

SPb HSE, 1 курс, осень 2021/22

Практика по алгоритмам #12

ДП по подмножествам

10 декабря

Собрано 11 декабря 2021 г. в 09:35

Содержание

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

ДП по подмножествам

1. Подмножества и надмножества

Оцените сложность кода:

```

1 for (a = 0; a < (1 << n); ++a)
2   for (b = 0; b < (1 << n); ++b) {
3     c = a & b
4     for (d = c; d > 0; --d &= c)
5       for (e = d; e > 0; --e &= d)
6         ; // сколько раз сюда попадём?
7   }
```

2. Операции с битами

Во всех задачах дано w -битное число, то есть число от 0 до $2^w - 1$.

- За $\mathcal{O}(1)$ проверить, является ли число степенью двойки.
- За $\mathcal{O}(1)$ проверить, правда ли, что в битовой записи никакие две единицы не идут подряд.
- За $\mathcal{O}(\log w)$ найти старший единичный бит.
- Номер младшего единичного бита за $\mathcal{O}(\log w)$.
- Младший единичный бит за $\mathcal{O}(1)$ (ищем 2^k).
- Номер младшего единичного бита за $\mathcal{O}(1)$ (ищем k). *Разрешён предподсчёт.*
- Чётность количества бит за $\mathcal{O}(\log w)$.
- (*) Количество единичных бит за $\mathcal{O}(\log w)$.

3. Паросочетания в произвольном графе

- Посчитать число паросочетаний.
- Найти максимальное по весу паросочетание.
- (*) Посчитать число совершенных паросочетаний для $n \leq 32$.

4. Независимые множества

Для каждого множества вершин в графе проверить, является ли оно независимым. $\mathcal{O}(2^n)$.

5. Независимые подмножества

Для каждого множества вершин в графе посчитать число независимых подмножеств.

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

6. Непересекающиеся подмножества

Выбрать максимальное число непересекающихся из m подмножеств B_1, B_2, \dots, B_m множества $A = \{1, 2, \dots, n\}$. $\mathcal{O}^*(2^{\min(n,m)})$. $n, m \leq 64$.

7. Свертка с единицей

Посчитать для каждого $A \subseteq \{1, 2, \dots, n\}$ $F[A] = \sum_{B \subseteq A} f[B]$.

(a) $\mathcal{O}(3^n)$, (b) (*) $\mathcal{O}(2^n n)$, (c) (*) обратное преобразование за $\mathcal{O}(2^n n)$.

8. Перестановки атакуют

Хотим из перестановки a получить b за 40 ходов.

За ход можно от x переходить к $x \circ p$ или $x \circ q$. Можно ли?

9. Замощение доминошками

Для каких ограничений мы умеем считать количество замощений «доминошками» 1×3 ?

10. (*) Покрытие строки

Покрыть строку s минимальным числом строк $t_i \in T$. Каждую из строк t_i мы или берём (тратим 1), или не берём (тратим 0). Если взяли, можно использовать несколько раз, как подстроку s . Нужно, чтобы каждый символ s был покрыт. $\mathcal{O}^*(2^{\min(|s|, |T|)})$.

11. (*) Покрытие кругами

Даны n точек (x_i, y_i) , покрыть их k кругами одинакового радиуса R , минимизировать R . Интересно наиболее быстрое решение.

12. (*) Гамильтоновы профили

Сколько есть гамильтоновых циклов по клеткам таблицы $n \times n$? ($n \leq 10$).

Представьте теперь себе задачу 8×10^9 . Предложите технические оптимизации.

13. (*) Максимальная клика

а) Найдите максимальную клику графа за $o(2^{n/2})$.

б) Найдите количество клик графа за $o(2^{n/2})$.

Разбор задач практики

1. Подмножества надмножеств подмножеств

Зафиксируем c , обозначим размер k , у него 3^k пар (d, e) , его можно получить 3^{n-k} способами (каждый элемент в a , в b или ни в одно), итого $\sum_k \binom{n}{k} 3^{n-k} 3^k = 3^n \cdot \sum_k \binom{n}{k} = 6^n$.

Другой способ: посмотрим на a, b, c, d, e , у каждого элемента 6 вариантов, где именно лежать – нигде, $a \setminus c, a \setminus c, b \setminus (a \cup c), e, d \setminus e, c \setminus d$.

2. Операции с битами

a) Является ли степенью двойки: $(x \& (x - 1)) == 0$ (выкинули младшую единичку).

b) Нет двух единицы подряд: $(x \& (x \gg 1)) == 0$.

c) Старший бит за $\mathcal{O}(\log w)$: бинпоиск, инвариант $2^L \leq x < 2^R$, ответ в L .

d) Номер младшего единичного бита за $\mathcal{O}(\log w)$: бинпоиск, инвариант $2^L - 1 \& x = 0 \wedge 2^R - 1 \& x \neq 0$, ответ в L .

e) Младший единичный бит за $\mathcal{O}(1)$: $x \& (\sim x + 1)$ или $x \& \sim(x - 1)$.

f) Номер младшего единичного бита за $\mathcal{O}(1)$: можно просто предсчитать $f[2^i] = i$. Для экономии памяти ищем такое m , что $2^i \bmod m$ различные для всех $i = 0..w-1$. Например, для $w = 64$ пойдет $m = 67$.

Предподсчитаем массив: $f[2^i \bmod m] = i$.

Запрос: применить предыдущую задачу и обратиться к массиву.

g) Чётность количества бит за $\mathcal{O}(\log w)$.

Ксорим половинки и чистим верхнюю: $(x \wedge (x \ll w/2)) \gg w/2$.

Чётность числа единичек сохранилась, число вдвое короче.

h) Число единичных бит за $\mathcal{O}(\log w)$.

Изначально число x разбито на группы по одному биту, каждая такая группа содержит двоичную запись числа, равного числу единичных бит в данной группе. Хотим научиться увеличивать вдвое размер таких групп.

Переход от размера k к размеру $2k$:

$$x = (x \& \underbrace{0\dots 01\dots 1}_k \dots \underbrace{0\dots 01\dots 1}_k) + ((x \gg k) \& \underbrace{0\dots 01\dots 1}_k \dots \underbrace{0\dots 01\dots 1}_k)$$

Как получить магические числа, состоящие из блоков нулей и единиц?

В обратном порядке:

$$\underbrace{0\dots 01\dots 1}_k \dots \underbrace{0\dots 01\dots 1}_k = \underbrace{0\dots 01\dots 1}_{2k} \dots \underbrace{0\dots 01\dots 1}_{2k} \wedge (\underbrace{0\dots 01\dots 1}_{2k} \dots \underbrace{0\dots 01\dots 1}_{2k} \ll k)$$

```

1 int bit_count_32(int a):
2     a = ((a >> 1) & 0x55555555) + (a & 0x55555555);
3     a = ((a >> 2) & 0x33333333) + (a & 0x33333333);
4     a = ((a >> 4) & 0xF0F0F0F) + (a & 0xF0F0F0F);
5     a = ((a >> 8) & 0xFF00FF) + (a & 0xFF00FF);
6     a = ((a >> 16) & 0xFFFF) + (a & 0xFFFF);

```

Более быстрые версии [здесь](#).

```

1 #pragma GCC target("popcnt") // сказать g++ "мой процессор это умеет за 1 такт"
2 std::popcount(int x) // наконец-то можно пользоваться стандартной функцией!

```

3. Паросочетания в произвольном графе

- a) $\text{count}[S]$ – число паросочетаний в подмножестве. Фиксируем $v \in S$.
Либо у v нет пары, тогда $\text{count}[S] += \text{count}[S \setminus \{v\}]$, либо переберём ребро $(v, u), u \in S$,
 $\text{count}[S] += \text{count}[S \setminus \{v, u\}]$. Итого $\mathcal{O}(2^n)$.
- b) Максимальное по весу точно так же, `relax` вместо `+=`.
- c) Переход: выбрать пару младшего бита. Ленивая динамика. Оценка времени работы: Посмотрим на все множества A , которые посетила наша ленивая динамика.
 $A = a_1 a_2 \dots a_n$. $a_i = 1 \Leftrightarrow i$ -я вершина покрыта паросочетанием.
Если $a_n = 1$, то $a_1 = 1 \Rightarrow F(n) = F(n-1) + F(n-2)$.

4. Независимые множества

Фиксируем $v \in S$ (младший бит, например).
 $\text{is_independent}[S] = \text{is_independent}[S \setminus \{v\}] \wedge S \cap N(v) = \emptyset$.

5. Независимые подмножества

- a) У нас уже есть динамика $\text{is_independent}[S]$, посчитанная за $\mathcal{O}(2^n)$.
За $\mathcal{O}(3^n)$ перебрать у каждого множества все подмножества, посчитать независимые.
- b) $\mathcal{O}(2^n)$. Фиксируем одну $v \in S$, либо независимое не содержит v , либо содержит, но тогда не содержит соседей v . $\text{count}[S] = \text{count}[S \setminus \{v\}] + \text{count}[S \setminus \{v\} \setminus N(v)]$.

Ленивое решение за $\mathcal{O}(2^{\min(n,m)})$: пишем рекурсивный перебор за 2^m с запоминанием.

6. Непересекающиеся подмножества

$\mathcal{O}(2^m)$ – рекурсивная процедура, каждое из m множеств берём или не берём.

За $\mathcal{O}(2^m)$ рюкзак по множествам.

Пусть $d[S]$ – максимальное число непересекающихся множеств с объединением S .

К каждому S пытаемся добавить каждое из множеств:

```

1 for (S = 0; S < (1 << n); S++)
2     for (i = 0; i < m; i++)
3         if ((S & A[i]) == 0)
4             relax(d[S | A[i]], d[S] + 1)

```

7. Свертка с единицей

$$F[A] = \sum_{B \subseteq A} f[B].$$

- a) $\mathcal{O}(3^n)$ – в лоб.
- b) $\mathcal{O}(2^n)$. Динамика $d[S, i]$ – сумма $f[A]$ по всем $A \subseteq S$, таким, что их хвост, начиная с бита i , совпадает с S . База динамики: $d[S, 0] = f[S]$. Переходы: $d[S, i + 1] = d[S, i]$;
 $\text{if } (\text{bit}(S, i) == 1) d[S, i + 1] += d[S - (1 \ll i), i]$. *Ответ*: $F[A] = d[A, n]$.

```

1 F = f
2 for k = 0..n-1
3     for A = 0..2^n-1
4         if A & 2^k
5             F[A] += F[A - 2^k]

```

- c) Обратим код выше (технически)

```

1 // F → f
2 for k = n-1..0
3     for A = 2^n-1..0

```

```

4     if A & 2^k
5         F[A] -= F[A - 2^k]

```

8. Перестановки атакуют

Meet in the middle – с каждого конца делаем по 20, получили два множества из 2^{20} перестановок, нужно проверить, пусто ли их пересечение.

9. Замоещение доминошками

Делаем, как на лекции. Число профилей, очевидно, $\mathcal{O}(wh4^w)$ (торчит две клетки). Можно оценить, как $\mathcal{O}(wh3^w)$ (так как не 4, а 3 варианта в каждом столбце).

Проведём эксперимент. Смотрим число профилей с ростом w в зависимости от $w \bmod 3$. Их количество увеличивается в 2, 1.7, 3 раз соответственно $\Rightarrow \approx 2.17^w$ профилей при больших w .

10. (*) Покрывание строки

Пусть $|s| = m$, $|T| = n$, w – размер машинного слова.

Решение $\mathcal{O}(mn + 2^n \frac{m}{w})$ – для каждого слова из T предподсчитали маску позиций в s , которые покроет слово. Рекурсивно перебираем, какие из слов T мы берём, при спуске в рекурсию поддерживаем объединение масок.

За $\mathcal{O}^*(2^{|s|})$. Динамика $\text{opt}[A, i]$ – минимальное число строк среди $T[1..i]$, которыми можно покрыть множество A позиций строки s . Опять рюкзак \Rightarrow Асимптотика $2^m n$.

11. (*) Покрывание кругами

Для каждого множества S предподсчитаем минимальный радиус *одного* круга, которым можно покрыть S . Мы это умеем делать за $\mathcal{O}(n^4)$ и за $\mathcal{O}(n \log^2 M)$ (вообще можно за $\mathcal{O}(n)$). Динамика $\text{minR}[S, i]$ – покрываем множество S ровно i кругами. Переход – добавить множество точек, покрытых ещё одним кругом. Получили $\mathcal{O}(2^n n^4 + 3^n k)$.

Заметим, что выгодно использовать только круги, опирающиеся на тройки и двойки вершин. Таких всего $m = \frac{n^3}{6} + \frac{n^2}{2}$. Будем использовать m в асимптотике. Предподсчитать $\forall S \text{ minR}[S]$ можно за $\mathcal{O}(m + 2^n n)$ динамикой. Мы получаем два решения: $\mathcal{O}(m + 2^n n + 3^n k)$ и $\mathcal{O}(2^n km)$.

Наконец, можно получить $\mathcal{O}(2^n m)$. Переберём все круги в порядке возрастания радиуса. Поддерживаем $\text{minCount}[S]$ – минимальное число кругов радиуса не больше текущего, нужных для покрытия множества S . Релаксируем текущим кругом все S (как в задаче о рюкзаке).

Бинпоиск по радиусу. Внутри только $z = n^2$ интересных кругов (опираются на две точки). Считаем $\text{minCount}[S]$, получаем $\mathcal{O}(2^n z)$ внутри бинпоиска. Всего $\mathcal{O}(2^n n^2 \log C)$.

В решениях, содержащих k в асимптотике можно улучшить $k \rightarrow k^{1/2}$, если заранее сделать `randomShuffle` кругов и смотреть на отклонение от среднего.

12. (*) Гамильтоновы профили

?

13. (*) Количество клик

На лекции была разобрана идея «перебор с запоминанием». Для любого порядка вершин мы получили время $2^{n/2}$. Давайте придумаем хороший порядок вершин.

Отсечение: если все рёбра присутствуют, то ответ = 2^V .

Иначе пусть нет ребра $a \leftrightarrow b$, поставим a и b в начало \Rightarrow два шага рекурсии приводят к ветвлению не в 4, а в 3 стороны \Rightarrow решаем $3^k = 2^{n-2k}$ (степень ветвления, остаток для

запоминания). Получаем $12^k = 2^n \Rightarrow k = \frac{1}{\log 12}n \Rightarrow (3^{1/\log 12})^n = 1.35859371^n$.

Ещё лучше. Пусть «отсутствующие рёбра» образуют паросочетание \Rightarrow мы можем по множеству вершин A за $\mathcal{O}(|A|)$ получить число подклик как $3^{|M|}2^{|A|-2|M|}$. Иначе есть a, b, c без рёбер (a, b) и (a, c) . 8 вариантов, 5 ветвлений $\Rightarrow 5^k = 2^{n-3k} \Rightarrow 40^k = 2^n \Rightarrow k = \frac{1}{\log 40}n \Rightarrow 1.3531^n$.

Пусть «отсутствующие рёбра» образуют дерево $\Rightarrow \forall A$ можем за $\mathcal{O}(|A|)$ насчитать динамику по дереву. Иначе есть цикл. Анализ показывает, что худший случай – цикл длины 4.

7 из 16 вариантов \Rightarrow получаем степень ветвления $x = 7^{1/4} = 1.62658$ и асимптотику 1.3309^n .

Домашнее задание

3.1. Обязательная часть

1. (1) Битовые прелести

Даны число x , за $\mathcal{O}(1)$ замените в нём все младшие нули на единицы.

Пример: 000101001110000 \rightarrow 000101001111111

2. (2) Фишки на матрице

Дана матрица $w \times h$. В каждой клетке матрицы стоит фишка одного из k типов.

Проходя по клетке, мы обязательно берем фишку и платим ее стоимость.

Стоимость фишки в клетке (i, j) равна $cost[i, j]$, независимо от ее типа.

Наша задача — смещаясь только вверх и вправо из клетки $(1, 1)$ в клетку (w, h) , собрать набор фишек, где будут все k типов, за минимальную стоимость.

Требуемое время работы программы в условии намеренно не указано.

Полный балл получают только решения, работающие для $w, h \geq 50$.

3. (3) Число совершенных паросочетаний

Дан двудольный граф, в первой доле $m \leq 15$ вершин, во второй $n \leq 1\,000$ вершин. Предложите алгоритм, который считает число паросочетаний, покрывающих все m вершин первой доли (совершенные паросочетания).

Асимптотика заранее неизвестна, ориентируйтесь на время работы на C++ ≈ 1 секунду. Можно оптимизировать константу. Решения лучше по константе получают больше баллов. Не нужно писать и запускать код, проведите аргументы, чем ваше решение хорошо по константе.

4. (4) Разбиение на пути

Посчитать за $\mathcal{O}^*(2^n)$ количество способов все вершины орграфа разбить на простые пути. Каждая вершина должна лежать ровно в одном пути, каждый путь содержит не менее двух вершин. Разбиения на пути различны, если различны множества использованных рёбер.

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

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

c) (+1) $\mathcal{O}^*(2^n)$, бьём на циклы.

5. (4) Расписание

Нужно составить расписание на один день в школе. Главная цель — чтобы все ушли домой пораньше, то есть минимизировать время окончания самого позднего урока. Уроки идут по расписанию, от звонка до звонка, все приходят утром к первому уроку.

За день обязательно провести некоторое множество уроков Q , каждый урок q_i задается тройкой $\langle a, b, C \rangle$, где a — учебная группа, b — преподаватель, C — множество аудиторий, в которых можно провести занятие. В каждый момент времени преподаватель может вести не более чем одну группу, а группа слушать не более, чем одного преподавателя, также в одной аудитории нельзя одновременно проводить больше одного занятия.

Составьте расписание за (2) $\mathcal{O}(9^{|Q|})$, (2) $\mathcal{O}(3^{|Q|})$.

В этой задаче полезно (но не обязательно) уметь за полином искать максимальное паросочетание в двудольном графе. Можно посмотреть в wiki и сослаться на существование решения.

3.2. Дополнительная часть

1. (3) Взвешивание

В волшебном мире «подмножества и их друзья» бывают всего два набора гирь:

a_1, a_2, \dots, a_n и b_1, b_2, \dots, b_n . У вас есть ряд из n с виду одинаковых гирь.

Известно, что в ряду гири лежат в правильном порядке – или a_1, \dots, a_n , или b_1, \dots, b_n .

Сделайте одно взвешивание на чашечных весах, чтобы понять, какой набор попал вам в руки? Каждую гирю можно положить на одну из чашей или не класть никуда. $n \leq 24$.

2. (3.5) Замощения конями

Сколько способов расставить k шахматных коней на доске $n \times n$ так, чтобы они не били друг друга?

- a) (1) $\mathcal{O}^*(4^n)$
- b) (0.5) $\mathcal{O}^*((4-\varepsilon)^n)$
- c) (1) $\mathcal{O}^*(3^n)$
- d) (1) $\mathcal{O}((3-\varepsilon)^n)$

3. (2) Перебор максимальных по включению клик

Дан граф $G = \langle V, E \rangle$ на n вершинах, найти для каждого $A \subseteq \{0, 1, \dots, n-1\}$ $list[A]$ – множество всех максимальных по включению подклик A . Время работы должно быть $\mathcal{O}^*(M)$, где $M = \sum_A |list[A]|$, т.е. оптимальным с точностью до полинома. $B \subseteq A$ называется подклик, если $\forall i, j \in B: (i, j) \in E$. Подклика $B \subseteq A$ называется максимальной по включению, если $\forall k \in A \setminus B: A \cup \{k\}$ не является подклик (нельзя к ней ничего добавить).

4. (4) Уменьшение алфавита

Дана строка длины $n \leq 10^6$ над алфавитом $k \leq 26$ и число $m \leq n$. Нужно выбрать некоторое множество букв A и заменить все вхождения букв из A в строку на пробелы. Таким образом нужно добиться, чтобы не было m подряд идущих не пробелов. Выберите минимальное по размеру такое A . $\mathcal{O}(n + 2^k)$. Частичный балл за $\mathcal{O}(n + 2^k \cdot poly(k))$.

5. (3) Махинации

У нас есть n типов товаров, суммарное число товаров A .

Наш магазин занимается махинациями.

В t -й момент времени, если у нас для каждого i от 1 до n есть хотя бы $a_{t,i}$ товаров типа i , мы можем взять $a_{t,1}$ товаров типа 1, $a_{t,2}$ товаров типа 2, \dots , $a_{t,n}$ товаров типа n и одновременно заменить их на $b_{t,1}$ товаров типа 1, $b_{t,2}$ товаров типа 2, \dots , $b_{t,n}$ товаров типа n .

Махинации махинациями, а закон сохранения действует: $\forall t \sum_i a_{t,i} = \sum_i b_{t,i}$.

Махинацию t мы можем применить только один раз и только в t -й момент времени.

Задача: можем ли мы после T моментов получить ровно c_1 товаров типа 1, \dots , c_n товаров типа n (сумма c_i равна A)? Нужно решение за $\mathcal{O}(2^{A+n}T)$.