|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | III етап Всеукраїнської олімпіади з інформатики (розбрі задач) |  |  |  | | --- | | **Задача A, 1 тур**     Гірший випадок буде коли перший гравець візьме максимум цукерок, а другий мінімум. Тому знайдемо максимум і мінімум в послідовності та виведемо їх різницю.  **Задача B, 1 тур**     Зчитаємо перший рядок  та виведемо його. Потім зчитаємо 3 останні в масив, відсортуємо його будь-яким алгоритмом(порівнювати рядки можна звичайними операторами < та >). Та виведемо їх як це просять в умові.  **Задача C, 1 тур**     Факторизуємо число звичайним алгоритмом пошуку простих дільників , тобто представимо число у вигляді: N= a1^b1\*a2^b2\*a3^b3\*…, де a1, a2, a3, … - прості числа. Тоді відповіддю буде число S = (a1^0+a1^1+a1^2+..+a1^b1)\*(a2^0+a2^1+a2^2+…+a2^b2)\*…  **Задача D, 1 тур**      Шуканий трикутник лежить у вершинах даного багатокутника. Алгоритм рішення такий: зафіксуємо одну вершину (припустимо це буде вершина i), будемо перебирати вершину  j від і+1 до N, а в кожній ітерації циклу будемо рушати деяку вершину К поки площа трикутника (i,j,k) – зростає. Максимум який буде досягатися при обрахунках і буде відповіддю на задачу. Нижче наведений псевдокод рішення: for i=1..N    {        k = i+1;        for j=i+1..N     {         If (k<=j) k= j+1;         last = -1;         while (k<=N)             {             temp = square(i,j,k);              if (temp > max) max = temp;             if (temp < last) break;             last = temp;             k = k+1;         }       k = k-1;     }        }   В Max – буде відповідь на задачу.   Примітка: функція square повинна вертати подвоєну площу трикутника, бо так вона буде ціла. Також для неї потрібно використовувати тип int64. Щоб порахувати подвоєну площу без дійсних чисел потрібно використовувати формулу:  2\*S = |(x1-x2)\*(y1+y2) + (x3-x2)\*(y3+y2) + (x1-x3)\*(y1+y3)|  **Задача E, 1 тур**      Перше рішення,яке набирало 30 балів є динамічне програмування за О(N\*N): A[1] = 0 A[i] = max(A[j]+|X[j]-X[i]|\*C[j])+T[i] 1<=j<i Позбудемось модуля , будемо мати дві формули з яких треба буде вибрати максимум: A[i] = max(max(A[j]+X[j]\*C[j]-X[i]\*C[j]),max(A[j]-X[j]\*C[j]+C[j]\*X[i]))+T[i] Нехай K1[i] = -C[i] , B1[i] = A[i] +X[i]\*C[i] , K2[i] = C[i] , B2[i] = A[i] -X[i]\*C[i], це є прямі з кутовими коефіцієнтами K1 та K2 та вільними  членами B1 та B2. Отже кожен раз нам буде потрібно знаходити пряму яка перетинає нашу в максимальній по У координаті. Та створювати нові прямі і додавати їх до множини. Це можна зробити звичайною множиною прямих в кожен момент вона повинна додавати прямі тільки тоді коли вони хоча-б при одному Х є найвищими. Аналогічно треба і прибирати прямі з множини якщо вони в усіх точках в яких Х від -1000000 до 1000000 є ця пряма має менший У ніж у якоїсь іншої.  a  На малюнку зображено червоними лініями ті прямі, які будуть входити в множини, а чорними ті, які не будуть.  **Задача A, 2 тур**     Відповідь на задачу розкладається на  2 випадки: K > N, відповідь: 2 K <= N, відповідь: (2\*N+K-1)/K,  **Задача B, 2 тур**     Задамо всі погані пари в матриці. Потім просто переберемо три смаки і за О(1)  перевіримо цю трійку на правильність.  **Задача C, 2 тур**     Задамо найкоротші відстані від вершин A та B алгоритмом Дейкстри(нехай це будуть масиви D1 та D2). Переберемо вершину в якій будемо купляти товар, нехай це буде першина V, тоді відповіддю буде Min(D1[v]+D2[v]+Cost[v]), Cost[v] – ціна покупки товару в веришні v.  **Задача D, 2 тур**      Якщо (a=1 та b=1) або (a=1 та c=1) або (b=1 та c=1) то відповідь “NO”. Якщо інші критерії не виконались і виконалось хоча-б одна умова:  a mod 2=0 або b mod 2=0 або c mod 2=0, то відповідь “YES” в усіх інших випадках “NO”.  **Задача E, 2 тур**      Нехай: - d1[i,j] – кількість клітинок в які ми можемо походити вгору з клітнки(i,j) не перетинаючи дерев. - d2[i,j] – кількість клітинок в які ми можемо походити вліво з клітнки(i,j) не перетинаючи дерев. -d3[i,j] – кількість клітинок в які ми можемо походити вниз з клітнки(i,j) не перетинаючи дерев. - d4[i,j] – кількість клітинок в які ми можемо походити вправо з клітнки(i,j) не перетинаючи дерев. - e1[i,j] = min(d1[i,j], d2[i,j]) - e2[i,j] = min(d3[i,j], d4[i,j])  b Синіми лініями позначено e1 для клітинок, червоними e2 для клітинки для яких ми будемо рахувати відповідь(тобто кількість трас кут яких співпадає з даною клітинко).     Очевидно, що другий кут траси повинен лежати на тій самій діагоналі що містить і перший кут.  Зобразимо це на нашому малюнку:  c  Тому будемо рахувати всі дані для кожної діагоналі окремо. Будемо для кожної діагоналі підтримувати ті сині лінії що зображені на малюнку. Для кожного Х нам потрібно знати кількість синіх ліній, що починаються на даному Х, щоб взнавати суму кількостей на деякому відрізку нам допоможе дерево Фенвіка. Також нам треба зберігати кучу(по мінімум на горі), в якій будемо зберігати Х, що будуть закінченнями синіх ліній. Коли якийсь елемент стає менше ні можна (тобто його Х менший ніж тий на якому ми знаходимось) то його треба викинути із кучі і змінити відповідний елемент в дереві Фенвіка. Коли ми хочемо порахувати відповідь для клітинки, то просто викинемо з кучі все, що нам не потрібно та візьмемо в дереві Фенвіка потрібну суму, тобто таку що довжина нашої траси буде не менше L і сторона не буде більше ніж це можливо. | |