Вказівки щодо розв'язання завдання № 24 відбірково-тренувальних зборів команди міста Києва

1. Шахи

Рівень 1. Можна здійснити повний перебір усіх варіантів розміщень коней з подальшою перевіркою коректності розташування. Для дошки $4\times N$ будемо мати 2^{4N} таких розміщень. Час на перевірку коректності розташування пропорційний N. Тому таке розв'язання виконує кількість операцій, що пропорційна $N\cdot 2^N$. Таким чином можна вирахувати значення кількостей для N від 2 до 7 за час туру олімпіади, задати ці кількості сталими у коді програми і для вказаних N виводити остачу від ділення на P попередньо обчислені кількості розташувань. Таке розв'язання набирає 15% балів.

Рівень 2. Знаючи кількість розташувань на дошці $4\times(N-1)$ для кожних можливих двох останніх стовпчиків, ми можемо досить швидко визначити такі кількості вже для дошки $4\times N$. Існує 2^8 варіантів розміщення коней у останніх двох стовпчиках, бо в кожній з 8 клітинок кінь може або бути або не бути. Зобразимо два останні стовпчики дошки, позначивши однаковими літерами поля літерами латиниці A, B, C, D. Для коней, розташованих на останніх стовпчиках дошки, бити один одного можуть лише пари коней, що знаходяться у клітинках, позначених у таблиці ліворуч однією літерою.При допустимих розташуваннях для

A	B
C	D
B	A
D	C

для кожної з цих пар маємо 3 варіанти: або немає коней на обох клітинках, або кінь є на першій (згори), або кінь є на другій. Тому загальна кількість позицій — допустимих розташувань коней на двох останніх стовпчиках дошки — дорівнює $3^4 = 81$. Для кожного такої позиції визначимо, які позиції можна отримати при коректному долученні ще одного стовпчика. Тоді для підрахунку кількості розташувань на дошці $4\times N$ для певної позиції k додаємо кількості розташувань на дошці $4\times(N-1)$ для тих позицій, з яких можна перейти у k долученням стовпчика і знаходимо остачу від ділення цієї суми на P. Кількість операцій у такому алгоритмі буде пропорційна $3^8\cdot N$, що має прийнятний час при $N \le 10000$. Таке розв'язання набирає 70% балів.

Рівень 3. Позначимо через A матрицю розміром 81×81, у якій елемент у рядку k і стовпчику j дорівнює 1, якщо з позиції j можна перейти до позиції k, та дорівнює 0, якщо такий перехід неможливий.

Позначимо через B_n матрицю розміром 81×1 (вектор-стовпчик, що має 81 координату), де значення у комірці $(k\,;1)$ дорівнює кількості розміщень у таблиці $4\times N$ з кінцевою позицією k. Усі координати B_2 дорівнюють 1.

Маємо: $A\cdot B_n = B_{n+1}$ і $A^{n-2}\cdot B_2 = B_n$ при довільному натуральному n.

Кількість допустимих розташувань — це сума всіх елементів B_N.

Найскладнішу частину алгоритму — підрахунок A^{N-2} — можна здійснити за час, що пропорційний $81^3\cdot \log_2 N$. Для цього спочатку обчислимо матриці A, A^2, A^4, ... , A^{2^s}, для всіх S, при яких 2^S не перевищує $N-2$. Кожну з цих матриць можна вираховувати як квадрат попередньої, отже кількість операцій для вираховування кожної з них пропорційна 81^3. Далі множимо між собою ті степені матриці A,

сума степенів яких дорівнює $N-2$. Ці степені відповідають відмінним від нуля розрядам двійкового розкладу $N-2$. Таке розв'язання набирає 100% балів.

2. Школа

Ототожнимо вершини кварталів, що належать місту, з вершинами деякого графа. Нехай дві вершини цього графа суміжні тоді й лише тоді, коли відповідні їм точки — суміжні вершини якогось кварталу або вершини сусідніх кварталів такі, що між ними є перехід. Крок за кроком будуватимемо щораз більшу підмножину вершин графа, для яких знатимемо найкоротший час, за який у ці вершини можна дістатися з початкової.

На першому кроці до підмножини включимо єдину точку — початкову вершину (дім Петрика). Для неї цей час відомий і дорівнює нулеві.

На кожному наступному кроці серед суміжних з уже долученими до підмножини вершин, але таких, що їх самих ще не долучено до підмножини, шукаємо вершину, до якої, якщо до неї йти *лише через уже долучені вершини,* дійти можна за найменший час. Цю вершину ми долучаємо до підмножини, а відповідний час — і є найменший час, за який до цієї вершини можна дійти з початкової.

Доведемо останнє висловлювання щодо мінімальності часу методом від супротивного. Якщо до вершини, що долучають, можна дійти за менший від знайденого час, то відповідний шлях десь виходить за межі підмножини (бо дім Петрика належить підмножині, а всі шляхи, що цілком належать підмножині, мають, з огляду на алгоритм, більший від обраного час). Для першої вершини шляху, що перебуває за межами підмножини, час буде навіть меншим, ніж для вершини, що ми розглядаємо, а це суперечить критерію вибору вершин.

Поки є вершини поза побудованою підмножиною, ми крок за кроком долучаємо до неї нову вершину. Всіх вершин — скінченна кількість. На певному кроці до підмножини буде долучено й вершину, що є пунктом призначення, та знайдено для неї відповідний час.

Щоб мати змогу швидко знаходити вершину, для якої час на шлях через уже долучені до підмножини точки найменший, на кожному кроці для кожної ще не долученої вершини зберігатимемо цей час. *У початковий момент цей час для всіх вершин, за виключенням пункту-старту, дорівнює* $+\infty$. Після кроку — по долученні нової вершини — для підтримання коректності цієї інформації досить оновити мінімальний час лише для суміжних з долученою вершин, де він насправді міг змінитись у зв'язку з долученням цієї вершини.

Нехай V — кількість вершин графа. Якщо мінімуми зберігати у звичайному масиві, на кожному кроці доведеться серед мінімумів шукати найменший за час $O(V)$. Якщо ж додатково використовувати *бінарну купу* (див. далі), на пошук найменшого та оновлення інформації на кожному кроці досить витратити час $O(\log V)$. Враховуючи, що кількість вершин графа дорівнює $4mn$, а кількість кроків може досягати кількості вершин, час роботи алгоритму можна оцінити як $O(mn \cdot \log(mn))$.

У розв'язанні задачі використано алгоритм Дейкстри.

Означення бінарної купи

Неспадною бінарною купою називають структуру даних — масив H з певною кількістю n елементів, які позначатимемо через H[1], H[2], … H[n], що впорядковано специфічним чином. Для довільного натурального k при $1 \le k \le n/2$ справджуються нерівності:

- $H[k] \le H[2k]$,
- $H[k] \le H[2k+1]$ при $2k+1 \le n$.

Такий масив зручно ототожнити з деревом, у якому нащадками вершини k є вузли $2k$ та $2k+1$ (якщо такі для даного k існують), а коренем — вершина, що відповідає елементу H[1]. У корені купи зберігається найменше з-поміж усіх значень у масиві (це випливає з означення купи), тому пошук найменшого елемента в цій структурі потребує часу $O(1)$. Покажемо, як за час $O(\log_2 n)$ оновити один з елементів купи, щоб зберегти її структурні властивості, а також, як видалити з купи елемент.

Коли один з елементів купи змінюється — в описаному алгоритмі він може лише зменшитись — єдиний «зв'язок», що може порушитись у купі — це нерівність між зміненим елементом та його батьком (батько може виявитись більшим від свого зміненого нащадка). Якщо ця нерівність справді порушується, розміняємо місцями змінений елемент та його батька. При цьому порушена нерівність відновлюється, і жодна інша не порушується, за винятком, можливо, нерівності між зміненим елементом та його новим, на рівень вищим, батьком. Якщо це так, знов переставимо їх. Продовжуватимемо виконувати цю операцію, поки батько зміненого елемента буде більшим від нього, або ж поки змінений елемент не стане коренем. На якомусь кроці це неодмінно трапиться, і тоді властивості купи буде відновлено. Номери щораз нових батьків зміненого елемента ми знаходимо шляхом ділення на 2 номера попереднього батька, кількість розмінів матиме порядок $O(\log_2 n)$.

Якщо змінений елемент збільшився, нерівність із батьківським елементом не порушується, зате можуть порушитись одна чи обидві нерівності, що зв'язують змінений елемент із його нащадками. В такому разі розміняємо місцями змінений елемент з меншим із двох його нащадків. В оновленій таким чином купі єдиними порушеними «зв'язками» знов-таки можуть виявитись лише нерівності між зміненим елементом та його (новими) нащадками. Якщо це так, знову переставимо його з меншим із нащадків. Продовжуватимемо виконувати цю операцію, поки властивості купи не буде відновлено. Таке рано чи пізно трапиться, бо кількість елементів купи скінченна, а номер місця, на яке переставляємо змінений елемент, щораз збільшується (з k номер перетворюється на $2k$ чи на $2k+1$). Кількість розмінів має, як і в попередньому випадку, порядок $O(\log_2 n)$.

Видаляють елемент k з купи таким чином. Переставимо його з останнім, n-м елементом. Уважатимемо, що розмір купи відтепер складає $n-1$, а також, що елемент k купи (який раніше був n-м) просто змінив своє значення — та застосуємо до нього описану вище процедуру. Так буде відновлено властивості купи. Вважають, що змінна, яка вказує на поточну кількість елементів купи, також входить до цієї структури даних.

У нашому розв'язку необхідно крім самої купи зберігати для кожної вершини номер елемента купи, де зберігається значення, що відповідає даній вершині, а також для кожного елемента купи зберігати координати (номер) вершини, що відповідає даному елементу. Цю інформацію, звісно, треба оновлювати при зміні купи. Масив же, що координатам (номеру) вершини ставить у відповідність тривалість відповідного проміжку часу, міститиме надлишкову інформацію, тому не потрібен. Вихідний вміст купи — 4*mn* значень «+∞» (на практиці це зазвичай просто достатньо великі числа), і насамперед оновлюють значення, що відповідає вершині, де розташовано дім Петрика.

Зауваження щодо лінійного «розв'язання»

Припущення, що завжди існує оптимальний шлях до школи, йдучи вздовж якого Петрик у жоден момент часу не прямуватиме на захід (ліворуч) або на південь (униз), хибне. Якби це припущення справджувалося (воно справджується для певних інших значень часових проміжків та обмежень), задачу можна було б розв'язати простіше.

Знову розглянемо місто як граф. Упорядкуємо вершини графа за довжиною найкоротшого шляху до них від помешкання Петрика. Вершиною нульового рівня назвемо сам Петриків дім, вершинами першого рівня — сусідні з домом верхню та праву вершини, другого рівня — три вершини, до яких можна потрапити з дому за два переміщення (вздовж кварталу або через вулицю) і т. д. Легко зрозуміти, що до вершини будь-якого ненульового рівня можна дістатися лише з двох вершин попереднього рівня, якщо дана вершина не прилягає до межі міста, й лише з однієї, якщо прилягає.

Знайдемо для кожної вершини графа найменший час, за який Петрик може дістатися до неї, за умови, що прямуватиме ввесь час на схід або на північ. Для вершини, в якій розташовано дім Петрика, цей час дорівнює нулю. Розглянемо тепер вершини від найближчих до найвіддаленіших — спочатку знайдемо шуканий час для всіх вершин першого рівня, далі для всіх вершин другого рівня, опісля — третього і т. д. Якщо Петрик може прямувати лише праворуч та вгору, то до вершини даного рівня він може потрапити лише з суміжних вершин попереднього. Для обох сусідніх (до даної) вершин попереднього рівня оптимальний час уже було пораховано, тому для даної його можна знайти, порівнявши час для обох варіантів шляху («знизу» та «зліва»). Якщо ж дана вершина прилягає до межі міста, існує лише одна суміжна до неї вершина попереднього рівня, тому оптимальний час дорівнюватиме часу, що витрачатиметься на шлях до тієї вершини та перехід із неї до даної.

На останньому кроці буде знайдено оптимальний час для єдиної вершини останнього рівня — школи Петрика. Враховуючи справедливість припущення про існування оптимального шляху до школи, йдучи вздовж якого Петрик завжди прямуватиме на схід чи на північ, саме це значення й буде відповіддю.

Час, необхідний для виконання поданого алгоритму, має лінійний щодо кількості кварталів порядок — $O(mn)$. Проте, цей алгоритм не є правильним. Він дає неправильну відповідь, наприклад, для таких вхідних даних:

```
2 2
1 300
300 1
```