



ELECOM
Paris



IP PARIS

TÉLÉCOM PARIS X

TÉLÉCODE

SÉANCE 3 | TÉLÉCODE

PREFIX SUMS & ALGORITHME DE KADANE

Présenté et rédigé par Léopold Bernard

PLAN

01
02
03
04
05
06

PRÉSENTATION DU PROBLÈME
ALGORITHMES NAÏFS
OPTIMISATION
IMPLÉMENTATION EN PYTHON
ALGORITHME DE KADANE
PROBLÈMES

1. PRÉSENTATION DU PROBLÈME

- Festival dure N jours (numérotés 1, ..., N)
- Tableau p : $p[i]$ personnes qui viennent le jour i
- M requêtes $\text{COMBIEN}(i, j) \rightarrow$ combien de personnes sont venues au festival entre le jour i et le jour j (j inclus)



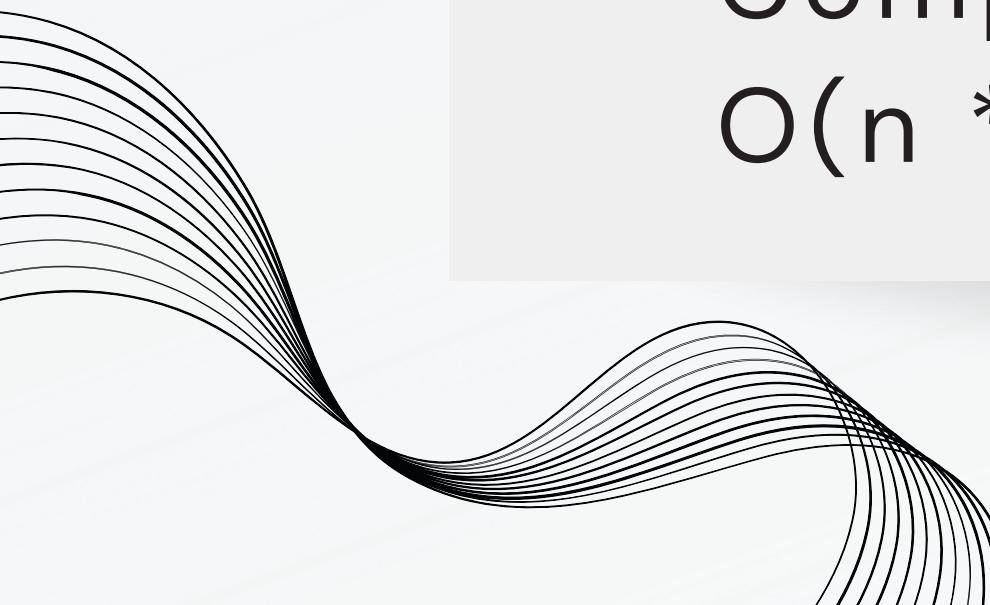
2. ALGORITHME NAÏF N°1

- Pour chaque requête COMBIEN(i, j) :
 - $S = 0$
 - Pour $k = i$ à j (inclus) :
 - $S += p[k]$
 - Renvoyer S
- Complexité spatiale : $O(n) \rightarrow OK$
- Complexité temporelle : $O(m * n) \rightarrow NON$



2. ALGORITHME NAÏF N°2

- Dans un tableau $M[i][j]$, pré-calculer les requêtes (i, j) “intelligemment” :
 - $M[i][i] = p[i]$
 - $M[i][j+1] = M[i][j] + p[j+1]$
- Complexité spatiale : $O(n * n) \rightarrow$ NON
- Complexité temporelle : initialisation en $O(n * n)$, requête en $O(1)$



3. OPTIMISATION

- Idée fondamentale :
 - $CB(i, j) = CB(0, j) - CB(0, i-1)$
- Le nombre de personnes qui sont venues entre les jours i et j sont celles qui sont venues depuis le début, dont on enlève celles qui sont venues avant le jour i
- Il suffit de garder un tableau des $CB(0, k)$



3. OPTIMISATION

- Au jour 0, personne n'est encore venu, donc on a $CB(0, 0) = 0$
- On construit le tableau prefix_CB :
 - $\text{prefix_CB}[0] = 0$
 - $\text{prefix_CB}[i+1] = \text{prefix_CB}[i] + p[i]$
- C'est un tableau des effectifs cumulés croissants (cf. statistiques) = prefix sums



4. IMPLÉMENTATION EN PYTHON

```
prefix_cb = [0] * (n+1)
for i in range (n) :
    prefix_cb[i+1] = prefix_cb[i] + p[i]

def combien(i, j) :
    return prefix_cb[j]-prefix_cb[i-1]
```

5. ALGORITHME DE KADANE

- Problème :
 - Entrée : liste L (de réels)
 - Objectif : trouver la **sous-séquence continue de somme maximale** dans L
- On peut le faire en un seul parcours de liste sans rien modifier à la liste originale
- COMMENT ? 5 minutes pour réfléchir



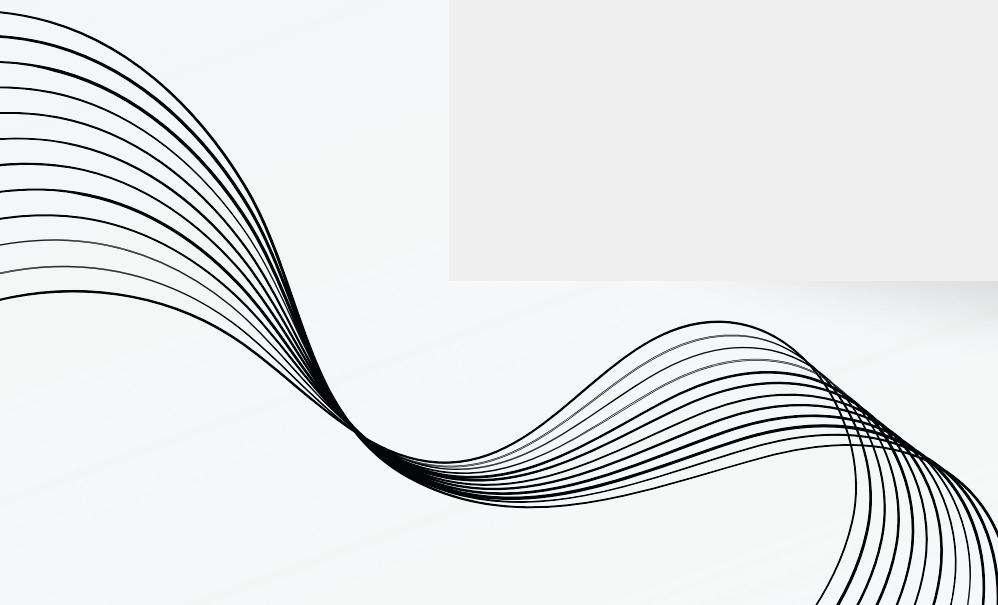
5. ALGORITHME DE KADANE

- Idée :
 - Garder une somme S en cours de calcul et un maximum M
 - Relancer un calcul de somme si on arrive à une somme actuelle négative, c'est-à-dire $S = \max(S, 0)$ à chaque itération
 - Toujours mettre à jour le max, $M = \max(M, S)$



5. KADANE : EXEMPLE

- Il y a 2 problèmes différents (à quelques cas près) selon si on a le droit de prendre une liste vide ou non (gestion des listes composées uniquement de négatifs différente)
- Exemple au tableau avec :
 - $L = [3, 1, -2, 4, -7, 9, -3, 4, 1]$



5. KADANE : IMPLÉMENTATION

- Complexité spatiale : $O(n)$ ($O(1)$ additionnelle)
- Complexité temporelle : $O(n)$

```
def kadane_with_empty(L, n) :  
    S, M = 0, 0  
    for i in range (n) :  
        S += L[i]  
        M = max(M, S)  
        S = max(S, 0)  
    return M  
  
def kadane_without_empty(L, n) :  
    S, M = 0, L[0]  
    for i in range (n) :  
        S += L[i]  
        M = max(M, S)  
        S = max(S, 0)  
    return M
```

6. PROBLÈMES

Leetcode :

53. Maximum subarray

1749. Maximum absolute sum of any subarray

Codeforces :

All pair segments : 2019/B

Password cracking : 2013/C

Maximize the root : 1997/D

Even positions : 1997/C