Université de Bordeaux
Nom: Cyril GAVOILLE
Numéro: 100

Licence Informatique, 3e année
10/05/2020

Techniques Algorithmiques et Programmation

Devoir Maison

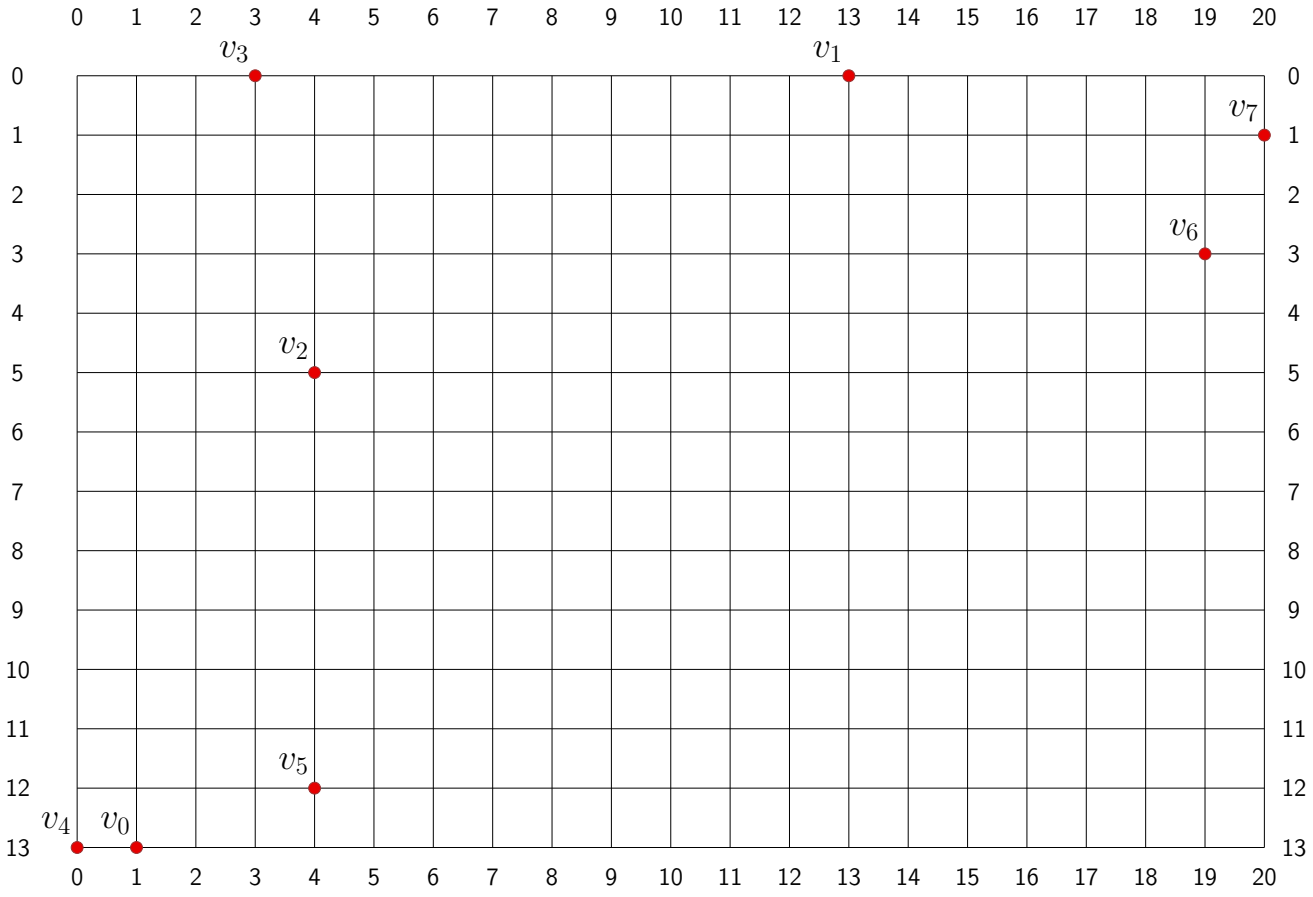
[durée: 8h, de 10h à 18h]

Toutes vos réponses sont à saisir dans le fichier `reponses.txt` accompagnant le sujet, et qui est à déposer sur la page Moodle du cours où figure également quelques consignes générales. Merci de ne pas renommer ce fichier réponses, ni de le compresser ou de changer son format. Vous pouvez mettre des commentaires directement dans `reponses.txt` si cela vous semble nécessaire. Enfin, ne cherchez pas à me rendre le sujet (.pdf) que je viens de vous envoyer.

Le sujet comporte **3** pages. Vous avez le droit d'utiliser les documents accessibles sur la page de l'UE, en particulier les notes de cours, ainsi que vos propres fichiers (notes et programmes de TP). Il s'agit d'un devoir individuel. Tout constat de manquement à cette consigne sera sanctionné.

Voyageur de commerce

On considère un ensemble de 8 points $V = \{v_0, v_1, \ldots, v_7\}$ placés sur une grille rectangulaire comme ci-dessous. La distance entre deux points de V est la distance dite de Manhattan, c'est-à-dire la longueur d'un plus court chemin qui suit la grille, chaque pas de la grille valant 1. Plus précisément, la distance entre les points $A = (x_A, y_A)$ et $B = (x_B, y_B)$ est $d(A, B) = |x_A - x_B| + |y_A - y_B|$. Notez bien que la grille ne comporte pas de diagonale et que toutes les distances sont entières.



Q1a. *Calculez $d(v_0, v_1)$.*

Q1b. *Calculez $d(v_0, v_2)$.*

Q2a. *Donnez les indices des deux points les plus proches.*

Q2b. *Calculez la distance correspondante.*

On dit qu'une suite $u_1, \ldots, u_k$ de k points sont « alignés » s'il existe un plus court chemin de u_1 à u_k passant par tous les points de la suite dans l'ordre $u_1, \ldots, u_k$.

Q3a. *Donnez une suite de trois points alignés (leurs indices).*

Q3b. *Donnez une suite de k points alignés avec le plus grand k possible. Pour simplifier la recherche, remarquez que tout plus court chemin sur la grille utilise, sur chaque dimension, des coordonnées toutes croissantes ou toutes décroissantes.*

On considère l'algorithme « du point le plus proche », un algorithme glouton vu en cours.

Q4a. *Donnez la tournée produite par cet algorithme depuis v_0, c'est-à-dire la liste des indices des points de la tournée. Il est inutile de remettre en fin de tournée le point de départ.*

Q4b. *Calculez la longueur de la tournée ainsi produite.*

La matrice des distances pour V est la matrice carrée 8×8, $M = (M_{i,j})$ où $M_{i,j} = d(v_j, v_i)$, donnant la distance entre chaque couple de points (v_j, v_i) de V. Il s'agit de calculer cette matrice. Mais comme la matrice est symétrique et que $M_{i,i} = 0$, seules certaines valeurs sont « utiles ». On notera L_i la i-ème ligne de la matrice M pour les valeurs $M_{i,j}$ avec $j < i$. On a donc :

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
L_1	$d(v_0, v_1)$						
L_2	$d(v_0, v_2)$	$d(v_1, v_2)$					
L_3	$d(v_0, v_3)$	$d(v_1, v_3)$	$d(v_2, v_3)$				
L_4	$d(v_0, v_4)$	$d(v_1, v_4)$	$d(v_2, v_4)$	$d(v_3, v_4)$			
L_5	$d(v_0, v_5)$	$d(v_1, v_5)$	$d(v_2, v_5)$	$d(v_3, v_5)$	$d(v_4, v_5)$		
L_6	$d(v_0, v_6)$	$d(v_1, v_6)$	$d(v_2, v_6)$	$d(v_3, v_6)$	$d(v_4, v_6)$	$d(v_5, v_6)$	
L_7	$d(v_0, v_7)$	$d(v_1, v_7)$	$d(v_2, v_7)$	$d(v_3, v_7)$	$d(v_4, v_7)$	$d(v_5, v_7)$	$d(v_6, v_7)$

Q5. *Calculez chacune des lignes $L_1, \ldots, L_7$.*

Il s'agit maintenant de construire un arbre couvrant de poids minimum pour V, plus précisément du graphe complet induit par V. Vous pouvez pour cela appliquer la méthode que vous souhaitez, en vous appuyant sur la matrice des distances que vous venez de calculer.

Q6a. *Donnez la liste des arêtes d'un arbre couvrant de poids minimum.*

Q6b. *Calculez le poids de cet arbre.*

Q6c. *Parmi les arêtes du graphe complet induit par V, combien n'appartiennent pas à l'arbre ?*

Q6d. *Donnez l'arête de poids minimum du graphe complet à ne pas être dans l'arbre. (En choisir une s'il y en a plusieurs.) Notez que lorsqu'on applique l'algorithme de Kruskal, c'est la première arête à former un cycle.*

On s'intéresse maintenant aux tournées produites par l'algorithme de 2-approximation vu en cours.

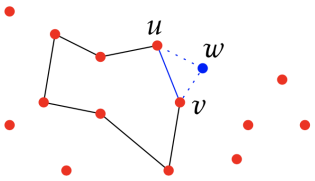
Q7a. *Donnez la tournée obtenue en effectuant un parcours en profondeur (DFS) depuis la racine v_0. Important : s'il y a un choix, votre parcours devra choisir le point voisin ayant le plus petit indice.*

Q7b. *Calculez la longueur de cette tournée.*

On s'intéresse maintenant à un autre algorithme produisant une tournée pour V. Il s'agit de la variante de l'heuristique dite d'« insertion aléatoire » présentée brièvement dans la section 3.4.3 des notes de cours (d'où a été copiée l'illustration ci-après).

Le principe est le suivant : on démarre avec une tournée initiale $P = v_0 - v_1 - v_2$, définie sur ces trois premiers points. Puis, pour chacun des points extérieurs à la tournée courante P, on « insère » le point w entre deux points consécutifs $u - v$ de P (voir la figure ci-après). Cela crée ainsi une nouvelle tournée avec un point de plus où l'arête $u - v$ de P a été remplacée par les arêtes $u - w$ et $w - v$.

Le point important, pour cette variante qu'on nommera « insertion minimum », est que le point w et l'arête $u - v$ de P sont choisis de façon à minimiser l'accroissement de la longueur de la tournée. On s'arrête bien évidemment lorsque tous les points on été ainsi insérés.



Q8a. *Donnez une tournée pour V obtenue par insertion minimum.*

Q8b. *Calculez la longueur de cette tournée.*

FIN.