TD1 – Backtracking

1 Premiers pas

Plusieurs sites recensent les annales des ICPC ; nous utiliserons le site de l'université de Tianjin, sur lequel vous devez *tous* vous enregistrer :

`http://acm.tju.edu.cn/toj/`

Sur ce site, vous pouvez trouver des tas d'énoncés de problèmes. Pour chaque problème, vous pouvez envoyer votre solution, qui sera alors vérifiée par un juge. Ce dernier essaie votre programme sur un jeu de test et répond une des choses suivantes :

- **Accepted (AC)** : C'est le mieux que le juge peut vous répondre : le programme est correct.
- **Presentation Error (PE)** : La réponse est bonne, mais le format de sortie incorrect (c'est-à-dire les espacements et nouvelles lignes).
- **Wrong Answer (WA)** : La réponse renvoyée par votre programme sur le jeu de test n'est pas correcte (impossible de savoir si c'est entièrement faux ou juste une erreur d'orthographe).
- **Runtime Error (RE)** : votre programme a planté durant son exécution (erreur de segmentation, division par zéro...)
- **Time Limit Exceeded (TLE)** : votre programme est trop gourmand en temps. À noter que le programme est interrompu par le juge quand il dépasse le temps limite. Il n'est pas possible de savoir si le programme est correct.
- **Output Limit Exceeded (OLE)** : votre programme a écrit trop de choses.
- **Compilation Error (CE)** : votre programme ne compile pas.

Les langages autorisés sont le C, C++, Pascal et Java. Nous vous demandons de programmer en C++. Les options de compilation sont :

```
g++ foo.cpp -o foo -O -ansi -Wall -lm -static
```

1.1 Hello world !

Voici un exemple d'énoncé de problème ; ils sont tous bâtis sur le même schéma :

This is the first problem for test.

Since all we know the ASCII code, your job is simple : *Input numbers and output corresponding messages.*

Input

The input will contain a list of positive integers separated by whitespaces(spaces, newlines, TABs). Please process to the end of file (EOF). The integers will be no less than 32.

Output

Output the corresponding message. Note there is NOT a newline character in the end of output.

Sample Input

```
72 101 108 108 111 44
32 119 111 114 108 100 33
```

Sample Output

```
Hello, world!
```

1.1.1 Entrées/Sorties

La lecture des données se fait sur l'entrée standard et la réponse doit être donnée sur la sortie standard. Généralement, vous pouvez utiliser au choix l'interface C (fonctions `scanf` et `printf`) ou l'interface C++ (flux `cin` et `cout`). Mais comme la version C++ est beaucoup plus lente, il faudra connaître les fonctions C de toute façon pour certains problèmes.
Toutes les fonctions C ont des pages man. Pour le C++, il y a une documentation sur tout ça et sur plein d'autres choses à l'adresse `http://www.cppreference.com/wiki/` (à mettre dans vos favoris !).
N'oubliez pas d'inclure le fichier d'en-tête :

```
#include <cstdio>       // Version C
#include <iostream>     // Version C++
```

2 Exploration exhaustive avec backtracking

Certains problèmes algorithmiques peuvent être résolus par des méthodes efficaces (algorithmes gloutons, programmation dynamique). Pour d'autres, les problèmes *NP*-complets en particulier, les seules solution que l'on connaisse sont en temps exponentiel. Une technique assez générale consiste en l'exploration d'un nombre exponentiel de possibilités.
Par exemple, pour le problème de trouver un 3-coloriage pour un graphe de taille n, on peut essayer les 3^n coloriages possibles c'est l'algorithme de *recherche exhaustive*. Mais ce n'est pas la solution la plus efficace : par exemple, si le graphe est une clique de taille 100, un programme peut détecter qu'il n'est pas 3-coloriable après avoir testé 3 sommets seulement.
Le *backtracking* consiste à explorer l'arbre des possibilités à l'aide d'un algorithme récursif, en rebroussant chemin dès qu'on détecte un sous arbre dont toutes les feuilles sont impossibles.

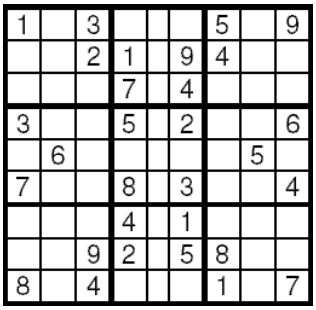
fonction ColorierRec(G, s)
  pour c variant de 1 à 3
    si aucun voisin de s dans G n'est de couleur c **alors**
      couleur[s] := c
      si sommet = $|G|$ **alors**
        Afficher le résultat
      sinon
        ColorierRec(G, $s + 1$)

Algorithm 1: 3-coloriage de graphe

En fonction de la taille des entrées, le backtracking peut être une solution pertinente ... ou non ! Souvent, les indications du problème permettent de déterminer la complexité maximale que doit avoir votre algorithme. Si après un (certain) temps de réflexion, vous ne trouvez pas de solution polynomiale, et que la taille des données laisse à penser que le backtracking peut fonctionner, et que vous soupçonnez le problème d'être *NP*-complet, tentez votre chance.

Exercice 1. *Sudoku*

Sudoku is a very simple task. A square table with 9 rows and 9 columns is divided to 9 smaller squares 3x3 as shown on the Figure. In some of the cells are written decimal digits from 1 to 9. The other cells are empty. The goal is to fill the empty cells with decimal digits from 1 to 9, one digit per cell, in such way that in each row, in each column and in each marked 3x3 subsquare, all the digits from 1 to 9 to appear. Write a program to solve a given Sudoku-task.



The input data will start with the number of the test cases. For each test case, 9 lines follow, corresponding to the rows of the table. On each line a string of exactly 9 decimal digits is given, corresponding to the cells in this line. If a cell is empty it is represented by 0.

For each test case your program should print the solution in the same format as the input data. The empty cells have to be filled according to the rules. If solutions is not unique, then the program may print any one of them.

Sample Input

```
1
103000509
002109400
000704000
300502006
060000050
700803004
000401000
009205800
804000107
```

Sample Output

```
143628579
572139468
986754231
391542786
468917352
725863914
237481695
619275843
854396127
```

1. Le problème du sudoku est *NP*-complet, quel serait un algorithme de recherche exhaustive pour ce problème ? Quelle est sa complexité ?

2. Proposez un algorithme de backtracking pour le sudoku, et écrivez le pseudo-code.

3 Salle machine

Vous retrouverez les énoncés de TD, ainsi que les liens vers les problèmes sur le wiki :

http ://perso.ens-lyon.fr/julien.robert/Teaching/ACM-2009-2010/wiki/

Commencez par vous inscrire sur ce wiki, puis sur le site de l'université de Tianjin.

3.1 Hello world !

Faites un programme pour résoudre le problème Hello World ! et soumettez le au juge pour vérifier votre solution. Essayez de rajouter des bugs pour obtenir les différentes insultes possibles du juge.

3.2 Sudoku

Le problème du sudoku est le n°1851 . Programmez et testez votre solution avec le juge.