Colorful

Christmas Contest 2024

Time Limit: 10s, **Memory Limit:** 512MiB

Финалните тестове можете да намерите тук. Задачата е отново отворена за предаване на решения (ползвайки финалните тестове). Можете да видите решенията на първите трима участници indjev99, anrieff, и Sorting.

Можете да изпращате контестации или въпроси (ако имате такива) до 23:00 на 5-ти Януари, 2025г. на thinkcreative@outlook.com. След това ще обявя класирането за финално и ще почна да се свързвам с хората, спечелили награди, за да организираме как да си ги получат.

В случай, че сте под 16 години, моля пишете ми на горния мейл за да участвате в надпреварата за награда за млад участник :))

На Ели ѝ омръзнаха бъговете на Gimp и безбожната цена на Photoshop, в следствие на което момичето реши да си напише собствена програма за редактиране на картинки.

Една от интересните функционалности на този тип програми е селектиране на региони: цъкайки някъде по картинката, програмата "селектира" регион, който е сходен по цвят. Това е полезно, когато искате да премахнете или копирате обект от картинка - примерно някоя от звездичките на американския флаг, кръста от английския, или кръга от японския. Програмите са "умни" и позволяват известна разлика в цветовете, както и значително по-неправилни форми - например, лицето на Мона Лиза, или пък луната от картината "Звездна нощ" на Ван Гог.

В своята програма Ели е имплементирала следния алгоритъм за намиране на регионите: цъкайки на даден пиксел, тя селектира избрания пиксел; неговите съседи по хоризонтала и вертикала, които имат сходен цвят с избрания пиксел; съседите на тези съседи, които имат сходен цвят с избрания пиксел и т.н. По-формално, алгоритъмът селектира свързаната компонента от пиксели, които съдържат първоначално избрания и са сходни по цвят с него.

Дефинираме два пиксела за "сходни по цвят", ако сумата от разликите на квадратите на (R, G, B) компонентите (червено, зелено и синьо) на двата пиксела е под някаква граница **K**. По-конкретно, ако два пиксела имат цветове ($\mathbf{R_1}$, $\mathbf{G_1}$, $\mathbf{B_1}$) и ($\mathbf{R_2}$, $\mathbf{G_2}$, $\mathbf{B_2}$), то казваме, че са "сходни по цвят", ако $(\mathbf{R_1} - \mathbf{R_2})^2 + (\mathbf{G_1} - \mathbf{G_2})^2 + (\mathbf{B_1} - \mathbf{B_2})^2 \leq \mathbf{K}$, за някакво цяло число **K**.

Забележете, че селектираната компонента не е задължително изпъкнала - може да има неправилна форма или дори "дупки" в себе си. Например, очите на Мона Лиза не биха били включени в селектираните пиксели, ако сме "цъкнали" на челото ѝ, но кожата около тях би била.

Крис беше скептична колко добре ще работи тази функционалност по начина, по който Ели я е имплементирала. За да ѝ докаже това, тя измисли следното предизвикателство! Крис си е избрала картинка с **N** реда и **M** колони, която Ели не знае. От Ели се иска да възстанови картинката ползвайки най-много **Q** от следните въпроси. Момичето назовава тройка $\mathbf{ROW_i}$ $\mathbf{COL_i}$ $\mathbf{K_i}$ - координатите ($0 \leq \mathbf{ROW_i} < \mathbf{N}$, $0 \leq \mathbf{COL_i} < \mathbf{M}$) на пиксел от картинката и числото $\mathbf{K_i} \leq \mathbf{L}$, което да се ползва за "сходност по цвят". Крис ѝ отговаря какъв е ($\mathbf{R_i}$, $\mathbf{G_i}$, $\mathbf{B_i}$) цветът на избрания пиксел, а както и кои са всички селектирани от алгоритъма на Ели пиксели с даденото $\mathbf{K_i}$. Забележете, че избраното **K** не може да бъде по-голямо от зададен за теста лимит **L**.

Вие искате да помогнете на Ели, като напишете програма, която "възстановява" картинката на Крис възможно най-добре.

Вход и Изход

При стартиране на програмата Ви, на стандартния вход (`stdin`) ще бъдат зададени целите числа **N**, **M**, **L**, и **Q** - съответно броят редове и колони на картинката, максималното $\mathbf{K_i}$, което можете да ползвате, и максималният брой въпроси, които можете да зададете.

Можете да задавате въпроси, като отпечатате на отделен ред на стандартния изход (`stdout`) тройка цели числа $\mathbf{ROW_i}$ $\mathbf{COL_i}$

$\mathbf{K_i}$ разделени с интервали. За всеки въпрос на стандартния вход (`stdin`) ще получите първо тройка цели числа между 0 и 255, включително, задаващи ($\mathbf{R_i}$, $\mathbf{G_i}$, $\mathbf{B_i}$) цвета на пиксела ($\mathbf{ROW_i}$, $\mathbf{COL_i}$), следвани от едно цяло число $\mathbf{S_i}$, задаващо колко пиксела включва компонентата на избрания пиксел, следвано от $\mathbf{S_i}$ * 2 цели числа, задаващи координатите им. Забележка: ползването на големи стойности на $\mathbf{K_i}$ обикновено води до повече върнати пиксели (в някои случаи – десетки хиляди), което може значително да забави решението ви при прекомерно ползване!

Когато сте готови (използвали сте всички **Q** въпроса, или сте решили, че нямате нужда от повече), на отделен ред на стандартния изход отпечатайте "Ready", след което **N** реда всеки с по 3 * **M** цели числа между 0 и 255, включително - (R, G, B) цветовете на всеки от пикселите на картинката според вашето решение.

За да се избегне забавяне на програмата в следствие на буфериране на изхода, след отпечатването на всеки въпрос (а както и на финалния отговор) `flush`-вайте изхода:

- C: `fflush(stdout);`
- C++: `cout << flush;`
- Java: `System.out.flush();`
- Python: `stdout.flush()`

Ограничения

- 1 ≤ **N**, **M** ≤ 512
- 10 ≤ **L** ≤ 5,000
- 1,000 ≤ **Q** ≤ 5,000
- 0 ≤ $\mathbf{R_i}$, $\mathbf{G_i}$, $\mathbf{B_i}$ ≤ 255
- 0 ≤ $\mathbf{ROW_i}$ < **N**
- 0 ≤ $\mathbf{COL_i}$ < **M**
- 0 ≤ $\mathbf{K_i}$ ≤ **L**

Примерен Вход	Примерен Изход
42 69 100 17 133 77 250 3 18 15 18 16 19 16 255 255 255 1 0 0 91 184 55 4 3 62 3 61 2 61 2 60 17 11 196 6 41 8 42 8 43 8 42 9 42 7 41 7	18 15 33 0 0 0 3 62 100 41 8 88 Ready 255 255 255 253 252 254 ... ...

Картинката има 42 реда и 69 колони. Максималната константа K, която можем да ползваме при намирането на свързаните компоненти, е 100. Имаме право да зададем най-много 17 въпроса. Задали сме 4 въпроса преди да изведем отговора. Цветът на пиксела (18, 15) от първия въпрос е (133, 77, 250). Има три пиксела със "сходни цветове" при константа K = 33 в неговата компонента: (18, 15), (18, 16) и (19, 16).

Оценяване

Ако използвате повече от **Q** въпроса; някой от въпросите е невалиден или не спазва зададения формат и ограничения; изведеният отговор е невалиден или не спазва зададения формат; използвате повече от разрешеното време; използвате повече от разрешената памет; или по някаква причина решението ви не завърши успешно (crash-не), то ще получите 0 точки за този тест.

В случай, че всичко е наред, точките за съответния тест ще бъдат формирани по следния начин. За всеки пиксел от вашата апроксимация на картинката ще сметнем "разстояние" от (R_1, G_1, B_1) цвета му до този на съответния му пиксел в оригиналната картинка (R_2, G_2, B_2) като сума от разликите в компонентите на квадрат: $(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2$. Нека *ourScore* е сумата от тези разстояния за цялата картинка, а *bestScore* е най-малката такава сума за този тест измежду всички участници. Точките ни за този тест ще бъдат равни на `5 * ((bestScore + 1) / (ourScore + 1))` - тоест релативно оценяване спрямо най-доброто решение.

Временно и Финално Класиране

Докато състезанието тече ще бъдат дадени 20 теста, на базата на които ще бъде налично временно класиране, достъпно за всички участници. След края на състезанието ще бъдат качени други 20 теста, на базата на които ще се изготви финално класиране. Забележете, че първите 20 няма да допринасят за точките на участниците във финалното класиране, но ще са донякъде подобни на финалните.

Визуализатор

За удобство на участниците задачата има визуализатор. След като предадете решение, тестовете, на които сте извели валиден отговор (отбелязани като зелени кръгчета на системата), стават линкове, чрез които можете да отворите визуализацията за съответния тест.