TP6 : Programmation dynamique

Le but de ce sujet est d'implanter l'algorithme du voyageur de commerce vu en cours.
Les fichiers fournis sont les suivants :

- `ensembles.cc/.h`, `matrices.cc/.h` et `points.cc/.h` pour la gestion des ensembles (*cf.* ci-dessous), des matrices et des points du plan (*cf.* ci-dessous) ;
- `tsp.cc`, **seul fichier à modifier** (accompagné de `tsp.h` à ne pas modifier), qui contiendra les implantations des algorithmes ;
- `tests.cc` et le `Makefile` pour tester les implantations effectuées.

Ensembles
On représente un ensemble S par un entier n : si n s'écrit $b_{t-1}\cdots b_0$ en binaire, alors n représente l'ensemble $\{i : b_i = 1\}$. Par exemple, 13 s'écrit 1101 en binaire, et représente donc l'ensemble $\{0,2,3\}$.
Les fichiers `ensembles.cc/.h` permettent de représenter les ensembles par des entiers. Le fichier `ensembles.h` contient une description des fonctions disponibles. *Vous pouvez regarder l'implantation pour comprendre comment gérer l'union de deux ensembles dans cette représentation, etc.*

Points dans le plan
Un point du plan est représenté par un élément de type `coord` qui est un `struct` ayant deux champs `float x` et `float y` pour représenter l'abscisse et l'ordonnée respectivement.

Exercice 1. *Exercice*

1. Calculer *à la main* les réponses aux questions suivantes, puis vérifier grâce au fichier de test : quel ensemble est représenté par l'entier 22 ? quel entier représente l'ensemble $\{1,3,5\}$? quel entier obtient-on si on rajoute l'entier 4 à l'ensemble représenté par l'entier 35 ?

Par simplicité, on implémente une **variante de l'algorithme TSP**[1] **présentée en Fig. 1 (p. 2)**. Des exemples de tests pour les questions 3 à 7 sont présentés en Fig. 3 (p. 3). Ils peuvent servir à vérifier vos résultats !

2. Compléter la fonction `float distance(coord, coord)` qui calcule la distance entre deux points.
3. Compléter le corps de la boucle `while` de la fonction `float tsp(int n, coord* P, float** D, int** Prec, int& min)` *en ignorant dans un premier temps les paramètres* `Prec` *et* `min`. On utilise la valeur `FLT_MAX` pour représenter $+\infty$; la boucle `while (U) { ... ; U = suivant(U, n-1, s); }` permet de parcourir tous les sous-ensembles $U \subset \{0,\dots,n-2\}$ de taille s.
4. Continuer à compléter la fonction `tsp` en calculant le minimum final *en ignorant toujours* `Prec` *et* `min`.
5. Finir de compléter `tsp` en remplissant `Prec` et en enregistrant l'indice du minimum final dans `min`.
6. Compléter la fonction `void tsp_rec(int* circuit, int n, float** D, int** Prec, int min)` qui reconstruit la solution optimale du problème. À ce stade, `Circuit.svg` doit contenir un chemin optimal, comme en Fig. 2.
7. *(bonus)* Implanter la fonction `float tsp_bruteforce(int n, coord* P)` qui calcule la longueur optimale en testant toutes les permutations possibles des points. *On pourra utiliser la fonction* `next_permutation` *de la bibliothèque* `algorithm` *dont une documentation est disponible à l'adresse* `https://fr.cppreference.com/w/cpp/algorithm/next_permutation`.
   Tester et comparer les temps de calculs avec la fonction `tsp`.

1. C'est un bon exercice que de comprendre pourquoi cette variante fait exactement le même calcul !

Algorithm: TSP(S)

$\Delta \leftarrow$ tableau à deux dimensions, indexé par les sous-ensembles de
    $\{0,\dots,n-2\}$ et par les sommets de s_0 à s_{n-1};
$\Delta[\emptyset, s_{n-1}] = 0$;
pour $s = 1$ *à* $n-1$ **faire**
    pour *tous les* $U \subset \{0,\dots,n-2\}$ *de taille* s **faire**
        $\Delta[U, s_{n-1}] = +\infty$;
        pour *tout* $s_j \in U$ **faire**
            $\Delta[U, s_j] = \min\{\Delta[U \setminus \{s_j\}, s_i] + \delta_{ij} : s_i \in U, i \neq j\}$;
        fin
    fin
fin
retourner $\min_j(\Delta[\{s_0,\dots,s_{n-2}\}, s_j) + \delta_{j,n-1}$;

FIGURE 1 – Variante de l'algorithme TSP.

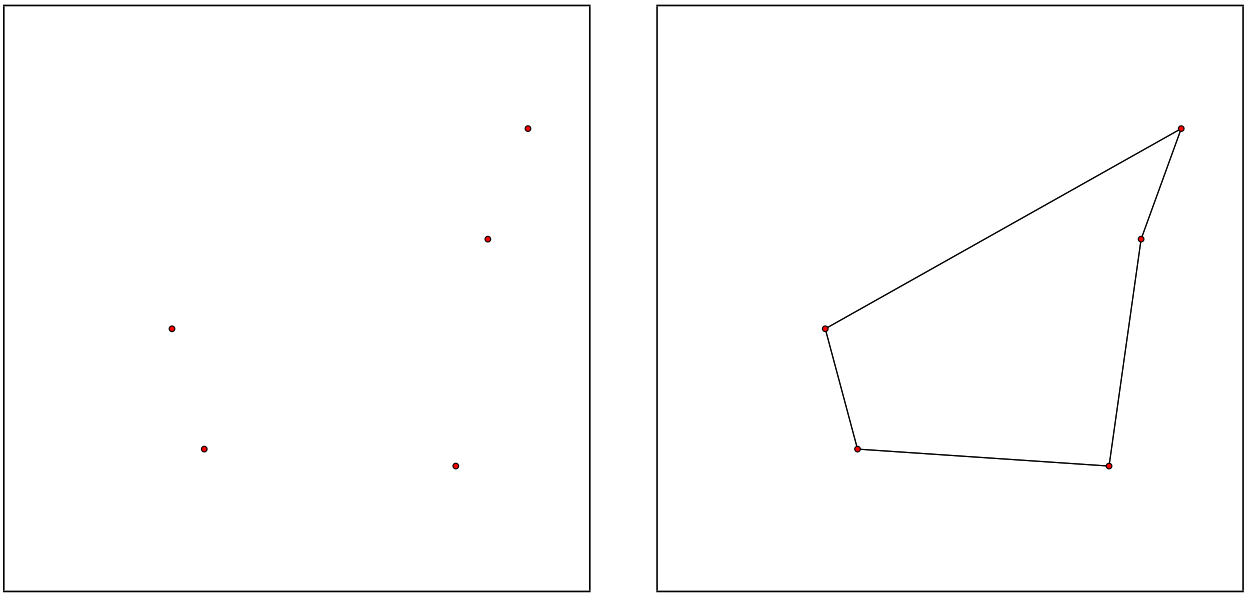

FIGURE 2 – Fichiers `Points.svg` (g) et `Circuit.svg` (d) pour 5 points, fixés.

EXEMPLE

```
Entrer le numéro de question à tester (0 pour quitter) : 3
Entrer un nombre de points (> 0) : 5
Type de points (1. fixés ; 2. aléatoires) : 1
5 points (cf Points.svg) : (840.188,394.383), (783.099,798.44), (911.648,197.551), (335.223,768.23),
    (277.775,553.97)

Contenu de la matrice D :
INF INF INF INF 0 0 4.62428e-44 0 584.617 INF INF INF INF 0 4.62428e-44 0
584.617 INF INF INF INF 0 4.62428e-44 0 INF 561.354 INF INF INF 0 4.62428e-44 0
INF 561.354 INF INF INF 0 4.62428e-44 0 969.425 992.687 INF INF INF 0 4.62428e-44 0
969.425 992.687 INF INF INF 0 4.62428e-44 0 INF INF 727.206 INF INF 0 4.62428e-44 0
INF INF 727.206 INF INF 0 4.62428e-44 0 936.608 INF 794.019 INF INF 0 4.62428e-44 0


Entrer le numéro de question à tester (0 pour quitter) : 4
Entrer un nombre de points (> 0) : 5
Type de points (1. fixés ; 2. aléatoires) : 1
5 points (cf Points.svg) : (840.188,394.383), (783.099,798.44), (911.648,197.551), (335.223,768.23),
    (277.775,553.97)

Longueur du circuit le plus court : 2015.4


Entrer le numéro de question à tester (0 pour quitter) : 5
Entrer un nombre de points (> 0) : 5
Type de points (1. fixés ; 2. aléatoires) : 1
5 points (cf Points.svg) : (840.188,394.383), (783.099,798.44), (911.648,197.551), (335.223,768.23),
    (277.775,553.97)

Longueur du circuit le plus court : 2015.4
Deuxième point du circuit : (335.223,768.23)
Contenu de la matrice Prec :
0 0 0 0 0 0 33 0 0 0 0 0 0 0 33 0
0 0 0 0 0 0 33 0 0 0 0 0 0 0 33 0
0 0 0 0 0 0 33 0 1 0 0 0 0 0 33 0
1 0 0 0 0 0 33 0 0 0 0 0 0 0 33 0
0 0 0 0 0 0 33 0 2 0 0 0 0 0 33 0


Entrer le numéro de question à tester (0 pour quitter) : 6
Entrer un nombre de points (> 0) : 5
Type de points (1. fixés ; 2. aléatoires) : 1
5 points (cf Points.svg) : (840.188,394.383), (783.099,798.44), (911.648,197.551), (335.223,768.23),
    (277.775,553.97)

Longueur du circuit le plus court : 2015.4
Circuit le plus court : 4 → 3 → 1 → 0 → 2 → 4
Circuit dessiné dans le fichier Circuit.svg


Entrer le numéro de question à tester (0 pour quitter) : 7
Entrer un nombre de points (> 0) : 5
Type de points (1. fixés ; 2. aléatoires) : 1
5 points (cf Points.svg) : (840.188,394.383), (783.099,798.44), (911.648,197.551), (335.223,768.23),
    (277.775,553.97)

Longueur du circuit le plus court :
    calculé par programmation dynamique : 2015.4
    calculé par recherche exhaustive : 2015.4
Comparaison des temps de calcul :
    programmation dynamique : 0.035 ms (dont reconstruction : 0.004 ms)
    recherche exhaustive : 0.09 ms
```

FIGURE 3 – Résultats des tests avec cinq points fixés.