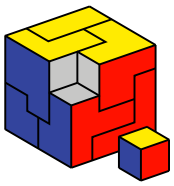
Prologin 2020



Concours national d'informatique

Épreuve écrite d'algorithmique

Soirée détente

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (4 heures).

Conseils

— Lisez bien tout le sujet avant de commencer.
— **Soignez la présentation** de votre copie.
— N'hésitez pas à poser des questions.
— Si vous avez fini en avance, relisez bien, ou préparez votre présentation pour l'entretien.
— N'oubliez pas de passer une bonne journée.

Remarques

— Le barème est donné à titre indicatif uniquement.
— Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
— Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
— Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe, sinon ça va barder.
— Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
— Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 Introduction

Comme chaque soir, vous regardez votre émission de télé préférée. Nécessairement, vous aurez besoin de remplir votre bol de chips, prendre une boisson au frigo et caresser le chat [1]. Vous souhaitez à tout prix éviter de manquer quelque chose, d'où votre obsession pour optimiser votre parcours tel que la distance totale soit minimisée.

On appelle dans ce sujet [2] un circuit hamiltonien un parcours qui commence à votre fauteuil, finit à votre fauteuil, et qui passe par tous les autres endroits par lesquels vous voulez passer (par exemple, le frigo, les chips et le chat) exactement une fois. De plus, on suppose que vous allez toujours en ligne droite d'un endroit à un autre. On supposera que chaque endroit est représenté par un point dans le plan. La longueur d'un circuit hamiltonien est alors défini comme la somme des longueurs de chaque segment.

Par exemple, la figure 1a est un cycle hamiltonien. En effet, on commence au fauteuil (le point O), puis on va aux chips (le point A), pour ensuite aller vers le chat (B) puis le frigo (C) et à la fin se rasseoir sur le fauteuil (O). Cependant, la figure 1b ne l'est pas, puisqu'on ne revient pas au fauteuil.

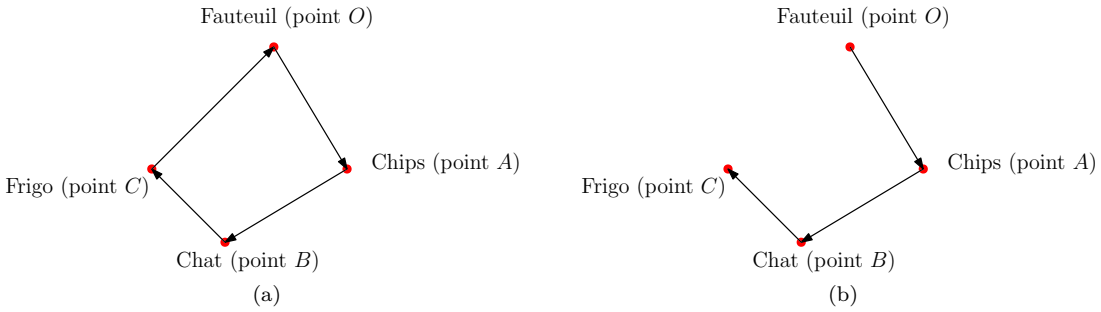


FIGURE 1 – (a) est un cycle hamiltonien tandis que (b) n'en est pas

3 Familiarisation avec les cycles hamiltoniens

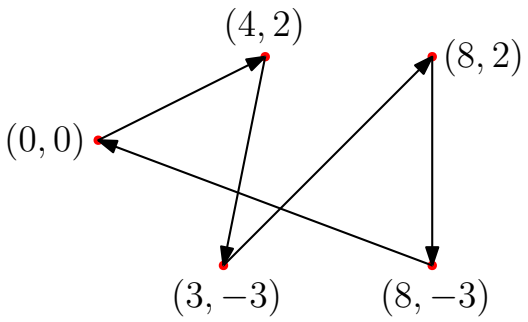


FIGURE 2

Question 1 (1 point)

> Quelle est la longueur du cycle hamiltonien dans la figure 2 ?

Question 2 (1 point)

> Quel est le cycle hamiltonien de longueur minimale dans la figure 2 ? Quelle est cette longueur ?

Question 3 (2 points)

> Si on désire minimiser la distance du cycle hamiltonien, pourquoi est-il autorisé de seulement prendre en considération les cycles hamiltoniens qui ne passent pas plusieurs fois par un point ?

1. Ou donner un coup de pied au chat, selon ce que vous avez mangé à midi.
2. Il s'agit d'une définition informelle, bien entendu.

En vue de la question précédente, on supposera qu'un circuit hamiltonien passe exactement une fois par chaque point. De plus, on remarque que l'endroit où on commence n'a que peu d'importance, seul l'ordre des points qu'on visite a une importance. Par exemple, on obtient la même longueur si on commence au point C, pour ensuite faire le cycle $C \to O \to A \to B \to C$. Aussi, le sens de parcours d'un cycle hamiltonien pour un ordre donné n'est pas important, puisque qu'on obtient la même longueur. En effet, la longueur de $O \to A \to B \to C \to O$ est la même que pour $O \to C \to B \to A \to O$.

Nous pouvons maintenant définir une manière de représenter un cycle hamiltonien. On peut représenter un cycle hamiltonien comme une permutation des points à visiter, c-à-d comme une suite de points où chaque point à visiter est présent exactement une seule fois. Par exemple, dans l'exemple du haut, le cycle hamiltonien peut être représenté comme (O, A, B, C). Cette permutation nous indique que le cycle commence en O, puis va vers A, vers B, vers C pour revenir vers O au final.

Pour la suite, on va représenter les n points d'un cycle avec des indices, i.e. s'il y a n points au total qu'il faut visiter, chaque point est représenté d'un indice $0, 1, \cdots, n-1$.

Question 4 (2 points)

> Écrivez une fonction `validPermutation(perm)` qui vérifie si le tableau `perm` est une permutation valide des nombres $0, 1, \cdots, n-1$, où n est le nombre d'éléments dans `perm`.

Question 5 (1 point)

> Analysez la complexité temporelle et spatiale de votre algorithme.

Question 6 (2 points)

> Écrivez une fonction `distance(permutation, coordonnees)` qui calcule la longueur du cycle eulérien correspondant. L'argument `permutation` représente la permutation des indices des points qu'il faut visiter, et `coordonnées` représente un tableau qui contient pour chaque point les coordonnées dans le plan.

Question 7 (1 point)

> Analysez la complexité temporelle et spatiale de votre algorithme.

Au vue des remarques précédentes, le cycle hamiltonien $(0, 3, 2, 5, 4, 1)$ peut être représenté de plusieurs manières différentes. Comme le sens de parcours peut être ignoré, $(0, 1, 4, 5, 2, 3)$ est aussi une représentation du même cycle hamiltonien. De plus, en utilisant le fait que pour le même cycle hamiltonien, on peut commencer à différents points, on a que $(3, 2, 5, 4, 1, 0)$ est aussi une représentation valide. Finalement, en combinant les deux observations, $(3, 0, 1, 4, 5, 2)$ est aussi valide. Comme autre exemple, le cycle hamiltonien dans la figure 1a peut être représenté avec une des permutations suivantes :

— (O, A, B, C),
— (A, B, C, O),
— (B, C, O, A),
— (C, O, A, B),
— (O, C, B, A),
— (A, O, C, B),
— (B, A, O, C), et
— (C, B, A, O).

A l'aide de ces observations, on peut définir que deux permutations qui représentent un cycle hamiltonien sont équivalentes si elles représentent le même cycle. Le but des prochaines questions est de déterminer si deux permutations sont équivalentes ou pas.

Question 8 (3 points)

> Pour commencer, écrivez une fonction qui, étant donné une permutation de $0, 1, 2, \cdots, n-1$ retourne une permutation équivalente qui commence par 0. Analysez la complexité temporelle et spatiale.

Question 9 (3 points)

> Écrivez une fonction qui pour une permutation de $0, 1, 2, \cdots, n-1$ avec $n \geq 3$ qui commence par un 0 retourne une permutation équivalente tel que le deuxième nombre de la permutation est plus petit que le dernier. Analysez la complexité temporelle et spatiale.

Dans la suite du sujet, on appelle forme normale d'une permutation qui représente un cycle hamiltonien une permutation de $0, 1, 2, \cdots, n-1$ qui commence avec 0 et, si la permutation a au moins 3 éléments, où le deuxième est plus petit que le dernier.

Question 10 (2 points)

> A l'aide des deux questions précédentes, écrivez un algorithme qui détermine si deux permutations de $0, 1, 2, \cdots, n-1$ sont équivalentes ou pas. Analysez la complexité temporelle et spatiale.

4 Générer aléatoirement des cycles hamiltoniens

Le but de cette partie est de générer une forme normale d'une permutation à partir d'une pièce pipée [3]. Dans un premier temps, on va construire une permutation aléatoire de $0, 1, 2, \cdots, n-1$ pour ensuite utiliser les questions de la partie précédente pour obtenir la forme normale du cycle hamiltonien correspondant.

Question 11 (3 points)

> Écrivez un algorithme qui retourne une permutation de $0, 1, 2, \cdots, n-1$. Vous avez à disposition une fonction `rand(k)` qui, pour un entier naturel k donné, retourne un entier $0 \leq x < k$ de manière uniforme et indépendante. Quelle est la complexité temporelle de votre algorithme ?

Question 12 (3 points)

> Écrivez un algorithme qui, étant donné un entier naturel k, retourne un entier $0 \leq x < k$ de manière indépendante et uniforme. Vous avez à disposition une pièce qui retourne pile de manière indépendante avec probabilité $\frac{1}{2}$. Quelle est la complexité temporelle de votre algorithme ?

Question 13 (3 points)

> Écrivez un algorithme qui retourne pile ou face de manière indépendante et uniforme [4]. Vous avez à disposition une pièce pipée qui retourne pile de manière indépendante avec probabilité $0 < p < 1$. Quelle est la complexité temporelle de votre algorithme ?

Question 14 (2 points)

> Quel est le nombre de cycles hamiltoniens pour n points ?

Question 15 (1 point)

> Justifiez que prendre la forme normale d'une permutation générée d'une manière uniforme résulte dans une forme normale d'une permutation qui représente un cycle hamiltonien générée d'une manière uniforme.

5 Algorithmes exacts

Trouver le cycle hamiltonien de longueur minimale pour des points dans le plan s'appelle le *problème du voyageur de commerce euclidien* que l'on abrège eTSP [5]. Cette partie propose de trouver des algorithmes exacts pour ce problème.

Question 16 (5 points)

3. C'est-à-dire qui retourne face avec une probabilité différente que pile.
4. Il est nécessaire que la probabilité soit *exactement* $\frac{1}{2}$, les approximations ne sont pas acceptées.
5. L'abréviation provient du nom anglais, le *euclidean travelling salesman problem*.

> Écrivez un algorithme, qui étant donné une liste de points dans le plan retourne la longueur minimale d'un cycle hamiltonien. Analysez la complexité temporelle et spatiale. Dans cette question, la rapidité de l'algorithme n'est pas prise en compte.

Question 17 (10 points)

> Écrivez un algorithme, qui étant donné une liste de points dans le plan retourne la longueur minimale d'un cycle hamiltonien en $O(n^2 \cdot 2^n)$ [6]. Quelle est la complexité spatiale de votre algorithme ?

6 Algorithmes d'approximation, première partie

Malheureusement, il n'existe pas à ce jour d'algorithmes rapides [7] pour ce problème. Une des plus grandes conjectures en informatique théorique dit qu'il n'existe pas d'algorithme rapide pour ce problème [8].

Cependant, il est courant d'utiliser des algorithmes d'approximation dans ce cas. Un algorithme d'approximation est un compromis entre rapidité et qualité du résultat. Dans le contexte de ce problème, un algorithme ne va pas forcément trouver la longueur minimale d'un cycle hamiltonien, mais la longueur d'un cycle hamiltonien court [9].

Un algorithme ρ-approché pour le travelling salesman problem est un algorithme qui pour chaque instance possible, retourne la longueur d'un cycle hamiltonien qui est au plus ρ fois aussi long que un cycle hamiltonien optimal.

Question 18 (1 point)

> Pourquoi ne peut-il pas y avoir un algorithme ρ-approché avec $\rho < 1$ pour le travelling salesman problem ?

Le but de cette partie et de la suivante est d'obtenir un algorithme 2-approché. Pour cela, un arbre couvrant minimal est utilisé qui est introduit dans cette partie.

Définissons en premier le concept de graphe euclidien. Un graphe euclidien consiste en un ensemble de sommets V qui est sous-ensemble fini de points du plan, formellement, on a que $V \subset \mathbb{R}^2$. De plus, chaque paire de sommets différents représente une arête non orientée, pondérée par la distance des deux points correspondants. Par exemple, dans la figure 3, on a $V = \{(0,0), (1,2), (2,1), (-4,3)\}$. Les sommets sont représentés par les points et les arêtes par des segments. Entre tous les sommets, il y a une arête, qui a un nombre réel inscrit à côté, il s'agit de sa pondération, qui correspond à la longueur du segment. Si on note E l'ensemble des arêtes, on peut noter $G = (V, E)$ le graphe.

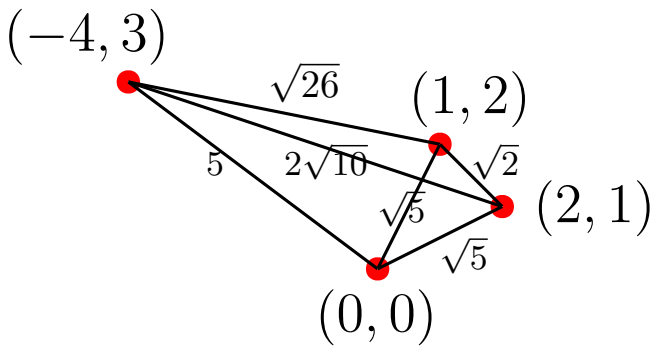
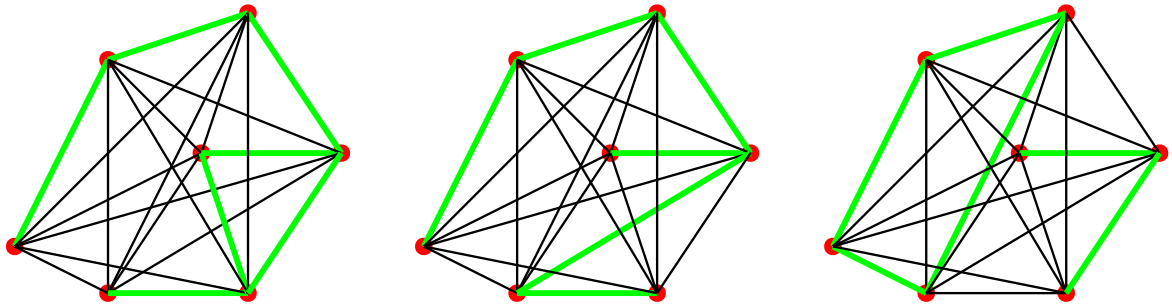


FIGURE 3 – Exemple d'un graphe euclidien

En fait, tout au long du sujet, on a travaillé sur des graphes euclidiens. Nous pouvons maintenant définir le eTSP de manière formelle. Étant donné un graphe euclidien, il faut trouver une permutation σ des sommets $v_1, v_2, v_3, \cdots, v_n$ telle que le réel $d(v_{\sigma_1}, v_{\sigma_2}) + d(v_{\sigma_2}, v_{\sigma_3}) + \cdots + d(v_{\sigma_{n-1}}, v_{\sigma_n}) + d(v_{\sigma_n}, v_{\sigma_1})$ est minimisé où la fonction d est la distance euclidenne [10].

6. Indice : programmation dynamique.
7. En temps polynomial pour être exact.
8. C'est en fait une formulation équivalente de la conjecture NP $\neq$ P.
9. Pour une certaine notion de *court*.
10. On rapelle que la distance euclidienne entre deux points (x_1, y_1) et (x_2, y_2) est $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Pour un graphe euclidien avec n sommets, un arbre couvrant est un ensemble de $n-1$ arêtes tel que entre chaque paire de sommets, il existe une suite d'arêtes dans l'ensemble que l'on peut parcourir pour aller du premier sommet jusqu'au deuxième.



(a) (b) (c)

FIGURE 4

Question 19 (2 points)

> Est-ce que les arêtes vertes qui sont mises en gras de la figure 4a forment un arbre couvrant ? Qu'en est-il pour la figure 4b ? Et la figure 4c ?

Pour chaque arbre couvrant, on associe un coût, qui est la somme des pondérations pour chaque arête dans l'ensemble. Il est souvent utile de vouloir minimiser ce coût. Un arbre couvrant de coût minimal est appellé un arbre couvrant minimal. Le but est d'obtenir un algorithme qui trouve un arbre couvrant minimal en temps polynomial.

Question 20 (4 points)

> Supposons que vous avez un arbre couvrant minimal pour un sous-ensemble S des sommets V. Construisez un arbre couvrant minimal pour les sommets dans S et un autre sommet dans V. Montrez que votre algorithme est correct.

Question 21 (6 points)

> À l'aide de la question précédente, proposez un algorithme qui trouve un arbre couvrant minimal pour un graphe euclidien avec n sommets en $O(n^c)$ pour une constante $c > 0$.

Question 22 (3 points)

> Montrez que l'algorithme précédent est correct, que sa complexité temporelle est bien en $O(n^c)$ et analysez sa complexité spatiale.

7 Algorithmes d'approximation, deuxième partie

Question 23 (7 points)

> En utilisant la partie d'avant, proposez un algorithme 2-approché rapide [11] pour le eTSP [12]. Analysez la complexité temporelle et spatiale de votre algorithme.

Question 24 (3 points)

> Montrez que votre algorithme est bien un algorithme 2-approché.

8 Partie bonus

Les questions suivantes sont plus dures que le reste du sujet. Il est fortement conseillé de finir les autres questions avant.

11. C'est-à-dire en temps polynomial.
12. Indice : parcourez l'arbre couvrant minimal d'une manière astucieuse.

Question bonus 25 (10 points)

> Supposons que les coordonnées des points sont entières et qu'on ne considère plus la distance euclidienne entre deux points mais la distance de Manhattan [13].
>
> Trouvez un algorithme $\frac{3}{2}$-approché en temps polynomial pour trouver la longueur du cycle hamiltonien de longueur minimale. Analysez la complexité temporelle et spatiale, et montrez qu'il s'agit bien d'un algorithme $\frac{3}{2}$-approché.
>
> Vous pouvez utiliser le fait qu'il est possible de trouver un couplage parfait de coût minimal en temps polynomial dans un graphe complet avec des pondérations des arêtes entières.

Question bonus 26 (6 points)

> Supposez que vous voulez visiter tous les sommets d'un polygone régulier avec n sommets inscrit dans un cercle de rayon $r > 0$. Quelle est l'espérance de la longueur du cycle hamiltonien, si vous sélectionner un cycle hamiltonien de manière uniforme ?

Question bonus 27 (42 points)

> Montrez que $\mathsf{P} \neq \mathsf{NP}$.

Question bonus 28 (1 point)

> Proposez un sujet écrit pour la prochaine épreuve régionale.

Question bonus 29 (-1 point)

> Quel est votre organisateur Prologin favori [14] ?

13. I.e., $d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$.
14. Indice : essayez de deviner l'organisateur qui aura la chance inouïe d'avoir le privilège de corriger votre copie.