# ОЛІМПІАДА З ІНФОРМАТИКИ У МІСТІ КИЄВІ У 2012–2013 НАВЧАЛЬНОМУ РОЦІ

Продовження, початок у №8 за 2013 рік, №1 за 2014 рік

**Знов'як Юрій Володимирович,**
*інженер з програмного забезпечення Google New York Center,*
**Мисак Данило Петрович,**
*керівник гуртка СШ №52 м. Києва,*
**Рибак Олександр Владиславович,**
*аспірант Інституту математики НАН України,*
**Рудик Олександр Борисович,**
*кандидат фізико-математичних наук, доцент кафедри методики природничо-математичної освіти і технологій Інституту післядипломної педагогічної освіти Київського університету імені Бориса Грінченка.*

**5. Вкладені множини (автор — Юрій Знов'як)**

**Динамічне програмування (неповне розв'язання)**

Позначимо через $f(n,Q)$ кількість послідовностей вкладених множин $S_1 \supseteq S_2 \supseteq \ldots \supseteq S_n$, при яких:

- $S_0 \supseteq S_1$;
- $H(S_1) \bmod 41 = h_1$, $H(S_2) \bmod 4 = h_2$, ..., $H(S_n) \bmod 41 = h_n$;
- $S_n = Q$.

Інакше кажучи, $f(n,Q)$ — відповідь на задачу з умови при $n=N$ і заданої кінцевої підмножини ($S_n=Q$). При такому означенні $f(n,Q)$ шукану кількість можна подати сумою:

$$\sum_{Q \subseteq S_0} f(N,Q). \qquad (1)$$

Тут додавання здійснено за всіма $Q$, що є підмножинами $S_0$.

Розглянемо приклад №1 з умови: $N=3$, $M=5$, $h_1=7$, $h_2=3$, $h_3=3$. Згідно з умовою $S_0=\{a_1, a_2, ..., a_5\}$. Існує лише один набір вкладених підмножин, що задовольняє умові. А саме: $S_1=\{a_1, a_2, a_3\}$, $S_2=S_3=\{a_1, a_2\}$.

Маємо:

$f(1, \{a_1, a_2, a_3\})=1$, $f(1, Q)=0$ при $Q \neq \{a_1, a_2, a_3\}$,

$f(2, \{a_1, a_2\})=1$, $f(2, Q)=0$ при $Q \neq \{a_1, a_2\}$,

$f(3, \{a_1, a_2\})=1$, $f(3, Q)=0$ при $Q \neq \{a_1, a_2\}$.

Зауважимо: $f(1, Q)$ — це кількість послідовностей з одного елемента із заданим останнім елементом, що відповідає властивості $H(Q) \bmod 41 = h_1$.

Маємо:

$f(1, Q)=1$ при $H(Q) \bmod 41 = h_1$, (2)

$f(1, Q)=0$ при $H(Q) \bmod 41 \neq h_1$. (3)

Покажемо, як обчислити $f(n, Q)$ при $n>1$, якщо відомі $f(n-1, R)$ при всіх $R$ $S_0$. Зауважимо: $f(n, Q)$ — це кількість різних послідовностей вкладених множин $S_1 \supseteq S_2 \supseteq \ldots \supseteq S_{n-1} \supseteq S_n$. Маємо рекурентне співвідношення:

$$f(n, S_n) = \sum_{S_n \subseteq S_{n-1} \subseteq S_0} f(n-1, S_{n-1}). \qquad (4)$$

Тут додавання здійснено за всіма $S_{n-1}$, що є підмножиною $S_0$ і містять $S_n$ як підмножину.

Використавши початкові значення (2–3) і рекурентну формулу (4), можна знайти всі доданки відповіді — суми (1).

Розглянемо питання реалізації. Як подати $f(n,Q)$? Доволі очевидним рішенням буде подання $f(n, Q)$ масивом:

f: array[1..N, 0..$2^M-1$] of Int64; //FreePascal

int64 f[N] [$2^M$]; //C++

Перший індекс — $n$, другий індекс — $H(Q)$:

$f(n,Q)$ =f[$n$] [$H(Q)$].

На жаль, таке розв'язання використовує занадто багато пам'яті.

**Динамічне програмування 2 (покращене неповне розв'язання)**

**Зауважимо:** $f(n,Q)=0$ при всіх $Q$ таких, що $H(Q) \bmod 41 \neq h_n$. Можна у 41 раз зменшити потребу щодо пам'яті:

f: array[1..N, 0.. $2^M$ div 41] of Int64; // FreePascal

int64 f[N] [$2^M/41$]; // C/C++

У цьому випадку величину $f(n,Q)$ зберігають не в $f[n][H(Q)]$, а в $f[n][H(Q) \text{ div } 41]$.

**Динамічне програмування 3 (повне розв'язання)**

Найтривалішим у поданому розв'язанні є обчислення згідно з рекурентною формулою (4) — необхідно швидко перелічити всі підмножини $S_{n-1}$ заданої множини $S_n$. На щастя, це зробити дуже легко!

Спочатку вирішимо, як подаватимемо множини. Найзручнішим буде працювати з множинами як зі звичайними цілими числами (LongInt у FreePascal або int в C++). Біт 0 відповідатиме елементу $a_1$, біт 1 — елементу $a_2$ і т.д. Наприклад, множині $\{a_1, a_3\}$ відповідає число $H(\{a_1, a_3\})=101_2=5_{10}$. Тут $H$ — функція з умови завдання.

Існує зв'язок між діями з множинами — аргументами функції $H$ — і побітовими діями з величинами функції $H$ (використано позначення C/C++):

$H(A \supseteq B)=H(A) | H(B)$;

$H(A \cap B)=H(A) \& H(B)$,

де:

| — операція побітового «або» (у FreePascal пишуть or );

& — операція побітового «і» (у FreePascal пишуть and);

Доведемо, що результатом $X\&(X-1)$ є відкидання наймолодшого одиничного біту числа. Подамо число $X$ у двійковій системі числення:

$$X=(x_k x_{k-1}\ldots x_1 x_0)_2=x_0\cdot 2^0+x_1\cdot 2^1+\ldots+x_k\cdot 2^k.$$

Без обмеження загальності міркувань будемо вважати, що саме $m$ наймолодших бітів $X$ — нулі. Інакше кажучи, $X$ ділиться без лишку на $2^m$, але не ділиться на $2^{m+1}$: $x_0=x_1=\ldots=x_{m-1}=0$, $x_m=1$.

Маємо:

$$X=(x_k x_{k-1}\ldots x_{m+1}10\ldots0)_2,$$
$$X-1=(x_k x_{k-1}\ldots x_{m+1}01\ldots1)_2.$$

Наймолодші $(m+1)$ біт чисел $X$ і $X-1$ відрізняються, а решта бітів — однакові. Тому

$$X\&(X-1)=(x_k x_{k-1}\ldots x_{m+1}00\ldots0)_2,$$

що й потрібно було довести. Саме ця рівність лежить в основі переліку підмножин даної множини.

Нехай $Q=\{a_r, a_s, a_t, \ldots, a_u\}$, і $R$ — непорожня підмножина $Q$.

Маємо:

$$H(Q)=2^r+2^s+2^t+\ldots+2^u;$$
$$H(\{a_r\})=(\mathbf{1}0\ldots0\,00\ldots000\ldots0\,00\ldots0)_2;$$
$$H(\{a_s\})=(00\ldots0\,\mathbf{1}0\ldots0\,00\ldots0\,00\ldots0)_2;$$
$$H(\{a_t\})=(00\ldots0\,00\ldots0\,\mathbf{1}0\ldots0\,00\ldots0)_2;$$
$$\ldots$$
$$H(\{a_u\})=(00\ldots0\,00\ldots0\,00\ldots0\,\mathbf{1}0\ldots0)_2;$$

$$H(Q)=(\mathbf{1}0\ldots0\,\mathbf{1}0\ldots0\,\mathbf{1}0\ldots0\,\mathbf{1}0\ldots0)_2;$$
$$H(R)=(\boldsymbol{x_1}0\ldots0\boldsymbol{x_2}0\ldots0\boldsymbol{x_3}0\ldots0\boldsymbol{x_m}0\ldots0)_2.$$

Як доведено, $H(R)\&(H(R)-1)$ відповідає тій множині $R$ без її елемента з найменшим номером. А якій множині відповідає $H(Q)\&(H(R)-1)$? Це число відповідає тій підмножині $Q$, яка лексикографічно йде безпосередньо перед множиною $R$!

Розглянемо приклад. $Q=\{a_2, a_3, a_5, a_6\}$, $H(Q)=110110_2=54_{10}$.

| $H(R)$ | $H(R)-1$ | $H(R-1)\,\&\,H(Q)$ |
|---|---|---|
| 000010 | 000001 | 000000 |
| 000100 | 000011 | 000010 |
| 000110 | 000101 | 000100 |
| 010000 | 001111 | 000110 |
| 010010 | 010001 | 010000 |
| 010100 | 010011 | 010010 |
| 010110 | 010101 | 010100 |
| 100000 | 011111 | 010110 |
| 100010 | 100001 | 100000 |
| 100100 | 100011 | 100010 |
| 100110 | 100101 | 100100 |
| 110000 | 101111 | 100110 |
| 110010 | 110001 | 110000 |
| 110100 | 110011 | 110010 |
| 110110 | 110101 | 110100 |

При $m$ — множині, чиї підмножини ми перелічуємо, $k$ — підмножині $m$, код програми мовою Pascal матиме такий вигляд:

```
k, m: LongInt;
...
k:=m;
while true do
begin
  ...
  if k=0 then break;
  k :=(k-1) AND m;
end;
```

або мовою C/C++:

```
for (int k=m; ; k= k - 1) &m) {
  ...
  if (k==0) break;
}
```

Використання такого підходу до перелічування підмножин виявляється достатньо для того, щоб розв'язати задачу за відведений час.

### 6. Портали (автор — Данило Мисак)

**Формулювання задачі в термінах теорії графів.** Задано незв'язний неорієнтований граф та вартість кожної його вершини. Необхідно зробити граф зв'язним, провівши ребра найменшої сумарної вартості за умови, що вартість нового ребра є сумою вартостей вершин, які це ребро сполучає.

Позначимо через $c$ кількість компонент зв'язності графа. Зауважимо таке:

- немає сенсу сполучати ребром вершини з однієї компоненти зв'язності;
- немає сенсу проводити більше ніж одне ребро між вершинами, що належать певним двом різним компонентам зв'язності;
- якщо $a$ — найменша вага вершини однієї компоненти зв'язності, а $b$ — найменша вага вершини іншої компоненти зв'язності, то ребро найменшої ваги, що сполучить вершини цих компонент зв'язності, має вагу $a+b$.

Перетворимо наш граф: замість кожної компоненти зв'язності залишимо вершину найменшої ваги у цій компоненті. Задачу зведено до побудови дерева із найменшою сумою ваг ребер на залишених $c$ вершинах.

Розглянемо довільне таке дерево і зафіксуємо його вершину М найменшої ваги (якщо таких вершин кілька, то довільну з них). Якщо дерево має ребро, що не виходить із цієї вершини, розглянемо вершини А та Б, які воно сполучає. Приберемо ребро. Дерево розпадеться на дві компоненти зв'язності, причому вершини А та Б належатимуть різним компонентам. Сполучимо тоді вершину М з тією з двох вершин А та Б, яка лежить в іншій, ніж М, компоненті зв'язності. Унаслідок цього, згідно з вибором вершини М, вага проведеного між компонентами зв'язності ребра не збільшилась, а з вершини М стало виходити на одне ребро більше. Отже, не збільшуючи сумарну вагу дерева, ми можемо змінити його так, щоб усі ребра дерева виходили з вершини М. Після цього як саме дерево, так, відповідно, і його вага відновлюються однозначно. Вага дерева дорівнює

$$(c-1)\cdot a_m+(a_1+a_2+\ldots+a_{m-1}+a_{m+1}+\ldots+a_c)=$$
$$=(c-2)\cdot a_m+(a_1+\ldots+a_c),$$

де $a_1, a_2, ..., a_c$ — ваги всіх $c$ вершин дерева, $a_m = \min\{a_1, a_2, ..., a_c\}$ — вага вершини M.

Залишилося реалізувати пошук компонент зв'язності початкового графа та найменшої ваги вершини у кожній компоненті. Найпростіше зробити це з допомогою пошуку в глибину, якщо рекурсивна функція повертає найменшу знайдену на даний момент вагу в поточній компоненті зв'язності. Складність алгоритму складає $O(n+m)$.

Для розв'язання задачі можна застосувати й загальний підхід до побудови т. зв. мінімального кістякового дерева, причому як уже після заміни компонент зв'язності окремими вершинами, так і без цього. В останньому випадку треба присвоїти усім уже наявним у графі ребрам нульову вартість.

Найшвидші реалізації традиційних алгоритмів побудови мінімального кістякового дерева (наприклад, Крускала або Прима) також дають повний бал.

**АВТОРСЬКІ РОЗВ'ЯЗАННЯ ЗАВДАНЬ III ЕТАПУ**

**1. Поліклініка**

```
/* GCC */
#include <stdio.h>
#define maxn 100000
int n, t1, t2, a[maxn], b[maxn], i, end,
          minMoment, minTime, ans;
int main()
{
  freopen("clinic.in", "r", stdin);
  freopen("clinic.out", "w", stdout);
  scanf("%d %d %d", &n, &t1, &t2);
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
  for (i = 0; i < n; i++)
    scanf("%d", &b[i]);
  if (a[0] >= t1) // Якщо перший пацієнт
// прийшов у момент початку прийому
// або пізніше, Петрику достатньо прийти
// у момент початку прийому, щоб одразу
// потрапити до лікаря
    ans = t1;
  else
  {
    end = t1; // Час, коли лікар
// закінчить огляд останнього розглянутого
// пацієнта (початкове значення — момент
// початку прийому t1)
    minTime = -1; // Найменший час
// очікування, який Петрик може забезпечити
// на даний момент (початкове значення -1
// умовно позначає, що Петрик ще нічого
// не встиг собі забезпечити)
    for (i = 0; i < n; i++)
// Розглядаємо одне за одним усіх пацієнтів
    {
      if (minTime == -1 || end - a[i] < minTime)
// Якщо, прийшовши водночас
// із i-м пацієнтом, Петрик забезпечить
// менший час очікування, ніж міг
// забезпечити раніше
      {
        minTime = end - a[i];
// Запам'ятовуємо цей час очікування
        minMoment = a[i];
// А також запам'ятовуємо відповідний
// момент, коли має прийти Петрик
      }
      end += b[i];   // Час, коли
// лікар закінчить огляд пацієнта,
// якого зараз розглядаємо
      if (end >= t2) // Якщо після
// цього лікар уже не вестиме прийом,
// виходимо з циклу
        break;
      if (i == n - 1 || a[i + 1] >= end)
// Якщо ми щойно розглянули останнього
// пацієнта з черги або якщо наступний
// пацієнт прийде не раніше, ніж лікар
// закінчить огляд розглянутого, Петрик
// може прийти в момент закінчення огляду
// розглянутого пацієнта — тоді він одразу
// потрапить до лікаря
      {
        minMoment = end;
        break;
      }
    }
    ans = minMoment; // Як відповідь
// слід вивести відповідний момент часу
  }
  printf("%d\n", ans);
  return 0;
}
```

**(Далі буде)**

★ ★ ★

Підписано до друку 14.03.2014 р. Формат 60x84 1/8. Папір офсет. Друк офсет. Умовн. друк. арк. 5,88.
Умовн. фарбо-відб. 11,76. Обл.-вид. арк. 8,54. Видавець: ФО–П Жугастрова О.В. Зам. №14–71.
Віддруковано у друкарні видавництва «Фенікс». Свід. ДК 271 від 7.12.2000 р.
Адреса видавця: вул. Половецька, 12/42, к. 88, м. Київ, 04107, Україна.
E-mail: csf221@rambler.ru, www.csf.vashpartner.com.

**Передплату на наш журнал можна оформити у будь-якому відділенні зв'язку. Наш індекс 74248**