TP 3
PLUS COURT CHEMIN

Introduction

Pour le dernier tp, vous devez construire un logiciel qui trouve le plus court chemin dans un univers représenté par une matrice. Cette matrice représente un plateau de jeu, un joueur peut se déplacer d'une case à l'autre. Ce plateau contient une case 'porte' indiquant où le joueur entre et sort de l'univers. Le joueur doit ramasser trois trésors dans l'univers et revenir à la porte en utilisant le plus court chemin possible.

Description

Entrées

Sur la ligne de commande, votre programme va recevoir un nom de fichier '.txt' contenant la description de l'univers. La première ligne du fichier indique la taille de l'univers (la matrice) à l'aide de deux valeurs entières : largeur t_x et hauteur t_y. Ensuite, chaque ligne du fichier contient la description d'une ligne de la matrice. Finalement, les trois dernières lignes du fichier vont contenir les coordonnées des trésors à raison d'une coordonnée par ligne. Chaque coordonnée est représentée par deux valeurs : numéro de colonne et numéro de ligne.

- Taille de l'univers : deux valeurs entières plus grande que 1. C'est valeurs seront séparées d'un espace. Si une des valeurs est incorrecte, une erreur est affichée et le programme termine.

  $$s_x > 1 \wedge s_y > 1$$

  Par exemple, pour une matrice de $s_x = 5$ colonnes par $s_y = 6$ lignes :

```
5 6
```

- Matrice : chaque ligne contient une suite de caractères représentant une ligne de la matrice. Chaque caractère est en majuscule et représente un terrain différent, il n'y a pas d'espace entre les caractères.
  - N : plaine
  - E : eau
  - F : forêt
  - R : route
  - P : porte, il ne doit y avoir qu'une seule porte, sinon une erreur est affichée un le programme termine.

Par exemple, nous ajoutons les lignes de description du terrain pour la matrice de 5 x 6 :

```
5 6
NNNNN
NEEFF
RRRFF
EENRF
NENNR
NPENN
```

- Coordonnée des trésors : trois lignes de deux coordonnées chacune. Chaque coordonnée est composée d'un numéro de colonne x_i suivit d'un numéro de ligne y_i (où i représente le numéro du trésor entre 1 et 3, le premier trésor est sur la première ligne, suivit du deuxième trésor sur la deuxième ligne et du troisième trésor sur la troisième ligne). Un trésor ne peut pas être dans l'eau.

$$t_i = (x_i, y_i) \quad \text{où} \quad 0 \leq x_i < s_x \wedge 0 \leq y_i < s_y$$

Si une coordonnée n'est pas correcte, alors un message d'erreur est affiché et le programme se termine.

Par exemple, si nous ajoutons les trésors $t_1 = (1,2)$, $t_2 = (3,0)$ et $t_3 = (4,3)$.

```
5 6
NNNNN
NEEFF
RRRFF
EENRF
NENNR
NPENN
1 2
3 0
4 3
```

Calcule

Votre logiciel doit trouver le plus court chemin visitant les trois trésors et retournant à la porte ($p = (x_p, y_p)$). Le déplacement entre deux cases consécutives peut être orthogonal (une case en haut, une case en bas, une case à gauche, une case à droite) ou il peut être diagonal. Le cout d'un déplacement dépend du terrain qu'il y a à la case de départ et à la case d'arrivée.

- Si la case d'arrivée est un 'F', alors un déplacement orthogonal coute 2 et un déplacement diagonal coute 2.8.
- Si la case de départ **et** d'arrivée est un 'R', alors un déplacement orthogonal coute 0.5 et un déplacement diagonal coute 0.7.
- Il est impossible d'avoir un déplacement vers une case d'arrivée 'E'.
- Tout autre déplacement coute 1 s'il est orthogonal et 1.4 s'il est diagonal.

Il est impossible d'aller vers le haut lorsque nous sommes sur la première ligne de la matrice et il est impossible d'aller vers le bas lorsque nous sommes sur la dernière ligne. Par contre, la première colonne de la matrice communique avec la dernière colonne de la matrice : la colonne 0 est voisine de la colonne $s_x - 1$.

Votre algorithme devra calculer les plus courts chemins entre tous les points de l'univers. Cela va permettre de trouver la distance entre la porte et chacun des trésors (entre p et chaque t_i) et aussi la distance entre chacun des trésors (entre chaque t_i). Ensuite, il doit vérifier toutes les combinaisons possibles pour l'ordre de parcours des trésors afin de trouver l'ordre le plus court.

Par exemple, si nous utilisons l'algorithme de Dijkstra à partir de la porte, nous obtenons les plus courtes distances suivantes pour chaque case de la matrice.

N 6.2	N 6.6	N 7.6	N 7.2 (t_2)	N 6.6
N 5.2	E ∞	E ∞	F 6.2	F 7.1
R 4.3	R 3.8 (t_1)	R 3.4	F 4.8	F 5.6
E ∞	E ∞	N 2.4	R 2.8	F 4.2 (t_3)
N 1.4	E ∞	N 1.4	N 2.4	R 2.4
N 1	P 0	E ∞	N 2.8	N 2

Ce qui nous donne les distances suivantes entre la porte et les trésors :

$$\left|\overrightarrow{p, t_1}\right| = 3.8$$

$$\left|\overrightarrow{p, t_2}\right| = 7.2$$

$$\left|\overrightarrow{p, t_3}\right| = 4.2$$

Si nous réutilisons l'algorithme de Dijkstra à partir du premier trésor :

N 2.4	N 2.8	N 3.8	N 3.8(t_2)	N 2.8
N 1.4	E ∞	E ∞	F 3.3	F 3.3
R 0.5	R 0 (t_1)	R 0.5	F 2.5	F 2.5
E ∞	E ∞	N 1.4	R 1.2	F 3.2(t_3)
N 2.9	E ∞	N 2.4	N 2.2	R 1.9
N 3.3	P 3.8	E ∞	N 3.2	N 2.9

Nous avons donc les distances suivantes :

$$\left|\overrightarrow{t_1, p}\right| = 3.8$$

$$\left|\overrightarrow{t_1, t_2}\right| = 3.8$$

$$\left|\overrightarrow{t_1, t_3}\right| = 3.2$$

Nous réutilisons l'algorithme de Dijkstra à partir du deuxième trésor :

N 2	N 2	N 1	N 0 (t_2)	N 1
N 2.4	E ∞	E ∞	F 2	F 2.8
R 3.4	R 3.8 (t_1)	R 3.4	F 4	F 4.8
E ∞	E ∞	N 4.4	R 4.1	F 6.1 (t_3)
N 5.8	E ∞	N 5.4	N 5.1	R 4.8
N 6.2	P 6.8	E ∞	N 6.1	N 5.8

Nous avons donc les distances suivantes :

$$\left|\overrightarrow{t_2, p}\right| = 6.8$$

$$\left|\overrightarrow{t_2, t_1}\right| = 3.8$$

$$\left|\overrightarrow{t_2, t_3}\right| = 6.1$$

Finalement, nous réutilisons l'algorithme de Dijkstra à partir du troisième trésor :

N 3.4	N 3.8	N 4.8	N 4.8 (t_2)	N 3.8
N 2.4	E ∞	E ∞	F 4.5	F 4
R 1.4	R 1.9 (t_1)	R 1.7	F 2.8	F 2
E ∞	E ∞	N 2	R 1	F 0 (t_3)
N 1.4	E ∞	N 2.4	N 1.4	R 1
N 2.4	P 2.8	E ∞	N 2.4	N 2

Nous avons donc les distances suivantes :

$$\left|\overrightarrow{t_3, p}\right| = 2.8$$

$$\left|\overrightarrow{t_3, t_1}\right| = 1.9$$

$$\left|\overrightarrow{t_3, t_2}\right| = 4.8$$

Si nous essayons toutes les combinaisons possibles pour passer par tous les trésors :

Ordre des trésors	Chemin	Calcule des distances	distance
1,2,3	$\left\|\overrightarrow{p, t_1}\right\|\left\|\overrightarrow{t_1, t_2}\right\|\left\|\overrightarrow{t_2, t_3}\right\|\left\|\overrightarrow{t_3, p}\right\|$	3.8+3.8+6.1+2.8	16.5
1,3,2	$\left\|\overrightarrow{p, t_1}\right\|\left\|\overrightarrow{t_1, t_3}\right\|\left\|\overrightarrow{t_3, t_2}\right\|\left\|\overrightarrow{t_2, p}\right\|$	3.8+3.2+4.8+6.8	18.6
2,1,3	$\left\|\overrightarrow{p, t_2}\right\|\left\|\overrightarrow{t_2, t_1}\right\|\left\|\overrightarrow{t_1, t_3}\right\|\left\|\overrightarrow{t_3, p}\right\|$	7.2+3.8+3.2+2.8	17
2,3,1	$\left\|\overrightarrow{p, t_2}\right\|\left\|\overrightarrow{t_2, t_3}\right\|\left\|\overrightarrow{t_3, t_1}\right\|\left\|\overrightarrow{t_1, p}\right\|$	7.2+6.1+1.9+3.8	19
3,1,2	$\left\|\overrightarrow{p, t_3}\right\|\left\|\overrightarrow{t_3, t_1}\right\|\left\|\overrightarrow{t_1, t_2}\right\|\left\|\overrightarrow{t_2, p}\right\|$	4.2+1.9+3.8+6.8	16.7
3,2,1	$\left\|\overrightarrow{p, t_3}\right\|\left\|\overrightarrow{t_3, t_2}\right\|\left\|\overrightarrow{t_2, t_1}\right\|\left\|\overrightarrow{t_1, p}\right\|$	4.2+4.8+3.8+3.8	16.6

L'ordre le plus court est donc 1,2,3.

S'il n'est pas possible de joindre un trésor, alors votre résultat ne passera pas par ce trésor. Vous devez passer par tout les trésors joignables. Il est à remarquer que vous pouvez utiliser l'algorithme de Floyd-Warshall à la place d'utiliser Dijkstra, dans ce cas, l'algorithme ne sera démarré qu'une seule fois.

Affichage

Finalement, votre programme doit afficher l'ordre de parcours des trésors avec le cout du parcours pour chaque trésor et le cout de parcours total. Dans notre exemple, vous affichez avec le format suivant :

```
Porte -> T1 : 3.8
T1 -> T2 : 3.8
T2 -> T3 : 6.1
T3 -> porte : 2.8
Total : 16.5
```

Si plus d'un chemin est le plus court, alors il faut afficher chaque chemin, un à la suite de l'autre.

Tests

Votre application doit contenir les tests nécessaires pour démontrer son bon fonctionnement. Ce sera votre responsabilité de prouver le bon fonctionnement, donc de construire les tests nécessaires pour couvrir les utilisations de votre logiciel.

Directive

1. Le tp est à faire seul ou en équipe de deux (maximum).
2. Commentaire :
    a. Commentez l'entête de chaque fonction. Ces commentaires doivent contenir la description de la fonction et le rôle de ces paramètres.
    b. Une ligne contient soit un commentaire, soit du code, pas les deux.
    c. Utilisez des noms d'identificateur significatif.
    d. Utilisez le français.
3. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaître que dans les `switch`.
    c. Un seul `return` par méthode.
4. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.

Remise

Vous devez remettre votre code (votre programme principal, vos modules, les tests et un **makefile**. Le tout doit être archivé (.tar) et ensuite compressé (.gz) sur Malt. Remettre le tp par l'entremise de Moodle. Remettez votre fichier `'*.tar.gz'`. Le tp est à remettre avant le 28 juillet 23:59.

Évaluation

- Fonctionnalité (10 pts) : Votre tp doit compiler sans erreur (il peut y avoir des warnings). J'utilise la ligne de compilation suivante : `make`
- Structure (2 pt) : Il faut avoir **plusieurs** fonctions et des classes. Construisez un code bien structuré.
- Lisibilité (3 pts) : commentaire, indentation et noms d'identificateur significatif.