Matthieu Boyer
Planche A

Sommes Partielles

1. Étant donné un tableau T trié dans l'ordre croissant, donnez un algorithme permettant de calculer la somme de toutes les sommes de sous-tableaux de T. On ne devra parcourir le tableau qu'une seule fois.
2. Donnez un algorithme calculant le nombre de sous-tableaux ayant somme k. On ne parcourra le tableau qu'un seule fois.

Solution: Sommes Partielles

1.
```python
def subarray(arr):
        n = len(arr)
        result = 0
        for i in range(n):
                result += arr[i] * (i + 1) * (n - i)
        return result
```

2.
```python
def subarraysum(arr, k):
        prev_sums = dict()
        res = 0
        curr = 0
        for i in range(n):
                curr += arr[i]
                if currsum == k:
                        res += 1
                if (curr - k) in prev_sums:
                        res += prev_sums[curr - k]
                prev_sums[curr] = prev_sums.get(curr, 0) + 1         return res
```

Matthieu Boyer
Planche B

Évaluation de Polynômes

Un polynômes est une fonction de la forme $P = x^n + \sum_{i=0}^{n-1} c_i x^i$. Soit $n = 2^k$. Soit $a_i \in \mathbb{R}$ pour $0 \leq i \leq n-1$.

1. Donnez une méthode pour évaluer $P(a_i)$ en $\mathcal{O}(n)$ opérations. Combien d'opérations seraient nécessaires pour calculer $P(a_0), \ldots, P(a_{n-1})$?
2. Considérez l'arbre dont les feuilles sont associées aux polynômes $x - a_j$ et les noeuds sont les produits des polynômes fils. Donnez un algorithme pour construire l'arbre en $\mathcal{O}(M(n) \log n)$ où $M(n)$ est le temps requis pour faire la multiplication de deux polynômes de degré n.
3. Donnez un algorithme calculant $P \mod P_u$ pour chaque noeud u de l'arbre ci-dessus en $\mathcal{O}(M(n) \log n)$.
4. En déduire un algorithme qui calcule tous les $P(a_i)$ en temps $\mathcal{O}(M(n) \log n)$.

Solution: Évaluation de Polynômes

Voir `https://mathexp.eu/bostan/publications/BoSc05.pdf` preuve de la première partie du théorème 1.

Matthieu Boyer
Planche C

MCBM

On considère le problème suivant :

Minimum Change Binary Matrix	
Entrée:	$M \in \mathcal{M}_{n,m}(\{0,1\}$
Sortie:	Le nombre minimum de coefficients à changer pour que chaque ligne et chaque colonne ait un nombre pair de 1.

Donnez un algorithme résolvant ce problème en temps linéaire.

Skyline

La skyline d'une ville est le contour extérieur de la ville quand on la regarde à distance. Supposons qu'on vous donne les localisations et hauteurs de tous les batîments (sous la forme (x_1, x_2, h)). Donnez un algorithme récursif qui renvoie la skyline des batiments.

Solution: MCBM

Il faut ajouter/enlever au plus un 1 par ligne et un 1 par colonne. On cherche les colonnes et les lignes ayant un nombre impair de 1. Il y en a le même nombre. On change la valeur aux intersections.

Solution: Skyline

L'algorithme est le suivant :

1. S'il n'y a qu'un seul bâtiment, on renvoie les coordonnées des bords gauches et droites du bâtiment avec la valeur de hauteur.
2. Sinon, on sépare la liste en deux et on s'appelle récursivement sur chaque côté.
3. On fusionne les deux listes en itérant sur chaque liste et en mettant à jour les hauteurs en cours pour chaque batiment rencontré.