NOM : ROUHLING   Prénom : Damien   Jury :

Algèbre ← Entourez l'épreuve → Analyse

Sujet choisi : 906 - Programmation dynamique : escemples et applications.

Autre sujet : Réf : Cormen
Paschos : Optimisation combinatoire 1
Kleinberg, Tardos ; Algorithm Design

I - Escemple introductif : multiplication de matrices

Déf 1 : Soit $(A_i)_{i\in[\![1,n]\!]}$ des matrices de taille respective $n_{i-1} \times n_i$, $i \in [\![1,n]\!]$.
On veut calculer le produit $A_1 \cdots A_n$ en minimisant le coût total.
On suppose que multiplier deusc matrices de taille respective $p\times q$ et $q \times r$ coûte pqr.

1 - Solution naïve

On calcule le coût de chaque parenthésage.

Prop 1 : le nombre de parenthésages de $A_1 \cdots A_n$ est
$$C(n) = \begin{cases} 1 & \text{si } n=1 \\ \sum_{k=0}^{n-1} C(k)C(n-k) & \text{sinon} \end{cases}$$

Rq 1 : $C(n)$ est le $n^{\text{è}}$ nombre de Catalan et $C(n) \underset{n\to+\infty}{\sim} \frac{4^n}{n\sqrt{n\pi}}$

2 - Sous structure optimale

Prop 2 : Si P_1P_2 est un parenthésage optimal de $A_1 \cdots A_n$ où $\begin{cases} P_1 \text{ est un parenthésage de } A_1 \cdots A_k \\ P_2 \text{ ——— } A_{k+1} \cdots A_n \end{cases}$
alors $\begin{cases} P_1 \text{ est optimal pour } A_1 \cdots A_k \\ P_2 \text{ ——— } A_{k+1} \cdots A_n \end{cases}$

Idée : esploiter cette propriété pour réduire l'espace de recherche

Cor 1 : Le coût optimal d'un parenthésage de $A_i \cdots A_j$ est :
$$c_{ij} = \begin{cases} 0 & \text{si } i=j \\ \min_{k\in[\![i,j-1]\!]} c_{ik} + c_{k+1\,j} + n_{i-1} n_k n_j & \text{sinon} \end{cases}$$

Prop 3 : c_{1n} est calculé en temps $O(n^3)$ par l'algorithme Algo 1 (Figure 1)

3 - Récupération d'une solution optimale

Rq 2 : Par un procédé de mémoïsation, le calcul de c_{1n} nous permet de récupérer une solution réalisant cet optimum.

Prop 4 : L'algorithme Algo 2 (Figure 2) appliqué à la matrice s calculée par Algo 1 donne une solution optimale pour un coût total de $O(n^3)$.

II - Une formalisation

1 - Processus, politiques et valuations

Déf 2 : Un processus est la donnée d'un quadruplet $(n, (E_i)_{i\in[\![0,n]\!]}, D, t)$ où :

- n est un nombre de périodes. On note $0 \ldots n$ les instants correspondants
- $(E_i)_{i\in[\![0,n]\!]}$ partitionne un ensemble E d'états avec $\begin{cases} E_i \text{ l'ensemble des états à l'instant } i \\ E_0 = \{e_0\} \text{ et } E_n = \{e_n\} \text{ sont des singletons} \end{cases}$
- D est un ensemble de décisions. On dispose d'un prédicat $D_e(d) \equiv$ "la décision d peut être prise dans l'état e"
- $t : E \times D \to E$ est une fonction de transition : $\forall i$ si $e \in E_i$ et $D_e(d)$ alors $t(e,d) \in E_{i+1}$

Esc 1 : la multiplication de matrice
$E_i = \{$parenthésages des sous-chaînes de longueur $\leq i\}$
$D_e(d) \equiv$ "d fisce un parenthésage pour toute sous-chaîne de longueur $i+1$ avec des parenthésages de e"

Déf 3: Une politique réalisable de $e_i \in E_i$ à $e_j \in E_j$ est une séquence $(d_{i+1} \ldots d_j)$ de décisions telle que $\forall k$ $D_{e_k}(d_{k+1})$ et $e_{k+1} = t(e_k, d_{k+1})$

On note $P(e_i, e_j)$ l'ensemble des politiques réalisables de e_i à e_j et $P = \bigcup_{e_i, e_j} P(e_i, e_j)$

Déf 4: Une valuation est une fonction $v : P \to O$ où O est un ensemble totalement ordonné.

On s'intéresse au problème d'optimisation suivant: minimiser/maximiser $v(p)$ pour $p \in P(e_0, e_n)$

Ex 2: la multiplication de matrices

$v(d_{i+1} \ldots d_j) = \sum_{k=i+1}^{j} \sum_{m \in d_k} \text{coût}(m)$

où coût(multiplication $(A_r \ldots A_s) \times (A_{s+1} \ldots A_t)$) $= n_{r-1} n_s n_t$

Déf 5: Une valuation v est monotone à droite (resp. à gauche) si $\forall e_i \in E_i, \forall e_j \in E_j, \forall p, p' \in P(e_i, e_j), \forall d \in D$ si $D_{e_j}(d)$ (resp. $\exists e_{i-1} \in E_{i-1}, e_i = t(e_{i-1}, d)$) et $v(p) \leq v(p')$ alors $v(p, d) \leq v(p', d)$ (resp. $v(d, p) \leq v(d, p')$)

Développement 1

Prop 5: Si v est monotone à droite (resp à gauche) alors il existe $p \in P(e_0, e_n)$ optimale, effective.

Rq 3: pour la multiplication de matrices, v est monotone à droite et la solution effective est donnée par Algo 1.

Rq 4: ce formalisme s'étend aux valuations à valeurs dans O partiellement ordonné.

2- Application au problème du sac à dos

Déf 6: On dispose de n objets de valeurs et masses respectives v_i et m_i, $i \in [\![1, n]\!]$. B est une masse limite. On veut choisir des objets afin de maximiser la valeur totale, sans dépasser B.

le processus est défini par:

$E_i = \{ (P, m), P \subseteq [\![1, i]\!]$ est de masse $\leq m \}$

$D = \{1, 0\}$

$$D_{(P,m)}(d) = \begin{cases} \text{vrai si } d = 0 \\ \text{vrai si } d = 1 \text{ et } m + m_{i+1} \leq B \\ \text{faux sinon} \end{cases}$$

On lui attribue la valuation $v(d_i \ldots d_j) = \sum_{d_k = 1} v_k$, monotone à droite et à gauche.

III- Applications de la programmation dynamique

1- Régression linéaire

Déf 7: Soit $(x_1, y_1) \ldots (x_n, y_n)$ un problème de régression linéaire La méthode des moindres carrés vise à trouver la droite d'équation $y = ax + b$ minimisant $\sum_{i=1}^{n} (y_i - ax_i - b)^2$

Rq 5: dans certains cas (Figure 3), définir plusieurs droite peut être souhaitable

Déf 8: Soit $(x_1, y_1) \ldots (x_n, y_n)$ un problème de régression linéaire avec $x_1 < x_2 < \ldots < x_n$.

On souhaite partitionner $[x_1, x_n]$ en segments afin de minimiser:

$$\sum_{s \text{ segment}} \text{erreur des moindres carrés sur } s + C \# \text{segment}$$

avec $C > 0$

Prop 6: On note e_{ij} l'erreur des moindres carrés sur $[x_i ; x_j]$. L'erreur minimale sur $[x_1, x_i]$ est alors

$$c_i = \begin{cases} 0 \text{ si } i = 0 \text{ ou } 1 \\ \min\limits_{j \in [\![1, i]\!]} e_{ji} + C + c_{j-1} \text{ sinon} \end{cases}$$

Cor 2: la programmation dynamique permet de calculer une partition optimale en $O(n^2)$

2_ Plus court chemin dans un graphe

Déf 9: Soit $G = (S, A)$ un graphe orienté et c une fonction de poids sur les arcs ($c : A \to \mathbb{R}$). Soit $(s, p) \in S^2$.
On veut calculer un chemin de poids minimal de s à p, s'il n'y a pas de cycle de poids négatif. Si un tel cycle existe, on veut ~~en~~ calculer un.

Développement 2

Application: l'algorithme de Bellman-Ford

IV_ les processus décision - hasard

1_ Définition

Déf 10: On adapte la définition de processus :
On ajoute des événements liés au hasard $h_i \in H_i$.
les décisions donnent des transitions $e_i \to h_i$.
Le hasard agit à la suite de chaque décision par une transition $h_i \to e_{i+1}$.
E_n n'est plus un singleton : c'est l'ensemble des états possibles à la fin du processus.

Déf 11: Une valuation est une fonction $v : E_n \to \mathbb{O}$.
le problème d'optimisation associé est la minimisation ou la mascimisation de l'espérance : $\mathbb{E}(p) = \sum_{e_n \in E_n} \mathbb{P}_p(e_n) v(e_n)$
où $\mathbb{P}_p(e_n)$ est la probabilité d'atteindre e_n en menant la politique p.

2_ Escemple : contrats d'assurance

Une assurance propose des contrats d'une année.
Un contrat coûte 1, un sinistre non couvert coûte 2.
La première année, la probabilité d'occurrence d'un sinistre est $\frac{3}{5}$.
En cas de sinistre la première année, la probabilité d'un sinistre la deuscième année est $\frac{1}{4}$. Sinon elle est inchangée.

La Figure 4 représente l'arbre correspondant.

Résolution:
$$v(h_i) = \sum_{\substack{e_{i+1} \text{ accessible} \\ \text{depuis } h_i}} \mathbb{P}(h_i \to e_{i+1}) v(e_{i+1})$$

$$v(e_i) = \min_{\substack{h_i \text{ accessible} \\ \text{depuis } e_i}} v(h_i) \quad (*)$$

On calcule une politique minimisant $v(e_0)$ en retenant des décisions réalisant le minimum dans $(*)$.

Ici, on souscrit au contrat la première année, et on ne le fait la deuscième année qu'en cas d'absence de sinistre la première année.

Figure 1 : Algo 1

```
C matrice de taille n×n
s matrice de taille (n-1)×(n-1)
pour i de 1 à n
  cii = 0
pour l de 2 à n                    //l longueur de sous-chaîne
  pour i de 1 à n-l+1
    j = i+l-1
    cij = ∞
    pour k de i à j-1          //parenthésage (Ai...Ak)(Ak+1...Aj)
      q = cik + ck+1j + ni-1 nk nj
      si q < cij        //minimisation et mémoïsation
        cij = q
        sij = k
```

Figure 2 : Algo 2

```
Aff(s, i, j) =
  si i = j afficher Ai
  sinon
    afficher (
    Aff(s, i, sij)
    Aff(s, sij+1, j)
    afficher )
```

Figure 3 :

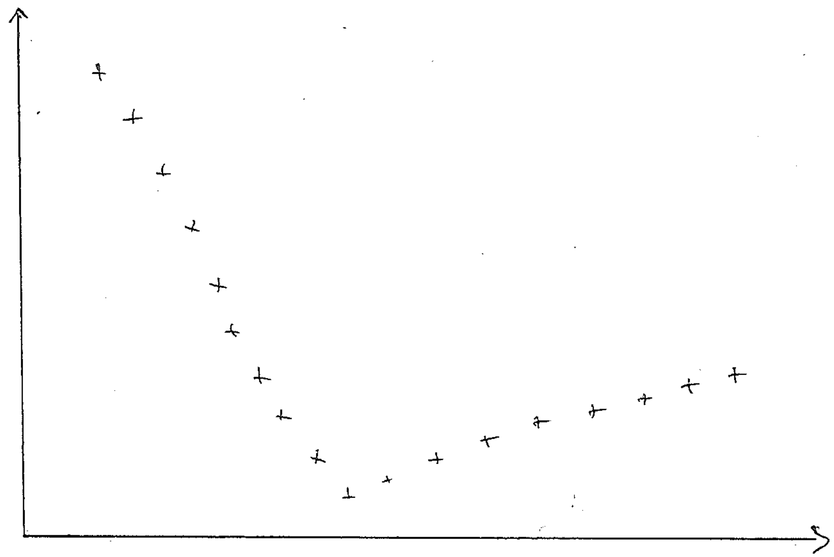
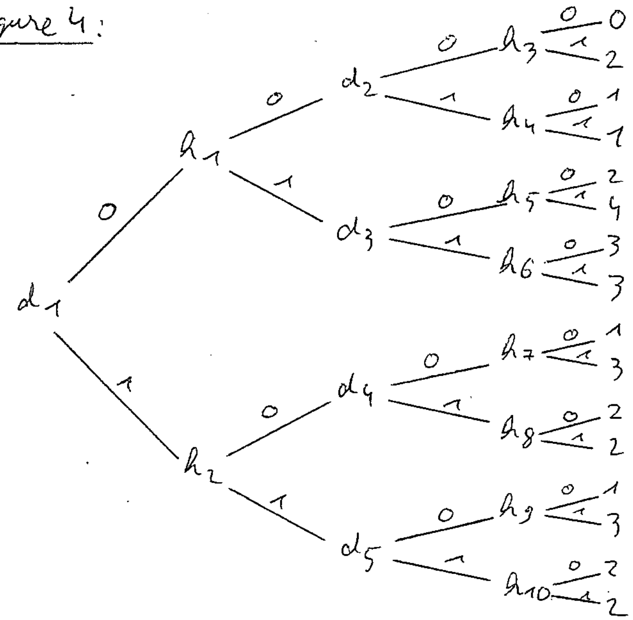


Figure 4 :



Légende :

$h \xrightarrow{1} d$: sinistre
$h \xrightarrow{0} d$: pas de sinistre
$d \xrightarrow{0} h$: pas de contrat
$d \xrightarrow{1} h$: contrat