа кубов натуральных чисел, $n \leq 10^6$.

2. (2) Центроиды

Дано дерево из n вершин.

Найти все вершины v : при удалении v размеры получившихся деревьев будут не более $\frac{2n}{3}$.

3. (2) Зануление массива

Дан циклический массив, перед i -м идёт элемент $(i-1) \bmod n$, после i -го идёт $(i+1) \bmod n$.

Рассмотрим операцию `zero(i) { ans += a[next(i)]*a[i]*a[prev(i)], a[i] = 0 }`

Найти последовательность операций, максимизирующую `ans`.

a) (1) $\mathcal{O}(n^3)$

b) (1) $\mathcal{O}(n)$

4. (2) Игра на массиве

Алиса и Боб играют в игру на массиве длины n . За ход можно съесть одно из крайних чисел. Алиса ходит первой. Результат игры – сумма чисел, съеденных Алисой. Задача Алисы – максимизировать результат, задача Боба – минимизировать результат.

$n \leq 100$. Найдите результат при оптимальной игре обоих.

5. (2) Сколько существует счастливых билетов?

Счастливым билет – строка из $2n$ цифр такая, что сумма цифр в первой половине равна сумме цифр во второй. Нужно решение за $\mathcal{O}(n^2)$.

6. (2) Разбивай и властвуй

Сколько разбиений числа N на неупорядоченные слагаемые?

Слагаемых должно быть от A до B , величины слагаемых от L до R .

$\mathcal{O}(N^3)$. (+1) балл за $o(N^3)$.

7. (1) Правильные скобочные подстроки

Сколько существует строк из n круглых скобок, которые являются подстрокой правильной скобочной последовательности?

3.2. Дополнительная часть

1. (3) Большая сумма кубов

Разбить число до 10^9 на сумму минимального числа кубов.

2. (3) Паросочетание в кактусе

Найти максимальное по весу паросочетание в кактусе за $\mathcal{O}(n)$.

Кактус – граф, в котором каждое ребро принадлежит не более чем одному циклу.

3. (4) Редукция дерева

Посчитать количество поддеревьев данного дерева из n вершин, содержащих корень дерева и ровно $k \leq n$ вершин. Поддеревом здесь называется связный подграф.

a) (2) $\mathcal{O}(n^3)$.

b) (3) $\mathcal{O}(n^2)$.

c) (4) $\mathcal{O}(nk)$.

4. (3) Упорядоченный Хаффман

Дана строка s над алфавитом Σ .

Каждой букве x из строки нужно сопоставить двоичную строку c_x так, чтобы:

a) $x < y \Rightarrow c_x < c_y$,

b) никакой c_x не является префиксом никакого c_y ,

c) $|c_{s_1}| + |c_{s_2}| + \dots + |c_{s_n}| \rightarrow \min$.

$\mathcal{O}(|\Sigma|^2 + |s|)$

5. (3) Идеальный шаблон

Найти минимальный шаблон такой, что под него не подходит первая строка, но при этом подходит вторая. $\mathcal{O}(n^3)$.