Introduction à l'algorithmique **Hiver 2024**

IFT2125-6001 Évaluation (7.5%)

Devoir 2 - L'eau, le feu et les nombres premiers

Consignes générales : À faire en équipe de 1 à 2 personnes. Remettre un seul pdf et les fichiers de code python et C++. Indiquez vos noms et matricules sur tous les fichiers (pdf et code).

Votre code doit obligatoirement passer les tests fournis avec le devoir, sinon une note de 0 sera attribué aux exercices dont les tests de base ne sont pas réussis. D'autres tests seront rajoutés lors de la correction. Seulement les librairies standards de Python et C++ sont permises, sauf si mentionné autrement.

Il va de soit que le code doit être clair, bien indenté et bien commenté.

1 Résolution de récurrences (25 points)

Question 1: (10 points) Montrez les démarches complètes qui permettent de trouver le polynôme caractéristique et les racines (avec leurs multiplicités) de la récurrence ci-dessous. Après avoir trouvé les racines, écrivez la forme générale de la récurrence avec des constantes génériques c_i.

$$t_n = 2t_{n-1} - 2t_{n-2} + 5n2^n$$

Indice : commencer par multiplier la récurrence par une constante et changer n par $n-1$

Attention : Vous ne devez pas répondre en utilisant simplement la formule en page 126 du livre de référence Bratley-Brassard, mais servez vous en pour valider votre propre réponse.

Question 2: (10 points) Résoudre la récurrence suivante exactement pour les puissances de 2 seulement, sans utiliser le théorème maître :

$$T(n) = \begin{cases} 0 & \text{si } n = 0, \\ 2 & \text{si } n = 1, \\ 9T(\frac{n}{2}) + 8(\frac{n}{2})^2 & \text{sinon} \end{cases}$$

Attention : Vous ne pouvez pas utiliser le théorême maître, mais servez vous en pour valider votre propre réponse.

Question 3: (5 points) Utiliser le théorème maître pour trouver l'ordre exacte des récurrences suivantes :

1. $t(n) = t(\frac{n}{2}) + 2n$
2. $t(n) = 3t(\frac{n}{2}) + 3n$
3. $t(n) = 4t(\frac{n}{2}) + n^2$
4. $t(n) = 2t(\frac{n}{2}) + 2n$
5. $t(n) = 2t(\frac{n}{3}) + 2n$

2 Nombres premiers - Code C++ (25 points)

Dans ce problème, on vous demande de trouver des séquences spéciales de nombres premiers en utilisant le langage C++. Vous pouvez bien sur réutiliser votre code du devoir 1 pour générer les nombres premiers. Votre algorithme devrait être efficace pour avoir tous les points.

Soit une séquence $S = S_0, S_1, ..., S_k$. La séquence S est spéciale si :

1. La séquence S est composé de nombres premiers seulement,
2. les termes de la séquence S sont ordonnés par ordre croissant,
3. $S_i = S_0 + i \times k$ où $k \in \mathbb{N}$ est une constante pour la séquence, et
4. tous les termes sont des permutations de chaque autre terme, en observant les chiffres.

Par exemple, la séquence $11483, 14813, 18143$ est spéciale car :

1. $11483, 14813, 18143$ sont tous des nombres premiers,
2. $11483, 14813, 18143$ sont ordonnés par ordre croissant,
3. $S_i = 11483 + i \times 3330$ où $3330 \in \mathbb{N}$: $11483 + 1 \times 3330 = 14813$ et $11483 + 2 \times 3330 = 18143$, et

4. tous les termes sont des permutations de $1, 1, 3, 4$ et 8.

On vous demande de trouver toutes les séquences de taille $|S| = 3$ contenant des termes strictement inférieurs à N. Pour $N = 10000$, il y a 2 séquences spéciales possibles et pour $N = 20000$, il y a 4 séquences spéciales possibles.

Code
Exemple d'appel :

```
Compilation :
g++ -o prime_numbers_sequences.exe prime_numbers_sequences.cpp
    PrimeSequenceCalculator.cpp

Execution :
.\prime_numbers_sequences.exe 20000
```

Remise
Compléter les fichiers *PrimeSequenceCalculator.cpp* et *PrimeSequenceCalculator.h* fournis, et ne remettre **uniquement** que ces fichiers. Ne **pas** remettre le fichier *prime_numbers_sequences.cpp*.

3 Ponts et piliers - Code Python (25 points)

Un pont tranversant une rivière est en cours de construction. Le pont aura une longueur totale de L mètres. Les piliers de support du pont doivent être espacés d'au moins D mètres (mesuré de milieu de pillier à milieu de pillier), pour laisser passer les bateaux et ne pas trop perturber le courant d'eau. Les piliers doivent aussi être espacés d'au moins C mètres du début et la fin du pont (mesuré du milieu du pillier au point de début ou de la fin), pour laisser la berge libre et laisser de la place à la nature. L'entreprise qui travaillait sur le projet a fait faillite et vous êtes en charge de continuer le projet. Cette entreprise a déjà posé quelque piliers (N) sans ordre précis, respectant tout de même les contraintes. Vous devez calculer le nombre maximal de piliers qui peuvent être rajouté dans la construction selon les données obtenues. On considère que $L, D, C \in \mathbb{N}^{\geq 1}$ et $N \in \mathbb{N}^{\geq 0}$.

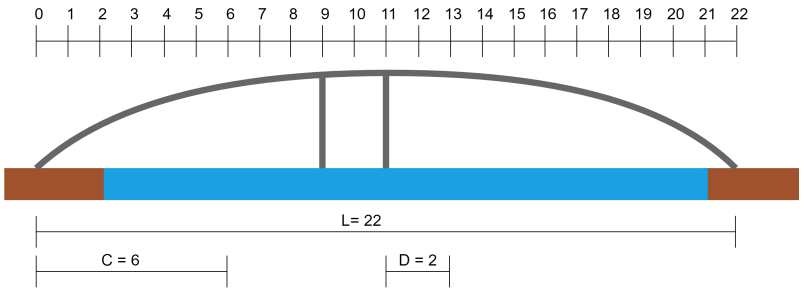
Données :

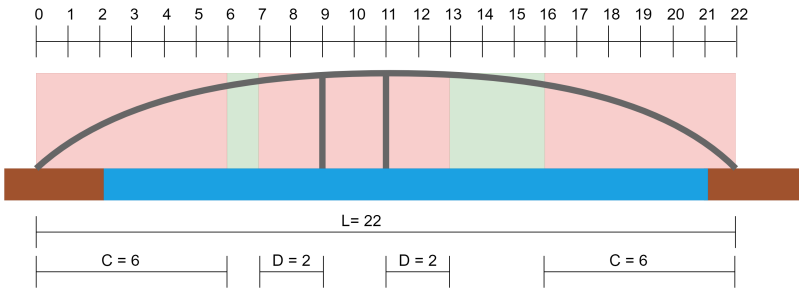
```
<longueur L> <distance entre piliers D> <distance entre piliers et les extremites C> <nb piliers déjà installés N>
<position du pilier 1>
<position du pilier 2>
...
<position du pilier N>
```

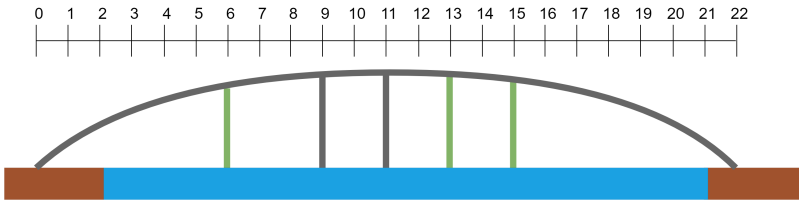
Exemples :

```
Exemple 1 :
22 2 6 2
11
9
--> doit donner 3 piliers additionnels

Exemple 1 :
47 5 6 0
--> doit donner 8 piliers additionnels
```







Code

Votre code doit lire un fichier spécifié en premier argument qui contient un problème et écrire la réponse dans un fichier spécifié en second argument.

Exemple d'appel :

```
python3 ponts_et_piliers.py input.txt output.txt
```

Un fichier *test_ponts_et_piliers.py* et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.

Exemple d'appel :

```
python3 test_ponts_et_piliers.py
```

Remise
Compléter le fichier *ponts_et_piliers.py* fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier *test_ponts_et_piliers.py*, ou n'importe quel autre fichier de test.

Bonus 10%
Nous allons accorder un bonus maximal de 10% (pour le devoir) aux équipes qui soumettent un code qui passe les tests sur Kattis pour le problème "Birds on a Wire". Une personne par équipe suffit. Votre nom doit apparaitre sur la liste (si il y a utilisation d'un pseudonyme, venir montrer à l'équipe d'enseignement que c'est votre compte.) Notez qu'il faudra modifier un tout petit peu le code pour répondre au problème sur Kattis, dont la lecture du problème qui se fait avec des input(), l'écriture de la réponse qui se fait en print() puis un paramètre du problème qui est fixe.

4 Chalet et poêle à bois - Code Python (25 points)

Vous et votre groupe allez au chalet pour J jours. Pour alimenter votre poêle à bois, vous avez accès à une réserve de bois contenant S sacs de bois. $S \leq J$. Chaque sac contient un certain nombre de bûches : $W[i]$ bûches pour le i-ème sac. La réserve de bois étant loin du chalet, vous ne pouvez prendre qu'un seul sac de bois par jour, au début de journée, peut-importe le nombre de bûches qu'il contient. À chaque cueillette, vous pouvez choisir le sac que vous voulez parmi les sacs disponibles restants. De plus, vous pouvez aller chercher un nouveau sac de bois seulement quand le dernier sac pris a été complètement vidé à la fin de la journée précédente. Pour pouvoir bien profiter de la chaleur du poêle à bois, vous devez décider quel est le nombre (constant, non variable) de bûches B que vous et votre groupe peuvent brûler par jour. Si vous brûlez votre bois trop lentement, il y aura du bois restant à la fin de votre séjour, mais si vous brûlez trop rapidement votre bois, il va manquer du bois dans les derniers jours. On cherche alors à trouver la meilleure cadence constante B (bûches/jour) $\in [1, max\{W\}]$ qui permettra de brûler tout le bois, le plus lentement possible. Il est évident que certaines journées, vous n'allez pas brûler B bûches s'il ne reste qu'un nombre inférieur à B de bûches. On peut

prendre pour acquis que tous les nombres considérés dans le problème sont des entiers ≥ 1.

Votre algorithme doit s'exécuter en $\mathcal{O}(S \times \log(max\{W\}))$. Votre algorithme ne doit pas s'exécuter en $\mathcal{O}(S \times max\{W\})$ avec une simple boucle qui teste tous les B possibles de 1 à $max\{W\}$.

Données :

```
<nb total de jours : J>
<nb de bûches dans le sac 0 : W[0]> <nb de bûches dans le sac 1 : W[1]> ... <nb de bûches dans le sac  S-1 : W[S-1]>
```

Exemples :

```
Exemple 1 :
8
11 3 6 7
--> doit donner B = 4 bûches par jour

Exemple 2 :
5
23 11 30 4 20
--> doit donner B = 30 bûches par jour
```

Code
Votre code doit lire un fichier spécifié en premier argument qui contient un problème et écrire la réponse dans un fichier spécifié en second argument.

Exemple d'appel :

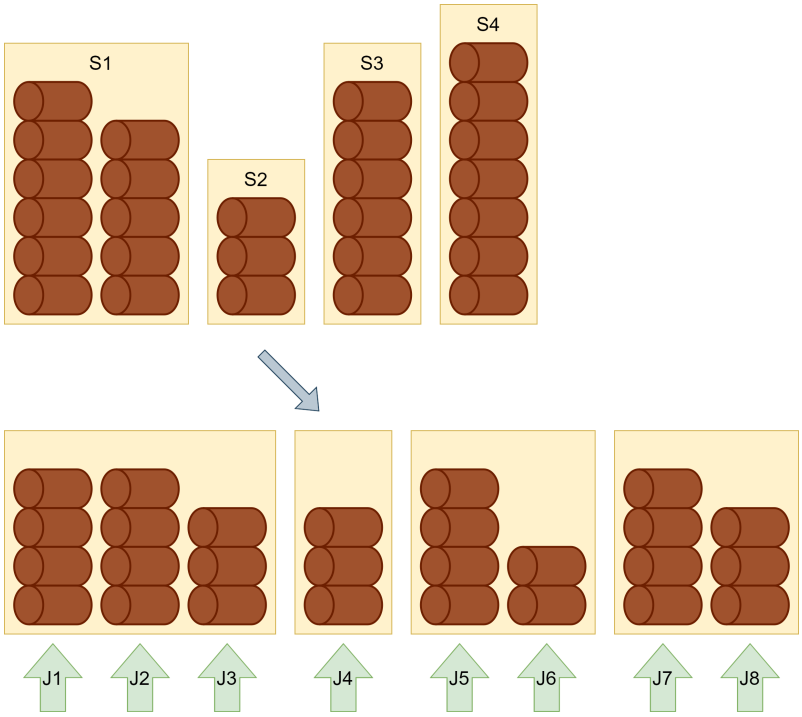
```
python3 firewood_bags.py input.txt output.txt
```

Un fichier *test_firewood_bags.py* et des fichiers d'entrées vous sont fournis pour vous aider à tester votre code.

Exemple d'appel :

```
python3 test_firewood_bags.py
```

Remise
Compléter le fichier *firewood_bags.py* fournis, et ne remettre **uniquement** que ce fichier. Ne **pas** remettre le fichier *test_firewood_bags.py*, ou n'importe quel autre fichier de test.


S1
S2
S3
S4
J1
J2
J3
J4
J5
J6
J7
J8