Ex 1 : Sac-à-dos

Q1 : Modélisation du problème

Un entier n : nombre d'objets

$(w_i)_{1 \leq i \leq n}$: poids des objets

$(v_i)_{1 \leq i \leq n}$: valeurs des objets

W : poids maximum

V : la valeur cible

Problème de décision : Existe-t-il un sous-ensemble $S \subseteq \{1, \dots, n\}$

tel que : $\sum_{i \in S} w_i \leq W$ et $\sum_{i \in S} v_i \geq V$

Problème d'optimisation : W est fixée

On cherche un sous-ensemble $S \subseteq \{1, \dots, n\}$ tel que $\sum_{i \in S} w_i \leq W$

qui maximise $\sum_{i \in S} v_i$

Q2 : Montrer que knapsack est NP-complet

N'a de sens que pour le problème de décision.

1°) Montrer qu'il est NP = on peut vérifier une solution en temps polyn.

Soit $(w_i)_{1\le i\le n}$ et $(v_i)_{1\le i\le n}$, W et V une instance de Knapsack.

Vérifier que S est une solution :

- $S \subseteq \{1,\dots n\}$ — $O(n)$
- $\sum_S w_i \le W$ — $O(n)$
- $\sum_S v_i \ge V$ — $O(n)$

) complexité linéaire donc polynomiale

Knapsack n'est pas plus complexe que NP

2°) Réduire un problème NP-difficile (ex Subset Sum) à Knapsack

$m, (a_i)_{1\le i\le m}, t$ ⟶ $n, (w_i)_{1\le i\le n}, (v_i)_{1\le i\le n}, W, V$

↑ transfos polynomiales

↑ solution

T

↓ résolution

S

Knapsack est au moins aussi dur qu'un problème NP-difficile.

$n = m$
$w_i = a_i$
$v_i = a_i$
$W = t$
$V = t$

⟶ $S \subseteq \{1,\dots n\} = T$

$\sum w_i \le W \Rightarrow \sum a_i \le t$

$\sum v_i \ge V \Rightarrow \sum a_i \ge t$

$\Rightarrow \sum a_i = t$

Branch and bound, Q4

On explore les objets par ordre décroissant de densité de valeur $\frac{v_i}{w_i}$
On explore d'abord les solutions qui consistent à prendre l'objet suivant.

Q5 : On classe les objets restant par densité de valeur décroissante. On les prend jusqu'à faire déborder le sac. Si la valeur cumulée des objets pris permet de faire mieux que la meilleure solution, on continue, sinon, on coupe la branche

Programmation dynamique

Q7 $V(i,j)$ = meilleure valeur en considérant les i premiers objet et un sac à dos de capacité j

$V(0,j) = 0$ pas d'objet → pas de valeur

$V(i,0) = 0$ pas de capacité → pas de valeur

si $1 \leq i$ et $w_i \leq j$

$$V(i,j) = \max\left(V(i-1,j),\ V(i-1, j-w_i) + v_i \right)$$

l'objet i ne fait pas partie de la sol. opt. — $V(i-1,j)$

l'objet i fait partie de la solution optimale — $V(i-1, j-w_i) + v_i$

si $1 \leq i$ et $1 \leq j < w_i$:

$$V(i,j) = V(i-1,j)$$

Q8 : Pour calculer $V(n,W)$ on doit remplir la matrice des $V(i,j)$ de taille $(n+1) \times (W+1)$

$\Rightarrow O(nW)$

Q9 : P = NP !

La complexité de l'algo en programmation dynamique est bien $O(n \times W)$

Mais la taille des données n'est pas n et W car W est un nombre dont la taille est $\log(W)$

Donc la complexité est bien exponentielle en fonction de la taille des données : $O(n \times e^{\log W})$