CS Games 2015 - Épreuve de parallélisme

Jean-Pierre Deschamps, Félix-Antoine Ouellet

14 Mars 2015

1 Mise en contexte

Le défi consistera à coder une application pouvant se connecter à un serveur simple et résoudre correctement des problèmes envoyés par celui-ci. Des points seront accordés pour les bonnes réponses tandis que de mauvaises réponses entraineront des pertes de points

Puisqu'il s'agit d'une épreuve de parallélisme, le serveur envoie **4** problèmes de la même catégorie à la fois. Les participants doivent donc résoudre les 4 problèmes et envoyer les solutions trouvées au serveur. Une bonne réponse ajoute X points tandis qu'une mauvaise en retire $X/2$. Le serveur répondra avec 4 nouveaux problèmes d'une nouvelle catégorie. Le but ultime sera de cumuler le plus grand nombre de points et ce, dans un temps déterminé.

Aussi, il est important de mentionner que le serveur n'attendra pas indéfiniment après les concurrents. Après un certain temps, celui-ci émettra 4 nouveaux problèmes et les points seront déduit présumant que les précédents n'ont pas été solutionnés assez rapidement.

Finalement, l'évaluation aura lieu sur un serveur dédié comportant 2 processeurs chacun, doté d'un seul coeur.

2 Catégories de problèmes

Les sous-sections suivantes décrivent chacunes des 6 catégories de problème que vous serez appelés à résoudre.

2.1 Labyrinthe

Dans ce problème, on vous demande d'identifier si un labyrinthe est solutionnable ou non. Vous recevrez donc 4 labyrinthes différents simultanément. Certains pourront être solutionnés, d'autres pas. Les labyrinthes seront encodés de la façon suivante :

taille	*données*	*taille*	*données*	*taille*	*données*	*taille*	*données*

Chacun des 4 ensembles *[taille, données]* se décompose comme suit. La première valeur sera un entier non signé sur 32 bits indiquant le nombre d'éléments dans une matrice **carrée**. Il y aura donc **N** entier non signé sur 32 bits. La valeur *1* représente une case navigable et la valeur *0* représente un mur. La position de départ sera toujours de *(1, 1)* et la position d'arrivée sera toujours de *($\sqrt{N}$-1, $\sqrt{N}$-1)* Cette matrice est donc mise à plat ligne par ligne. Ainsi, pour une taille N nous avons ceci :

N	*ligne 1*	*ligne 2*	...	*ligne $\sqrt{N}$*

Par exemple, les données suivante :

36	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

représente le labyrinthe suivant :

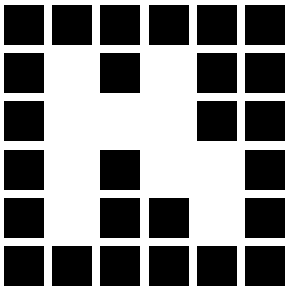


Figure 1: Un labyrinthe 6x6

La figure représente une image zoomée de 6x6 pixels. Les carrés noirs sont des murs, donc impossible de s'y déplacer. Les carrés les blancs sont le chemin navigable. Le point de départ sera donc le carré blanc en haut à gauche. Le point d'arrivée sera donc le carré blanc en bas à droite. La réponse sera *True* car il est possible de naviguer du point de départ au point d'arrivée.

La réponse attendue pour cette catégorie de problèmes est un ensemble de 4 valeurs booléennes : *True* si le labyrinthe comporte une solution, *False* sinon.

2.2 Sudoku

Le sudoku est un jeu en forme de grille défini en 1979 par l'Américain Howard Garns, mais inspiré du carré latin, ainsi que du problème des 36 officiers du mathématicien suisse Leonhard Euler. Le but du jeu est de remplir la grille avec une série de chiffres, de lettres ou de symboles tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou

dans une même sous-grille. La plupart du temps, les symboles sont des chiffres allant de 1 à 9, les sous-grilles étant alors des carrés de 3x3. Cependant, il est possible de généraliser ce principe à de plus petites ou plus grandes grilles.

Dans ce problème, il ne sera pas question de solutionner un sudoku mais plutôt de valider si les solutions proposées respectent les règles du jeu. Vous recevrez donc 4 sudokus différents simultanément. Certains seront valides, d'autres pas. Les sudoku seront encodés de la façon suivante :

taille	*données*	*taille*	*données*	*taille*	*données*	*taille*	*données*

Chacun des 4 ensembles *[taille, données]* se décompose ainsi. La première valeur sera un entier non signé sur 32 bits indiquant le nombre totals d'éléments au sudoku. Il y aura donc **N** entier non signé sur 32 bits.

Par exemple, les données suivantes :

16	*3*	*1*	*2*	*4*	*4*	*2*	*3*	*1*	*1*	*3*	*4*	*2*	*2*	*4*	*1*	*3*

représente le sudoku en pseudo-base 4 suivant :

3	1	2	4
4	2	3	1
1	3	4	2
2	4	1	3

Table 1: Un sudoku 4x4 pseudo-base 4

La réponse attendue pour cette catégorie de problèmes est un ensemble de 4 valeurs booléennes : *True* si la grille est une solution valide, *False* sinon.

2.3 Valeur dans un tableau

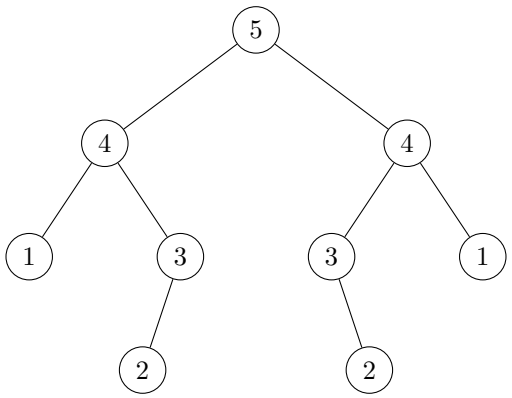
Dans ce problème, on demande simplement si une valeur donnée est présente ou non dans un tableau. Les valeurs à rechercher ainsi que les tableaux seront encodés de la façon suivante :

valeur	*taille*	*données*	*valeur*	*taille*	*données*	*valeur*	*taille*	*données*	*valeur*	*taille*	*données*

La réponse attendue pour cette catégorie de problèmes est un ensemble de 4 valeurs booléennes : *True* si le tableau comporte la valeur données, *False* sinon.

2.4 Arbre symétrique

Dans ce problème, on demande d'identifier quels arbres dans un ensemble d'arbres donnés sont symétriques. En d'autres termes, quels sont les arbres dans lesquelles le sous-arbre de droite de la racine est le miroir du sous-arbre de gauche. La figure ci-dessous offre un exemple d'arbre symétrique qui pourrait vous être transmis:



De plus, il est important de préciser que les arbres utilisés dans ce problème ne contiennent que des valeurs entières positives.

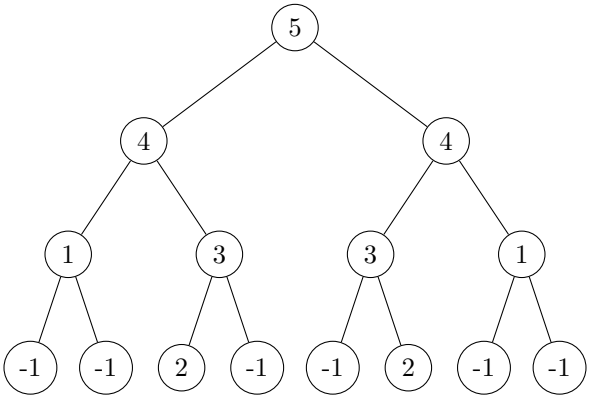
Au niveau du message envoyé, le client peut s'attendre à voir les données sous la forme suivante.

Taille	*Données*	···	*Taille*	*Données*

On doit donc s'attendre à retrouver quatre fois un entier non signé de 32 bits indiquant le nombre d'éléments de l'arbre se retrouvant dans la section données qui le suit. Les données de l'arbre sont des entiers signés de 32 bits.

Au niveau du contenu d'une section données, il faut préciser deux choses. Premièrement, même si l'arbre ne contient que des valeurs positives dans ses noeuds, le client va retrouver des valeurs négatives dans la section données. Ainsi, les liens nuls seront caractérisés par une valeur de -1. Deuxièment, les valeurs des noeuds seront présentés suivant un parcours en profondeur, c'est-à-dire ils sont en pré-ordre.

La combinaison de ces deux éléments fera en sorte qu'un arbre dans le cadre de ce problème ressemblera à celui présenté ci-dessous:



Ceci résultera en un section données dans un message ressemblant à celui présenté ci-dessous:

5	*4*	*1*	*-1*	*-1*	*3*	*2*	*-1*	*4*	*3*	*-1*	*2*	*1*	*-1*	*-1*

La réponse attendue pour cette catégorie de problèmes est un ensemble de 4 valeurs booléennes : *True* si l'arbre est symétrique, *False* sinon.

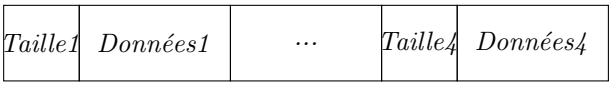
2.5 Chaînes pareilles

Dans le cadre de ce problème, on vous demandera d'identifier quelle chaîne de caractères diffère des autres. Ainsi, vous recevrez quatre versions d'une certaine chaîne de caractères. Dans chacune de ces chaînes, les caractères ont été mélangés de façon aléatoire et certains de ces caractères ont été mis en majuscules. De plus, dans une de ces chaînes, certains caractères ont été remplacés par des nouveaux caractères. Les caractères sont encodés en UTF8. Ce qui distingue une chaîne d'une autre sont les caractères différents. Les changement de case (MAJUSCULE, minuscule) ainsi que les changement de position d'un caractère ne sont pas considérés comme des changements pour cette catégorie. Ainsi, seul l'ajout ou le retrait d'un caractère est considéré comme un changement valable. Un exemple concret de ce que vous pourriez recevoir est donné ci-dessous

PASSword
WoRdPaSs
dpRAosWS
pawossrX

La réponse pour ce problème sera *False, False, False, True*.

Finalement, un problème de cette catégorie sera envoyé dans un message de la forme suivante:



La première valeur sera un entier non signé sur 32 bits indiquant la taille des chaînes de caractères. Il sera suivi de quatre chaînes de caractères.

La réponse attendue pour cette catégorie de problèmes est un ensemble de 4 valeurs booléennes : *True* si la chaîne se distingue des autres, selon les règles citées plus haut, par rapport aux autres, *False* sinon.

2.6 Encodage

Le but de ce problème est de déterminer si une chaîne de caractères, une fois décodée, est d'une longueur donnée.

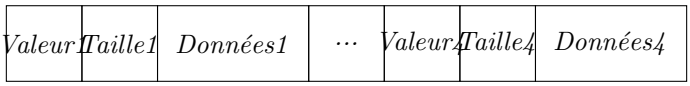
L'algorithme utilisé pour encoder les chaînes de caractères est l'algorithme Run-Length Encoding (RLE). Brièvement, cet algorithme transforme une chaîne de caractères contenant plusieurs plages de caractères répétés en une chaîne de caractères où ces plages ont été remplacées par le nombre de répétitions et le caractère répété. Pour donner un exemple concret de la chose, la chaîne de caractère suivante:

aaaaaabbcccccddddddddddddd

se fera transformer en la chaîne encodée suivante:

6a2b5c13d

Dans un message, un problème de ce type aura le format suivant:



Comme on peut le constater, le message commencera par un entier non signé sur 32 bits indiquant la taille de la chaîne de caractères recherchée. Ensuite, on retrouvera un entier non signé sur 32 bits indiquant la taille de la chaîne de caractères encodée et la chaîne, encodée en UTF8, en tant que telle le tout 4 fois de suite.

La réponse attendue pour cette catégorie de problèmes est un ensemble de 4 valeurs booléennes : *True* si la chaîne de caractères décodée correspond à la valeur donnée, *False* sinon.

3 Protocole réseau

Lors de la connexion au serveur, celui-ci se mettra immédiatement à envoyer des problèmes à résoudre. Pour ce faire, le serveur utilise un protocole brut très simple. Toute les valeurs sont encodées en *little-endian*. Les valeurs booléennes devront être encodées la forme d'un nombre non signé de 8 bits, où 1 représente *True* et où 0 représente *False*.

En général, le serveur envoie les problèmes sous cette forme :

ID	*... bloc de données ...*

ID est un identifiant encodé sous la forme d'un entier non signé sur 32 bits. Il ne pourra prendre qu'une valeur dans l'intervalle $[0, 5]$. Cet intervalle est garanti par le serveur. Cette valeur indique le type de problème à résoudre reçu. L'association entre valeur et type de problème est décrite dans le tableau ci-dessous :

ID	**Type**
0	Labyrinthe
1	Sudoku
2	Valeur dans un tableau
3	Arbre symétrique
4	Chaînes pareilles
5	Encodage

Le *bloc de données* est propre à chacune des catégories de problème. De façon générale, ce bloc sera une suite de 4 ensembles *taille, données* où *taille* est un entier non signé sur 32 bits indiquant le nombre d'éléments séquentiel composant *données*. La définition complète de chacun des problèmes est décrite ci-dessous.

3.1 Labyrinthe

ID	N	donnée	...	donnée	N	donnée	...	donnée	N	donnée	...	donnée	N	donnée	...	donnée

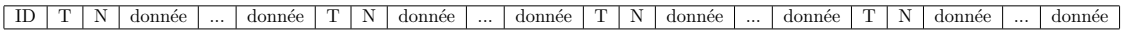
- *ID* est un entier non signé sur 32 bits ayant toujours la valeur 0.
- *N* sont des entiers non signés sur 32 bits indiquant le nombre de données. Ils sont présents au nombre de 4 dans tous le message.
- *donnée* sont des entiers non signés sur 32 bits représentant les cases du labyrinthe. Ils sont présents *N* fois. Ils ne peuvent prendre que les valeurs 0 et 1.

3.2 Sudoku

ID	N	donnée	...	donnée	N	donnée	...	donnée	N	donnée	...	donnée	N	donnée	...	donnée

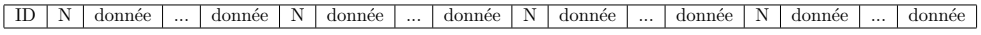
- *ID* est un entier non signé sur 32 bits ayant toujours la valeur 1.
- *N* sont des entiers non signés sur 32 bits indiquant la taille totale du sudoku. Ils sont présents au nombre de 4 dans tous le message.
- *donnée* sont des entiers non signés sur 32 bits représentant les valeurs dans la grille de sudoku. Ils sont présents *N* fois.

3.3 Valeur dans un tableau



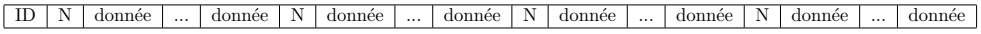
- *ID* est un entier non signé sur 32 bits ayant toujours la valeur 2.
- *T* sont des entiers non signés sur 32 bits indiquant la valeur à trouver dans le tableau. Ils sont présents au nombre de 4 dans tous le message.
- *N* sont des entiers non signés sur 32 bits indiquant la taille totale du tableau. Ils sont présents au nombre de 4 dans tous le message
- *donnée* sont des entiers non signés sur 32 bits représentant les valeurs dans le tableau. Ils sont présents *N* fois.

3.4 Arbre symétrique



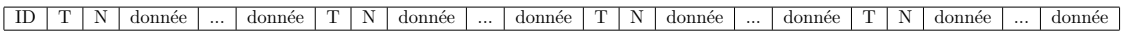
- *ID* est un entier non signé sur 32 bits ayant toujours la valeur 3.
- *N* sont des entiers non signés sur 32 bits indiquant la taille totale de l'arbre. Ils sont présents au nombre de 4 dans tous le message.
- *donnée* sont des entiers signées sur 32 bits représentant les valeurs dans l'arbre. Ils sont présents *N* fois. Ils sont garantis de prendre que des valeurs strictement positives et -1.

3.5 Chaînes pareilles



- *ID* est un entier non signé sur 32 bits ayant toujours la valeur 4.
- *N* sont des entiers non signés sur 32 bits indiquant la longueur totale de la chaîne. Ils sont présents au nombre de 4 dans tous le message.
- *donnée* sont des caractères UTF8. Ils sont présents *N* fois.

3.6 Encodage



- *ID* est un entier non signé sur 32 bits ayant toujours la valeur 5.

- T sont des entiers non signés sur 32 bits indiquant la longueur recherchée. Ils sont présents au nombre de 4 dans tous le message.
- N sont des entiers non signés sur 32 bits indiquant la longueur totale de la chaîne. Ils sont présents au nombre de 4 dans tous le message.
- *donnée* sont des caractères UFT8. Ils sont présents N fois.

3.7 Formats des réponses

donnée	donnée	donnée	donnée

- *donnée* sont des booléens encodés sous la forme de nombre entiers non signés sur 8 bits. Ils sont présents *4* fois. Ils ne peuvent prendre que les valeurs 0 et 1.

4 Programmes

Un programme client de démonstration vous sera fourni. Celui-ci permet simplement de se connecter au serveur, recevoir une valeur et emmètre une réponse aléatoire en utilisant la bibliothèque Boost. Ce client n'a besoin que d'un seul paramètre : l'adresse du serveur. Le code source vous est fourni. Il est possible de le démarrer avec la ligne de commande suivante :

you@0xD:~$./client localhost

Un programme serveur de test vous sera fourni. Celui-ci permet de simuler l'environnement d'évaluation sur lequel votre programme sera testé. Ce serveur a besoin des noms de fichiers contenant les données des problèmes dans l'ordre suivant : Labyrinthe, Sudoku, Valeur dans un tableau, Arbre Symétrique, Chaînes pareilles et Encodage. Il est possible de le démarrer avec la ligne de commande suivante :

you@0xD:~$./server maze_small.bin sudoku_small.bin array_small.bin password_small.bin tree_small.bin RLE_small.bin

Deux jeux de données vous seront fournis, l'un contenant quelques petits problèmes, l'autre un plus grand nombre de gros problèmes. L'évaluation finale sera faite avec un jeu de données gardé secrètes spécialement concocté.

5 Remarques

Nous tenons à remercier chaleureusement Reginald "Reggie" Fils-Aimé pour son support dans la conception de cette épreuve. Sa contribution, bien que modeste, est essentiel au bon fonctionnement du programme. Vous comprendrez

rapidement pourquoi. Nous tenons aussi à remercier tout aussi chaleureusement Patrice Roy pour ses idées ayant menées à cette compétition. Merci Pat! En cas d'accident, veuillez garder votre calme. Ne paniquez pas. Demeurez à votre siège. Attendez les instructions. Mangez votre main droite. Procédez dans l'ordre et suivez à la lettre les instructions. Paniquez.

Merci de votre collaboration.
0xD

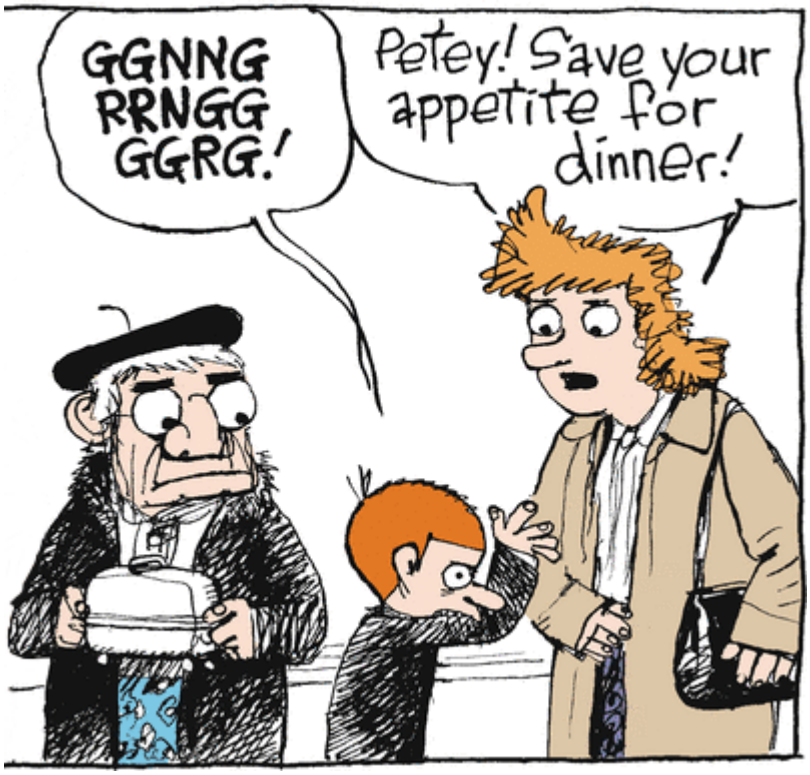


Figure 2: Thompson, Richard. "Cul de Sac", 23 Novembre 2011