Plan

- Programmation dynamique
  - Partition
  - Sac à dos
  - Voyageur de commerce
- Branch & bound
- Comparaison

- Partition

Programmation dynamique

Problème de la partition

- Problème de décision
- Construction de 2 sous-ensembles E_1, E_2
- $|E_1| = |E_2|$

Algorithme

- si le poids total est impair, renvoyer faux
- création de T : (n, P/2+1)-matrice de bool
- initialisation
  - T[0,0] = vrai ; pour $j \in [0,P/2]$: T[0,j] = faux
- pour $i \in [1,n]$
  - pour $j \in [0,P/2]$
    - $T(i, j) \leftarrow [(j = 0) \vee (j = p(a_i)) \vee (T(i-1, j)) \vee (T(i-1, j-p(a_i)))]$
- renvoyer T[n, P/2]

Implémentation

Performances

- Complexité
  - Temps : O(n.P)
  - Espace : (n, P/2+1)-matrice

Tests

- Sac à dos

Programmation dynamique

Problème du sac à dos

- Problème d'optimisation
- Remplir un sac à dos avec *n* objets
- Volume limité
- Choix d'une solution à utilité maximale

Algorithme

- création de T une (n+1, volumeMax)-matrice
- initialisation : pour $j \in [1, volumeMax]$
  - T[0,j] = 0
- pour $i \in \{1, \ldots, n\}$
  - pour $j \in \{1, \ldots, volumeMax\}$
    - pour $k \in \{0, \ldots, volumeMax/volume[i]\}$
      - $T[i, j] \leftarrow \max(T[i, j], T[i-1, j-k \times volume[i]] + k \times utilite[i])$
- renvoyer T[n,volumeMax]

Implémentation

Performances

- Complexité
  - Temps : $O(n.V^2)$
  - Espace : (n+1, volumeMax)-matrice

Tests

- Voyageur de commerce

Programmation dynamique

Problème du voyageur de commerce

- Passer une seule fois par chaque ville
- Voyage de coût minimum
- cycle hamiltonien de poids minimum

Algorithme

Implémentation

Performances

Tests

BRANCH & BOUND