# TP : Programmation Dynamique

□ Objectifs

- Répondre à un problème en utilisant la programmation dynamique

□ Pré-requis

- Récursivité
- Programmation dynamique

# 1 Description du problème

On dispose d'une barre de longueur donnée que l'on peut revendre d'un seul tenant ou en morceaux, l'objectif étant de **maximiser le gain**. Par exemple, on dispose d'une barre de longueur 8 et des prix de chaque longueur suivants :

| Longueur du morceau | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Prix du morceau | 0 | 2 | 3 | 8 | 10 | 11 | 15 | 16 | 19 |

En coupant 2 morceaux de longueurs 1 et 7, le gain est de 2 + 16 = 18.

En coupant 3 morceaux de longueurs 1, 2 et 5, le gain est de 2 + 3 + 11 = 16.

**Question 01** Donner une ou plusieurs solutions optimales.

# 2 Détermination d'un algorithme naïf

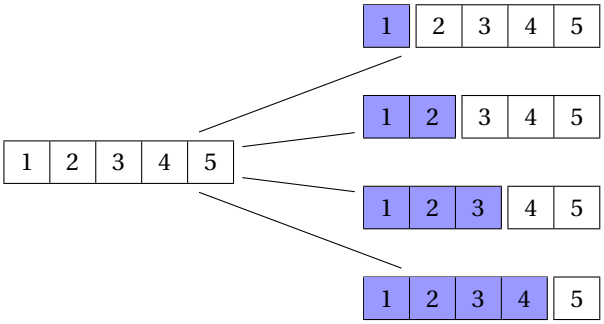
## 2.1 Combien de découpes ?

On cherche dans un premier temps à déterminer l'ensemble des découpes possibles.

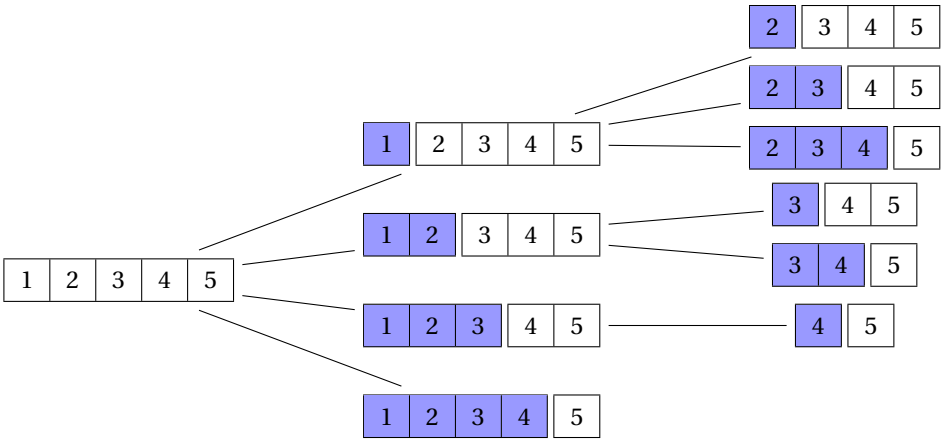
Pour l'exemple, on considère une barre de longueur 5, représentée de la manière suivante : | 1 | 2 | 3 | 4 | 5 |

En découpant une première fois ce morceau de longueur 5, on peut obtenir 2 morceaux de tailles différentes :

- un morceau bleu que l'on conserve
- un morceau blanc que l'on continue à découper, **s'il est de longueur supérieure ou égale à 2**




Une seconde découpe donne les possibilités suivantes :




**Question 02** Construire l'arbre comportant toutes les découpes possibles.

**Question 03** Quelle est la taille de cet arbre ? En déduire le nombre de découpes possibles, et en donner la liste.

**Question 04** Pour une barre de longueur $n$, combien y a-t-il de découpes possibles ?

## Calcul du meilleur gain

Le calcul du gain maximum se fait récursivement : pour chaque morceau découpé (ce qui correspond à un noeud sur l'arbre précédent), le meilleur gain est égal au maximum, pour chaque découpe possible, de la somme du prix de la partie découpée et du meilleur gain de ce qui reste à découper.

Prenons pour exemple le cas où l'on coupe un premier morceau de longueur 1 : | 1 | 2 | 3 | 4 | 5 |
Le gain maximum suite à cette découpe est égal à la somme :

- du prix de la partie découpée, soit le prix d'un morceau de longueur 1 : | 1 |
- du meilleur gain de la partie restante, de longueur 4 : | 2 | 3 | 4 | 5 |

L'algorithme naïf qui en découle est le suivant. Il fait intervenir une fonction récursive `COUPE(L, prix)` dont le rôle est de retourner le gain maximum pour une barre de longueur `L` avec une liste de prix `prix` :

```
L : longueur de la barre
prix : une liste de prix pour chaque longueur

COUPE(L, prix):
  Si L <= 0:
    Retourner 0

  meilleur = 0

  Pour decoupe allant de 1 à L:
    meilleur = max(meilleur, prix[decoupe] + coupe(L - decoupe, prix))

  Retourner meilleur
```

**Question 05** Appliquer cet algorithme à la main sur l'exemple avec une barre de longueur 5, et les prix donnés au début du TP. On indiquera la valeur du gain maximum sur chaque noeud de l'arbre fait à la question 2).

**Question 06** Pour une barre de longueur 5, combien d'appels récursifs sont effectués au total ? combien d'appels récursifs successifs au maximum ?

**Question 07** Même question pour une barre de longueur $n$.

**Question 08** Si l'on double la longueur de la barre (qui passe d'une longueur $n$ à une longueur $2n$), par quel nombre multiplie-t-on le nombre total d'appels récursifs ?

**Question 09** Que peut-on dire de l'efficacité de cet algorithme ?

**Question 10** Coder la fonction `coupe` avec Python.

**Question 11** Vérifier les résultats des questions 1) et 5) en exécutant la fonction `coupe` .

# 3 Amélioration de l'algorithme

Afin de réduire le nombre d'appels récursifs et donc le temps de calcul, on se propose de coder la fonction `coupe` de façon dynamique :

- On initialise une liste de même taille que la liste des prix, qui contiendra les gains maximum pour une longueur donnée
- Lors de l'appel à la fonction `coupe`, on vérifie sur le gain maximum pour la longueur `L` donnée n'est pas déjà calculée dans cette liste, auquel cas on la retourne sans calcul supplémentaire
- Sinon, on calcule ce gain maximum et on le stocke dans la liste

**Question 12** Coder la fonction `coupe` en utilisant la programmation dynamique (version *Top Down* ou *Bottom Up*, au choix).

**Question 13** ***(Bonus)*** Écrire une fonction `decouper(L, prix)` qui affiche la découpe maximisant le gain.

```
>>> decouper(5, [0, 2, 3, 8, 10, 11, 15, 16, 19])
Gain maximum : 12
1 morceau de longueur 1 (gain : 1 x 2 = 2)
1 morceau de longueur 4 (gain : 1 x 10 = 10)
>>> decouper(8, [0, 2, 3, 8, 10, 11, 15, 16, 19])
Gain maximum : 20
2 morceaux de longueur 4 (gain : 2 x 10 = 20)
```