Prologin 2010

# Concours National d'Informatique

## Sujet de demi-finale Paris II / Lille

27 Février 2010

# Norsk skogkatt

## 1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre demi-finale. 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.
- 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 la demi-finale et la date, en haut de votre copie.
- Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le nous, nous pouvons ajouter des questions plus difficiles.
- Tous les langages sont autorisés. Néanmoins, veuillez préciser celui que vous utilisez.
- Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
- 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.

# 2 Sujet

## Introduction

Un chat, ça aime bien grimper aux arbres et sauter de branches en branches. Surtout un puissant chat des forêts norvégiennes[1] ! Mais un chat, c'est toujours un peu paresseux. Ne comptez pas sur ces partisans du moindre effort pour se fatiguer plus que nécessaire !

Chaque saut qu'il effectue lui demande un certain effort, en fonction de sa longueur, de sa hauteur, de s'il monte ou s'il descend...De plus, pour le chat l'effort global pour effectuer un trajet est égal au *maximum* des efforts nécessaires pour effectuer tous les sauts qui le composent. Le but de ce problème est de résoudre le *problème du chat*, à savoir calculer l'effort minimal qu'il doit fournir pour aller entre deux branches.

Pour simplifier, modélisons l'ensemble des branches sur lesquelles il peut sauter par un ensemble de points dans l'espace. Chaque point est représenté par ses trois coordonnées $(x, y, z)$ où $z$ est l'altitude du point et où le plan $(x, y)$ est parallèle au sol. On considère donc que le chat peut sauter de points en points, et qu'il commence toujours à partir de la branche numéro 0.

### Question 1 (2 points)

Proposer une ou des structures de données pour stocker l'ensemble des branches, et pour stocker la description d'un trajet de branches en branches.

Vous pourrez utiliser la constante `N` pour le nombre total de branches, et supposer que toutes les coordonnées sont entières. Pour fixer les idées, vous pouvez considérer que `N` est plus petit que 1000.

Attention, cette question est importante car vous utiliserez votre solution pour écrire les fonctions des questions suivantes. Relisez donc entièrement le sujet (et les questions qui suivent) avant de répondre.

### Question 2 (2 points)

Spécifions un peu l'effort que doit fournir notre chat norvégien pour sauter d'un point $A(x_0, y_0, z_0)$ à un point $B(x_1, y_1, z_1)$. Si le point $B$ est plus bas que le point $A$, l'effort qu'il doit fournir est égal à la distance entre les deux points. Si en revanche $B$ est plus haut que $A$, l'effort à fournir est égal à la distance entre les deux points additionné de la différence de hauteur. Écrivez donc une fonction `effort(x0, y0, z0, x1, y1, z1)` retournant l'effort à fournir pour sauter de $A$ vers $B$.

Notez que les algorithmes demandés dans la suite du problème ne doivent pas être spécifiques à cette fonction `effort` et doivent fonctionner quelque soit le chat.

### Question 2 (2 points)

Écrivez une fonction qui prend en entrée l'ensemble des branches et la description d'un trajet, et qui renvoie l'énergie (l'effort) que le chat a dû dépenser pour effectuer ce trajet.

### Question 3 (3 points)

Imaginons le temps d'une question que notre chat veuille se promener sans but particulier. Il part donc de la branche numéro 0 et se déplace ensuite suivant l'algorithme suivant : lorsqu'il est sur une branche, il choisit comme prochaine branche la branche qui lui demande le moins d'effort, et qui n'a pas été visitée auparavant, jusqu'à retourner sur sa branche de départ. Le seul cas où il retourne sur une branche déjà visitée est donc quand il revient sur sa branche de départ.

[1] Norsk skogkatt

Écrivez une fonction qui renvoie l'énergie (l'effort) que le chat a dû dépenser pour faire ce trajet, en fonction des positions des branches.

## Question 4 (4 points)

Vous remarquerez qu'avec le mode de déplacement du chat de la question précédente, il est possible que le chat ait à fournir un très gros effort au cours du trajet ! En fait, le chat refuse de dépenser plus d'énergie qu'une constante $E$.

Écrivez une fonction qui renvoie l'ensemble des branches qui lui sont accessibles à partir de sa branche de départ.

Écrivez également une fonction qui renvoie l'ensemble des branches qui lui sont accessibles et desquelles il peut revenir à sa branche de départ ! Donnez un contre-exemple qui montre que cet ensemble de branches et différent de celui renvoyé par la fonction précédente.

## Question 5 (4 points)

Cette question vous propose de mettre au point un premier algorithme pour résoudre le *problème du chat*, à savoir aller vers la branche de son choix en déployant le moins d'effort possible.

En fait, il vous suffit d'utiliser intelligemment la première fonction que vous avez écrite dans la question précédente pour plusieurs valeurs de $E$ que votre algorithme choisira astucieusement.

Écrivez donc une fonction qui renvoie l'effort minimal qu'il doit déployer pour aller vers la branche de son choix, et estimez grossièrement le nombre d'opérations effectuées par votre algorithme en fonction de `N` et de `C`. `C` est l'effort maximal (tel que renvoyé par la fonction `effort`) nécessaire pour sauter entre deux branches. Des réponses à cette questions sont possibles sans utiliser la constante `C`.

Le barème de cette question valorise les algorithmes les plus rapides.

## Question 6 (5 points)

Nous allons à partir de cette question chercher un algorithme plus rapide pour résoudre le *problème du chat*.

### Partie 1 (3 points)

Notre algorithme va reposer sur une structure de données permettant de maintenir une partition de l'ensemble des branches. Un partition est un ensemble d'ensembles de branches $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ vérifiant les conditions suivantes :

- Chaque $C_i$ est un ensemble de branches ;
- Une branche ne peut pas se trouver dans deux ensembles $C_i$ et $C_j$ en même temps ;
- Chaque branche est dans un ensemble.

On veut pouvoir effectuer efficacement les opérations suivantes sur $\mathcal{C}$ :

- Construction : construire `N` ensembles $C_1, C_2, \ldots, C_N$ tels chaque $C_i$ ne contienne que la branche numéro $i$ ;
- Trouver : étant donné une branche de numéro $j$, trouver le numéro de l'ensemble auquel elle appartient ;
- Fusionner : fusionner (faire l'union de) deux ensembles $C_a$ et $C_b$.

Vous remarquerez que les trois opérations décrites conservent les propriété de la partition.

Mettez au point une structure de données permettant de faire ces trois opérations efficacement[2], et estimez grossièrement le nombre d'opération effectuées si l'on effectue un seul appel à Construction, au plus `K` appels à Trouver et `N` appels à Fusionner, en fonction de `N` et `K`.

[2]Pour votre information, ce problème s'appelle *Union-find*

**Partie 2** (2 points)

Utilisez ce que vous venez de faire pour résoudre le *problème du chat*. Pour cela, réfléchissez calmement à ce que vous avez fait dans la question 5 et essayez de l'améliorer en utilisant la structure de donnée décrite précédemment. Analysez enfin grossièrement le nombre d'opérations effectuées par votre algorithme et comparez avec la question 5.

## Questions bonus

Ces questions peuvent vous rapporter des points seulement si vous avez répondu juste à toutes les questions précédentes.

### Question bonus 1

Supposons maintenant que le chat ne commence pas toujours à la branche numéro zéro. Avec ce que nous venons de faire, il faut relancer l'algorithme à chaque fois que le point de départ du chat change. Supposons que le chat veuille se déplacer entre `Q` paires de branches, avec `Q` pouvant être très grand. Avez-vous un meilleur algorithme à proposer que de relancer `Q` fois l'algorithme de la question 6, avec éventuellement une phase de pré-calcul[3] ?

### Question bonus 2

Devinez le nom du chat en photo[4]. Réponses acceptées par email après la demi-finale :)

Si vous avez fini, vérifiez qu'il n'y a pas d'erreur dans ce que vous avez fait, et demandez une nouvelle question !

[3] Indice : notre meilleure complexité en utilisant des algorithmes usuels est de `N` * Log(`N`) + `Q` * Log(`N`)

[4] Indice : `md5(nom_du_chat) = 094c688da065dc5daa8a1dece0a2dde9`