TP n° 1 – Algorithmes dynamiques et mémoïsation

thomas.h@ens.fr
thomash.fr/teaching/

1 Oups !

Question 1. Yorel Reivax implémente le code suivant, qui crée une fonction `fibo0` telle que `fibo0 n` retourne le n-ième élément de la suite de Fibonacci. Quel est le problème de son implémentation ?

$$Fib_n = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ Fib_{n-1} + Fib_{n-2} & \text{sinon.} \end{cases}$$

```
let rec fibo0 n = match n with
| 0 -> 0
| 1 -> 1
| n -> fibo0 (n - 1) + fibo0 (n - 2)
```

Question 2. Écrire une fonction `fibo1` de complexité linéaire.

Question 3. * Écrire une fonction `fibo2` de complexité logarithmique.

2 Mémoïsation

La mémoïsation est une technique qui consiste à mémoriser le résultat d'un appel de fonction pour pouvoir renvoyer le résultat en temps constant si la fonction est appelée avec les mêmes paramètres une seconde fois.

On utilise souvent une table de hachage pour stocker les valeurs d'entrée et les valeurs de sortie associées, cependant dans le cas où les valeurs d'entrée sont des entiers de taille raisonnable on peut utiliser de simples tableaux à la place.

Question 4. Écrire une fonction `fibo3` qui soit une version mémoïsée de `fibo0`.

Quelle est sa complexité ?

Question 5. La société Nigol Corp. fabrique des distributeurs de boissons et vous demande de coder la partie « rendre la monnaie ». Vous devez donc étant donné un montant renvoyer la plus petite liste de pièces représentant ce montant. La société Nigol Corp. étant implantée dans de nombreux pays aux systèmes monétaires variés, votre fonction devra prendre en paramètre la liste des pièces qui existent dans le pays.

Quelle est la complexité de votre fonction ?

Question 6. Pouvez-vous écrire une fonction `memoise` telle que `memoise f` renvoie une version mémoïsée de `f` ?

3 Algorithmes dynamiques

Il n'est pas toujours utile de mémoriser tous les résultats précédemment calculés, notamment lorsqu'une fonction n'est appelée qu'une seule fois, la mémoïsation consomme alors plus de mémoire que nécessaire.

Les algorithmes dynamiques reprennnent l'idée de la mémoïsation mais en ne mémorisant que les résultats intermédiaires nécessaires au calcul du résultat final et en les oubliant tous une fois le résultat final trouvé. Ils évitent ainsi de recalculer les solutions des sous-problèmes tout en utilisant peu de mémoire.

Question 7. On dispose d'une liste de matrices $A_1, \ldots, A_n$ de tailles variées et on souhaite calculer le produit $A_1 \times \cdots \times A_n$. On suppose que le nombre d'opérations nécessaires pour calculer le produit de deux matrices de taille n, p et p, m est nmp [1]. Le nombre d'opérations pour calculer le grand produit dépend donc du parenthésage utilisé.

Écrire une fonction qui calcule le parenthésage optimal pour minimiser le nombre total d'opérations. On traitera chaque facteur de la liste $A_1, \ldots, A_n$ comme un sous-problème.

Quelle est la complexité de votre fonction ?

Question 8. Écrire une fonction `organise` qui étant donnée une liste d'activités (chaque activité est représentée par une heure de début et une heure de fin) renvoie une liste d'activités compatibles (dont les horaires ne se recoupent pas) maximisant le temps total es activités.

Quelle est la complexité de votre fonction ?

Question 9. La distance de Levenshtein entre deux mots est le nombre minimal d'insertions, délétions ou substitutions de lettre nécessaires pour passer de l'un à l'autre [2]. Écrivez une fonction `levenshtein` qui calcule la distance de Levenshtein entre deux mots [3]. On utilisera un tableau à deux dimensions pour stocker les résultats intermédiaires.

Quelle est la complexité de la fonction `levenshtein` ?

Question 10. Pouvez-vous réécrire la fonction `levenshtein` pour stocker moins de résultats intermédiaires ?

4 Fin

Question 11. * Le grand roi Sunil I$^{\text{er}}$ dispose d'un grand jardin rectangulaire planté d'arbres centenaires dans lequel il souhaite mettre une piscine (rectangulaire aussi) la plus grande possible. Il vous a donc chargé de trouver le plus rectangle possible dans son jardin qui ne contienne pas d'arbres [4] pour y mettre la piscine.

Écrire une fonction qui prend un tableau bidimensionel indiquant la position des arbres et renvoie la taille du plus grand rectangle sans arbres.

1. Si quelqu'un n'est pas d'accord, qu'il code l'algorithme de Coppersmith-Winograd.
2. Par exemple pour passer de `abcde` à `bfdef`, il faut au minimum 3 opérations : `abcde`→`bcde`→`bfde`→`bfdef`
3. Essayez de trouver une relation de récurrence entre $L(a_1 \ldots a_{n-1}, b_1 \ldots b_m)$, $L(a_1 \ldots a_n, b_1 \ldots b_{m-1})$, $L(a_1 \ldots a_{n-1}, b_1 \ldots b_{m-1})$ et $L(a_1 \ldots a_n, b_1 \ldots b_m)$.
4. il tient beaucoup à ses arbres