N° d'anonymat:

Année Universitaire 2021-2022
Session 2 de printemps

Parcours/Étape: L3 Informatique **Code UE:** 4TIN602U
Épreuve: Techniques Algorithmiques et Programmation
Date: 19/04/2022 **Heure:** 11h30 **Durée:** 1h30
Documents: une seule feuille A4 recto-verso autorisée.
Épreuve de M. Cyril Gavoille

université de BORDEAUX

Collège Sciences et Technologie

RÉPONDRE DIRECTEMENT SUR LE SUJET
QUI EST À RENDRE DANS LA FEUILLE DOUBLE D'EXAMEN

Questions de cours

Pour chacune des questions suivantes, vous devez indiquer si l'affirmation est « vraie » ou « fausse » tout en justifiant votre réponse par une explication, un exemple ou un contre-exemple. Sans justification, pas de point.

Question 1. *Une instance particulière d'un problème indécidable peut être résolue par un programme* `C`.

Question 2. *Un algorithme pour un problème donné peut boucler à l'infini sur certaines de ses instances.*

Question 3. *Pour un problème donné, une heuristique a toujours une complexité en temps plus faible qu'un algorithme exact.*

Question 4. *L'heuristique h dans A* peut être monotone et sous-estimer toutes les distances.*

À propos du voyageur de commerce

Dans cette partie on supposera que pour le voyageur de commerce les points sont des points du plan et que la distance d entre deux points est la distance euclidienne.

Question 5. *Rappelez en quelques lignes le principe de l'algorithme* ApproxMST *vu en cours permettant de calculer une tournée pour toute entrée (V, d) du voyageur de commerce.*

Question 6. *Donner le facteur d'approximation de l'algorithme* ApproxMST*.*

Question 7. *Ce facteur d'approximation est-il valable pour d'autres fonctions de distance $d' \neq d$, c'est-à-dire différente de la distance euclidienne ? Justifiez.*

On considère maintenant un ensemble V_7 particulier contenant 7 points, $V_7 = \{v_0, v_1, \ldots, v_6\}$, placés aux sommets d'un hexagone régulier de côté unité avec un point placé au centre. Plus précisément, on construit V_7 à partir d'un point arbitraire v_0, en traçant un cercle de centre v_0 et de rayon unité, et en plaçant successivement et de manière régulière sur le cercle les 6 points $v_1, v_2, \ldots, v_6$ dans cet ordre. Il est important de remarquer que chaque triplet de points $(v_0, v_i, v_{(i \bmod 6)+1})$, $1 \leqslant i \leqslant 6$, forme un triangle équilatéral.

Question 8. *Dessinez les points de V_7 ainsi qu'une tournée de longueur optimale pour (V_7, d). Prenez ce que vous voulez comme unité. On demande bien sûr un dessin approximatif.*

Dans la suite on notera $\text{OPT}(V_7, d)$ la longueur optimale de la tournée pour (V_7, d).

Question 9. *Calculez la longueur théorique de votre tournée et prouvez quelle est de longueur optimale, c'est-à-dire de longueur* $\text{OPT}(V_7, d)$.

Question 10. *Donnez le principe d'un algorithme permettant de calculer un arbre couvrant de poids minimum.*

Question 11. *Donnez deux arbres (différents) couvrant V_7 et qui soient de poids minimum. Dessinez les et précisez leurs poids.*

Étant donné un arbre T couvrant V_7 et un point $v_i \in V_7$, on note $\ell(T, v_i)$ la longueur de la tournée pour (V_7, d) obtenue en visitant tous les points de V_7 selon l'ordre de leur première visite lors d'un parcours en profondeur d'abord quelconque de T à partir du point v_i. Attention ! Il y a plusieurs parcours possibles de T depuis v_i, notamment si v_i à plusieurs voisins dans T. La longueur $\ell(T, v_i)$ représente donc la tournée obtenue selon le pire des parcours en profondeur possibles de T en partant de v_i.

Question 12. *Montrez qu'il existe un arbre T tel que $\ell(T, v_0) = \text{OPT}(V_7, d)$, c'est-à-dire un arbre T couvrant V_7 tel que l'ordre de première visite de T depuis v_0 selon n'importe quel parcours en profondeur d'abord donne toujours une tournée de longueur optimale. Dessinez T et la tournée correspondante. Précisez l'ordre de visite des points. Justifiez.*

On admettra sans preuve que la plus grande diagonale d'un losange obtenu en collant deux triangles équilatéraux de côté unité, vaut $\sqrt{3} > 3/2$.

Question 13. *Montrez qu'il existe un arbre T' de poids minimum tel que $\ell(T', v_0) \geqslant 8 + 2\sqrt{3}$. Justifiez.*

Question 14. *En déduire qu'il existe une instance (V, d) pour laquelle l'algorithme ApproxMST a un facteur d'approximation $> 11/7$. Justifiez.*

Le nombre de sous-ensembles

On s'intéresse à certains sous-ensembles de $\{1, \ldots, n\}$ où $n \geqslant 1$ est un entier. Plus précisément, on s'intéresse à tous ceux ayant exactement k éléments, avec bien sûr $0 \leqslant k \leqslant n$. On notera $b(n, k)$ leur nombre. Dit autrement, $b(n, k)$ représente le nombre de sous-ensembles de $\{1, \ldots, n\}$ à k éléments.

Question 15. *Donnez tous les sous-ensembles de* $\{1, 2, 3, 4\}$ *à* 2 *éléments. En déduire* $b(4, 2)$.

Question 16. *En remarquant que dans un graphe complet sur* n *points (ou clique à* n *sommets), il y a autant d'arêtes que de paires de points, donnez une formule close pour* $b(n, 2)$.

Afin d'établir une formule de récurrence pour $b(n, k)$, on remarque qu'il y a deux catégories de sous-ensembles de $\{1, \ldots, n\}$ à k éléments, en supposant $0 < k < n$:

- Ceux qui possèdent n : il y en a exactement $b(n-1, k-1)$, car pour obtenir un sous-ensemble de cette catégorie il suffit de prendre un sous-ensemble de $\{1, \ldots, n-1\}$ à $k-1$ éléments et d'y ajouter n.
- Ceux qui ne possèdent pas n : il y en a exactement $b(n-1, k)$, car tout sous-ensemble de $\{1, \ldots, n-1\}$ à k éléments appartient à cette catégorie.

Pour tous les autres cas, c'est-à-dire lorsque $k = 0$, $n = 1$ ou $n = k$, on peut vérifier facilement que $b(n, k) = 1$.

Question 17. *En vous appuyant sur la discussion précédente, écrire en* `C` *une fonction récursive* `long b_rec(int n, int k)` *renvoyant l'entier* $b(n, k)$ *pour des entiers* n, k *tels que* $n \geqslant 1$ *et* $0 \leqslant k \leqslant n$.

Question 18. *Construisez l'arbre des appels pour* `b_rec(4,2)` *en observant qu'il possède* 6 *feuilles.*

On admettra sans preuve que $b(n,k) = \Theta(2^n/\sqrt{n})$, lorsque $k = \lfloor n/2 \rfloor$.

Question 19. *En analysant le nombre de sommets de l'arbre des appels, en particulier son nombre de feuilles, montrez que la complexité de* `b_rec(n,k)` *peut-être exponentielle en* n.

Question 20. *Montrez que pour chaque* n *suffisamment grand, l'arbre des appels de* `b_rec(n,n/2)` *contient plusieurs fois les mêmes nœuds internes, c'est-à-dire des sous-appels qui ne sont pas des feuilles et avec les mêmes paramètres.*

Pour supprimer les calculs inutiles de `b_rec(n,k)`, et être beaucoup plus efficace, on va utiliser la technique de mémorisation (ou de programmation dynamique).

Question 21. *Écrire en* `C` *une nouvelle fonction non-récursive* `b_prog_dyn(n,k)` *renvoyant* $b(n,k)$ *à l'aide d'un algorithme basé sur la programmation dynamique, c'est-à-dire utilisant une table de mémorisation. (Vous pouvez aussi vous contenter de présenter, avec suffisamment de détails, le principe de votre algorithme.)*

Question 22. *Quelles sont les complexités en temps et en espace de votre fonction* `b_prog_dyn(n,k)` *exprimée en fonction de* n *et* k *?*

FIN.