Вказівки щодо розв'язання завдання № 28 відбірково-тренувальних зборів команди міста Києва

1. Військові навчання

Відстань, про яку йдеться в умові задачі, ще називають *мангеттенською*. Вулиці Мангеттену — району Нью-Йорка — переважно перетинаються під прямим кутом, бо прокладені з півночі на південь або зі сходу на захід. Запроваджена в умові задачі відстань між перехрестями — природна з погляду пішохода чи водія, які не можуть прямувати навпростець через хмарочоси.

Множина точок, що розташовані від точки (x_0, y_0) на відстані, що не перевищує d, є квадратом з вершинами: $(x_0 \pm d, y_0)$ та $(x_0, y_0 \pm d)$. Зазвичай, працювати з фігурами такого вигляду незручно, тому повернемо систему координат на 45° і здійснимо гомотетію (розтягування) з центром у початку координат і коефіцієнтом $2^{½}$. Для цього застосуємо таке лінійне перетворення координат $(x, y) \to (x', y')$:

$$x' = x + y = 2^{½} (x \cos 45° + y \sin 45°);$$
$$y' = – x + y = 2^{½} (– x \sin 45° + y \cos 45°).$$

При цьому:

- пара цілих чисел (x, y) відображається у пару цілих чисел (x', y') *однакової парності*;
- квадрат з вершинами: $(x_0 \pm d, y_0)$ або $(x_0, y_0 \pm d)$ відображається у квадрат з вершинами $(x'_0 \pm d, y'_0 \pm d)$ при $x'_0 = x_0 + y_0$, $y'_0 = – x_0 + y_0$.

Отже, для вирішення задачі нам потрібно реалізувати структуру даних, яка б могла підтримувати такі типи запитів: додати точку з цілими координатами, видалити точку, порахувати кількість точок у заданому квадраті. Існують декілька варіантів такої структури.

Авторське розв'язання. Двовимірне дерево Фенвіка

Деревом Фенвіка (суматором) називають структуру даних, яка дозволяє для лінійного масиву змінювати значення у його комірці та шукати суму на проміжку $1..X$. Обидві операції виконують за час $O(\log M)$, де M — кількість комірок у масиві. Двовимірний суматор — це узагальнення одновимірного суматора. Ця структура може змінювати значення у комірці двовимірного масиву і шукати суму у прямокутнику з протилежними вершинами (1, 1) і (X, Y). Кожну операцію виконують за час $O(\log^2 M)$. Прочитати детальніше про цю структуру даних можна, наприклад, за такою адресою: http://e-maxx.ru/algo/fenwick_tree.

Для того, щоб знайти суму чисел у довільному прямокутнику з протилежними вершинами (X_1,Y_1), (X_2,Y_2) при $X_1 \le X_2$ і $Y_1 \le Y_2$, можна скористатись формулою включення-виключення. Позначимо через $\text{sum}(X,Y)$ суму чисел у прямокутнику з протилежними вершинами (1, 1) та (X, Y). Тоді шукана сума дорівнює $\text{sum}(X_2,Y_2) – \text{sum}(X_1 – 1,Y_2) – \text{sum}(X_2,Y_1 – 1) + \text{sum}(X_1 – 1,Y_1 – 1)$.

Отже, сумарна складність нашого алгоритму $O(N \log^2 R)$, де N – кількість запитів, а R — довжина діапазону координат X або Y. У нашому випадку $R \le 600$.

Альтернативне розв'язання

Очевидним розв'язанням з часовою складністю $O(N^2)$ є таке:

- при кожному висаджуванні солдата запам'ятовуємо його координати та номер за допомогою певного масиву;
- при кожному відкликанні солдата вилучаємо його координати й номер з масиву;
- при кожному запиті щодо кількості солдатів, розташованих від даної точки на відстані, що не перевищує d, аналізуємо елементи масиву і для кожного солдата перевіряємо, на якій відстані від даної точки він розташований.

Перші два типи запитів виконують за сталий час, якщо зберігати:

- у двовимірному масиві `A[x][y]` — кількість солдатів з координатами (x,y);
- у масивах `X[i]` та `Y[i]` — координати x та y солдата з номером i або –1000, якщо такого солдата немає.

«Висадити» солдата s у точку (x, y) можна таким чином:

- `inc(A[x][y]); X[s]:=x; Y[s]:=y;` — мовою Pascal;
- `++A[x][y]; X[s]=x; Y[s]=y;` — мовою C++.

«Відкликати» солдата s можна таким чином:

- `if X[s]<>-1000 then begin dec(A[X[s]][Y[s]]); X[s]:=-1000; Y[s]:=-1000 end;` — мовою Pascal;
- `if (X[s]!=-1000) { --A[X[s]][Y[s]]; X[s]=Y[s]=-1000; }` — C++.

Обидві ці дії виконуються за сталий час.

Для пришвидшення відповіді на третій запит можна зробити таке:

1. Виберемо сталу K.
2. Кожного солдата будемо вважати або «старим», або «новим»:
   - у комірках таблиці A будемо зберігати кількість «старих» солдатів у точках з координатами, що є індексами комірки;
   - у комірках допоміжного лінійного масиву довжини K будемо зберігати інформацію про «нових» солдатів — тих, кого висаджено на полігон останніми.
3. За таблицею A побудуємо допоміжну таблицю B, у комірці (X,Y) якої буде стояти сума чисел у прямокутнику з протилежними комірками (1,1) та (X,Y) таблиці A. Це можна зробити згідно з такою формулою:
$$B(X,Y) = A(X,Y) + B(X-1,Y) + B(X,Y-1) - B(X-1,Y-1).$$
4. При запиті першого типу будемо долучати подію «прийшов новий солдат з такими координатами і таким номером» у проміжний масив.
5. При запиті другого типу будемо долучати подію «видалено солдата з такими координатами і таким номером» у проміжний масив. Якщо солдата з таким номером не було на полігоні, не робитимемо нічого.
6. Якщо розмір допоміжного масиву дорівнює K, переносимо інформацію з нього у таблицю A, перебудуємо таблицю B й очистимо допоміжний масив.
7. При кожному запиті третього типу будемо шукати суму чисел у відповідному квадраті у таблиці A, застосовуючи формулу включення-виключення для таблиці B та аналізуючи елементи допоміжного масиву (будемо збільшувати чи зменшувати відповідь на 1, якщо висаджують чи відкликають солдата з відповідного квадрата).

Запити перших двох типів, як і раніше, виконують за сталий час, а запит третього типу виконують за час $O(K)$. Додатково через кожні K запитів перших двох типів перебудовуємо таблицю B за час $O(R^2)$, де R, як і раніше, — довжина

діапазону координат X або Y. Отримуємо ефективність за часом $O(NK + R^2N/K)$.

Потрібно лише вдало підібрати сталу K. Можна вважати що $R^2 = 360\ 000$. При $K = 600$ маємо: $NK \le 30\ 000\ 000$, $R^2N/K \le 30\ 000\ 000$, $NK + R^2N/K \le 60\ 000\ 000$.

2. Розклад уроків

В умові зазначено, що з кожного предмету треба провести не більше ніж T уроків. Доведемо, що за цієї умови завжди можна знайти потрібний розклад.

Назвемо клас *насиченим*, якщо у цьому класі треба провести саме T уроків. Предмет назвемо *насиченим*, якщо з цього предмету в усіх класах разом потрібно провести саме T уроків. Інакше кажучи, насичений клас не може пропустити жодного уроку, а насичений предмет на кожному уроці має вивчатися в якомусь класі.

Спочатку знайдемо розклад для першого уроку, задовольнивши такі вимоги:

- кожний насичений предмет вивчається у деякому класі;
- кожний насичений клас має якийсь урок.

Після цього:

- кожний урок, який буде у першому стовпчику розкладу, вилучимо з відповідного рядка умови (якщо число, що позначає внесений до розкладу урок, зустрічається у відповідному рядку умови більше одного разу, ми зменшимо кількість таких чисел у цьому рядку на 1);
- інші уроки будемо намагатися розмістити в решті $(T - 1)$ стовпчиках таблиці розміру $N \times T$, що задає розклад (див. формат вихідного файлу).

При складанні першого стовпчику задіяні всі насичені уроки, після цього з кожного предмету буде потрібно провести не більше ніж $(T - 1)$ урок. Тому в такий же спосіб ми можемо побудувати розклад для другого і наступних уроків. Перед побудовою кожного стовпчика розкладу будемо поновлювати списки насичених класів і предметів. Насиченими будемо вважати ті класи, яким залишилося стільки уроків, скільки є ще незаповнених стовпчиків у розкладі. Аналогічно поновимо перелік насичених предметів. На кожному етапі будемо використовувати, звісно, оновлені списки насичених класів і предметів. Кількість незаповнених стовпчиків таблиці-розкладу після кожного заповнення буде зменшуватися, тому за T кроків ми побудуємо весь розклад.

Залишилось описати, як скласти розклад для першого уроку. Це завдання зводиться до класичної задачі пошуку паросполучення у дводольному графі. Розглянемо граф, у якому одні вершини відповідають класам, а інші — предметам. З'єднаємо ребрами такі пари вершин U, V, що U відповідає деякому класу, а V відповідає предмету, з якого у цьому класі потрібно провести хоча б один урок. *Паросполученням* назвемо таку сукупність ребер графа, у якій жодні два ребра не мають спільних вершин. Це якраз відповідає таким двом умовам:

- ніякий предмет не може вивчатися у двох класах одночасно;
- у кожному класі в один і той самий час може бути урок лише з одного предмету.

Потрібно розподілити предмети таким чином, щоб це відповідало деякому паросполученню, і включити у нього всі насичені вершини, тобто вершини, які відповідають насиченим класам або предметам. Покажемо, як це зробити.

Сукупність вершин, які відповідають класам, назвемо *нижньою долею*, а

сукупність вершин, які відповідають предметам, — *верхньою долею*. Спочатку побудуємо паросполучення, в яке з нижньої долі входять усі насичені вершини і лише вони. Уявімо, що вже є паросполучення, в яке з нижньої долі входять лише насичені вершини, але не всі. Можна вважати, що таке паросполучення існує і спочатку, але воно не містить жодного ребра. Покажемо, як збільшити кількість ребер у цьому паросполученні на 1. Для цього перетворимо граф на орієнтований. Усі ребра, які входять до паросполучення, орієнтуємо «згори вниз» — від предметів до класів, а решту ребер орієнтуємо «знизу вгору» — від класів до предметів. Виберемо довільну насичену вершину U з нижньої долі, яка ще не входить у паросполучення. Знайдемо таку вершину V з верхньої долі, яка також не входить у паросполучення і до якої можна дійти від U, рухаючись ребрами у відповідності до їх орієнтації (доведення існування такої вершини V подано у наступному абзаці). Вершину V та ланцюг ребер, що веде до неї, можна знайти за допомогою відомої процедури *пошуку в ширину*. Нехай U, V_1, U_1, V_2, U_2, ... , V_k, U_k, V — послідовні вершини, які лежать на згаданому ланцюжку ребер. Також можливий випадок, коли $k = 0$, тобто коли з U можна безпосередньо дістатися до V. Згідно з побудовою, вершини V_1, V_2, ... , V_k розташовані у верхній долі, а U_1, U_2, ... , U_k — у нижній. До паросполучення входять k ребер (V_1, U_1), ... , (V_k, U_k). Замінимо їх на ($k + 1$) ребро (U, V_1), (U_1, V_2), ... , (U_k, V), що не належать до нього. У результаті отримаємо паросполучення, яке містить на одне ребро більше і в яке з нижньої долі залучено лише насичені вершини. У випадку $k = 0$ ми просто долучимо до паросполучення ребро (U, V). Повторюючи вказане перетворення, побудуємо паросполучення, якому належать усі насичені вершини першої долі.

Доведемо від супротивного, що потрібна вершина V на кожному кроці знайдеться. Припустимо, що це не так. Нехай Y_1, Y_2, ... , Y_l — усі вершини *верхньої* долі, яких можна досягнути з U, рухаючись орієнтованими ребрами. Згідно з припущенням, усі ці вершини належать до паросполучення. Нехай (Y_1, X_1), (Y_2, X_2), ... , (Y_l, X_l) — ребра паросполучення, які містять згадані вершини. Кожної вершину X_i ($1 \le i \le l$) також можна досягнути з U, рухаючись орієнтованими ребрами. Для цього потрібно лише вийти з відповідної Y_i. Вершини U, X_1, X_2, ... , X_l є насиченими, тобто з кожної з них, без урахування орієнтації, виходить по T ребер. Усі ці ребра мають у якості інших кінців лише вершини Y_1, Y_2, ... , Y_l, інакше у верхній долі існувала би ще одна вершина, досяжна з U. Тому до вершин Y_1, Y_2, ... , Y_l разом входять, без урахування орієнтації, не менше ніж $(l + 1)\cdot T$ ребер. Але це неможливо, бо до кожної з цих l вершин може входити щонайбільше T ребер. Отримана суперечність свідчить про хибність припущення про відсутність потрібної вершини V.

Тепер, якщо це не вийшло відразу, перебудуємо паросполучення таким чином, щоб і всі насичені вершини *верхньої* долі увійшли до нього. Змінимо орієнтацію всіх ребер. Розглянемо насичену вершину *V верхньої* долі, яка не належить паросполученню. Нам достатньо знайти:

- або вершину U в нижній долі, яка не входить до паросполучення і є досяжною з V згідно з новою орієнтацією ребер;
- або ненасичену вершину W у верхній долі, яка знову-таки досяжна з V.

Існування однієї з таких вершин доводиться точно так само, як до цього ми доводили існування вершини V, досяжної з U. У першому випадку в нас буде ланцюг V, U_1, V_1, U_2, V_2, ... , U_k, V_k, U, в якому ребра (U_1, V_1), ... , (V_k, U_k) можна

замінити на (V, U_1), (V_1, U_2), ... , (V_k, U). У другому випадку отримаємо ланцюг V, U_1, V_1, U_2, V_2, ... , U_k, W, в якому ребра (U_1, V_1), (U_2, V_2), ... , (V_k, W) можна замінити на (V, U_1), (V_1, U_2), ..., (V_{k-1}, U_k). У цьому (другому) випадку ми не збільшимо кількість ребер, але долучимо до паросполучення насичену вершину V замість ненасиченої W. Будемо повторювати описане перетворення, поки не долучимо всі насичені вершини *верхньої* долі до пар.

Після вказаних процедур ми отримаємо розподіл предметів для першого уроку. Повторюючи їх, можна побудувати весь розклад.

Оцінимо кількість операцій, яку вимагає вказаний алгоритм. Пошук у ширину потребує $O(e)$ операцій, де e — кількість ребер у графі. Наш граф містить $O(NT)$ ребер. Для побудови паросполучення ми застосовуємо пошук у ширину $O(N)$ разів, бо маємо не більше ніж N насичених класів, а тому й не більше ніж N насичених предметів. Отже, пошук одного паросполучення потребує $O(N^2T)$ операцій. Для багатьох графів справджується оцінка $O(NT)$. Така оцінка досягається завдяки тому, що на більшості пошуків у ширину проглядаються далеко не всі ребра. Тому для одного паросполучення маємо теоретичну оцінку $O(N^2T)$ і практичну — $O(NT)$. Для побудови розкладу ми шукаємо T паросполучень. Отже, для розв'язання задачі теоретично потрібно $O(N^2T^2)$ операцій, а на практиці — $O(NT^2)$.

Зробимо ряд зауважень щодо можливого *практичного використання* поданого розв'язання задачі.

1. Описаний алгоритм на кожному етапі намагається навантажити лише насичені класи. Тому багато «вікон» виникає саме на початку розкладу. Для того, щоб розклад виглядав більш природно, його можна формувати з кінця.

2. Обмеження, що кожен предмет веде лише один учитель, не є істотним полегшенням задачі. Воно лише робить умову задачі зручнішою для сприйняття. До даного обмеження можна звести випадок, коли предмет ведуть k вчителів, але їх сумарне навантаження не перевищує kT уроків. Для цього треба лише розподілити, які вчителі будуть вести свій предмет у яких класах і для кожного вчителя занумерувати його предмет окремим числом. Розподіляти треба так, щоб кожному вчителеві дісталося не більше T уроків. Схоже перетворення можна зробити, коли один вчитель веде декілька уроків. Тоді всі його уроки потрібно позначити одним числом.

3. Стовпчики побудованої таблиці-розкладу можна переставляти. Тому розв'язання можна пристосувати до складнішої ситуації, коли якийсь учитель не може бути у школі під час проведення деякого уроку. Нехай s – кількість учителів, а k_1, k_2, ... , k_s – кількості уроків, які вони мають провести. Нехай для i-того учителя $(1 \le i \le s)$ є t_i номерів уроків, на яких він не може бути. Тоді якщо $k_1t_1 + k_2t_2 + ... + k_st_s < T$, то завжди можна підібрати гарний розклад. Дійсно, будемо переставляти стовпчики розкладу циклічно. Тоді i-тий учитель може «зіпсувати» не більше k_it_i зсувів, отже потрібний варіант залишиться.

Деякі інші застосування паросполучень описані у статтях:

[1] О. Аксенов, О. Рибак, *Застосування однієї графової теореми при розв'язуванні деяких задач*, У світі математики, 8 (2002), № 3, 55-58, доступна за адресою http://mathsociety.kiev.ua/rybaka01.pdf.

[2] О. Рибак, *Ідеальні розбиття підмножин*, У світі математики, 10 (2004), № 2,

29–35, доступна за адресою http://mathsociety.kiev.ua/rybaka02.doc.