Zbigniew Czech
Treść zadania, Opracowanie

Piotr Stańczyk
Program

OI, Etap II, dzień próbny, 06.02.2007

Grzbiety i doliny

Bajtazar lubi wędrować po górach. Podczas swoich wędrówek odwiedza wszystkie okoliczne grzbiety i doliny. Dlatego, aby dobrze zaplanować czas, musi wiedzieć, ile grzbietów oraz dolin znajduje się na obszarze, po którym będzie wędrował. Twoim zadaniem jest pomóc Bajtazarowi.

Bajtazar dostarczył Ci mapę obszaru, który wybrał jako cel swojej kolejnej wyprawy. Mapa ma kształt kwadratu o wymiarach $n \times n$. Dla każdego pola (i,j) tego kwadratu (dla $i,j \in \{1,\ldots,n\}$) podana jest jego wysokość $w_{(i,j)}$.

Mówimy, że pola sąsiadują ze sobą, jeżeli mają wspólny bok lub wierzchołek (tzn. pole (i,j) sąsiaduje z polami $(i-1,j-1)$, $(i-1,j)$, $(i-1,j+1)$, $(i,j-1)$, $(i,j+1)$, $(i+1,j-1)$, $(i+1,j)$, $(i+1,j+1)$, o ile pola o takich współrzędnych mieszczą się na mapie).

Mówimy, że zbiór pól S tworzy grzbiet (dolinę), jeżeli:

- *wszystkie pola z S mają tę samą wysokość,*
- *zbiór S stanowi spójny kawałek mapy (z każdego pola ze zbioru S można przejść do każdego innego pola z S, przechodząc tylko z sąsiedniego pola na sąsiednie i nie wychodząc poza zbiór S),*
- *jeśli $s \in S$ oraz pole $s' \notin S$ sąsiaduje z s, to $w_s > w_{s'}$ (dla grzbietu) lub $w_s < w_{s'}$ (dla doliny).*

W szczególności, w przypadku, gdy cały obszar przedstawiony na mapie ma tę samą wysokość, jest on jednocześnie grzbietem i doliną.

Twoim zadaniem jest wyznaczyć liczbę grzbietów oraz dolin dla krajobrazu opisanego przez otrzymaną mapę.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia opis mapy,*
- *obliczy liczbę grzbietów i dolin dla krajobrazu opisanego przez tę mapę,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita n ($2 \leqslant n \leqslant 1\,000$) oznaczająca rozmiar mapy. W każdym z kolejnych n wierszy znajduje się opis kolejnego wiersza mapy. W $i+1$-szym wierszu (dla $i \in \{1,\ldots,n\}$) znajduje się n liczb całkowitych $w_{(i,1)},\ldots,w_{(i,n)}$ ($0 \leqslant w_{(i,j)} \leqslant 1\,000\,000\,000$), pooddzielanych pojedynczymi odstępami. Są to wysokości kolejnych pól w i-tym wierszu mapy.

Wyjście

Pierwszy i jedyny wiersz wyjścia powinien zawierać dwie liczby całkowite oddzielone pojedynczym odstępem — liczbę grzbietów, a następnie liczbę dolin dla krajobrazu opisanego przez mapę.

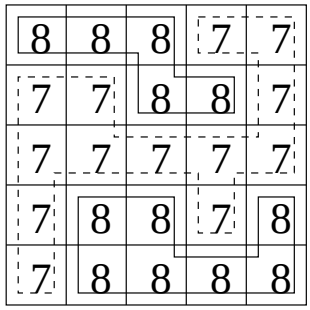
Przykład

Dla danych wejściowych:

```
5
8 8 8 7 7
7 7 8 8 7
7 7 7 7 7
7 8 8 7 8
7 8 8 8 8
```

poprawnym wynikiem jest:

```
2 1
```

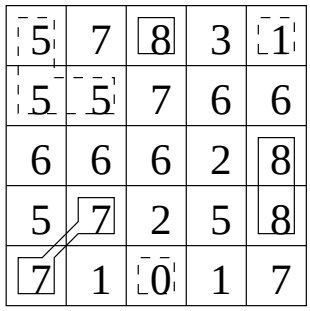


Natomiast dla danych:

```
5
5 7 8 3 1
5 5 7 6 6
6 6 6 2 8
5 7 2 5 8
7 1 0 1 7
```

poprawnym wynikiem jest:

```
3 3
```



Na powyższych rysunkach zaznaczone są grzbiety (linia ciągła) i doliny (linia przerywana).

Rozwiązanie

Rozwiązanie wzorcowe

Na wstępie wprowadźmy pojęcie *osiągalności pola*, które będzie pomocne w dalszej analizie.

Definicja 1 *Pole k jest osiągalne z pola p*, jeżeli ma taką samą wysokość co pole p oraz jeżeli do pola k można dojść z pola p, poruszając się tylko pomiędzy sąsiednimi polami o tej samej wysokości.
W szczególności, pole p jest osiągalne z pola p.

W rozwiązaniu wzorcowym liczbę grzbietów oraz dolin wyznaczamy w dwóch niezależnych fazach. Poniżej przedstawiamy sposób wyznaczania liczby dolin; podobnie można wyznaczyć liczbę grzbietów. Zadanie zinterpretujemy w języku grafów. Każde pole mapy Bajtazara potraktujemy jako wierzchołek grafu G. Krawędzie w tym grafie reprezentują możliwe ruchy pomiędzy sąsiednimi polami. Z każdego pola (wierzchołka

grafu) nieleżącego na brzegu mapy wychodzi zatem osiem krawędzi. Dolina w tej interpretacji jest maksymalnym, spójnym (jednokawałkowym) zbiorem pól o tej samej wysokości w, otoczonym polami o wysokości większej od w.

Zastanówmy się, w jaki sposób można stwierdzić, czy dane pole k należy do doliny. W tym celu można wyznaczyć wszystkie pola osiągalne z pola k, a następnie sprawdzić, czy wszystkie one sąsiadują z polami o nie mniejszej wysokości. Jeśli którekolwiek pole sąsiaduje z polem o wysokości mniejszej, to pole k (oraz każde inne pole osiągalne z pola k) nie należy do doliny. W przeciwnym razie, pole k oraz wszystkie pola z niego osiągalne należą do jednej, tej samej doliny. Co więcej, nie ma dalszych pól należących do tej doliny, bo gdyby takowe istniały, to byłyby osiągalne z pola k.

Powstaje pytanie, w jaki sposób efektywnie przeprowadzić opisaną wyżej procedurę. Okazuje się, że wystarczy w tym celu zastosować przeszukiwanie grafu G wszerz. Przeszukiwanie rozpoczynamy w wierzchołku k i odwiedzamy wszystkie osiągalne wierzchołki o tej samej wysokości co wierzchołek wyjściowy, odnajdując obszar *potencjalnej doliny* zawierającej k. Jeśli jakakolwiek krawędź wychodząca z wyznaczonego obszaru dochodzi do wierzchołka o mniejszej wysokości od wierzchołka k, to potencjalna dolina nie jest w rzeczywistości doliną i ani k, ani żaden z jej pozostałych wierzchołków nie należy do doliny. W przeciwnym razie potencjalna dolina okazuje się być rzeczywiście doliną. Dla każdego zbioru pól osiągalnych wystarczy wykonać tylko jedno przeszukiwanie — liczba dolin jest równa liczbie wykonanych przeszukiwań, dla których znaleziono rzeczywistą dolinę.

Ponieważ każdy wierzchołek w grafie G ma ograniczony stopień, nie większy niż osiem, więc sumaryczny czas wykonania przeszukiwań jest liniowy względem liczby wierzchołków. Tym samym otrzymujemy algorytm o złożoności $O(n^2)$, zapisany w plikach `grz.cpp` i `grz1.pas`.

Rozwiązanie alternatywne

Zadanie można także rozwiązać, używając struktury danych *Find-Union*[1] zamiast reprezentacji grafowej. Struktura umożliwia efektywny podział pól na rozłączne zbiory pól wzajemnie osiągalnych. W tym celu na początku działania algorytmu każde pole mapy reprezentowane jest jako jednoelementowy zbiór. Następny krok algorytmu polega na przeanalizowaniu wszystkich par sąsiadujących pól oraz połączeniu zbiorów, do których należą pola o tej samej wysokości. Na koniec wystarczy dla każdego uzyskanego zbioru sprawdzić, czy wszystkie pola należące do niego sąsiadują z polami nie niższymi od siebie — jeśli tak, to rozpatrywany zbiór stanowi dolinę. Oczywiście test sprawdzający, czy obszar jest grzbietem, można zrealizować analogicznie. Uzyskany w ten sposób algorytm ma złożoność $O(n^2 \log^* n)$. Rozwiązanie zostało zapisane w pliku `grzs1.cpp`.

Rozwiązanie niepoprawne

Głównym źródłem rozwiązań niepoprawnych było stosowanie rekurencyjnego przeszukiwania grafu, na przykład za pomocą przeszukiwania w głąb. Podejście to może spowodować

[1] Struktury *Find-Union* służą do przechowywania elementów pogrupowanych w rozłączne zbiory i efektywnego wykonywania operacji: połączenia zbiorów (*Union*) oraz zidentyfikowania zbioru, do którego należy element (*Find*). Więcej na ich temat można przeczytać w książkach [14] i [19].

przepełnienie stosu. Jest ono wysoce prawdopodobne dla testów typu „wąż” (patrz dalej). Rozwiązanie takie zostało zaimplementowane w pliku `grzb1.cpp`.

Testy

Testy do zadania zostały wygenerowane losowo, przy użyciu pięciu niezależnych metod:

- **płaski** — generator testów zawierających płaski obszar o zadanej wielkości oraz wysokości,
- **wąż** — generuje test o zadanej wielkości, umieszcza w nim „węża” (ciąg pól o tej samej wysokości i długości rzędu $O(n^2)$) oraz otacza go polami o losowych wysokościach,
- **losowy** — generuje planszę o zadanej wielkości i losowych wysokościach pól,
- **szachownica** — generuje planszę o zadanej wielkości, zawierającą dwie różne wysokości pól ułożone we wzór szachownicy,
- **losowy obszar** — generuje planszę o zadanej wielkości, wypełniając ją losowymi spójnymi obszarami o zadanej maksymalnej wielkości oraz wysokości.

Poniższa lista zawiera podstawowe informacje dotyczące testów. Przez n została oznaczona wielkość mapy, a przez h — maksymalna wysokość pól. W ostatniej kolumnie podany został rodzaj generatora użyty do stworzenia testu.

Nazwa	n	h	Opis
grz1a.in	8	2	szachownica
grz1b.in	10	100	wąż
grz2a.in	15	100	płaski
grz2b.in	16	267	szachownica
grz2c.in	13	20	losowy
grz3a.in	20	9852	płaski
grz3b.in	50	29	wąż
grz4.in	100	96 500	losowy
grz5.in	120	9 999	losowy obszar
grz6.in	200	1 000 000	losowy obszar
grz7.in	400	9 000 000	losowy
grz8a.in	800	100 000 000	losowy
grz8b.in	800	1 000 000	losowy obszar
grz9a.in	1 000	854	szachownica
grz9b.in	750	1 000 000 000	losowy
grz10a.in	1 000	50	wąż
grz10b.in	1 000	1 000 000 000	płaski