# Первый курс, осенний семестр 2018/19
# Практика по алгоритмам #11

## ДП последнее!

## 17 ноября

Собрано 20 ноября 2018 г. в 00:18

---

## Содержание

# ДП последнее!

1. **Покраска в 4 цвета**
   За $\mathcal{O}^*(2^n)$ проверить, можно ли покрасить граф в 4 цвета.
   *Подсказка: в два цвета красить очень просто.*

2. **Операции с битами**
   Во всех задачах дано $w$-битное число, то есть число от 0 до $2^w - 1$.
   a) За $\mathcal{O}(1)$ проверить, является ли число степенью двойки.
   b) За $\mathcal{O}(1)$ проверить, правда ли, что в битовой записи никакие две единицы не идут подряд.
   c) За $\mathcal{O}(\log w)$ найти старший единичный бит.
   d) Номер младшего единичного бита за $\mathcal{O}(\log w)$.
   e) Младший единичный бит за $\mathcal{O}(1)$ (ищем $2^k$).
   f) Чётность количества бит за $\mathcal{O}(\log w)$.
   g) Количество единичных бит за $\mathcal{O}(\log w)$.

3. **Паросочетания в произвольном графе**
   a) Посчитать число паросочетаний.
   b) Найти максимальное по весу паросочетание.

4. **Матричные узоры**
   Клетки поля $n \times 5$ покрашены в чёрный и белый цвета. Будем называть получившийся узор красивым, если он не содержит одноцветного квадрата $2 \times 2$. Вычислите число красивых узоров по модулю небольшого простого числа за время $\mathcal{O}(\log n)$.

5. **Разрезание на прямоугольники**
   Вам дана доска фанеры размера $n \times m$. В нее было вбито несколько гвоздей с целыми координатами (от них остались некрасивые дырки). Сколькими способами можно разрезать доску на прямоугольники с целыми сторонами так, чтобы ни один из гвоздей не попал внутрь прямоугольника. Время: $\mathcal{O}(4^m nm)$.

6. **Свертка с единицей**
   Посчитать для каждого $A \subseteq \{1, 2, \ldots, n\}$ $F[A] = \sum_{B \subseteq A} f[B]$. (a) $\mathcal{O}(3^n)$, (b) $\mathcal{O}(2^n n)$.

7. **(*) Свертка**
   $(f * g)[A] = \sum_{B \cup C = A} f[B] \cdot g[C]$. Даны $f$ и $g$, посчитать $f * g$.

8. **(*) Покрытие кругами**
   Даны $n$ точек $(x_i, y_i)$, покрыть их $k$ кругами одинакового радиуса $R$, минимизировать $R$.

9. **(*) Число клик**
   Найти количество клик графа за $o(2^{n/2})$.

# Разбор задач практики

1. **Покраска в 4 цвета**
Переберём множество вершин, которое покрасится в два цвета. Остальные вершины тоже должны покраситься в два цвета. Проверим для каждого как угодно за полином.

2. **Операции с битами**
   a) Является ли степенью двойки: `(x & (x - 1)) == 0` (выкинули младшую единичку).
   b) Нет двух единицы подряд: `(x & (x >> 1)) == 0`.
   c) Старший бит за $\mathcal{O}(\log w)$:
   бинпоиск, инвариант `(1 << l) <= x, (1 << r) > x`, ответ в `l`.
   d) Номер младшего единичного бита за $\mathcal{O}(\log w)$:
   бинпоиск, инвариант `((1 << l) - 1) & x == 0, ((1 << r) - 1) & x != 0`, ответ в `l`.
   e) Младший единичный бита за $\mathcal{O}(1)$: `x & (~x + 1)` или `x & ~(x - 1)`.
   f) Чётность количества бит за $\mathcal{O}(\log w)$.
   Ксорим половинки и чистим верхнюю: `(x ^ (x << w/2)) >> w/2`. Четность числа единичек сохранилась, число вдвое короче.
   g) 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 $2k$:
   $\texttt{x = x \& } \underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{k}\ldots\underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{k} \texttt{ + (x >> k) \& } \underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{k}\ldots\underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{k}.$
   Как получить магические числа, состоящие из блоков нулей и единиц?
   В обратном порядке:
   $\underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{k}\ldots\underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{k} \texttt{ = } \underbrace{0\ldots0}_{2k}\underbrace{1\ldots1}_{2k}\ldots\underbrace{0\ldots0}_{2k}\underbrace{1\ldots1}_{2k} \texttt{ \^{} (}\underbrace{0\ldots0}_{2k}\underbrace{1\ldots1}_{2k}\ldots\underbrace{0\ldots0}_{2k}\underbrace{1\ldots1}_{2k} \texttt{ << k)}$

```
int bit_count_32( int a ) {
    a = ((a >>  1) & 0x55555555) + (a & 0x55555555);
    a = ((a >>  2) & 0x33333333) + (a & 0x33333333);
    a = ((a >>  4) & 0xF0F0F0F)  + (a & 0xF0F0F0F);
    a = ((a >>  8) & 0xFF00FF)   + (a & 0xFF00FF);
    a = ((a >> 16) & 0xFFFF)     + (a & 0xFFFF);
}
```
   Более быстрые версии здесь.

3. **Паросочетания в произвольном графе**
   a) `count`$[S]$ – число паросочетаний в подмножестве. Фиксируем $v \in S$.
   Либо у $v$ нет пары, тогда `count`$[S]$ `+=` `count`$[S \setminus \{v\}]$, либо переберем ребро $(v, u), u \in S$, `count`$[S]$ `+=` `count`$[S \setminus \{v, u\}]$. Итого $\mathcal{O}(2^n n)$.
   b) Максимальное по весу точно так же, `relax` вместо `+=`.

4. **Матричные узоры**
Профиль – раскраска столбца. Для каждой пары профилей выясним, могут ли они быть соседними. $A[i, j]$ – могут ли быть соседними профили $i$ и $j$. Заметим, что динамику по такому профилю можно выразить через умножение на $A$, возведём её в степень $n$.

5. **Разрезание на прямоугольники**
Профиль – кодирование горизонтальных разрезов столбца. Посмотрим на два соседних профиля. Если в них есть на одних и тех же позициях пара горизонтальных разрезов, и между этими разрезами нет других разрезов ни в одном из профилей, то в этом месте может быть вертикальный разрез. Если всего таких мест $k$, то способов перейти от одного профиля к другому $2^k$. А дырки запрещают нам отсутствие каких-то разрезов внутри профиля и между соседними.
$d[i][mask]$ – число способов разрезать первые $i$ столбцов, последний имеет профиль $mask$. Состояний $\mathcal{O}(2^m n)$, пересчитываем через все профили предыдущего столбца за $\mathcal{O}(2^m)$, для каждого находим места возможных вертикальных разрезов за $\mathcal{O}(m)$, всё вместе за $\mathcal{O}(4^m nm)$.

6. **Свертка с единицей**
$F[A] = \sum\limits_{B \subseteq A} f[B]$.
   a) $\mathcal{O}(3^n)$ – в лоб.
   b) $\mathcal{O}(2^n n)$. Динамика `d[S, i]` – сумма `f[A]` по всем `A` $\in$ `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]; if (bit(S, i) == 1) d[S, i + 1] += d[S - (1 << i), i]`. Ответ: `F[A] = d[A, n]`.

7. **Свертка**
Пусть $h = f * g$. Пусть $H[A] = \sum\limits_{B \subseteq A} h[B]$. Аналогично определим $F[A]$ и $G[A]$.
Тогда $H[A] = \sum\limits_{B \subseteq A} \sum\limits_{C \cup D = B} f[C] \cdot g[D] = \sum\limits_{C, D \subseteq A} f[C] \cdot g[D] = (\sum\limits_{C \subseteq A} f[C])(\sum\limits_{D \subseteq A} g[D]) = F[A] \cdot G[A]$.
За $\mathcal{O}(2^n n)$ считаем $F$ и $G$, за $\mathcal{O}(2^n)$ считаем $H$. Осталось по $H$ получить $h$. Несложно убедиться, что $h[A] = \sum\limits_{B \subseteq A} (-1)^{|A|-|B|} H[B]$.
То есть $h'[A] = (-1)^{|A|} h[A] = \sum\limits_{B \subseteq A} (-1)^{|B|} H[B]$, обозначим $H'[B] = (-1)^{|B|} H[B]$.
Получили $\varphi \colon h \to H$, то $\varphi \colon H' \to h'$.

8. **Покрытие 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 $\mathcal{O}(n \log^2 M)$. Динамика `min_radius[S, i]` – покрываем множество `S` `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, опирающиейся на тройки и двойки вершин. Таких всего $\frac{n^3}{6} + \frac{n^2}{2}$. Тогда для каждого множества минимальный радиус можно предподсчитать за $\mathcal{O}(n^3 + 2^n n)$, и мы получаем два решения: $\mathcal{O}(2^n n + 3^n k)$ и $\mathcal{O}(2^n k n^3)$.

   Наконец, можно получить $\mathcal{O}(2^n n^3)$. Переберем все круги в порядке возрастания радиуса. Будем поддерживать величину `min_count[S]` – минимальное число кругов радиуса не больше текущего, нужных для покрытия множества `S`. Релаксируем текущим кругом все `S` (как в задаче о рюкзаке).

9. **Число клик**
Гроб. Тем не менее придумывается, зная только пройденное на лекции.

# 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 → 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.
3. **(3) Разбиение на пути**
   Посчитать за $\mathcal{O}^*(2^n)$количество способов все вершины **ор**графа 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) **(1)** $\mathcal{O}(3^n)$, b) **(2)** $\mathcal{O}^*(2^n)$
4. **(2) Число совершенных паросочетаний**
   Дан двудольный граф, в первой доле $m \leqslant 15$ вершин, во второй $n \leqslant 1000$ вершин. Предложите алгоритм, который считает число паросочетаний, покрывающих все $m$ вершин первой доли (совершенные паросочетания).
   Асимптотика заранее неизвестна, но таймлимит для реалиазации на языке `C++` – 1 секунда.
5. **(3) 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 a) **(2)** $\mathcal{O}(9^{|Q|})$, b) **(3)** $\mathcal{O}(3^{|Q|})$.

   Для решения этой задачи полезно уметь за полином искать максимальное паросочетание в двудольном графе. Можно посмотреть формулировку задачи в википедии и сослаться на существование решения.

## 3.2. Дополнительная часть

1. **(2) Замощения длинными доминошками**
   Дано поле $n \times n$ с дырками. Сколько способов замостить его фигурами $1 \times 3$? $\mathcal{O}^*(3^n)$

2. **(3) 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 $M = \sum\limits_{A} |list[A]|$, т.е. оптимальным с точностью до полинома. $B \subseteq A$ называется подкликой, если $\forall i, j \in B \colon (i, j) \in E$. Подклика $B \subseteq A$ называется максимальной по включению, если $\forall k \in A \setminus B \colon A \cup \{k\}$ не является подкликой (нельзя к ней ничего добавить).

3. **(4) Уменьшение алфавита**
   Дана строка длины $n \leqslant 10^6$ над алфавитом $k \leqslant 26$ и число $m \leqslant n$. 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 $A$. $\mathcal{O}(n + 2^k)$.

4. **(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: $\forall t \sum\limits_i a_{t,i} = \sum\limits_i b_{t,i}$.

   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 $\mathcal{O}(2^{A+n}T)$.