TD 2

N’oubliez pas de mettre en œuvre vos idées (code)

MISE EN JAMBE

On dispose de N pièces et l'on doit rendre la monnaie.

Prenez soin de bien définir les données que vous utilisez (soyez général ! Ne pas considérer de cas particulier)

Version très brutale

Écrire un algorithme qui utilise N itérations

Commentaire ?

Version un peu plus générale

Soit pièce un tableau tel que piece[i] correspond à la valeur de la ième pièce et somme la somme à rendre ; résoudre la famille de pb

Vers une solution plus élégante

A l'image de l'exemple vu en cours, on souhaite représenter l'espace des solutions possibles et se donner une manière d’énumérer

Réfléchir à une représentation astucieuse de cela (utiliser un tableau de N booléen/utiliser une représentation binaire), expliquer le principe.

Approche gloutonne

On souhaite résoudre maintenant le problème des pièces de monnaie. Formulez le problème en termes de données et résultats, puis expliquez comment appliquer la stratégie de construction.

APPROCHE FORCE BRUTE

On souhaite résoudre le problème suivant (pas la classe des problèmes de ce type) avec une approche « force brute ».

DEUX+CINQ=SEPT

```
    D   E   U   X
+   C   I   N   Q
```

S E P T

Chaque chiffre est remplacé par une lettre, ce problème peut alors se ramener à un système de relations entre lettres ou entre mots.

Attention rien ne dit qu'un même chiffre n'est pas associé à deux lettres différentes.

Il s'agit de trouver à quoi corresponde chaque lettre en terme de chiffre pour que la somme soit juste. Une solution évidente est que toutes les lettres correspondent à la valeur nulle.

On souhaite résoudre le problème en faisant appel à un algorithme « force brute ».

Questions introductives (3 points)

Qu'est-ce qu'un algorithme de force brute ? Quel est son principe général ?

Quelles sont les lettres représentant l'espace d'états (par opposition à celle se déduisant du choix de deux autres) ? Combien y en a t-il ? (Dit autrement quelles sont les inconnues du système)

Combien de valeurs peuvent prendre chacune de ces lettres ?

Quelle est la taille de l'espace d'état, la complexité d'un algorithme de force brute sur le problème ?

Construction de l'algorithme (1+3+3 points)

On représente chaque lettre du problème à trouver par une variable du même nom. Par exemple la lettre I sera associé à une variable I (ou i) dans l'algorithme. Comment itérer sur cet espace d'états (cf questions 1 et 2) ?
Remarque : la manière la plus simple n'est peut être pas la meilleure mais si elle permet de répondre à la question elle est satisfaisante *ici*.

On suppose l'existence d'une fonction OK qui, étant données les valeurs de ces lettres (cf point 1) représentant l'espace d'états, retourne un booléen indiquant si ces valeurs sont solutions du problème ou non.

Donner l'algorithme de force brute qui utilise cette méthode/fonction

Ecrire la méthode ok (définir les paramètres et leur type, le type du résultat et l'algorithme associé, ...)

Coder cette solution.

SAC A DOS GLOUTON.

On dispose d'un ensemble (ci-dessous) de n items de poids w_i (weight) et de coût p_i (profit) :

i	1	2	3	4

w_i	20	50	30	100
p_i	10	5	50	20

Soit C la « capacité » maximale du sac à dos, C=80. Soit x_i une variable valant 1 si le $i^{ème}$ item est pris (0 sinon).

L'objectif est de maximiser $\sum_{i=1}^{n} p_i x_i$ sous la contrainte $\sum_{i=1}^{n} w_i x_i \leq C$.

On choisit une approche gloutonne pour laquelle on va maximiser le profit (donc choisir les items les plus profitables).

A quoi correspondent les données de l'énoncé après la première phase de l'algorithme, quelle est la classe de complexité de cette première phase (expliquez) ?

i	1	2	3	4
w_i				
p_i				

Comment s'applique ensuite la deuxième partie de l'algorithme glouton, dérouler « manuellement » son exécution ici à partir des données de la question précédente
Important : si vous ne savez pas répondre à la question précédente, vous utiliserez comme données celles ci-dessous pour montrer que vous savez quand même appliquer un algorithme glouton.

i	*1*	*2*	*3*	*4*
w_i	*20*	*50*	*30*	*100*
p_i	*10*	*5*	*50*	*20*

Quelle est la complexité de cette $2^{ème}$ partie ?

Quelle est la complexité de l'algorithme glouton appliqué à ce problème (on notera n le nombre d'items)?

Coder cet exemple