TD 01 – Introduction

Exercice 1. *Cherchez la star...*

Dans un groupe de n personnes (numérotées de 1 à n pour les distinguer), une *star* est quelqu'un qui ne connaît personne mais que tous les autres connaissent. Pour démasquer une star, s'il en existe une, vous avez juste le droit de poser des questions à n'importe quel individu i du groupe, du type « est-ce que vous connaissez j ? ». On suppose que les individus répondent toujours la vérité. On veut un algorithme qui trouve une star s'il en existe une, et qui garantit qu'il n'y en a pas sinon, en posant le moins de questions possibles.

1. Combien peut-il y avoir de stars dans le groupe ?
2. Écrire le meilleur algorithme que vous pouvez et donner sa complexité en nombre de questions (on peut y arriver en $\mathcal{O}(n)$ questions).
3. Donner une borne inférieure sur la complexité (en nombre de questions) de tout algorithme résolvant le problème.
   (*Difficile*) Prouver que la complexité exacte de ce problème est $3n - \lfloor \log_2(n) \rfloor - 3$.

Exercice 2. *Le grand saut*

Le problème est de déterminer à partir de quel étage d'un immeuble, sauter par une fenêtre est fatal. Vous êtes dans un immeuble à n étages (numérotés de 1 à n) et vous disposez de k étudiants. Les étudiants sont classés par notes de partiel croissantes. Il n'y a qu'une opération possible pour tester si la hauteur d'un étage est fatale : faire sauter le premier étudiant de la liste par la fenêtre. S'il survit, vous pouvez le réutiliser ensuite (évidemment, l'étudiant survivant reprend sa place initiale dans la liste triée), sinon vous ne pouvez plus.
Vous devez proposer un algorithme pour trouver la hauteur à partir de laquelle un saut est fatal (l'algorithme doit renvoyer $(n+1)$ si on survit encore en sautant du n-ème étage) en faisant le minimum de sauts.

1. Si $k \geq \lceil \log_2(n) \rceil$, proposer un algorithme en $\mathcal{O}(\log_2(n))$ sauts.
2. Si $k < \lceil \log_2(n) \rceil$, proposer un algorithme en $\mathcal{O}(k + \frac{n}{2^{k-1}})$ sauts.
3. Si $k = 2$, proposer un algorithme en $\mathcal{O}(\sqrt{n})$ sauts.

Exercice 3. *Bricolage*

Dans une boîte à outils, vous disposez de n écrous de diamètres tous différents et des n boulons correspondants. Mais tout est mélangé et vous voulez appareiller chaque écrou avec le boulon qui lui correspond. Les différences de diamètre entre les écrous sont tellement minimes qu'il n'est pas possible de déterminer à l'œil nu si un écrou est plus grand qu'un autre. Il en va de même avec les boulons. Par conséquent, le seul type d'opération autorisé consiste à essayer un écrou avec un boulon, ce qui peut amener trois réponses possibles : soit l'écrou est strictement plus large que le boulon, soit il est strictement moins large, soit ils ont exactement le même diamètre.

1. Écrire un algorithme simple en $\mathcal{O}(n^2)$ essais qui appareille chaque écrou avec son boulon.
2. Supposons qu'au lieu de vouloir appareiller tous les boulons et écrous, vous voulez juste trouver le plus petit écrou et le boulon correspondant. Montrer que vous pouvez résoudre ce problème en moins de $2n - 2$ essais.
3. Prouver que tout algorithme qui appareille tous les écrous avec tous les boulons doit effectuer $\Omega(n \log n)$ essais dans le pire des cas.

Problème ouvert : proposer un algorithme en $o(n^2)$ essais pour résoudre ce problème.