SPb HSE, 1 курс, весна 2024/25
Практика по алгоритмам #22

Жадные алгоритмы
12 марта

Собрано 12 марта 2025 г. в 09:55

Содержание

Жадные алгоритмы

1. **2-приближение Vertex Cover**

   Найдите вершинное покрытие не более, чем вдвое превышающее оптимальное. $\mathcal{O}(V+E)$.

2. **Неприближаемость коммивояжера**

   Покажите, что нельзя за полином найти приближение задачи о коммивояжере на полном графе без неравенства $\triangle$.

3. **Непрерывный рюкзак (повторение лекции)**

   Вор с рюкзаком вместимости W снова оказался в ломбарде. Перед ним n слитков, i-й из них имеет стоимость c_i и объём (вес) w_i.

   Однако в этот раз он взял с собой ножовку по металлу, поэтому от каждого слитка он может отрезать и взять с собой произвольную долю $0 \leqslant \alpha_i \leqslant 1$. Такой кусок имеет объём $\alpha_i w_i$ и стоит $\alpha_i c_i$.

   Какое максимальное количество денег сможет выручить вор?

4. **Хаффман**

   a) Постройте код Хаффмана за $\mathcal{O}(n)$, если частоты букв даны в отсортированном порядке.
   b) Докажите, что если частоты всех символов $< 1/3$, то нет символа с длиной кода 1.
   c) Какова максимальная длина кода над алфавитом размера m? Приведите пример частот.

5. **Сбалансированное разбиение**

   Удалить из дерева размера n вершину, чтобы оставшиеся компоненты были размера $\leqslant \frac{1}{2}n$.

6. **Белоснежка и n гномов**

   Даны n гномов. Если i-го гнома укладывать спать a_i минут, он потом спит b_i минут.
   Можно ли сделать так, чтобы в какой-то момент все гномы спали?

7. **Башня из стойких спортсменов**

   Есть n спортсменов. i-й спортсмен имеет массу m_i и может держать на своих плечах суммарную массу s_i.

   a) Могут ли все спортсмены встать в башню? $\mathcal{O}(n \log n)$.
   b) Какое максимальное число спортсменов могут выстроиться в башню? $\mathcal{O}(n^2)$.

8. **Вылезти из ямы**

   Однажды на РОИ n школьников упали в яму глубины S.
   Каждый школьник имеет рост (от ног до плеч) h_i и длину рук l_i.
   Школьники могут вставать друг другу на плечи, верхний школьник может вытянуть руки.

   a) Могут ли выбраться все школьники? $\mathcal{O}(n \log n)$.
   b) Какое максимальное число школьников может выбраться? $\mathcal{O}(n^2)$.

9. **Задача о жадном выборе заявок**

   Учёные хотят прочесть доклады, i-й запланировал свой доклад на время $[a_i, b_i]$.

   a) Какое максимальное число докладов можно прочитать, если есть только одна аудитория? Угадайте жадность (практики, потренируйтесь заранее валить кривые жадности). Важно доказательство корректности.

b) Найти минимальное число аудиторий, чтобы прочитать все доклады (два решения).
c) Найти максимальное число докладов, если есть k аудиторий (два решения).

У последней заявки есть переформулировка через маршрутку: пассажир = отрезок, k = вместимость, кого взять, чтобы больше заработать?

10. **Два станка и отношения**

Неправильное решение задачи «Два станка» использует следующий компаратор:
$(a_1, b_1) \preccurlyeq (a_2, b_2) \iff \min(a_1, b_2) \geqslant \min(a_2, b_1)$. Докажите, что такое отношение нетранзитивно.

11. **Диаметр дерева**

Найти диаметр дерева (две самые удалённые друг от другая вершины).

12. **Разбиение на подпоследовательности**

a) Разбить массив на минимальное число возрастающих подпоследовательностей.
b) (*) Выбрать из массива k возрастающих непересекающихся подпоследовательностей максимальной суммарной длины. Найти только длину за $\mathcal{O}(kn \log n)$.
c) (**) Найти и длину, и сами последовательности за $\mathcal{O}(n\sqrt{n})$.

13. **(*) Ценные отрезки**

Даны отрезки на прямой с весами.
Выберите максимальное по сумме весов множество непересекающихся отрезков. $\mathcal{O}(n \log n)$.

14. **(*) То ли шубы, то ли вёдра**

У Пети есть n ведёрок. У каждого есть размер a_i.
Петя может поставить i-е внутрь j-го iff $a_i + d \leqslant a_j$, где d – константа.
Внутрь можно ставить только одно, получается ведёрки можно разбить на стеки.
Петя минималист и хочет, чтобы ведёрки занимали как можно меньше места.
Помогите ему так расположить ведёрки внутри друг друга так, чтобы
минимизировать $\sum a_i$ внешних ведёрок. $\mathcal{O}(n \log n)$.

15. **(*) Жадность, а не поток**

Жила-была матрица, да потерялась. Остались от неё только суммы в строках и столбцах и информация о том, что все элементы были от 0 до 100.
Восстановите любую такую матрицу. $n \leqslant 1\,000$.

16. **(*) Расход топлива**

Машина тратит единицу топлива на километр, имеет бак объёма k и находится в начале прямой дороги в точке 0. Для всех $i \in \mathbb{N} \cup \{0\}$ на i-м километре дороги есть заправочная станция со своей положительной ценой c_i. Определите за время $\mathcal{O}(n)$, как проехать n километров за минимальную стоимость.

17. **(*) Приближенное независимое множество**

Выбрать в графе независимое множество размера $\lceil \frac{V}{D+1} \rceil$, где D – максимальная степень.

Разбор задач практики

1. **2-приближение Vertex Cover**

   Жадность. Смотрим на любое непокрытое ребро. В оптимальном покрытии есть хотя бы один из концов ребра. А мы не знаем, какой взять, возьмём оба.

2. **Неприближаемость коммивояжера**

   Сведём гамильтонов цикл в неорграфе к приближению коммивояжера. Берём граф. Создаём матрицу весов: нет ребра $\Rightarrow w_{ij} = 1$, есть ребро $\Rightarrow w_{ij} = 0$. $\exists$ гамильтонов цикл $\Rightarrow$ вес для комивояжера $= 0$, иначе 1, что в бесконечность раз больше.

   Если нас просят тест, где все веса положительны заменим $\langle 0, 1\rangle$ на $\langle 1, Cn+1\rangle$.

3. **Непрерывный рюкзак (повторение лекции)**

   Удельная стоимость слитка: $q_i = c_i/w_i$. Отсортируем в порядке убывания q_i, набираем жадно.

   Доказательство: индукция по n. Пронумеруем предметы так, чтобы $q_1 \geqslant q_2 \geqslant q_n$. Рассмотрим оптимальный ответ, если в нём целиком взят первый слиток, выкинем его из доступных слитков, и положим $W' = W - w_1$ — теперь у нас $n-1$ слиток.

   Если первый слиток взят не целиком, возьмём его целиком, а то, что не вмещается в рюкзак (если такое появилось) — обрежем. Цена рюкзака при этом не уменьшилась.

   База: $n = 1$. Тут слиток может быть не взят целиком, когда $w_1 > W$.

4. **Хаффман**

   a) **Построение за $\mathcal{O}(n)$.**
   Две очереди: для старых символов и для новых.
   Частоты новых символов всегда возрастают $\Rightarrow$ очередь новых символов будет всегда в возрастающем порядке.
   Так что можно находить минимальный символ за $\mathcal{O}(1)$, это минимум из первых элементов обеих очередей.

   b) **У редких символов длинные коды.**
   Пусть есть символ с кодом длины 1, кончается в вершине x. Его частота $< 1/3$, $\Rightarrow$ на последнем шаге он объединился с вершиной y частотой $> 2/3$.
   Исходные символы редкие $\Rightarrow$ y получилась слиянием каких-то a и b.
   Пусть a более частый. Частота $y > 2/3 \Rightarrow$ у a частота $> 1/3$.
   Тогда перед слиянием a и b в y минимальные частоты были у x и b, противоречие.

   c) **Max длина кода**.
   Если частоты $1, 1, 2, 4, \ldots, 2^{m-2}$, то есть код длины $m-1$.
   Больше в полном двоичном дереве с m листьями не бывает.

5. **Сбалансированное разбиение**

   Пойдем сверху вниз по дереву. Пока существует ребенок с поддеревом размера $> \frac{n}{2}$, идем в него. Остановились в вершине v, ее и надо удалить. Раз остановились, все дети $\leqslant \frac{n}{2}$. Раз не остановились раньше, ее поддерево $> \frac{n}{2} \Rightarrow$ выше нее осталось $\leqslant \frac{n}{2}$.

6. **Белоснежка и n гномов**

   У первого уложенного $b_i \geqslant \sum\limits_{j \neq i} a_j \Leftrightarrow a_i + b_i \geqslant \sum a_j$.

Отсюда мысль – сортируем по убыванию $a_i + b_i$.
Действительно, если у двух подряд уложенных гномов $a_i + b_i < a_{i+1} + b_{i+1}$, то их можно поменять местами:
$b_i \geqslant \sum\limits_{j>i} a_j \Rightarrow b_i \geqslant \sum\limits_{j>i+1} a_j$ (можно i передвинуть вперед, вообще если гнома можно положить перед каким-то множеством, то перед меньшим тем более можно),
$a_i + b_i \geqslant \sum\limits_{j\geqslant i} a_j \Rightarrow a_{i+1} + b_{i+1} \geqslant \sum\limits_{j\geqslant i} a_j$ (можно передвинуть $(i+1)$ назад, вообще если можно гнома положить перед каким-то множеством, то гнома с большей суммой тем более можно).

7. **Башня из стойких спортсменов**

   В каком порядке расставлять? Внизу может быть любой i:
   $s_i \geqslant \sum_{j\neq i} m_j \Leftrightarrow s_i + m_i \geqslant \sum_j m_j = \texttt{CONST} \Rightarrow$ давайте вниз ставить $i = argmin_j(s_j + m_j)$.
   $\Rightarrow$ каких бы спортсменов мы ни выбрали, их нужно ставить сверху вниз в порядке $(s_j+m_j) \nearrow$
   Отсортируем в таком порядке. Будем строить башню сверху вниз. Далее динамика:

   $f_{n,k}$ – минимальный вес башни, если из первых n спортсменов взяли в башню k.
   $f_{n,k} = \min(f_{n-1,k}, f_{n-1,k-1} + m_n)$ (второй переход возможен только при $s_n \geqslant f_{n-1,k-1}$).

8. **Вылезти из ямы**

   a) Школьник, вылезающий последним, должен уметь в одиночку выбираться из ямы, для остальных глубина ямы меньше за счёт стоящих ниже $\Rightarrow$ сортируем по возрастанию h_i+l_i.
   Доказательство: вниз можно поставить любого i: $h_i + l_i \geqslant H$, например, $argmax$.

   b) Перебираем школьников по возрастанию $h_i + l_i$.
   Те, кто не вылезет, будут стоять на дне и помогать следующим. Динамика $f_{n,k}$ – максимальная сумма h_i «оставшихся в яме навсегда школьников», если из первых n вылезут k.
   $f_{n,k} = \max(f_{n-1,k} + h_n, f_{n-1,k-1})$. Второй переход возможен только при $h_n + f_{n-1,k-1} \geqslant H$.

9. **Задача о жадном выборе заявок**

   a) Одна аудитория: max число непересекающихся отрезков. Просматриваем отрезки в порядке $R_i \nearrow$. Каждый жадно берём, если можем.

   b) min число аудиторий.
   Способ 1. Сортируем события «начало отрезка», «конец отрезка». Находим точку, покрытую max числом отрезков.
   Восстановление ответа: зная k, поддерживаем `vector` номеров свободных аудиторий, когда открывается отрезок, кладём его в любую свободную. $\mathcal{O}(sort + n)$.

   Способ 2. Идем по докладам по возрастанию правого конца.
   Храним `set` правых концов последних докладов каждой аудитории.
   Пришел новый доклад $[l, r]$, добавим его куда можем с максимальным концом $r_i < l$
   (то есть, к `--set.upper_bound(`l`)`). $\mathcal{O}(n \log n)$.
   Корректность: пусть мы добавили $[l, r]$ в S, пусть оптимально было в S взять какой-то из более поздних отрезков – $[x, y]$, а $[l, r]$ или никуда не взять, или взять в T.
   $r \leqslant b$, $T.right \leqslant S.right \leqslant x \Rightarrow$ сделаем замену.

   c) *Пробуем сортировать по* **правому** *концу.* Жадно берём отрезки в ответ. Будем поддерживать ответ в виде «k стеков, в каждом стеке отрезки попарно непересекаются».
   Для каждого стека важен только правый конец последнего отрезка $\Rightarrow$
   храним `set<int> s` правых концов и очередной отрезок $[L_i, R_i]$ пытаемся дописать к стеку `s.upper_bound(`L_i`)`.

Пробуем сортировать по **левому** *концу.* Жадно берём отрезки в ответ.
Проблема случится, если после добавления $\langle L_i, R_i \rangle$ точка L_i покрыта $k+1$ отрезком.
Нужно кого-то выкинуть. Думаем о будущем $\Rightarrow$ выкидываем $\max R_j \Rightarrow$
нужен `set<int> s` правых концов.
При проверке покрытости L_i удаляем из `s` все $R_j < L_i$, смотрим на `s.size()`.

10. **Два станка и отношения**

Например, $(4, 5), (1, 1), (2, 3)$.

```
x : 4 5
y : 1 1
z : 2 3
x = y, y = z
z < x
```

11. **Диаметр дерева**

$a = Farthest(0), b = Farthest(a)$

Вершины a, b образуют диаметр.

12. **Разбиение на подпоследовательности**

a) Храним `set` концов набранных последовательностей.

Пришел новый элемент x, добавим его к такой последовательности S, что он подойдет, и элемент перед ним максимален. Т.е. к `set.lower_bound`(x).
Если в оптимуме в S добавили y, а x добавили в какую-то T, то можно поменять местами хвосты S и T.

b) (*) Вспомним, как мы решаем задачу за $\mathcal{O}(n \log n)$ для $k = 1$. Динамика
$tend[\texttt{len}]$ – минимальный конец последовательности длины len.
Теперь храним k последовательностей $\texttt{end}_i[\texttt{len}]$ для $i = 1..k$.
Причём последовательности упорядочены по длине, end_1 – самая длинная.
Приходит новый элемент a_i.
Найдём бинпоиском его позицию p в $\texttt{end}_1$: $\texttt{end}_1[p-1] < a_i \leqslant \texttt{end}_1[p]$.
Заменяем $\texttt{end}_1[p]$ на a_i. Получается, мы как бы вытолкнули из $\texttt{end}_1$ элемент $\texttt{end}_1[p]$.
Добавим его таким же образом в $\texttt{end}_2$. И так далее...
В итоге мы получим не последовательности, а какой-то мусор. Зато суммарная длина этого мусора в точности равна ответу на задачу. Без доказательства.

c) (*) Найти и длину, и сами последовательности за $\mathcal{O}(n\sqrt{n})$.

Сложно... очень.

13. (*) **Ценные отрезки**

Это не жадность, а динамика. Да, задача с подколом.

Отсортируем по правому концу r_i. Посчитаем динамику.
$f[i]$ – максимальная сумма весов, если самый правый взятый – i-й отрезок.
$f[i] = w_i + \max_{j:\, r_j < l_i} f[j]$, то есть, максимум на префиксе. $\mathcal{O}(n \log n)$.

14. (*) **То ли шубы, то ли вёдра**

Перебираем по убыванию. Храним `set` вёдер, куда можно вставить новое. Вставляем в самое узкое ведро (`lower_bound`). Если никуда нельзя вставить, ну что ж, добавляемся в `set`.

Заметим, что перебирать по возрастанию не верно. Так как, нам нужно не только количество, но и вес концов минимизировать.

15. **(*) Жадность, а не поток**

    Храним `set` сумм в столбцах. Перебираем строки. Очередную строку выгодно распеределить по максимальны столбцам из `set`-а.

16. **(*) Расход топлива**

    Выгодно покупать бензин «в будушем» на одной из k последних заправок.

17. **(*) Приближенное независимое множество**

    Наберем независимое множество произвольным образом, пока набирается. Т.е. идем по вершинам, если вершина не вычеркнута, возьмем ее в ответ и вычеркнем всех соседей.
    Каждый раз, когда мы берем в ответ одну вершину, из графа уходит $\leqslant D+1$ вершин. Значит, когда кончатся все вершины, наберем $\lceil \frac{V}{D+1} \rceil$.

Домашнее задание

3.1. Обязательная часть

1. **(2) Задания и дедлайны**

   Вы – студент, у вас есть куча дзшек. У каждого задания есть t_i – сколько часов его нужно делать. Чтобы i-е задание зачли, его нужно *начать* делать не позже дедлайна d_i. После того, как вы начали делать задание, его обязательно довести до конца.

   a) **(1)** Можно ли выполнить все задания в срок?
   b) **(1)** Сколько максимум заданий можно выполнить в срок? $\mathcal{O}(n^2)$.

2. **(2) Хаффман и длины кодов**

   Если длина текста n, какова максимальная длина кода одного символа? Приведите пример. Оцениваются только длины кода $\Omega(\log n)$.

   (+1) За оптимальную константу и доказательство оптимальности.

3. **(2) Сбалансированное разделение**

   Под «поддеревом» в данной задаче мы понимаем связный подграф дерева. Дано дерево, разбить все его рёбра на два поддерева, пересекающиеся по одной вершине. Размеры поддеревьев должны отличаться не более чем в 2 раза.

4. **(3) Авторитеты**

   Есть n человек. Человек i готов примкнуть к нашей коалиции, если наш авторитет хотя бы a_i, при этом он изменит `(+=)` наш авторитет на b_i. Наш изначальный авторитет равен A. $b_i \in \mathbb{Z}$ (может быть как положительным, так и отрицательным).

   a) **(2)** Можем ли завербовать всех людей?
   b) **(1)** Какое максимальное число людей мы можем завербовать? $\mathcal{O}(n^2)$.

3.2. Дополнительная часть

1. **(1.5) Хаффман: у частых символов короткие коды**

   Доказать, что если есть символ частоты $> \frac{2}{5}$, то есть символ с длиной кода один.

2. **(3) Жадность между зависимыми задачами**

   Дано корневое дерево зависимостей между задачами. Время выполнение i-й задачи $= t_i$.
   Ориентация дерева: изначально можно выполнить только корень.
   Штраф за задачу $= T_i \cdot fee_i$, где $fee_i \geqslant 0$ даны, T_i – момент начало выполнения.
   Нужно выполнить задачи, минимизируя суммарный штраф. $\mathcal{O}(n \log n)$.

3. **(3) Школьники, глубокая яма и $\mathcal{O}(n \log n)$**

   Какое максимальное число школьников сможет выбраться из ямы?
   Решите за $\mathcal{O}(n \log n)$. **(1.5)**.
   Докажите корректность решения. **(1.5)**.

4. **(3) Исключительные предметы**

   Есть n предметов, у каждого есть стоимость a_i. Можно объединить любые два и получить $x = a_i \,\texttt{xor}\, a_j$ (x добавляется в массив, a_i и a_j удаляются). Уже объединенные предметы объединять повторно нельзя. Требуется сделать сколько-то объединений, чтобы максимальная стоимость стала как можно меньше.

Закончилась практика?
Отнеси лист на второй этаж в эко-мусорку