Programmation dynamique

Exercice 1

Q 1.1

$\text{nb}(1) = 1, \text{nb}(2) = 2$

Q 1.2

- Si n est en monome:
  Alors tous les autres élèves peuvent être dans les $\text{nb}(n-1)$ autre configurations
- Si n est en binome:
  Alors il est nécessairement avec un autre élève. Il y aura alors $\text{nb}(n-2)$ configurations pour les autres élèves. Il peut e mettre avec (n-1) élèves, d'où le résultat.

Q 1.3

On en déduit que:

$$\text{nb}(n) = \text{nb}(n-1) + (n-1) \times \text{nb}(n-2)$$

Q 1.4

```
t = [1, 2]

def nb(n):
  if len(t) > n:
    return t[n]

  tmp = nb(n-1) + (n-1) * nb(n-2)
  t.append(tmp)
  return tmp


def nb(n):
  t = [1, 2] + [0] * (n-2)

  for i in range(3, n+1):
    t[i] = t[i-1] + (i-1) * t[i-2]
  return t[n]
```

Exercice 2

Q 2.1

Avec valeurs de l'exemple, $n = 4$, stratégie gloutonne donne $8 \times 1 + 1 \times 1 = 9$ alors que la meilleure option est $5 \times 2 = 10$

Q 2.2

1.

$r(i)$ le morceau maximal que l'on peut obtenir pour une corde de i mètres

2.

$r(i) = \max_{1 \leq j \leq i} r(i-j) + p_j$, en initialisant $r(0) = 0$

3.

```
DecoupeCorde(p,n)
  Entrée p[1..n] un tab de prix, avec p[i] = p_i
  sortie: Découpe optimale à partir d'une corde de longueur n
  L[0] <- 0
  pour i allant de 1 à n faire:
    r[i] <- - inf
    pour j allant de 1 à i faire:
      r[i] <- max(r[i]n r[i - j] + p[j])
  renvoyer p[n]
```

Q 2.3


```
DecoupeCorde(p,n)
  Entrée p[1..n] un tab de prix, avec p[i] = p_i
  sortie: Découpe optimale à partir d'une corde de longueur n
  L[0] <- 0
  pour i allant de 1 à n faire:
    r[i] <- - inf
    pour j allant de 1 à i faire:
      r[i] <- max(r[i]n r[i - j] + p[j])
  renvoyer p[n]
```


Exercice 3

$\mathrm{opt}(n) = 0$

Q 3.1

$\mathrm{opt}(i) = \min_{i<j\leq n}\{c_{ij} + \mathrm{opt}(j)\}$

Q 3.2

```
pour i allant de n-1 à 1 faire
  opt[i] := + inf
  pour j allant de i + 1 à n faire
    si c [i, j] + opt[j] < opt[i] alors
      opt[0] := c[i, j] + opt[j]
```

$O(n^2)$

Q 3.3

opt[n] := 0, S[n] := n

```
pour i allant de n-1 à 1 faire
  opt[i] := + inf
  pour j allant de i + 1 à n faire
    si c [i, j] + opt[j] < opt[i] alors
      opt[0] := c[i, j] + opt[j]
      s[i] := j
```

Exercice 4

Q 4.1