Mémoïsation

ÉCOLE POLYTECHNIQUE ParisTech

Conception et mise en œuvre d'algorithmes

Programmation Dynamique - leçon 5-2.b

Benjamin Werner

Ecole Polytechnique

Deux approches

ÉCOLE POLYTECHNIQUE ParisTech

equation récursive : $M(n) = \max_{1 \leq p \leq n}(P(p) + M(n-p))$

Récursif

```
public static int max(Integer n, int[] P){
    if (n == 0) { return (0); }
    int max = 0;
    int res = 0;
    for (int j = 1; j <= n; j++) {
      if (j < P.length) {
        res = P[j] + max(n - j, P);
        if (res > max) { max = res; }}}
    return (max);
  }
```

Itératif avec tableau

```
public static int max(Integer n, int[] P){
   int[] max = new int[n+1];
   max[0] = 0;
   for (int i = 1; i <= n; i++) {
     max[i] = 0;
     if (i < P.length) { max[i] = P[i]; }
     for (int j = 1; j < i; j++) {
       if (j < P.length && P[j] + max[i-j] > max[i])
        max[i] = P[j] + max[i-j];
     } }
   return(max[n]);
 }
```

max(n) ⟶ max(n-1) ⟶ max(n-2) ⟶ ... max(0)

max[0] max[1] max[2] ... max[n]

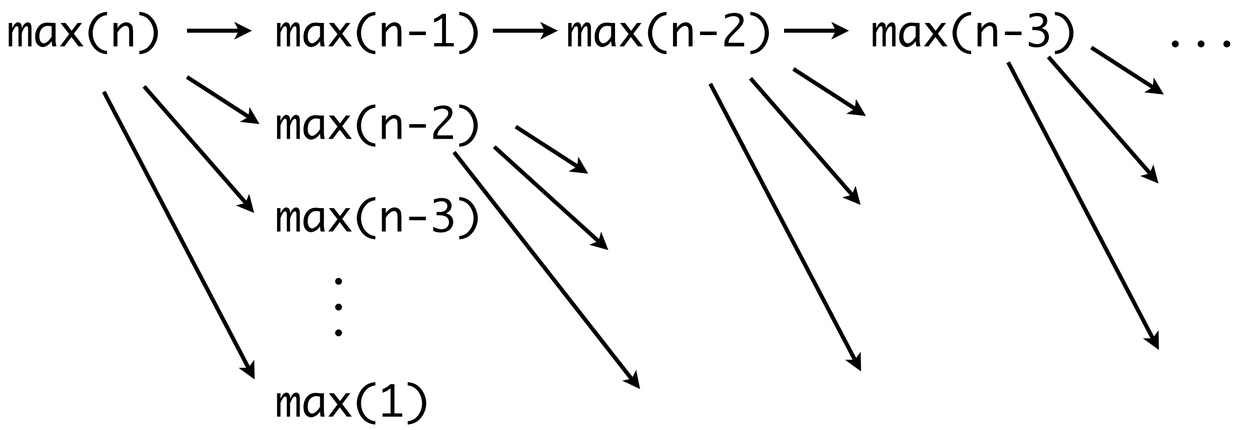
Haut en bas *(top-down)*
descendant

Bas en haut *(bottom-up)*
ascendant

Retour sur le programme récursif

ÉCOLE POLYTECHNIQUE ParisTech

```
public static int max(Integer n, int[] P){
    if (n == 0) { return (0); }
    int max = 0;
    int res = 0;
    for (int j = 1; j <= n; j++) {
      if (j < P.length) {
        res = P[j] + max(n - j, P);
        if (res > max) { max = res; }}}
    return (max);
  }
```



Peut-on ne faire les calculs qu'une fois ?

Retour sur le programme récursif

ÉCOLE POLYTECHNIQUE ParisTech

On a besoin d'un tableau

vaut -1 si pas encore calculé

si déjà calculé, on arrête

```
public static int max(Integer n, int[] P){
  if (n ==0) { return (0);}
    int max = 0;
    int res = 0;
    for (int j = 1; j <= n; j++) {
      if (j < P.length) {
        res = P[j] + max(n - j, P);
        if (res > max) { max = res; }}}
    return (max);
  }
```

Quand un calcul est fait on le note

Peut-on ne faire les calculs qu'une fois ?

```
int[] vmax = new int[m];
for (int i=0; i<m; i++) vmax[i] = -1;


public static int max(Integer n, int[] P){
   if (vmax[n] >=0) return (vmax[n]);
   if (n == 0) return (0);
   int max = 0;
   int res = 0;
   for (int j = 1; j <= n; j++) {
     if (j < P.length) {
       res = P[j] + max(n - j, P);
       if (res > max) { max = res; }}}
   vmax[n] = max;
   return (max);
 }
```

Retour sur le programme récursif

ÉCOLE POLYTECHNIQUE ParisTech

On a besoin d'un tableau

vaut -1 si pas encore calculé

si déjà calculé, on arrête

Quand un calcul est fait on le note

On ne fait les calculs qu'une fois !

```
int[] vmax = new int[m];
for (int i=0; i<m; i++) vmax[i] = -1;



public static int max(Integer n, int[] P){
    if (vmax[n] >=0) return (vmax[n]);
    if (n == 0) return (0);
    int max = 0;
    int res = 0;
    for (int j = 1; j <= n; j++) {
      if (j < P.length) {
        res = P[j] + max(n - j, P);
        if (res > max) { max = res; }}}
    vmax[n] = max;
    return (max);
  }
```

Retour sur le programme récursif

ÉCOLE POLYTECHNIQUE ParisTech

On a besoin d'un tableau

si déjà calculé, on arrête

Quand un calcul est fait on le note

Technique de la mémoïsation

Efficacité comparable à la version itérative/bas-en-haut

```
int[] vmax = new int[m];
for (int i=0; i<m; i++) vmax[i] = -1;



public static int max(Integer n, int[] P){
    if (vmax[n] >=0) return (vmax[n]);
    if (n == 0) return (0);
    int max = 0;
    int res = 0;
    for (int j = 1; j <= n; j++) {
      if (j < P.length) {
        res = P[j] + max(n - j, P);
        if (res > max) { max = res; }}}
    vmax[n] = max;
    return (max);
  }
```

Que choisir ?

ÉCOLE POLYTECHNIQUE ParisTech

Très largement une question de goût

Peut être influencé par le langage de programmation, ou le problème traité

Pour l'ascendant (itératif)

- Un peu plus rapide en général (on ne gère pas de pile récursive)
- Montre clairement le principe de la programmation dynamique
- Un peu plus naturel pour reconstruire la solution

Pour le descendant (récursif avec mémoïsation)

- Elégance
- plus efficace sur certains problèmes, lorsque l'on n'a pas besoin de tous les résultats intermédiaires