Travaux dirigés n^o 10
Programmation dynamique

Exercice 10.1 — Suite de Fibonnacci

On rappelle la définition de la suite de Fibonacci : $\begin{cases} Fib(0) = 0 \\ Fib(1) = 1 \\ \forall N \geq 0,\ \ Fib(N+2) = Fib(N) + Fib(N+1) \end{cases}$

1. Proposez un algorithme itératif Fib(N) utilisant la programmmation dynamique pour calculer le N$^{\text{ème}}$ nombre de Fibonacci.
2. Calculez la complexité de votre algorithme en considérant bien sûr que l'on manipule des grands entiers. *Rappel* : une approximation du terme $Fib(N)$ est $2^{0,694N}$.

Exercice 10.2 — Sac-à-dos discret

On veut résoudre le problème de sac-à-dos suivant (par comparaison au problème du TD précédent, les items ne sont plus fractionnables) :

Données :

— une collection d'items i pour i variant de 1 à n
— pour chaque item, un couple valeur/poids (val_i, w_i)
— un poids maximal W

Problème : Quel est le plus grand bénéfice en valeur que l'on peut réaliser en sélectionnant des items à la condition que la somme de leurs poids n'excède pas le poids W ?

Considérons par exemple le tableau de couples valeur/poids suivant :

$$[(30,5),(20,2),(45,1),(10,4),(25,3),(5,2),(15,2),(10,1),(35,4)]$$

avec le poids $W = 7$.

1. Sur l'exemple précédent, on peut réaliser un bénéfice maximal égal à 100. Précisez avec quels items.
2. Proposez un algorithme récursif pour résoudre ce problème. Que faudrait-il lui ajouter pour le rendre efficace ?
3. Proposez un algorithme itératif utilisant cette fois la programmation dynamique pour ce problème.
4. Comparez la complexité des deux algorithmes obtenus.
5. *(facultatif)* Adaptez votre dernier algorithme afin qu'il retourne (au moins) une liste d'items réalisant la solution. On pourrait également chercher à obtenir **toutes** les listes d'items réalisant la solution.

Exercice 10.3 — Partition d'un entier

Ce problème consiste à énumérer toutes les façons de décomposer un entier en somme d'entiers positifs. Pour commencer, nous vous proposons seulement de compter le nombre de solutions.

> **Donnée :** un entier strictement positif N
> **Problème :** évaluer le nombre de partitions de l'entier N

Par exemple, si $N = 5$, la réponse est 7. En effet, en énumérant les partitions, on obtient :

```
1  + 1  + 1  + 1  + 1
2  + 1  + 1  + 1
2  + 2  + 1
3  + 1  + 1
3  + 2
4  + 1
5
```

1. Proposez le pseudo-code d'un algorithme récursif qui donne le nombre de partitions d'un entier donné. Quelle est sa complexité ?
2. Trouvez un algorithme itératif utilisant la programmation dynamique pour répondre à ce problème. Quelle est sa complexité ?
3. *(facultatif)* Faites en sorte d'énumérer toutes les partitions d'un entier donné.