ЗАВДАННЯ XXIII МІЖНАРОДНОЇ ОЛІМПІАДИ З ІНФОРМАТИКИ ТА РЕКОМЕНДАЦІЇ ЩОДО ЇХ РОЗВ'ЯЗАННЯ

Гуржій А.М., Бондаренко В.В.

(Продовження, початок у №6 за 2011 рік)

ЗАВДАННЯ ДРУГОГО ТУРУ

1. ПІДЗЕМНЕ МІСТО КРОКОДИЛІВ

Археолог Бенджама рятується втечею після спроби дослідити таємне підземне місто крокодилів. Місто складається з N кімнат і M двохсторонніх коридорів, кожен з яких з'єднує пару різних кімнат. Кожна пара кімнат з'єднана не більше, ніж одним коридором. Для кожного з коридорів відомо час, який потрібно щоб по ньому пробігти. Тільки K з N кімнат мають виходи з міста. Бенджама починає з кімнати з номером 0. Вона хоче дістатись виходу якомога швидше.

Страж міста крокодилів хоче, щоб Бенджама не змогла втекти. Зі своєї печери страж контролює секретні двері, які можуть заблокувати рівно *один* коридор. Тобто, коли вони блокують деякий коридор, то попередній заблокований коридор розблокується.

Бенджама знаходиться у такий ситуації: кожного разу, коли вона намагається залишити якусь кімнату, страж може обрати один з коридорів, що пов'язані з цією кімнатою, і заблокувати його. Після цього Бенджама вибирає один з незаблокованих коридорів і біжить по ньому у другу кімнату. Як тільки Бенджама вбігає у якийсь коридор, страж не може заблокувати цей коридор, доки Бенджама не вибіжить з нього на другому кінці цього коридору. Тільки після того, як вона вбіжить у чергову кімнату, страж знову може обрати якийсь зв'язаний з новою кімнатою коридор (у тому числі і той, по якому Бенджама тільки що пробігла) і заблокувати його, і т. д.

Бенджама хотіла б заздалегідь придумати який не будь нескладний план втечі. А саме, їй потрібен набір інструкцій, котрий буде повідомляти їй, що робити, коли вона потрапляє у якусь кімнату. Позначимо цю кімнату як A. Якщо з кімнати A є вихід з міста, то очевидно, що ніяких інструкцій не потрібно — Бенджама може втекти. Інакше, інструкція для кімнати A повинна мати одну з таких форм:

- «Якщо ти коли-небудь потрапиш у кімнату A, втікай по коридору, що веде в кімнату B. Однак, якщо цей коридор заблоковано, втікай по коридору, що веде в кімнату C».
- «Не хвилюйся з приводу кімнати A за планом ти ніколи до неї не потрапиш».

Зауважимо, що у деяких випадках (наприклад, якщо за вашим планом Бенджама буде бігати по циклу) страж може перешкодити Бенджамі добігти до виходу. План називається *гарним*, якщо Бенджама гарантовано потрапить у вихід за скінчений час незалежно від дій стража. Для даного гарного плану позначимо як T такий мінімальний час, що коли сплине T одиниць часу Бенджама *гарантовано* зможе вийти. У цьому випадку ми говоримо, що *для виконання даного гарного плану потрібно витратити час* T.

Завдання

Написати процедуру travel_plan(N,M,R,L,K,P), якій передають такі параметри:

- N — кількість кімнат. Кімнати нумеруються числами від 0 до (N−1).
- M — кількість коридорів. Коридори нумеруються числами від 0 до (M−1).
- R — двовимірний масив цілих чисел, який описує коридори. Для $0 \le i < M$, коридор з номером i з'єднує дві різні кімнати з номерами $R[i][0]$ та $R[i][1]$. Ніякі два коридори не з'єднують одну й ту саму пару кімнат.
- L — одновимірний масив цілих чисел, у якому міститься час, необхідний для того, щоб пробігти по кожному з коридорів. Для $0 \le i < M$, значення $1 \le L[i] \le 1\,000\,000\,000$ є час, який потребує Бенджама, щоб пробігти по i-му коридору.
- K — кількість кімнат, у яких є вихід з міста. Гарантується, що $1 \le K < N$.
- P — одновимірний масив з K різних цілих чисел, що відповідають кімнатам з виходами. Для $0 \le i < K$ значення $P[i]$ є номером i-ї кімнати з виходом. Кімната з номером 0 ніколи не буває кімнатою з виходом.

Ваша процедура повинна повертати мінімальний можливий час T, для якого існує такий гарний план втечі, що для виконання цього плану потрібно витратити час T.

Гарантується, що з кожною кімнатою, що не містить виходу, зв'язано принаймні два коридори. Також гарантується, що у кожному тесті існує гарний план втечі, для якого $T \le 1\,000\,000\,000$.

Приклади

Приклад 1

Розглянемо приклад, зображений на рис. 1, де N=5, M=4, K=3 і

$$R = \begin{matrix} 0\,1 \\ 0\,2 \\ 3\,2 \\ 2\,4 \end{matrix} \quad L = \begin{matrix} 2 \\ 3 \\ 1 \\ 4 \end{matrix} \quad P = \begin{matrix} 1 \\ 3 \\ 4 \end{matrix}$$

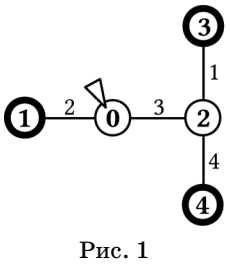

Рис. 1

Кімнати показано кружками, а коридори, що їх з'єднують — відрізками. Кімнати з виходами обведено жирним контуром. Бенджама починає рух у кімнаті з номером 0 (її відмічено трикутником). Оптимальний план втечі такий:

- Якщо ти коли-небудь потрапиш у кімнату з номером 0, втікай по коридору, що веде до кімнати з номером 1. Однак, якщо цей коридор заблоковано, втікай по коридору, що веде до кімнати з номером 2.
- Якщо ти коли-небудь потрапиш у кімнату з номером 2, втікай по коридору, що веде до кімнати з номером 3. Однак, якщо цей коридор заблоковано, втікай по коридору, що веде до кімнати з номером 4.

У найгіршому випадку Бенджама потрапить у кімнату з виходом за 7 одиниць часу. Таким чином, процедура travel_plan має повертати число 7.

Приклад 2

Розглянемо приклад, зображений на рис. 2, де N=5, M=7, K=2 і

$$\boldsymbol{R}=\begin{matrix}0\ 2\\0\ 3\\3\ 2\\2\ 1\\0\ 1\\0\ 4\\3\ 4\end{matrix}\quad \boldsymbol{L}=\begin{matrix}4\\3\\2\\10\\100\\7\\9\end{matrix}\quad \boldsymbol{P}=\begin{matrix}1\\3\end{matrix}$$

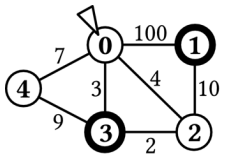

Рис. 2

Оптимальний план втечі описується так:

- Якщо ти коли-небудь потрапиш у кімнату з номером 0, втікай по коридору, що веде до кімнати з номером 3. Однак, якщо цей коридор заблоковано, втікай по коридору, що веде до кімнати з номером 2.
- Якщо ти коли-небудь потрапиш у кімнату з номером 2, втікай по коридору, що веде до кімнати з номером 3. Однак, якщо цей коридор заблоковано, втікай по коридору, що веде до кімнати з номером 1.
- Не хвилюйся з приводу кімнати з номером 4 — за планом ти ніколи не потрапиш до неї.

Бенджама потрапить у якусь з кімнат з виходом непізніше, ніж коли сплине 14 одиниць часу. Тому процедура travel_plan має повертати значення 14.

Підзадачі

Підзадача 1 (46 балів)

- $3 \le N \le 1\,000$.
- Підземне місто представляє собою дерево. Тобто, $M=N-1$ і для кожної пари кімнат з номерами i та j існує послідовність коридорів, що з'єднують кімнати з номерами i та j.
- Кожна кімната з виходом з'єднана рівно з однією іншою кімнатою.
- Решта кімнат з'єднані коридорами з трьома або більше відмінними від неї кімнатами.

Підзадача 2 (43 бали)

- $3 \le N \le 1\,000$.
- $2 \le N \le 100\,000$.

Підзадача 3 (11 балів)

- $3 \le N \le 100\,000$.
- $2 \le N \le 1\,000\,000$.

Рекомендації щодо розв'язання

Підзадача 1: Дерева

У цій задачі ми можемо користатись тим, що відповідний граф є деревом і кімнати з виходами є всіма його листами. Цей випадок є більш простим ніж загальний, але буде корисним для наступних підзадач. Для початку, нехай $L(u, v)$ позначає час подорожі по коридору що з'єднує вузол u з вузлом v. Для простоти, ми розташуємо корінь дерева у початковій кімнаті (кімната 0).

Важливим спостереженням тоді буде те, що у довільному гарному плані втечі, інструкція у кожному вузлі, якщо він досяжний з кореня, буде завжди спрямовувати Бенжданму рухатись далі від кореня (інакше, її можна буде заставити циклічно рухатись підземним містом). Це призводить до простого розв'язку з використанням динамічного програмування. Нехай $T[u]$ позначає найкращий час, необхідний щоб дістатися кімнати з виходом. По-перше, $T[u]=0$, якщо u є листом. Інакше, якщо u має дітей $v_1,\ldots,v_k$, впорядкованих так що $T[v_1]+L(u,v_1) \le T[v_2]+L(u,v_2) \le \ldots$ $T[v_k]+L(u,v_k)$, то $T[u]=T[v_2]+L(u,v_2)$. Умова гарантує, що $k \ge 2$.

Індуктивно просто довести, що $T[u]$ насправді є найкращим часом для u. Спочатку доведемо, що Бенджама може дістатись кімнати з виходом u за час $T[u]$ незалежно від того, що робить страж, потім доведемо, що страж може заставити Бенджаму провести час $T[u]$ у підземному місті. Очевидно, якщо u є листовим вузлом, а, отже, і кімнатою з виходом, то $T[u]$ є 0. Припустивши індуктивно, що $T[v_i]$ вже обчислено потрібним чином для кожного сина v_i вузла u, маємо, що Бенджама може дістатись виходу з u через $(u; v_i)$ за час $T[v_i]+L(u; v_i)$, якщо цей коридор не заблоковано. Оскільки злий крокодил може блокувати тільки один коридор у той самий момент часу, він може змусити Бенджаму витратити час $T[v_2]+L(u; v_2)$, заблокувавши $(u; v_1)$, оскільки блокування іншого коридору допоможе Бенджамі швидше дістатись виходу, за час $T[v_1]+L(u; v_1)$.

Щоб реалізувати цей динамічний підхід, будемо обходити дерево, виконуючи спочатку обробку синів вузла, а потім самого вузла. Остаточна відповідь буде зберігатись у $T[0]$. Обчислення для кожного вузла включає знаходження другого найменшого значення, що можна зробити за час $O(d_u)$. Тут d_u позначає вихідний степінь вузла u. Отже загальний час виконання буде

$$C\sum_u d_u = O(N),$$

для деякої додатної константи C, оскільки сума степенів вузлів дерева дорівнює $2(N-1)$.

Підзадачі 2 та 3: Графи загального виду

Проблема з узагальненням нашого алгоритму на загальний випадок полягає здебільш у відсутності чіткого відчуття напрямку, адже у нашому поточному алгоритмі ми знали, що Бенджама завжди повинна рухатись від кореня, відповідно до структури дерева.

Після деяких роздумів, можна помітити схожість між алгоритмом Дейкстри пошуку найкоротшого шляху і нашим алгоритмом для дерев. Насправді, алгоритм ітеративно збільшує граничний набір вершин,

і в кожен момент часу вузол u буде у цьому наборі, якщо для нього вже визначено $T[u]$. З цієї точки зору наш алгоритм може розглядатись як виконання алгоритму Дейкстри, починаючи з кімнат з виходом. Алгоритм є стандартним за винятком того, як визначається вартість для кожного вузла.

Розглянемо наступний алгоритм: для всіх вузлів u встановимо $T[u]$ в ∞ за винятком випадків, коли u є кімнатою з виходом, тоді нехай $T[u]=0$. На початку граничний набір S буде містити тільки кімнати з виходом. Під час виконання алгоритму будемо підтримувати умову, що для $w \notin S$, вартість w може бути обрахована формуванням списку $\{v \in N(w) : T[v]+L(w,v)\}$, де — множина сусідів w, сортуванням списку, та поверненням другого значення (тобто, другого найменшого значення у цьому списку). Коли вузол u потрапляє до границі (він має найменшу вартість серед не граничних вузлів, $T[u]$ набуває значення вартості u на цей момент.

Твердження 1. Для кожного вузла u Бенджама може дістатися виходу, почавши з вузла u за час $T[u]$ незалежно від того, що робить страж. Більш того, страж може змусити Бенджаму провести у підземному місті час $T[u]$.

Це твердження можна довести індуктивним методом, аналогічним до випадку з деревом, і помітивши, що як і в алгоритмі Дейкстри, як тільки вузол потрапляє до граничного набору, його вартість не може бути зменшена, оскільки ребра мають додатню вартість.

Деталі реалізації. Для кожного вузла, ми можемо підтримувати його поточну вартість, зберігаючи два числа — найменше значення і друге найменше, що можна змінювати за час $O(1)$, якщо змінюються сусідні значення. У такому випадку алгоритм Дейкстри потребує час $O((M+N)\log N)$ з використанням кучі або $O(N^2)$ без використання.

2. ТАНЦЮЮЧІ СЛОНИ

Танцюючі слони — це видовищне шоу в Паттаї, у якому приймають участь N слонів, що танцюють на одній лінії, що зветься сценою.

У результаті багатолітніх тренувань слони, що беруть участь у шоу, вивчили велику кількість танцювальних рухів. Усе шоу складається з послідовності актів. У кожному акті тільки один слон виконує один танцювальний рух, у результаті якого він може пересунутись на іншу позицію на сцені.

Постановники шоу хочуть зробити фотоальбом, який би містив фотографії всього шоу. Після кожного акту вони хочуть зробити фотографії усіх слонів.

Удовільний момент часу протягом шоу деяка кількість слонів може знаходитись в одній і тій самій позиції — це означає, що слони просто стоять поруч.

Одна фотокамера може фотографувати групу слонів тоді і тільки тоді, коли всі позиції, у яких знаходяться слони, лежать на відрізку довжини L (обидві границі відрізка включаються в нього). Так як слони можуть розташовуватись вздовж усієї сцени, то може знадобитись декілька фотокамер, щоб сфотографувати всіх слонів одночасно.

Наприклад, припустимо, що L=10 і слони розташовані на сцені в позиціях 10, 15, 17, і 20 відповідно. У цей момент достатньо однієї фотокамери, щоб сфотографувати всіх слонів, як це показано нижче (рис. 3). Слонів зображено як трикутники; фотокамери зображено як трапеції.

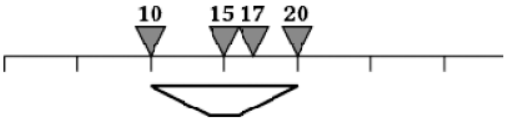

Рис. 3

У наступному акті слон, що знадиться в позиції 15, у результаті танцювального руху пересувається в позицію 32. Після цього акту необхідно вже не менш двох фотокамер для того, щоб сфотографувати всіх слонів одночасно (рис. 4).

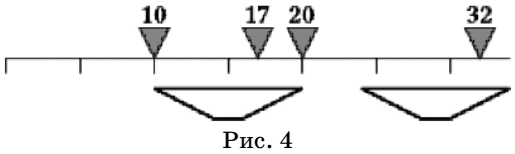

Рис. 4

У наступному акті слон, що знаходиться в позиції 10, пересувається в позицію 7. У даному випадку знадобиться 3 фотокамери для того, щоб сфотографувати всіх слонів (рис. 5).

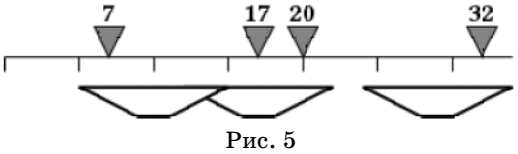

Рис. 5

У цій задачі, яка є інтерактивною, ви повинні визначити *мінімальну* кількість фотокамер, що необхідно для того, щоб зробити фотографії після кожного акту шоу. Слід зауважити, що кількість необхідних фотокамер може збільшуватись, зменшуватись, або залишатись тією самою від акту до акту.

Завдання

Написати такі процедури.

1. Процедуру init(N,L,X), якій передають такі параметри:

- N — кількість слонів. Слони нумеруються від 0 до (N–1).
- L — довжина відрізка, що можна сфотографувати однією фотокамерою. Гарантується, що ціле число L задовольняє обмеженню: $0 \le L \le 1\,000\,000\,000$.
- X — одновимірний масив цілих чисел, що задають початкові позиції слонів. Для кожного i ($0 \le i < N$) слон з номером i знаходиться в позиції $X[i]$. Початкові позиції є впорядкованими. Точніше, гарантується, що $0 \le X[0] \le \ldots \le X[N-1] \le 1\,000\,000\,000$. Слід зауважити, що протягом танцю при зміні позиції слонів може змінюватись і порядок їх слідування.

Цю процедуру буде викликано рівно один раз (до всіх викликів процедури update). Ця процедура не повертає ніякого значення.

2. Процедуру update(i,y), якій передають такі параметри:

- i — номер слона, який буде пересуватись у цьому акті.
- y — позиція, у якій слон з номером i буде стояти після цього акту. Гарантується, що ціле число y задовольняє обмеженню: $0 \le y \le 1\,000\,000\,000$.

Цю процедуру буде викликано багато разів. Кожен виклик відповідає одному акту, який знаходиться в послідовності актів після всіх попередніх актів, для яких раніше було викликано цю процедуру. Кожен виклик має повернути *мінімальну кількість фотокамер*, що необхідні для фотографування після відповідного акту.

Приклад

Розглянемо приклад, де $N=4$, $L=10$ і початкові позиції слонів такі:

$$X = \begin{matrix} 10 \\ 15 \\ 17 \\ 20 \end{matrix}$$

Спочатку буде викликано процедуру init з вищеописаними параметрами. Після цього буде викликатись процедура update, один раз для кожного акту. Нижче наведено приклад послідовності викликів і значень, що треба повертати:

Акт	Параметри виклику	Значення, що треба повернути
1	update(2,16)	1
2	update(1,25)	2
3	update(3,35)	2
4	update(0,38)	2
5	update(2,0)	3

Підзадачі

Підзадача 1 (10 балів)

- Є рівно два слони ($N=2$).
- На початку і після кожного акту всі позиції слонів відрізняються.
- Процедуру update буде викликано не більше 100 разів.

Підзадача 2 (16 балів)

- $1 \le N \le 100$.
- На початку і після кожного акту всі позиції слонів відрізняються.
- Процедуру update буде викликано не більше 100 разів.

Підзадача 3 (24 балів)

- $1 \le N \le 50\,000$.
- На початку і після кожного акту всі позиції слонів відрізняються.
- Процедуру update буде викликано не більше 50 000 разів.

Підзадача 4 (47 балів)

- $1 \le N \le 70\,000$.
- Слони можуть займати одну й ту саму позицію.
- Процедуру update буде викликано не більше 70 000 разів.

Підзадача 5 (3 бала)

- $1 \le N \le 150\,000$.
- Слони можуть займати одну й ту саму позицію.
- Процедуру update буде викликано не більше 150 000 разів.

Рекомендації щодо розв'язання

Підзадача 1

У підзадачі 1 може бути тільки 2 слони. Отже, можна просто визначити кількість потрібних камер за константний час. А саме, нам знадобиться тільки одна камера, якщо слони знаходяться на відстані не більше L один від одного, та дві в протилежному випадку.

Підзадачі 2 і 3

Якщо слони знаходяться в позиціях $x_0, x_1, \ldots, x_{N-1}$, таких що $x_i \le x_{i+1}$ для всіх $0 \le i < N-1$, ми можемо обчислити мінімальну кількість необхідних камер, використовуючи жадібний алгоритм. Почнемо з пустого набору камер. Поки поточний набір камер не покриває всіх слонів, будемо обирати слона, який ще не покривається і має мінімальну позицію x_j, і додавати камеру щоб сфотографувати слонів у позиціях з відрізка $[x_j, x_j+L]$. Цю процедуру можна реалізувати, щоб вона виконувалась за час $O(N)$, один раз проходячи по списку відсортованих позицій слонів.

Кожного разу, коли слон рухається, ми можемо внести зміни до сортованого списку позицій за час $O(N)$. Отже, маємо розв'язок задачі, що потребує часу $O(NM)$, і якого достатньо для повного розв'язання підзадачі 2, а після оптимізації і підзадачі 3. Однак для розв'язання підзадач 4 і 5 потрібен швидший алгоритм.

Підзадачі 4 і 5

Групування слонів

Щоб знайти кількість камер, замість проходу по всіх слонах, побудуємо структуру даних, що дозволить нам «перестрибувати» через багатьох з них.

Будемо підтримувати k груп слонів $B_0, B_1, \ldots, B_{k-1}$ так, щоб групи з меншими індексами містили слонів з меншими позиціями, тобто для довільного $0 \le b < k-1$, для всіх $x_i \in B_b$ та $x_j \in B_{b+1}$, $x_i \le x_j$. Також, слони в кожній групі відсортовані по їхніх позиціях.

Метою є впевнитись, що для знаходження потрібної кількості камер, достатньо відвідати кожну групу тільки один раз. Для простоти ми завжди будемо розташовувати камери так, що сама ліва позиція, що фотографується камерою, є позицією якогось слона.

Розглянемо групу b, що містить p слонів. Позначимо список індексів слонів з B_b як $e_0, e_1, \ldots, e_{p-1}$ (при цьому, $x_{e_i} \le x_{e_{i+1}}$). Для заданого слона e_i ми хотіли би швидко отримувати відповіді на такі два запитання:

- Q1: Якщо ми захочемо сфотографувати всіх слонів починаючи з e_i (тобто слонів з множини $\{e_0, e_1, \ldots, e_{p-1}\}$), скільки камер нам буде потрібно?
- Q2: Яку найбільшу позицію фотографує цей набір камер?

Для слона e, позначимо відповідь на запитання Q1 як $J(e)$, а відповідь на запитання Q2 як $T(e)$.

Якщо в нас є відповіді на ці запитання для кожного слона в кожній групі, ми можемо знайти кількість камер за час $O(k\log N)$ так.

Почнемо розмістивши камеру відносно першого слона в групі B_0 так, щоб позиція слона була найлівішою позицією, що фотографує ця камера. Тепер розглянемо розміщення камери відносно слона e в групі B_i у той самий спосіб. Ми знаємо, що для фотографування всіх слонів у B_i, нам потрібно використати $J(e)$ камер і ці камери покриють позиції до $T(e)$. Знайдемо у наступній групі B_{i+1} першого слона e_0, що не потрапляє до цих камер, шукаючи бінарним пошуком слона з мінімальною позицією, що більше $T(e)$. Після цього розташуємо камеру відповідно відносно слона e_0 з групи B_{i+1}.

Будемо повторювати цей крок поки не дійдемо до останньої групи. Оскільки кожен крок виконується за час $O(\log N)$ (завдяки бінарному пошуку), на все нам знадобиться час $O(k\log N)$.

Ми можемо попередньо обчислити відповіді на запитання Q1 і Q2 для всіх B_i слонів за час $O(|B_i|)$ проходячи по всіх слонах від e_j до e_{p-1} і e_0 зберігаючи вказівник на першого слона з-за меж діапазону $x_{e_j}+L$. Більше деталей можна знайти в останньому розділі.

Важливо зазначити, що ми можемо обробляти кожну групу незалежно від усіх інших груп.

Внесення змін до структур даних

Коли слон e рухається, нам буде необхідно вносити зміни до двох груп: поточної групи B_i та нової групи B_j. Це можна зробити за час, що пропорційний розміру групи. Щоб знайти поточну групу, у якій знаходиться e, ми можемо зберігати вказівник з e, але у будь-якому випадку потрібно час $O(k)$, щоб знайти нову групу. Отже, на внесення змін потрібен час $O(k+|B_i|+|B_j)$.

Зауважте, що час прямо залежить від розміру кожної з груп. На початку ми будемо мати в кожній групі приблизно N/k слонів, однак кількість може зростати, оскільки слони можуть рухатись. Щоб підтримувати розмір кожної з груп обмеженим зверху $O(N/k)$, будемо перебудовувати всю структуру даних після внесення кожних $[N/k]$ змін. Перебудова займе час $O(N)$.

Вибір привильного параметра

Нам потрібно обробити M змін і відповісти на одне запитання після кожної з цих змін. Загальний час виконання буде

$$O(M\cdot k\log N)+O(M\cdot(k+N/k))+O(M\cdot(N/(N/k))),$$

де перший доданок визначає загальний час на обробку запитів, другий визначає загальний час внесення змін, а останній — загальний час на перебудову.

Вибір $k=\sqrt{N}$ дасть загальний час виконання

$$O(M\sqrt{N}\log N),$$

якого достатньо, щоб отримати всі бали за цю задачу. Однак, неефективна реалізація може завадити отримати бали за підзадачу 5. Наприклад, використання структури даних set з C++ Standard Template Library може додати множник $\log n$ до часу, необхідного на перебудову структури даних. Цього можна уникнути використовуючи звичайні масиви.

Обробка кожної з груп

Щоб спростити подання, додамо неіснуючого слона e_p в позицію $x_{e_{p-1}}+L+1$. Також нехай $y_j=x_{e_j}$ буде позицією j-го зліва слона з групи B_i.

Розглянемо кожного слона j від самого правого e_{p-1} до самого лівого. Також будемо підтримувати індекс t, що вказує на самого лівого слона e_t, чия позиція $y_t>y_j+L$. На початку $j=p-1$, та $t=p$.

Для кожного слона e_j будемо обчислювати $J(e_j)$ та $last(e_j)$, самого лівого слона, що потрапляє до самої правої камери з набору камер, що фотографують $\{e_j, e_{j+1}, \ldots, e_p\}$.

Для доданого вузла ми встановимо $J(e_p)=0$ і $last(e_p)=e_p$. Для слона e_j ми перевіримо, чи потрібно змінювати t, тобто чи більше позиція e_{t-1} ніж y_j+L. Якщо так, то знайдемо найменше t, таке що $y_t>y_j+L$. Встановимо $J(e_j)=J(e_t)+1$ та $last(e_j)=last(e_t)$.

В кінці, для кожного слона e_j, такого що $last(e_j)$ вказує на доданого слона e_p, змінимо $last(e_j)$ на e_j.

Цей процес можна виконати за один прохід по всіх слонах у групі B_i. Також зауважте, що вказівник t рухається через кожного слона тільки раз. Таким чином, ми довели, що час виконання буде $O(|B_i|)$, як стверджувалось раніше.

Щоб відповісти на запитання Q2, ми для кожного слона e_j будемо повертати $y_{last(e_j)}+L$.

3. ПАПУГИ

Яні любить птахів. З тих пір як вона прочитала про протокол IP overAvianCarriers (IPoAC), вона провела багато часу, дресируючи зграю розумних папуг для передачі повідомлень на далекі відстані.

Яні мріє використати своїх птахів, щоб передати повідомлення M до дуже далекої країни. Її повідомлення M є послідовністю з N (необов'язково різних) цілих чисел, кожне від 0 до 255 включно. У Яні є K спеціально натренованих папуг. Усі папуги мають однаковий вигляд, Яні їх не відрізняє. Кожен птах може запам'ятовувати одне ціле число від 0 до R включно.

Спочатку Яні використовувала просту схему: щоб надіслати повідомлення, Яні ретельно випускала птахів одного за одним. Перед тим, як кожен з птахів летів, Яні вчила птаха черговому числу з послідовності, що утворює повідомлення. На жаль, ця схема не працювала. Виявилось, що всі птахи приліталють у пункт призначення, але вони не обов'язково прилітають у тому самому порядку, у якому вони відлітали. Використовуючи цю схему, Яні могла відновити всі числа, які вона відправляла, але у неї не виходило розташувати їх у правильному порядку.

Щоб здійснити мрію, Яні потрібно знайти кращу схему, і для цього Яні потребує вашої допомоги. Маючи повідомлення M, вона планує випускати птахів одного за одним, як і раніше. Необхідно написати програму, яка буде виконувати дві окремі операції:

- По-перше, програма має читати повідомлення M і перетворювати його в послідовність з не більше ніж K цілих чисел від от 0 до R, котрим Яні зможе навчити птахів.
- По-друге, програма має читати послідовність з цілих чисел від 0 до R, що отримуються, коли птахи досягають пункту призначення, після чого перетворювати їх назад у вихідне повідомлення M.

Гарантується, що всі папуги завжди прилітають у пункт призначення і що кожен з них пам'ятає про вивчене ним число. Яні ще раз нагадує, що папуги можуть прилітати у довільному порядку. Необхідно звернути увагу, що у Яні є тільки K папуг, тобто послідовність цілих чисел від 0 до R, у яку перетворюється повідомлення, має містити не більше ніж K цілих чисел.

Завдання

Написати дві окремі процедури. Одна з них буде використовуватись при відправленні (encoder), а інша при отриманні (decoder).

Весь процес показано на рис. 6.

Напишіть 2 такі процедури:

1. Процедуру decode(N,M), якій передають такі параметри:

- N — довжина повідомлення.
- M — одновимірний масив з N цілих чисел, які задають повідомлення. Гарантується, що $0 \le M[i] \le 255$ для $0 \le i < N$.

Ця процедура має кодувати повідомлення M у послідовність, яка буде передаватися за допомогою папуг і яка складається з цілих чисел від 0 до R включно. Щоб повідомити цю послідовність, процедура encode має викликати процедуру send(a) для кожного цілого числа a, якому ви бажаєте навчити чергового птаха.

2. Процедуру decode(N,L,X), якій передають такі параметри:

- N — довжина вихідного повідомлення.
- L — довжина отриманого повідомлення (кількість відправлених птахів).
- X — одновимірний масив з L цілих чисел, які було отримано. Числа $X[i]$ для $0 \le i < L$ є тими самими числами, які згенерувала процедура encode, але вони, можливо, розташовані в іншому порядку.

Ця процедура має відновлювати вихідне повідомлення. Щоб його повернути, процедура decode має викликати процедуру output(b) для кожного цілого числа b з розкодованого повідомлення у тому порядку, у якому вони утворюють вихідне повідомлення.

Необхідно звернути увагу, що значення R і K не передаються як вхідні параметри (див. опис підзадач нижче).

Щоб правильно розв'язувати певну підзадачу, ваші процедури мають задовольняти такі умови:

- Всі цілі числа, що відправляються процедурою encode за допомогою процедури send, мають бути у діапазоні, що вказано в підзадачі.
- Кількість викликів процедури send, які робить ваша процедура encode, має не перевищувати граничне значення K, що вказано в підзадачі. Необхідно звернути увагу, що значення K залежить від довжини повідомлення.
- Процедура decode має правильно відновлювати вихідне повідомлення M і викликати процедуру output(b) рівно N разів зі значеннями b, що дорівнюють числам $M[0]$, $M[1]$, …, $M[N-1]$ відповідно.

У останній підзадачі бали, що було отримано в результаті оцінювання, залежать від відношення між довжинами закодованого і вихідного повідомлень.

Приклад

Розглянемо приклад, де $N=3$ і:

$$M = \begin{matrix} 10 \\ 30 \\ 20 \end{matrix}$$

Процедура encode(N,M), використовуючи невідомий метод, може закодувати це повідомлення за допомогою такої послідовності чисел: (7, 3, 2, 70, 15, 20, 3). Щоб повідомити цю послідовність, вона має викликати процедуру send у такій послідовності:

send(7)
send(3)
send(2)
send(70)
send(15)
send(20)
send(3).

Припустимо, що після того як всі папуги досягли пункту призначення, було отримано наступний спи-

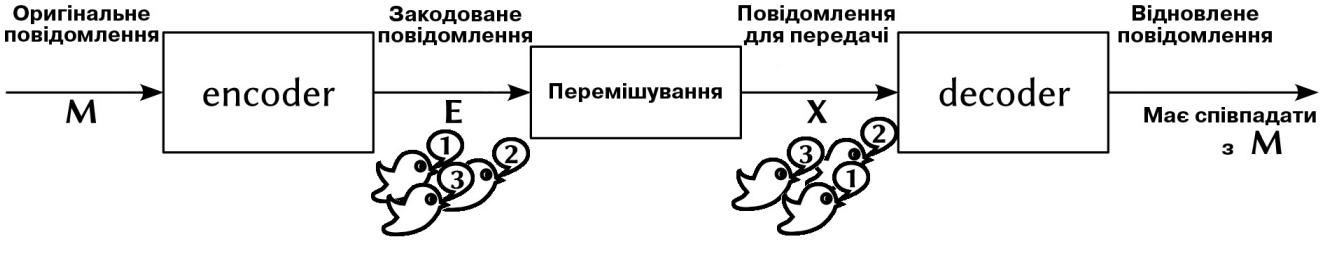

Рис. 6

сок чисел: (3, 20, 70, 15, 2, 3, 7). Процедуру decode буде викликано з N=3, L=7 і

$$X = \begin{matrix} 3 \\ 20 \\ 70 \\ 15 \\ 2 \\ 3 \end{matrix}$$

Процедура decode має відновити вихідне повідомлення, тобто (10, 30, 20). Вона повідомить результат, викликаючи процедуру output у такій послідовності:

output(10)
output(30)
output(20).

Підзадачі

Підзадача 1 (17 балів)

- N=8 і кожне ціле число в масиві M дорівнює 0 або 1.
- Кожне закодоване ціле число має бути в діапазоні від 0 до R=65535 включно.
- Кількість викликів процедури send має бути не більше $K=10\times N$.

Підзадача 2 (17 балів)

- $1\le N\le 16$.
- Кожне закодоване ціле число має бути в діапазоні від 0 до R=65535 включно.
- Кількість викликів процедури send має бути не більше $K=10\times N$.

Підзадача 3 (18 балів)

- $1\le N\le 16$.
- Кожне закодоване ціле число має бути в діапазоні від 0 до R=255 включно.
- Кількість викликів процедури send має бути не більше $K=10\times N$.

Підзадача 4 (29 балів)

- $1\le N\le 32$.
- Кожне закодоване ціле число має бути в діапазоні від 0 до R=255 включно.
- Кількість викликів процедури send має бути не більше $K=10\times N$.

Підзадача 5 (до 19 балів)

- $16\le N\le 64$.
- Кожне закодоване ціле число має бути в діапазоні від 0 до R=255 включно.
- Кількість викликів процедури send має бути не більше $K=15\times N$.
- *Увага*: кількість балів за цю підзадачу залежить від відношення між довжинами закодованого і вихідного повідомлень.

Для кожного тесту з номером t в цій підзадачі нехай величина $P_t=L_t/N_t$ буде дорівнювати відношенню між довжиною закодованої послідовності L_t і довжиною вихідної послідовності N_t. Нехай P буде максимумом серед всіх P_t. Кількість балів, що буде отримано в результаті оцінювання за цю підзадачу, буде визначатися такими правилами:

- Якщо $P\le 5$, розв'язок отримує за цю підзадачу всі 19 балів.
- Якщо $5<P\le 6$, розв'язок отримує за цю підзадачу 18 балів.
- Якщо $6<P\le 7$, розв'язок отримує за цю підзадачу 17 балів.
- Якщо $7<P\le 15$, кількість балів, що розв'язок отримує за цю підзадачу, буде дорівнювати значенню виразу $1+2\times(15-P)$, за округленому вниз до найближчого цілого числа.
- Якщо $P>15$ або хоча б одна з ваших відповідей не є правильною, розв'язок отримує за цю підзадачу 0 балів.

- *Увага*: Довільний правильний розв'язок для підзадач від 1 до 4 розв'язує всі попередні підзадачі. Однак, через більш високе обмеження на K, правильний розв'язок для п'ятої підзадачі може не розв'язувати підзадачі з 1 по 4. Є можливість розв'язати всі підзадачі одним розв'язком.

Рекомендації щодо розв'язання

У цій задачі задано повідомлення M довжини N, яке складається з послідовності цілих чисел між 0 та 255, включно, та для якого потрібно знайти схему кодування і декодування, у якій закодоване повідомлення X є інваріантним до перестановок. Закодоване повідомлення має бути якомога коротшим.

Є багато шляхів розв'язання цієї задачі. Розглянемо, як можна розв'язати перші кілька підзадач.

Підзадача 1

Для цієї підзадачі згадаємо, що оригінальне повідомлення M є послідовністю тільки з двох можливих чисел, 0 або 1. Припустивши, що оригінальне повідомлення проіндексовано від 0 до $N-1$, ми можемо вирішити надсилати номери позицій, де в оригінальному повідомленні стоять одиниці. Ця техніка використовує для відправки повідомлення не більше N цілих чисел.

Підзадача 2

Для підзадачі 2, оскільки числа у вихідному повідомленні M можуть бути від 0 до 255, вони можуть бути закодовані з використанням тільки 8 біт. Однак, ця підзадача дозволяє використовувати у закодованому повідомленні E до 16 бітів на число. Отже, ми зможемо закодувати позицію кожного з чисел, використавши старші 8 бітів, і використати молодші 8 бітів, щоб закодувати актичні числа з оригінального повідомлення M, тобто $E_i=256i+\mathrm{M}_i$, де E_i містить i-те число з оригінального повідомлення.

Зауважте, що закодовані числа є впорядкованими, тобто $E_i<E_{i+1}$, для $0\le i<N-1$. Отже, коли ми захочемо розкодувати повідомлення з перемішаного масиву X, нам потрібно буде його відсортувати для отримання E. Після цього отримати оригінальне повідомлення досить просто.

Цей метод використовує не більше N папуг, що достатньо для розв'язання задачі.

Підзадача 3

Роздуми про перші дві підзадачі могли познайомити нас із підходом. Далі можна рухатись багатьма шляхами. Однак, щоб формалізувати наш підхід, поміркуємо спочатку про загальне подання послідовності.

Оскільки ми будемо тримати цілі числа в закодованому масиві, природнім поданням буде використання відсортованої послідовності. Зауважте, що це подання є дуже стійким до перестановок. Знаючи, що закодований масив E є відсортованим, за допомогою сортування перемішаного масиву X ми можемо просто відновити закодований масив E, як у підзадачі 2.

Спробуємо розширити підхід з підзадачі 2. Однак для задач 3–5 ми не маємо зайвих бітів для зберігання інформації про позицію. У цих випадках будемо розглядати оригінальне повідомлення як послідовність двійкових цифр (або бітів), 0 та 1, подаючи кожне число за допомогою 8 бітів. Отже,оригінальне повідомлення довжини N буде мати $8N$ бітів.

Ця ідея використовується для розв'язання підзадачі 3. Зауважте, що оригінальне повідомлення M містить щонайбільше $16\times8=128$ бітів. Нехай B_i буде i-м бітом оригінального повідомлення, з індексом від 0 до $8N-1$. У цьому випадку, достатньо взяти 7 бітів для подання всіх позицій кожного з бітів, і використовувати останній біт для запису самих даних. Отже, схема кодування буде такою: $E_i=2i+B_i$, яка використовує 7 бітів для подання позиції і ще біт для зберігання даних. Довжина закодованого повідомлення для кожного тесту з цієї підзадачі буде $8N$ байтів.

Підзадача 4

Для цієї підзадачі є $32\times8=256$ позицій бітів, тобто треба 8 бітів тільки для запису позицій. Отже, не існує способу включити в кодування інформацію, що несуть біти з оригінального повідомлення. Якщо ми згадаємо техніку розв'язання підзадачі 1, ми зможемо покращити використану в підзадачі 3 техніку, кодуючи позиції тільки одиниць. Цей підхід відсилає не більше $8N$ байтів.

Підзадача 5

Розглянемо кілька потенційних розв'язків, що можуть набрати певну кількість балів.

• Підхід, що дає відношення 12

Ми збираємося узагальнити розв'язок з попередньої підзадачі, оскільки тепер ми маємо більше ніж 256 бітів, а саме 512.

Щоб розв'язати цю проблему, ми розділимо повідомлення на 256 пар бітів. Визначимо, що i-та бітова пара P_i буде (M_{2i}, M_{2i+1}). Знову, бітова пара P буде мати індекс від 0 до $4N-1$.

Зауважте, що є 4 можливих значення для кожної пари бітів, що можна представити цілими числами від 0 до 3. Одним з методів є надіслати позицію бітової пари (що використає 8 бітів для кодування) як закодовані дані, а дані з безпосередньо бітової пари будуть закодовані як кількість відсилань такої позиції.

Використовуючи цей підхід для кожної пари бітів, нам доведеться надіслати до 3 байтів. Отже, ми надішлемо щонайбільше $(3/2)\times8N=12N$ байтів.

• Підхід, що дає відношення 6.25

Просте спостереження дозволить нам зменшити розмір закодованих повідомлень приблизно вдвічі. Помітимо, що найкращий випадок (у термінах довжини послідовності) для попереднього метода кодування, це коли оригінальне повідомлення містить тільки нульові біти, а найгірший, коли повідомлення містить тільки одиничні біти. Спробуємо знайти середину цих двох випадків.

Розглянемо дві схеми кодування, назвемо їх **ОДИН** і **НУЛЬ**. Для пари бітів (b_{2i}, b_{2i+1}), схема **ОДИН** надсилає $2b_{2i}+b_{2i+1}$ копій i, а схема НУЛЬ надсилає $3-(2b_{2i}+b_{2i+1})$ копій. Зауважте, що обидві схеми працюють, якщо декодувальник знає, яку схему використовував кодувальник при кодуванні.

Для кожної пари бітів сумарна кількість закодованих байтів буде завжди $2b_{2i}+b_{2i+1}+3-(2b_{2i}+b_{2i+1})=3$ байти. Для всіх пар, сума буде $3(8N/2)=12N$. Тепер, якщо ми оберемо схему з меншою кількістю закодованих даних, нам знадобиться щонайбільше половина цієї кількості, тобто не більше $6N$ байтів.

Є багато способів повідомити декодувальнику, яку схему обрав кодувальник, без суттєвого впливу на ефективність. Наприклад, ми можемо додати 4 копії числа 255, якщо використовується схема **НУЛЬ**. Це погіршить відношення на $4/N$. Однак, оскільки $N\geq16$, найгірше відношення буде не більше 6.25.

• Кращі підходи

Є багато інших підходів, що дають краще відношення стискання.

Один байт на 4 числа

Розглянемо скільки існує закодованих повідомлень довжини не більше ніж 7, що складаються тільки з чисел 0, 1, 2 та 3 (0, 0, 1, 1, 1, 1, 3 є одним з таких повідомлень). Щоб це проаналізувати, можливо, простіше розглянути еквівалентну схему: закодовані повідомлення довжини рівно 7, що складаються з чисел 0, 1, 2, 3 та **ПУСТО**. Їх існує

$$\binom{4+7}{4}=\binom{11}{4}=330,$$

чого достатньо для кодування одного байту оригінального повідомлення.

Оскільки оригінальне повідомлення має довжину не більше 64 байт, можна виділити 4 числа на байт (0, 1, 2, та 3 на перший байт наприклад), і таким чином чисел від 0 до 255 буде достатньо. Ця схема дає відношення не більше 7.

Оптимальне відношення

Теоретично, ми можемо закодувати 64 байти даних з використанням мінімум 261 байтів у закодованому повідомленні без втрати інформації. Існує

$$\binom{256+261}{256}=\binom{517}{256}\approx1.47\times10^{154}$$

різних закодованих повідомлень, що використовують не більше 261 байтів. Це трохи більше ніж $256^{64}\approx1.34\times10^{154}$, кількість різних 64-байтних оригінальних повідомлень. Рівень стиснення, що буде досягнуто у цьому випадку, буде приблизно 4.08. Для менших даних відношення буде навіть меншим.

Цей метод дозводить отримати всі бали за задачу, але його реалізація є достатньо важкою.