Вказівки щодо розв'язання завдання № 31 відбірково-тренувальних зборів команди міста Києва

1. Фарбування

Найефективніше *в умовах олімпіади* розв’язання ґрунтується на наслідку теореми Редфілда — Пойя. Теоретичний матеріал українською мовою вичерпно подано за такою адресою: http://kievoi.narod.ru/lectures/polya.html. Алгоритм, на відміну від теорії, легкий для сприйняття і програмного втілення:

1. Зчитавши вхідні дані, встановити кількість граней і відношення суміжності.
2. За допомогою оптимізованого перебору встановити групу симетрій S — взаємно однозначних відображень множини граней многогранника у себе, що зберігають відношення суміжності (наявності спільного ребра, — і кількість елементів цієї групи $|S|$.

Примітка. *Кроки 1 і 2 реалізовано у розв'язанні задачі 16.2 відбірково-тренувальних зборів (Многогранник–2, III етап 2005 року), авторське розв'язання є в архіві тестових файлів.*

3. При всіх k в межах від 1 до $|S|$ змінній p_k надати величину 0.
4. Для кожного елемента групи симетрій:
   - знайти j — кількість циклів, на які він розкладається (фрагмент задачі 8.1 відбірково-тренувальних зборів);
   - збільшити на 1 величину p_j .
6. Обчислити $n_{\text{out}} = (p_1 \cdot m + p_2 \cdot m^2 + p_3 \cdot m^3 + \ldots + p_{|S|} \cdot m^{|S|}) : |S|$.
7. Вивести величину n_{out}.

2. Фортеця

Точки, в яких можна розташовувати вежі, будемо називати *відміченими*. Впорядкуємо ці точки за зростанням першої координати, а при однакових перших координатах — за зростанням другої. Згідно з нашим впорядкуванням занумеруємо відмічені точки натуральними числами від 1 до n, де n — кількість точок. Розглянемо довільний невироджений опуклий многокутник — реалізацію розв’язання задачі: всі вершини цього многокутника розташовані у відмічених точках, а його межа містить найбільшу можливу кількість таких точок. Нехай A — вершина многокутника, яка має найменший номер серед усіх його вершин, а C — вершина, яка має найбільший номер.

Введемо деякі означення. Ламану лінію $V_0V_1...V_k$ вважатимемо *опуклою вгору*, якщо при всіх $j = 1, ..., k - 1$ промінь, проведений з вершини V_j вертикально вниз, перетинає відрізок $V_{j-1}V_{j+1}$. Аналогічно, ламану$V_0V_1...V_k$ вважатимемо *опуклою вниз*, якщо при всіх $j = 1, ..., k - 1$ промінь, проведений з вершини V_j вертикально вгору, перетинає відрізок $V_{j-1}V_{j+1}$. У тому випадку, коли ламана є відрізком, вважатимемо її опуклою одночасно вгору і вниз.

Вершини A та C розділяють межу многокутника на дві ламані лінії. За вибором A та C, одна з ламаних опукла вниз, а інша — вгору. Позначимо їх як l_1 та l_2 відповідно. Очевидно, що при русі від A до C вздовж l_1 або l_2 відмічені точки на цих лініях лежать у порядку зростання номерів. Многокутник невироджений, тому можливі лише такі три

випадки:

1) l_1 складається з декількох відрізків, а l_2 є відрізком;

2) l_1 є відрізком, а l_2 складається з декількох відрізків;

3) обидві ламані l_1 та l_2 складаються з декількох відрізків.

У кожному з випадків ламані лінії, що складаються з декількох відрізків, мають містити максимальну кількість відмічених точок серед ліній свого типу. Наприклад, у першому випадку l_1 містить максимальну кількість відмічених точок серед усіх ламаних, які сполучають A та C, складаються з декількох відрізків і опуклі вгору. У другому випадку l_2 містить максимальну кількість таких точок серед усіх ламаних, які сполучають A та C, складаються з декількох відрізків і опуклі вниз. Те ж саме для третього випадку. Якщо одна зі згаданих умов не справджується, відповідну ламану можна замінити на лінію того самого типу з більшою кількістю точок. Тоді новий многокутник також буде невиродженим і опуклим, але міститиме більшу кількість відмічених точок. А це суперечить припущенню про максимальність відмічених точок для нашого многокутника. Отже, для кожної пари A, C достатньо знайти, яку найбільшу кількість відмічених точок може містити ламана, що сполучає A та C, складається з декількох відрізків і опукла вгору. Те ж саме для ламаних, опуклих униз.

Назвемо ламану лінію *правильною*, якщо вона задовольняє такі умови:

- всі її вершини є відміченими точками;
- при русі вздовж ламаної від кінця з меншим номером до кінця з більшим номером номери відмічених точок на цій ламаній зростають;
- ламана опукла вгору;
- ламана не є відрізком.

Трійку відмічених точок (A, B, C) назвемо *впорядкованою*, якщо A має менший номер, ніж B, а B — менший номер, ніж C. Нехай $F(A, B, C)$ — найбільша можлива кількість відмічених точок, розташованих на *правильній* ламаній, на якій A є першою, B — передостанньою, а C — останньою відміченою точкою. Якщо відповідної ламаної не існує, покладемо $F(A, B, C) = 0$.

Перед тим, як шукати $F(A, B, C)$ для всіх упорядкованих трійок, зробимо декілька спостережень. Впорядковану трійку (A, B, C) називатимемо *правою*, якщо ламана ABC опукла вгору. Якщо при цьому B не лежить на відрізку AC, трійку називатимемо *строго правою*. Нехай $S(B, C)$ — кількість відмічених точок на відрізку BC. Згідно з даними означеннями, на правильній ламаній трійка (A, B, C) має бути строго правою. Тому для інших трійок маємо $F(A, B, C) = 0$. Також потрібної ламаної не існує, якщо $S(B, C) > 2$ — у цьому випадку B не може бути передостанньою точкою. З іншого боку, якщо (A, B, C) строго права і $S(B, C) = 2$, то $F(A, B, C) > 0$. Справді, у цьому випадку поставлені умови задовольняє принаймні одна ламана — об'єднання відрізків AB та BC. У зв'язку з цим строго праві трійки (A, B, C), для яких $S(B, C) = 2$, називатимемо *перспективними*.

Перейдемо до підрахунку $F(A, B, C)$. Кожне значення A розглянемо окремо. Для всіх B та C, для яких (A, B, C) — впорядкована трійка, знайдемо значення $F(A, B, C)$ методом динамічного програмування. А саме, за певного значення C нам потрібно буде розрахувати $F(A, B, C)$ для всіх таких B, що (A, B, C) є впорядкованою трійкою. Ми зробимо це на основі значень $F(A, P, Q)$, розрахованих для всіх впорядкованих трійок (A, P, Q), де P та Q мають менші номери, ніж B.

У випадку перспективної трійки (A, B, C) визначимо $F(A, B, C)$ наступним чином.

Нехай $G(A, B, C)$ дорівнює максимуму величини $F(A, D, B)$, взятому по всіх відмічених точках D, для яких (D, B, C) є впорядкованою правою трійкою. Якщо таких точок D немає, покладемо $G(A, B, C) = 0$. Тепер нехай

$$F(A, B, C) = \max\{G(A, B, C); S(A, C)\} + 1.$$

Дійсно, потрібну ламану для трійки (A, B, C) можна отримати, якщо приєднати відрізок BC до відповідної ламаної для (A, D, B) або до відрізка AB. Залишилось обрати варіант, який дасть лінію з найбільшою кількістю відмічених точок.

Як уже зазначалося, якщо (A, B, C) не перспективна, то $F(A, B, C) = 0$.

Нехай у парі (A, C) точка A має менший номер, ніж C. Для кожної такої пари нехай $U(A, C) = \max\{F(A, B, C)$, де (A, B, C) — впорядкована трійка$\}$. А якщо для даних A та C впорядкованих трійок (A, B, C) немає, покладемо $U(A, C) = 0$. Тоді $U(A, C)$ дорівнює максимальній кількості відмічених точок, що лежать на деякій ламаній з кінцями в A та C, яка опукла вгору і не є відрізком.

Точно так само знайдемо $V(A, C)$ — максимальну кількість відмічених точок, що лежать на деякій ламаній з кінцями в A та C, яка опукла вниз і не є відрізком. Цю задачу можна звести до попередньої, якщо замінити всі координати точок на протилежні за знаком.

Для кожної пари (A, C), де A має менший номер, ніж C, обчислимо

$$M(A, C) = \max\{U(A, C) + S(A, C), S(A, C) + V(A, C), U(A, C) + V(A, C)\} - 2.$$

Величина $M(A, C)$ дорівнює найбільшій кількості відмічених точок, що знаходяться на межі невиродженого опуклого многокутника, в якому вершиною з найменшим номером є A, а з найбільшим — C. Взявши максимум величини $M(A, C)$ по всіх парах (A, C), отримаємо відповідь на питання задачі.

Вкажемо деякі прийоми, що роблять розв'язок зручнішим та швидшим.

Перевірити, чи є впорядкована трійка (A, B, C) правою, можна за допомогою псевдоскалярного добутку. *Псевдоскалярним добутком* векторів (x_1, y_1) та (x_2, y_2) називають величину $x_1y_2 - y_1x_2$. Нехай точки A, B, C мають координати (x_a, y_a), (x_b, y_b), (x_c, y_c) відповідно. Відомо, що трійка (A, B, C) є правою тоді і тільки тоді, коли псевдоскалярний добуток векторів AB та AC — недодатний. Тобто при

$$(x_b - x_a)(y_c - y_a) - (y_b - y_a)(x_c - x_a) \le 0.$$

Якщо $(x_b - x_a)(y_c - y_a) - (y_b - y_a)(x_c - x_a) < 0$, трійка є строго правою. Цей спосіб перевірки не потребує ділення, а тому не вимагає розглядання випадку, коли дільник є нулем.

При розрахунку величин $S(A, B)$ та $F(A, B, C)$ можна не враховувати найправішу відмічену точку на відповідному відрізку чи ламаній. Тоді згадані величини будуть на 1 меншими. У цьому випадку не потрібно віднімати 2 при розрахунку $M(A, C)$, що спрощує формулу.

Для прискорення обчислення $G(A, B, C)$ застосуємо наступну процедуру. Для кожної відміченої точки B розглянемо всі точки, що мають менший номер, ніж B. З них залишимо лише ті точки D, для яких на відрізку DB немає інших відмічених точок. Позначимо залишені точки як $D_1, ..., D_h$. Нехай з точки B проведено промінь, який спочатку направлений вертикально вниз, а потім обертається за годинниковою стрілкою. Точки $D_1, ..., D_h$ відсортуємо в порядку попадання на цей промінь. Нехай

$$F_{\max}(A, D_i, B) = \max\{F(A, D_1, B); ...; F(A, D_i, B)\}.$$

Якщо деяка точка D_i має не більший номер, ніж A, величина $F(A, D_i, B)$ для неї не визначена. Тому при розгляданні певної A вказані точки D_i можна виключити з відсортованого списку. Для всіх D_i значення $F_{\max}(A, D_i, B)$ можна отримати за лінійний від h час, якщо використати співвідношення $F_{\max}(A, D_1, B) = F(A, D_1, B)$ та $F_{\max}(A, D_i, B) = \max\{F_{\max}(A, D_{i-1}, B); F(A, D_i, B)\}$ для $i > 1$. Тепер достатньо знайти найбільше i, для якого (D_i, B, C) — права трійка. Це можна зробити за допомогою двійкового пошуку. Маємо $G(A, B, C) = F_{\max}(A, D_i, B)$.

Оцінимо кількість операцій, які здійснює описаний алгоритм. Кількість впорядкованих трійок, для яких ми розраховуємо $F(A, B, C)$, має порядок $O(n^3)$. Для кожної трійки (A, B, C) потрібно порівняти порядку n значень $F(A, D, B)$, тобто виконати $O(n)$ операцій. Якщо застосувати сортування та двійковий пошук, ця кількість зменшиться до $O(\log_2 n)$. Отже, для всіх трійок треба $O(n^3 \log_2 n)$ операцій. Інші розрахунки потребують меншої кількості дій. Тому загальна кількість виконаних операцій становить $O(n^3 \log_2 n)$.

3. Вкладені множини

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:

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

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:

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

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 $S_0 = \{a_1, a_2, \ldots, a_5\}$. 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:

$$f(1, \{a_1, a_2, a_3\}) = 1, \quad f(1, Q) = 0 \text{ при } Q \neq \{a_1, a_2, a_3\},$$
$$f(2, \{a_1, a_2\}) = 1, \quad f(2, Q) = 0 \text{ при } Q \neq \{a_1, a_2\},$$
$$f(3, \{a_1, a_2\}) = 1, \quad f(3, Q) = 0 \text{ при } Q \neq \{a_1, a_2\}.$$

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

$$f(1, Q) = 1 \text{ при } H(Q) \bmod 41 = h_1, \qquad (2)$$
$$f(1, Q) = 0 \text{ при } H(Q) \bmod 41 \neq h_1. \qquad (3)$$

Покажемо, як обчислити $f(n, Q)$ при $n > 1$, якщо відомі $f(n - 1, R)$ при всіх $R \supseteq S_0$. 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`][`$H(Q)$ `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++):

$$H(A \cup B) = H(A) \mid H(B);$$

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

де:

| — операція побітового «або» (мовою Free Pascal пишуть `or`);

& — операція побітового «і» (мовою Free Pascal 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:

$$X = (x_k x_{k-1} \ldots x_{m+1} 10 \ldots 0)_2$$

$$X - 1 = (x_k x_{k-1} \ldots x_{m+1} 01 \ldots 1)_2$$

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

$$X \,\&\, (X - 1) = (x_k x_{k-1} \ldots x_{m+1} 00 \ldots 0)_2,$$

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:

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

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

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 виглядатиме так:

```
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 в відведений час.