Matthieu Boyer
Planche A

Primauté

On note $\mathcal{P}$ l'ensemble des nombres premiers.

1. En remarquant que $p \in \mathcal{P}$ si et seulement si $\forall 1 < k < p, k \not| p$, donnez un algorithme en $\mathcal{O}(n)$ vérifiant si $p \in \mathcal{P}$
2. Montrez que $p \in \mathcal{P}$ si et seulement si $\forall 1 < k < \sqrt{p}, k \not| p$. Déduisez-en un algorithme en $\mathcal{O}(\sqrt{n})$ vérifiant si $p \in \mathcal{P}$.
3. En utilisant le petit théorème de Fermat : si p est premier, alors pour tout $1 \leq a < p, a^{p-1} = 1 \mod p$, donnez un algorithme vérifiant si p est premier. Quelle est sa complexité ? Que ce passe-t-il si on choisit de fixer a ?
4. On note que $B = \left\{a \mid a^{N-1} = 1 \mod N\right\}$ est un sous-groupe de $\mathbb{Z}/N\mathbb{Z}^\times$, et donc $|B| \leq N - 1/2$. Montrer que dans ce cas, on a une probabilité d'erreur plus petite que $\frac{1}{2^k}$.

Solution: Primauté

Matthieu Boyer
Planche B

Mon beau sapin

On vous donne un tableau T tel que $T_i = (x_i, y_i)$ représente la position de l'arbre i dans votre jardin. Vous souhaitez mettre une clôture autour de votre de jardin avec le moins de corde possible. Le jardin est bien clos si tous les arbres sont à l'intérieur de la cloture.

1. Donnez l'ensemble des coordonnées d'arbres qui sont exactement sur le bord de la cloture.
2. Comment adapteriez-vous votre algorithme si vos arbres peuvent ne pas être au sol ?

Solution: Mon beau sapin

On fait un parcours de Graham :

```
def compare(p1, p2, p3): \# O(1)
        return (p3[1] - p2[1])*(p2[0]-p1[0]) - (p2[1]-p1[1])*(p3[0]-p2[0])

def outerTrees(trees):
        upper, lower = [], []
        for point in sorted(trees): \# O(nlog n)
                while len(lower) >= 2 and compare_slopes(lower[-2], lower[-1], point) < 0: lower.po
                while len(upper) >= 2 and compare_slopes(upper[-2], upper[-1], point) > 0: upper.po
                lower.append(tuple(point))
                upper.append(tuple(point))
        return list(set(lower + upper))
```

Le parcours de Graham s'adapte aisément pour de plus grandes dimensions en considérant des déterminants. On pourrait aussi regarder l'algorithme QuickHull `https://en.wikipedia.org/wiki/Quickhull`

Matthieu Boyer
Planche C

Demain il pleut

Étant donné une matrice 2D W de taille $m \times n$ qui représente une carte constituées de cases de terre et d'eau ($W_{i,j} = 0$ si $W_{i,j}$ est de la terre, sinon $W_{i,j} = 1$), on cherche à assigner une hauteur à chaque case pour obtenir le plus haut pic possible sur la carte. L'assignation h doit vérifier :

- $h_{i,j} \geq 0$
- La hauteur d'une case d'eau est de 0
- La différence de hauteur entre deux cases qui partagent un côté ne peut pas être plus grande que 1.

Renvoyez la hauteur maximale. Quelle est la complexité de votre algorithme ?

Solution: Demain il pleut

On peut procéder en faisant un Parcours en Largeur sur le graphe d'adjacence depuis chaque case d'eau, ou en faisant de la programmation dynamique.