Лекция по алгоритмам #9
Тема: Динамика на подмножествах

28 октября

Собрано 27 декабря 2014 г. в 02:47

Содержание

1 Множества и битовые операции

$A \subset \{0, 1, \ldots, n-1\}$
Пример: $A = \{2, 3\}\ 2^2 + 2^3$

Побитовые операторы:

1. $<<$ — сдвиг влево, дописывает справа нули;
2. $>>$ — сдвиг вправо, дописывает слева нули;
3. $\sim A = \text{not } A$;
4. $A\&B = A \text{ and } B$;
5. $A|B = A \text{ or } B$;
6. $A\hat{}B = A \text{ xor } B$;

Наличие элемента в множестве: $x \in A : (A >> x)\&1$
Добавление элемента: $A \cup \{x\} : A|(1 << x)$
Удаление элемента: $A \backslash \{x\}$: $A\& \sim (1 << x)$
Пересечение: $A\&B$
Объединение: $A|B$
Вычитание: $A\& \sim B$
Является ли одно множество подмножеством другого: $A \subset B\ (A\&B) == A$
Замена на противоположный: $A\hat{}(1 << x)$

2 Решение задач с использованием битовых операций

1. **Перебор множеств, сумма**

   FOR $A = 0 \ldots 2^N - 1$
   $A\& \sim (A-1) = 2^i$, где i — минимальный единичный бит.

   Предподсчет:

```
sum[0] = 0
for (i = 0; i < n; ++i)
        sum[1 << i] = w[i]
```

   1 способ:

```
for (A = 1; A < (1 << n); ++A)
        B = A & ~(A - 1)
        sum[A] = sum[B] + sum[A ^ B]
```

   2 способ:

```
bit = 0
for (A = 1; A < (1 << n); ++A)
        if ((1 << (bit + 1)) == A)
                ++bit
        sum[A] = w[bit] + sum[A ^ (1 << bit)]
```

3 способ:

```
REC(i, A, sum)
        if (i < 0)
                s[A] = sum;
                return;
        REC(i - 1, A, sum)
        REC(i - 1, A|(1 << i), sum + w[i])
```

Запуск: $REC(n-1, 0, 0)$

2. **Число бит в множестве**

   $num[A] = num[A >> 1] + (A\&1)$

3. **Переворот множества**

   Пример: 01011 11010
   $rev[A] = rev[A >> 1] + ((A\&1) << (n-1))$

4 **Покрытие множества минимальным количеством множеств**

(a) $B, A_1, A_2, \ldots, A_m \subset \{0, 1, \ldots, n-1\}$
$B = \cup A_i \ : \ |I| = \min$
$union[I] = A[bit] | union[I \hat{} 2^{bit}]$, где bit — старший или младший бит $(0 \le bit \le m-1)$
$I = 0, \ldots, 2^m - 1$
После этого проверяем, лежит ли B в $union[I]$.
Асимптотика $\mathcal{O}(2^m)$

(b) $f[i, B]\ min$
i — количество рассмотренных подмножеств
B — объединение
f — сколько взяли

$f[i, B] \to f[i+1, B|A_i], f[i+1, B]$

$Mem = \mathcal{O}(2^n)$ при хранении только последних 2 строчек
Асимптотика $\mathcal{O}(m \times 2^n)$

5 **Vertex Coloring**

$edge(a, b)$: $c[a] \neq c[b], c[v] \in \{1, \ldots, k\}$ $k \min$
$k[A] = \min\limits_{B \subset A, good(B)=1} (k[A \hat{} B] + 1)$
$good(B) = 1$ — можно покрасить множество B в один цвет
$good[B]$ $\mathcal{O}(n^2 \times 2^n)$

(a) $\mathcal{O}(4^n)$

```
for (A = 1; A < (1 << n); ++A)
        for (B = 1; B < (1 << n); ++B)
                if (B subset A)
                        ...
```

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

```
for (A = 1; A < (1 << n); ++A)
        for (B = A; B > 0; --B, B &= A)
                ...
```

(c) $good[A]$ за $\mathcal{O}(2^n)$
$good[A] = good[A \hat{} 2^{bit}]$ and $(adj[bit] \& A) == 0$, $adj[bit]$ — соседи bit

Перебор всех надмножеств

```
for (D = A; D < (1 << n); ++D, D |= A)
```

(d) Заметим, что найти минимальную раскраску для множества A мы можем, найдя его максимальное по включению независимое подмножество(не является подмножеством другого независимого множества)
Задача нахождения всех max по включению множеств решается за $\mathcal{O}(3^{n/3})$
Так что задачу min раскраски можно решить за $\mathcal{O}(2.44^n)$

3 Задача нахождения гамильтонова пути и цикла

$is[A, v] = 0$ or 1 — посетили множество вершин A, v — последняя вершина в обходе

1. $T = \mathcal{O}(2^n \times n^2)$
   $Mem = \mathcal{O}(2^n \times n)$

```
for (A = 0; A < (1 << n); ++A)
        for (v = 0; v < n; ++v)
                if (is[A, v])
                        for (x = 0; x < n; ++x)
                                if (c[v, x] == 1 and A & (1 << x) == 0)
                                        is[A | (1 << x), x] = 1
```

2. $T = \mathcal{O}(2^n \times n)$
   $Mem = \mathcal{O}(2^n)$

   $end[A] = 01\ldots$ (n цифр) — множество вершин, на которые мог закончиться путь A
   $adj[x]$ — соседи вершины x

```
for (A = 0; A < (1 << n); ++A)
        for (x = 0; x < n; ++x)
                if (A & (1 << x) == 0 and (end[A] & adj[x]) != 0)
                        end[A | (1 << x)] |= (1 << x)
```

   Инициализация для нахождения пути: $end[2^x] = 2^x, x = 0, \ldots, n-1$
   Инициализация для нахождения цикла: $end[2^0] = 2^0$, т.е. начинаем цикл с вершины $v = 0$, поэтому в конце условие на достижимость данной вершины из $end[2^n - 1]$