Programmation Dynamique

Alain Camanes

alain.camanes@free.fr

Stanislas

Option Informatique
2021-2022

Plan

Méthodes de programmation

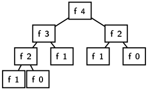
↪ Diviser pour régner. Diviser le problème en sous-problèmes *indépendants*, résoudre les sous-problèmes, combiner les solutions.

↪ Algorithmes gloutons. Optimiser localement chacun des choix. . . en espérant que ce soit le mieux globalement !

↪ Programmation dynamique. Diviser le problème en sous-problèmes *reliés*, mémoriser les solutions des sous-problèmes, combiner les solutions dans le bon ordre.
Bellman 1940

Retour sur Fibonacci

```ocaml
let rec fibo n = match n with
    |0 -> 0 |1 -> 1
    |_ -> fibo (n-1) + fibo (n-2);;
```



Stocker pour Mémoriser !

```ocaml
let rec fibo n =
  let f = Array.make (n+1) (-1) in
  f.(0) <- 0; f.(1) <- 1;
  let rec aux i tab =
    if tab.(i) <> (-1) then tab.(i)
    else (tab.(i) <- aux (i-1) tab + aux (i-2) tab;
          tab.(i))
  in aux n f;;
```

Plan

Découpe de barre

↪ *Barre de métal* de longueur n - Vente à la découpe.

↪ *Prix* en fonction de la longueur de coupe.

ℓ	1	2	3	5	7	11	$\cdots$	i
prix	1	5	6	12	12	13	$\cdots$	$p(i)$

↪ Exemple. $n = 10$.

$p(5) + p(5) = 24.$ 

$p(2) + p(3) + p(5) = 23.$ 

$p(1) + p(2) + p(7) = 18.$ 

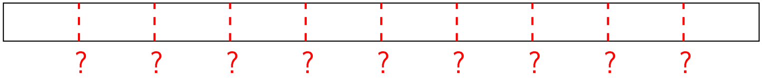
$p(2) + p(2) + p(2) + p(2) + p(2) = 25.$ 

↪ Comment découper la barre pour en tirer un revenu maximum ?

Approche naïve

$\hookrightarrow$ Problème discret.

Toutes les unités : découper *ou* ne pas découper...

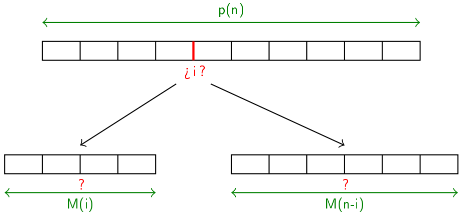


$\hookrightarrow$ Nombre de possibilités.

$$2^{n-1}.$$

Sous-problèmes

$\hookrightarrow$ Couper la barre *en* 2 puis découper les parties *optimalement*.



$\hookrightarrow$ M(n) prix optimal pour une barre de longueur n.

$\hookrightarrow$ Équations.

$$M(n) = \max\left\{\max_{0<i<n}\{M(i)+M(n-i)\}, p(n)\right\}.$$

ou encore en regardant le lieu de la première découpe,

$$M(n) = \max\left\{\max_{0<i<n}\{p(i)+M(n-i)\}, p(n)\right\}.$$

Nouvelle approche naïve

↪ Définition récursive.

```
let rec prix_optimal n p =
  let maximum = ref p.(n) in
    for i = 1 to n do
      maximum := max !maximum
        (p.(i) + (prix_optimal (n-i) p))
    done;
    !maximum;;
```

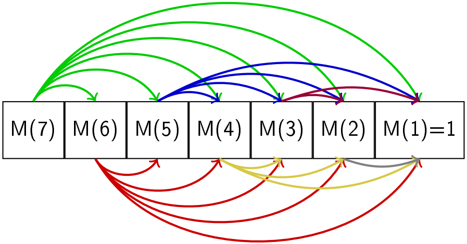
↪ Complexité exponentielle. On calcule plusieurs fois la même quantité.

$$C_{n+1} = 1 + \sum_{k=0}^{n} C_k.$$

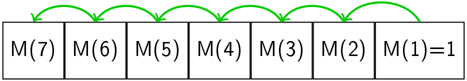
↪ Idée. Dépenser de l'*espace* plutôt que du *temps*.

Barre de longueur 7

↪ Mémoïsation : *de haut en bas.*



↪ Programmation dynamique : *de bas en haut.*



Plan

Mémoïsation - *Top-down*

Approche récursive. Tableau auxiliaire contenant les résultats.

```
let decoupe_mem n p =
  (* r.(i) contient -1 ou M(i) *)
  let r = Array.make (n+1) (-1) in
  (* Cas de base : longueur de barre = 0 *)
  r.(0) <- 0;
  (* Remplissage recursif du tableau *)
  let rec aux i r p =
    (* Si r.(i) <> -1, M(i) a deja ete calcule *)
    if r.(i) >= 0 then r.(i)
    (* Sinon, utilisation de la formule de rec. *)
    else (for j = 1 to i do
            r.(i) <- max r.(i)
              (p.(j) + (aux (i-j) r p))
          done;
          r.(i);)
  (* Appel de la fonction auxiliaire recursive *)
  in aux n r p;;
```

Programmation dynamique (*Bottom-up*)

Approche itérative. Tableau auxiliaire contenant les résultats.

```
let dynamique n p =
  (* r.(i) contient -1 ou M(i) *)
  let r = Array.make (n+1) (-1) in
    (* Cas de base : longueur de barre = 0 *)
    r.(0) <- 0;
    (* Remplissage des cases du tableau *)
    for i = 1 to n do
      (* Utilisation de la formule de rec. *)
      for j = 1 to i do
        r.(i) <- max r.(i) (p.(j) + r.(i-j));
      done;
    done;
  r.(n);;
```

Complexité

$\hookrightarrow$ Temps. $O(n^2)$.

$\hookrightarrow$ Espace. $O(n)$.

Solution optimale

```ocaml
type decoupe = {prix:int array; coupe:int array};;
```

```ocaml
let decoupe_dyn n p =
  let r = Array.make (n+1) (-1)
  and c = Array.make (n+1) 0 in r.(0) <- 0;
  (* c.(i) : 1ere decoupe pour barre de taille i *)
  for i = 1 to n do
    for j = 1 to i do
      if r.(i) < p.(j) + r.(i-j) then
        (r.(i) <- (p.(j) + r.(i-j)); c.(i) <- j;)
    done; done; {prix = r; coupe = c};;
```

```ocaml
(* Abscisses successives de coupe *)
let rec abscisses c lg =
  if c.(lg) = 0 then []
  else c.(lg)::(abscisses c (lg - c.(lg)));;
```

```ocaml
let bar = decoupe_dyn 7 p in abscisses bar.coupe 7;;
```

Programmation dynamique - Résumé

↪ Identifier une *famille* de sous-problèmes.

↪ Identifier les *liens* entre ces sous-problèmes.
Formule de récurrence.

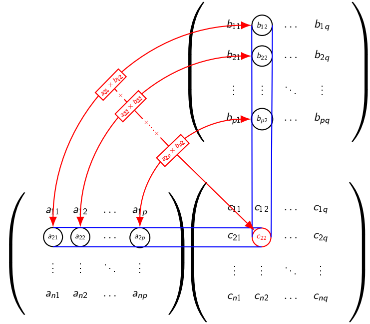
↪ Choix d'un *ordre* de parcours : *Mémoïsation* ou *Prog. dyn.* ?

- Choix du programmeur et du langage.
- Mémoïsation : coût du récursif *mais* ne calcule que le nécessaire.

Plan

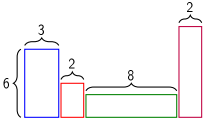
Multiplication matricielle : Comment parenthéser ?

↪ Produit de deux matrices.



Complexité : $O(n \cdot p \cdot q)$.

↪ Exemple.



$$\Big[(6 \times 3)\big[(3 \times 2)(2 \times 8)\big]\Big](8 \times 2) \rightarrow 288$$

$$\Big[\big[(6 \times 3)(3 \times 2)\big](2 \times 8)\Big](8 \times 2) \rightarrow 228$$

$$(6 \times 3)\Big[(3 \times 2)\big[(2 \times 8)(8 \times 2)\big]\Big] \rightarrow 80$$

Méthode naïve

$\hookrightarrow$ Dénombrement du nombre de parenthésages de $n+1$ matrices ($P(0)=0$).

$$P(n) = \begin{cases} 1 & \text{si } n = 0 \\ \sum_{k=0}^{n} P(k)P(n-k) & \text{si } n \geqslant 1 \end{cases}$$

$\hookrightarrow$ Bijection avec les arbres binaires $\rightarrow$ Nombres de Catalan.

$$P(n) = \frac{1}{n+1}\binom{2n}{n} \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}.$$

Découpage

$\hookrightarrow$ Première parenthèse fermante.

$$(A_0 \times \cdots \times A_k) \times (A_{k+1} \times \cdots \times A_{n-1}).$$

$\hookrightarrow$ *Notation.* A_i de dimension $a_i \times a_{i+1}$.

$\hookrightarrow$ Équation.

$$C(i,j) = \begin{cases} 0 \text{ si } i = j \\ \min\limits_{i \leqslant k < j} \{C(i,k) + C(k+1,j) + a_i \cdot a_{k+1} \cdot a_{j+1}\} \end{cases}$$

$\hookrightarrow$ Calcul selon les *$j - i$ croissants*.

$\hookrightarrow$ $s(i,j)$: indice où le maximum est atteint.

$\hookrightarrow$ *Récursion* pour déterminer le parenthésage optimal.

Parenthésage des produits matriciels

```ocaml
let parenthesage a =
  let n = Array.length a - 1 in
  let c = Array.make_matrix n n (-1) and
      s = Array.make_matrix n n (-1) in
  (* Initialisation des couts *)
  for i = 0 to n-1 do c.(i).(i) <- 0 done;
  (* l : distance entre i et j *)
  for l = 1 to n do
    for i = 0 to n-1-l do
      let j = i + l in
      (* Remplissage de la case (i, j) *)
      for k = i to j-1 do
let q = c.(i).(k)+c.(k+1).(j)+a.(i)*a.(k+1)*a.(j+1)
      in (* Comparaison au calcul précédent *)
        if (c.(i).(j) < 0 || q < c.(i).(j)) then
          (c.(i).(j) <- q; s.(i).(j) <- k)
      done; done; done; c, s;;
```

Complexité

$\hookrightarrow$ Espace. Stockage des $C(i,j)$.

$$O(n^2)$$

$\hookrightarrow$ Temps. $C(i,j)$ coûte $O(j-i)$.

$$O(n^3)$$

Plan

An interesting question is, "Where did the name, dynamic programming, come from ?" The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word "research". I'm not using the term lightly ; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term research in his presence. You can imagine how he felt, then, about the term mathematical. [...] What title, what name, could I choose ? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word "programming". I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying. I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.

Richard Bellman, Eye of the Hurricane : An Autobiography (1984)