Розбір задачі «Газон у Потоколяндії»

Будемо косити газон за таким алгоритмом: за першу годину косимо перші k метрів, за другу годину останні k метрів, потім найлівіший відрізок з k метрів ніколи не покошеної трави, потім найправіший відрізок з k метрів ніколи не покошеної трави. Тоді в якийсь момент залишиться ділянка довжиною менше, або рівно k метрів, яку ми можемо скосити через те, що або правіше, або лівіше цієї ділянки трава вже виросла. Частковий випадок коли $k \cdot 2 > n$. Оскільки ми не можемо за другу годину покосити останні k метрів, то відповідь на такий випадок — 3.

Рішення, які не розглядали цей випадок, набирали 76 балів.

Розбір задачі «Мафія у Потоколяндії»

Припустимо, що ми наймаємо t тестувальників. Тоді сумарно буде написано $t \cdot a + t \cdot (t-1) \cdot b$ слів. $t \cdot a$ тому що кожен з t тестувальників напише по a слів у загальний чат, а також кожному $(t-1)$-ому іншому тестувальнику b слів у особистий.

Отже, потрібно знайти мінімальне таке t, що $t \cdot a + t \cdot (t-1) \cdot b \geqslant n$. Можна помітити, що якщо $b > 0$, то $t \leqslant 2 \cdot 10^6$, і таке t можна перебрати циклом.

Якщо b рівне нулю, то маємо формулу $t \cdot a \geqslant n$. Мінімальне t можна знайти формулою $\frac{n+a-1}{a}$.

Розбір задачі «Козак Вус та міньйони»

Для початку відсортуємо масив за зростанням b_i, а при рівності за неспаданням a_i. Стверджується, що для будь-якого k, ми можемо досягти $f(k)$, якщо серед перших k міньйонів виберемо у загін або всіх k міньйонів, або $(k-1)$ перших міньйонів і ще їх лідера серед інших $n-k+1$.

Це ми можемо стверджувати, адже, якщо ми виберемо у загін менше ніж $(k-1)$-ого серед перших k, то можна з більшою вигодою замінити міньйона на одного із перших k.

Отже, $f(k) = min(pref_k + diff_k, pref_{k-1} + suf_k)$, де $pref_i$ — сума всіх b_j для $j \leqslant i$, $diff_i$ — мінімальна різниця $(a_j - b_j)$ для усіх $j \leqslant i$, suf_i — мінімальне a_j для $j \geqslant i$.

Розбір задачі «Футбол у Потоколяндії»

Перейдемо до формальної умови задачі. В нас є масив a та запити двох типів: додати до a_i та відняти від a_{i+1} число k, а також знаходити на відрізку з l по r підвідрізок з сумою w.

Порахуємо масив часткових сум $psum$ для префікса такий, що $psum_i$ буде рівне сумі всіх елементів масиву a з індексами не більшими за i. Після цього наші запити будуть мати наступний вигляд: змінити один елемент масиву $psum$ та знайти на відрізку два індекси i та j, що $psum_j$ - $psum_i = w$.

Будемо підтримувати множину пар індексів, для яких виконується попередня умова, а також не існує індексу i_1, що $j > i_1 > i$ і $psum_j - psum_{i_1} = w$ та не існує індексу j_1, що $j > j_1 > i$ і $psum_{j_1} - psum_i = w$. Не складно довести, що якщо на відрізку існує хоч одна добра пара індексів, то існує й така пара, що належить нашій множині. При зміні елемента зруйнується не більше двох пар індексів з множини й утвориться нових теж не більше двох. Для кожного першого індексу з множини пар можна в дереві відрізків зберігати другий і таким чином замінити всі запити другого типу на запити знаходження мінімума на відрізку.

Розбір задачі «Козак Вус та картопля»

Знайдемо значення найкоротших відстаней від кожного міста до кожного бункера. Нехай $dist_{ij}$ — найкоротша відстань від міста з номером j до бункера з номером i. Щоб порахувати ці значення достатньо запустити алгоритм Дейкстри s разів (для кожного бункера як можливого початку) по розвернутим ребрам (тоді рахуватиметься не відстань від бункерів до міст, а навпаки).

Помітимо, що відповідь можна шукати за допомогою бінарного пошуку. Справді, якщо Козаки встигають заховати всі мішки з картоплею за x годин, то за y, де $y > x$ теж встигнуть. Опишемо функцію $check(x)$ яка визначатиме, чи достатньо x годин для того, щоб заховати всі мішки з картоплею у бункери. Для кожного міста i від 1 до n розглянемо $m(i)$ — бітову маску з s бітів: j-ий біт цієї маски буде рівний 1 лише тоді, якщо $dist_{ji} \leqslant x$. Іншими словами, це бітова маска що визначає бункери до яких можна заховати мішки з картоплею з міста з номером i. Помітимо, що групу міст з однаковим значенням m можна умовно об'єднати в одне місто, просумувавши кількість мішків з картоплею в них. Розглянемо двочастковий граф, де в лівій частини є вершини пронумеровані від

0 до $2^s - 1$, а у правій — від 0 до $s - 1$. Тепер для кожної вершини лівої частини створимо стільки її копій, скільки мішків з картоплею у містах з відповідним значенням m сумарно; для кожної вершини правої частини створимо стільки її копій, скільки мішків з картоплею може містити відповідний цій вершині бункер. Проведемо ребра між вершинами різних частин так, що ребро з вершини лівої частини яка є копією вершини з номером m веде у вершину правої частини, що є копією вершини з номером t тоді й тільки тоді, якщо у m біт з номером t рівний 1. Легко помітити, що кількість пар максимального паросполучення має бути рівною розміру лівої частини, щоб x годин було достатньо, щоб заховати всі мішки з картоплею у бункери. Іншими словами, на цьому графі має існувати повне паросполучення.

Скористаємось теоремою Холла для перевірки існування повного паросполучення.

Трохи поміркувавши :) (проаналізувавши отриманий вище граф) можна помітити, що достатньо перебрати бітову маску бункерів та перевірити чи кількість мішків з картоплею, що можна заховати лише у бункери з цієї маски, не перевищує суми розмірів бункерів з цієї маски. Якщо ця умова виконається для всіх масок від 0 до $2^s - 1$, то за згаданою вище теоремою можна стверджувати, що повне паросполучення на вигаданому вище графі існує.

Порахуємо масив c розміру 2^s, де c_i — сума значень p_j для всіх міст з номером j, що $m(j) = i$.

Тоді порахуємо масив d розміру 2^s, де d_i — сума значень c_j для всіх j, що $i \; and \; j = j$. Тут and — операція побітового І.

Помітимо, що необхідне нам значення кількості мішків з картоплею, що можна заховати лише у бункери з маски m рівне d_m для будь-якого m. Після цього легко визначити чи достатньо x годин, щоб заховати всі мішки з картоплею у бункери.

Порахувати значення масиву d, знаючи значення масиву c, достатньо швидко можна наступним чином:

Здійснимо s ітерацій, де після ітерації з номером i значенням d_x буде $\sum c_y$, таких що $x \; and \; y = y$, та починаючи з біта з номером i, біти чисел x та y збігаються (біти нумеруються з 0).

Очевидно, що перед першою ітерацією $d_x = c_x$. На ітерації з номером i перерахуємо d_x.

Якщо в числі x біт з номером i рівний 0, то $d_{new,x} = d_x$.

Якщо в числі x біт з номером i рівний 1, то $d_{new,x} = d_x + d_{x \; xor \; 2^i}$.

Отже, легко реалізувати цей алгоритм наступним чином:

```
for (int i=0;i<(1<<s);i++){
    g[i]=c[i];
}
for (int i=0;i<s;i++){
    for (int mask=0;mask<(1<<s);mask++){
        if (mask&(1<<i)){
            d[mask]+=d[mask^(1<<i)];
        }
    }
}
```

Складність $O(s \cdot m \cdot log(n) + log(n \cdot w) \cdot s \cdot (n + 2^s))$.