INF2010 - Structures de données et algorithmes

Été 2020

Travail Pratique 5

Graphes

Ce travail pratique sera assez différent des précédents. On vous demandait de rouler des tests localement et vous aviez accès au code du test lui-même pour vérifier si vous aviez la bonne logique. Ici on utlisera la plateforme HackerRank, c'est un site extrêmement utilisé pour se pratiquer pour des entrevues ou simplement pour se pratiquer à résoudre des problèmes algorithmiques.

Directives pour commencer le laboratoire :

- Rendez-vous sur le lien : www.hackerrank.com/poly-inf2010-tp5
- Faites-vous un compte HackerRank, rappelez-vous de votre nom d'utilisateur, il sera utile pour la remise
- Il est fortement recommandé de faire les énoncés dans cet ordre :
    - https://www.hackerrank.com/contests/poly-inf2010-tp5/challenges/pays-et-continents/
    - https://www.hackerrank.com/contests/poly-inf2010-tp5/challenges/trouver-la-sortie

Directives de remise :

- Télécharger vos deux fichiers Java (vous pouvez simplement faire copier-coller de l'éditeur en ligne dans un fichier sur votre ordinateur local). Vous pouvez leur donner comme noms :
    - PaysEtContinent.java
    - TrouverLaSortie.java
- Mettez clairement vos matricules ainsi que le nom d'utilisateur que nous devons vérifier le score sur HackerRank dans les commentaires de votre code (dès le début de préférence)
- Remettez-les comme à l'habitude sur moodle

Barème de correction :

Pays Et Continent	/10
Trouver La Sortie	/9
Style	/1

Le point sur le style est comme à l'habitude, il faut que le code soit clair et respecte les bonnes pratiques, le copier-coller (la duplication de code) est généralement une mauvaise pratique.

Les deux énoncés se retrouvent en annexes, mais il est recommandé de les visionner sur HackerRank.

Pays et continents

Cet algorithme permet de regrouper des pays selon leur continent en sélectionnant un algorithme *breadth-first search* ou bien *depth-first search*.

- **Carte du monde** `world` : Surface plane, plus précisément un carré, entourée d'eau
- **Continent** : Territoire entouré d'eau
- **Pays** : Sous-partie indissociée d'un continent

Chaque pays porte un nom distinct, soit un entier positif non-nul unique. Un pays ne peut faire partie que d'un seul continent.

Input Format

La première ligne contient le booléen `isBfs` qui détermine si votre algorithme doit être de type *breadth-first search* ou bien *depth-first search*.
Les lignes suivantes contiennent la carte du monde `world` de format **N** x **N**.
Chaque case représente une sous-région du monde `world[i][j]` ayant la valeur **k_ij**.

Une valeur de 0 représente une sous-région d'eau. Une valeur positive non-nulle représente une sous-région du pays dénommé par cette même valeur.

Constraints

0 < **N** < 10^6
0 <= **k_ij** < 2^32

Output Format

Le format de sortie doit respecter le format suivant :

- Chaque ligne représente un continent.
- Les continents sont écrits de gauche à droite, de haut en bas.
- Chaque continent doit contenir ses pays en ordre croissant.
- Tous les pays d'un même continent sont séparés uniquement par des espaces.

Sample Input 0

```
1
1 1 1 0 0 2
1 1 0 0 2 2
3 3 3 0 0 2
```

Sample Output 0

```
1 3
2
```

Sample Input 1

```
0
1 1 1 1 0 2
1 3 3 1 0 2
1 1 1 2 2 2
```

Sample Output 1

```
1 2 3
```

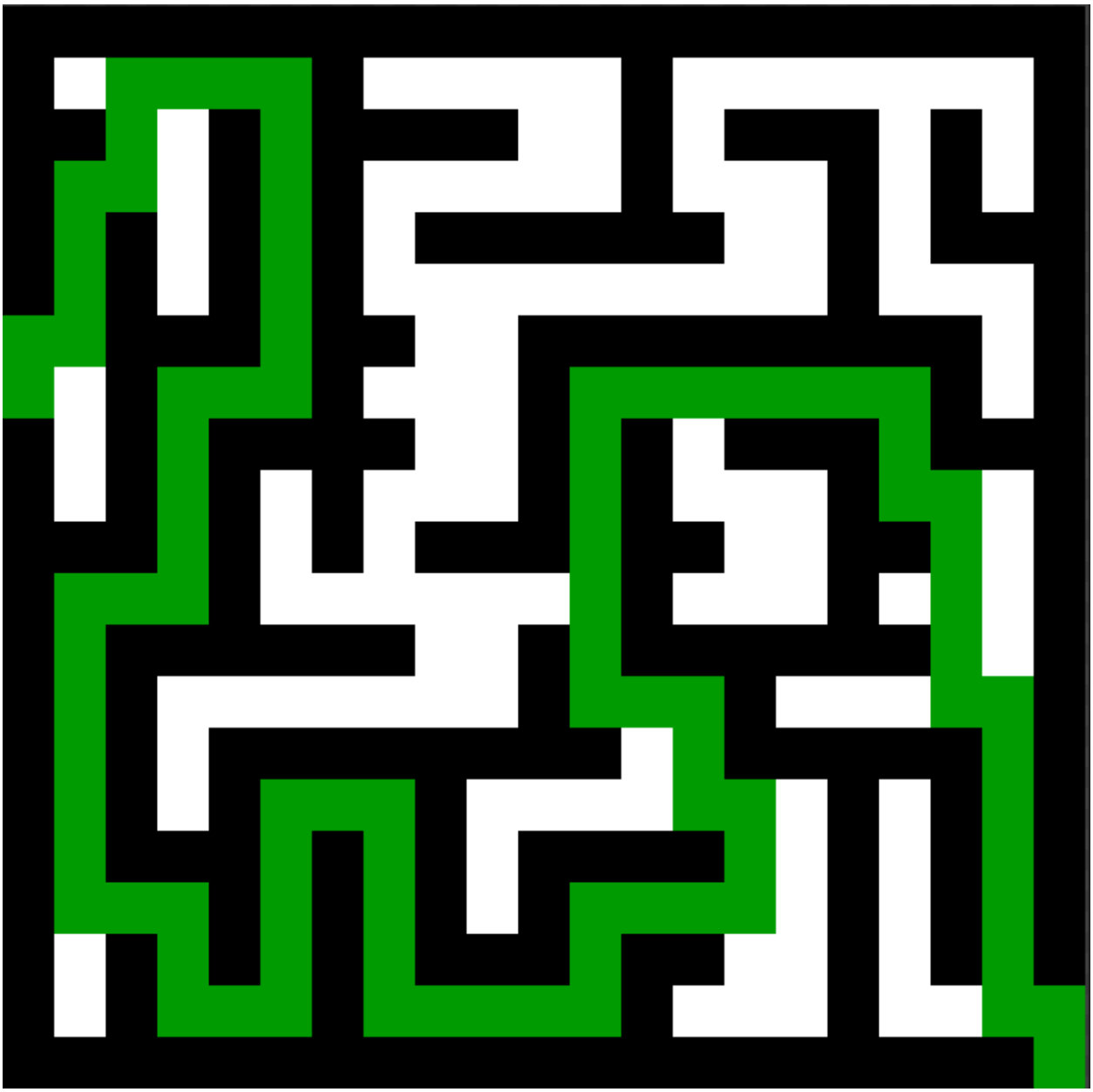
Trouver la sortie

Cet algorithme permet de trouver la longueur du chemin le plus court pour sortir d'un labyrinthe.

- Carte de jeu `board` : Planche de jeu composée de carreau de labyrinthe
- Carreau de labyrinthe `MazeTile` : Sous-partie de la planche de jeu représentant un morceau de plancher, un mur ou une entrée/sortie

Votre chemin doit commencer au point d'entrée et terminer au point de sortie . Les points d'entrée et sortie sont interchangeables. Votre chemin ne peut pas passer sur un carreau de labyrinthe qui est un mur.

Solution pour le 1



Input Format

Chaque ligne représente une partie de la carte de jeu `board` de format **N** x **N**
Chaque sous-région de la carte de jeu `board[i][j]` représente un carreau de labyrinthe.

Un carreau peut avoir trois valeurs différentes :

- 0 : Plancher
- 1 : Mur
- 2 : Entrée ou sortie

Constraints

0 < **N** < 10^6

Output Format

Imprimer l'entier représentant la longueur du chemin le plus court pour se rendre de l'entrée à la sortie du labyrinthe fourni.

Sample Input 0

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1
1 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1
1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1
1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1
2 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1
1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 1 1 0 1 1 1
1 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1
1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1
1 0 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1
1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1
1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1
1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1
1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1
1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1
1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
```

Sample Output 0

```
91
```

Sample Input 1

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
1 0 1 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 0 0 1
1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1
1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1
1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1
0 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1
2 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1
1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 0 1
1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1
1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1
1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1
1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1
1 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 2
1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 0 1
1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Sample Output 1

```
56
```