TD 8. Programmation dynamique

Exercice 1. *Coefficients binomiaux*

Les coefficients binomiaux $\binom{n}{k}$ vérifient la récurrence suivante : $\binom{n}{0} = \binom{n}{n} = 1$ et $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ pour $0 < k < n$.

1. Écrire un algorithme de programmation dynamique pour calculer $\binom{n}{k}$ et analyser sa complexité.

2. Écrire une variante de l'algorithme qui minimise l'espace mémoire utilisé.

Exercice 2. *Stratégie étudiante*

Une étudiante prévoit son programme de révision. Chaque jour peut être un jour de travail *tranquille*, un jour de travail *soutenu*, ou un jour de *repos*. Mais avant un jour *soutenu*, il lui faut un jour de *repos*. De plus, elle sait estimer, pour le jour i, le nombre t_i de points qu'elle obtiendra avec un travail *tranquille* et le nombre s_i de points avec un travail *soutenu*. Un jour de repos ne lui fait gagner aucun point. Par exemple, sur la semaine suivante, sa stratégie optimale est d'être en repos les jours 1 et 4, en travail tranquille le jour 3, et en travail soutenu les jours 2 et 5, ce qui lui rapportera $2,5$ points.

Jour	1	2	3	4	5
t	0,3	0,5	0,4	0,6	0,2
s	0,5	1,2	1	0,8	0,9

1. Montrer que l'algorithme suivant n'est pas optimal (n est le nombre de jours).

```
i ← 1
Tant que i ≤ n :
    Si i ≤ n−1 et s_{i+1} > t_i + t_{i+1} :
        Repos le jour i et travail soutenu le jour i+1
        i ← i+2
    Sinon :
        Travail tranquille le jour i
        i ← i+1
```

2. Proposer une formule récursive pour calculer le nombre p_i maximal de points obtenu en travaillant jusqu'au jour i.

3. Décrire un algorithme de programmation dynamique pour calculer p_n et analyser sa complexité en temps et en espace.

4. Peut-on réduire sa complexité en espace ?

5. Décrire un algorithme de calcul d'une stratégie optimale.

Exercice 3. *Le retour du sac-à-dos*

On rappelle le *problème du sac-à-dos* : étant donné n objets (t_0, v_0), …, (t_{n-1}, v_{n-1}) et une taille S, on cherche un sous-ensemble $I \subset \{0, \dots, n-1\}$ qui maximise la *valeur* totale $\sum_{i\in I} v_i$ tout en respectant la contrainte $\sum_{i\in I} t_i \leq S$. On note $V_{\max}$ la valeur maximale qu'on peut atteindre.

1. Pour $m < n$ et $t \leq S$, on note $V(m, t)$ la valeur maximale que l'on peut atteindre en ne prenant que des objets parmi (t_0, v_0), …, (t_m, v_m), et avec une taille totale maximale $\leq t$.
   - **i.** Exprimer $V_{\max}$ avec $V(m, t)$ pour un m et un t bien choisis.
   - **ii.** Que vaut $V(m, t)$ si $t < t_m$? Et si $m = 0$?
   - **iii.** Donner une formule récursive pour $V(m, t)$. *Distinguer deux cas : on choisit l'objet m ou non.*
   - **iv.** En déduire un algorithme de calcul de $V_{\max}$ et analyser sa complexité.
   - **v.** Comparer le résultat avec l'approche par recherche exhaustive.
2. On souhaite effectuer un algorithme de reconstruction, pour obtenir une solution.
   - **i.** Modifier l'algorithme précédent pour calculer, pour tout m et t, un booléen $C_{[m,t]}$ qui indique si l'objet m est choisi pour atteindre la valeur $V(m, t)$.
   - **ii.** Utiliser le tableau des $C_{[m,t]}$ pour reconstruire la solution.
3. *(bonus)* Décrire une variante des algorithmes précédents qui calcule la taille $S(m, v)$ minimale d'un sac-à-dos de valeur v constitué des objets 0 à m, pour tout m et v.

Exercice 4. *Voyageur de commerce*

On rappelle le problème du voyageur de commerce : étant donné n points $p_i = (x_i, y_i)$, on cherche un chemin $p_{i_0} \to p_{i_1} \to \cdots \to p_{i_{n-1}} \to p_{i_0}$ de longueur totale minimale, où la longueur entre p_i et p_j est $\delta_{i,j} = \sqrt{x_j - x_i)^2 + (y_j - y_i)^2}$. On note L la longueur minimale d'un plus court chemin.

1. Pourquoi peut-on fixer $i_0 = 0$?
2. Soit P l'ensemble des points, et $U \subset P$ tel que p_0, $p_j \in U$. On note $\Delta(U, p_j)$ la longueur du plus court chemin de p_0 à p_j qui passe par chaque sommet de U une fois exactement.
   - **i.** Exprimer L en fonction de la fonction Δ et de $\delta_{j,0}$.
   - **ii.** Que vaut $\Delta(\{p_0\}, p_0)$?
   - **iii.** Montrer que $\Delta(U, p_j) = \min\{\Delta(U \setminus \{p_j\}, p_i) + \delta_{i,j} : p_i \in U, i \neq 0, j\}$.
3.
   - **i.** Écrire un algorithme de programmation dynamique pour le voyageur de commerce.
   - **ii.** Analyser sa complexité. *Indication : combien P a-t-il de sous-ensembles ?*
   - **iii.** Comparer le résultat avec l'approche par recherche exhaustive.