Introduction to Algorithms Winter 2024

IFT2125-6001 Evaluation (7.5%)

Homework 2 - Water, fire and prime numbers

General instructions: To be done in teams of 1 to 2 people. Submit a single pdf and the python and C++ code files. Indicate your names and registration numbers on all files (pdf and code).

Your code must pass the tests supplied with the assignment, otherwise a mark of 0 will be given to exercises for which the basic tests are not passed. Other tests will be added during correction. Only standard Python and C++ libraries are allowed, unless otherwise stated.

It goes without saying that the code must be clear, well indented and well commented.

1 Résolution de récurrences (25 points)

Question 1: (10 points) Show the complete procedures for finding the characteristic polynomial and roots (with their multiplicities) of the recurrence below. After finding the roots, write the general form of the recurrence with generic constants c_i.

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

Hint: Start by multiplying the recurrence by a constant and changing n to $n-1$
.

Caution: Don't simply answer using the formula on page 126 of the Bratley-Brassard reference book, but use it to validate your own answer.

Question 2: (10 points) Solve the following recurrence exactly, for powers of 2 only, without using the master theorem:

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

Note: You can't use the master theorem for the exercice, but you can use it to validate your own answer.

Question 3: (5 points) Use the master theorem to find the exact order of the following recurrences:

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 Prime numbers - C++ code (25 points)

In this problem, you're asked to find special sequences of prime numbers using the C++ language. You can of course reuse your code from assignment 1 to generate the prime numbers. Your algorithm should be efficient in order to get all the points of the exercice.

Consider a sequence $S = S_0, S_1, ..., S_k$. The sequence S is special if :

1. The sequence S is composed of prime numbers only,
2. the terms of the sequence S are ordered in ascending order,
3. $S_i = S_0 + i \times k$ where $k \in \mathbb{N}$ is a constant for the sequence, and
4. all terms are permutations of every other term, observing the digits.

For example, the sequence $11483, 14813, 18143$ is special because :

1. $11483, 14813, 18143$ are all prime numbers,
2. $11483, 14813, 18143$ are ordered in ascending order,
3. $S_i = 11483 + i \times 3330$ where $3330 \in \mathbb{N}$: $11483 + 1 \times 3330 = 14813$ and $11483 + 2 \times 3330 = 18143$, and
4. all terms are permutations of $1, 1, 3, 4$ and 8.

You are asked to find all sequences of size $|S| = 3$ containing terms strictly less than N. For $N = 10000$, there are 2 possible special sequences and for $N = 20000$, there are 4 possible special sequences.

Code
Call example :

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

Call example :
.\prime_numbers_sequences.exe 20000
```

Submission
Complete the *PrimeSequenceCalculator.cpp* and *PrimeSequenceCalculator.h* files supplied, and hand over these files only. Do not submit the file *prime_numbers_sequences.cpp*.

3 Bridges and pillars - Python Code (25 points)

A river-crossing bridge is currently under construction. The bridge will have a total length of L meters. The bridge's support pillars must be spaced at least D meters apart (measured from mid-pillar to mid-pillar), to let boats pass and not disturb the water current too much. The pillars must also be spaced at least C meters from the beginning and end of the bridge (measured from the middle of the pillar to the beginning or end point), to leave the bank clear and leave room for nature. The company working on the project has gone bankrupt, and you're in charge of continuing the project. The company has already installed a number of pillars (N) in no particular order, while respecting the constraints. You need to calculate the maximum number of pillars that can be added to the construction according to the data obtained. Consider $L, D, C \in \mathbb{N}^{\geq 1}$ and $N \in \mathbb{N}^{\geq 0}$.

Data :

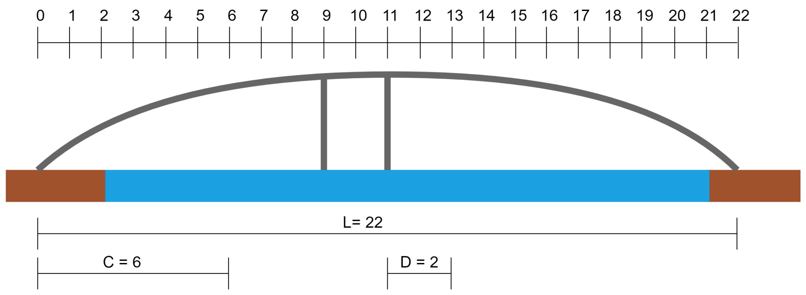
```
<length L> <distance between pillars D> <distance between pillars and ends C> <nb pillars already installed N>
<position of pillar 1>
<position of pillar 2>
...
<position of pillar N>
```

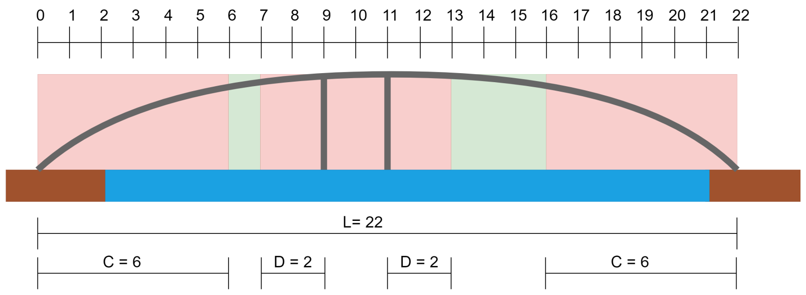
Examples:

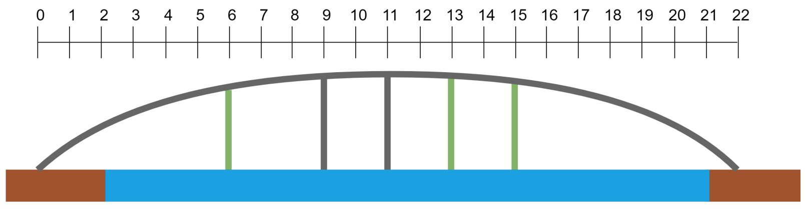
```
Example 1 :
```

```
22 2 6 2
11
9
--> must give 3 additional pillars

Example 1 :
47 5 6 0
--> must give 8 additional pillars
```







Code

Your code must read a file specified in the first argument that contains a problem. Write the answer to a file specified in the second argument.

Example call:

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

An *test_ponts_et_piliers.py* file and input files are provided to help you test your code.

Example call :

```
python3 test_ponts_and_piliers.py
```

Submission
Complete the file *ponts_et_piliers.py* provided, and hand over **only** this file. Do not hand over the *test_ponts_et_piliers.py* file, or any other test file.

Bonus 10%
We'll award a maximum bonus of 10% (for the assignment) to teams that submit code that passes the tests on Kattis for the "Birds on a Wire" problem. One person per team is enough. Your name must appear on the list (if you use a pseudonym, come and show the teaching team that it's your account). Note that you'll need to modify the code a little to answer the problem on Kattis, including reading the problem with input(), writing the answer with print() and a fixed problem parameter.

4 Chalet and wood stove - Python code (25 points)

You and your group are going to the chalet for J days. To fuel your wood stove, you have access to a wood supply containing S bags of wood. $S \leq J$. Each bag contains a certain number of logs: $W[i]$ logs for the i-th bag. As the wood storage is far from the chalet, you can only pick up one bag of wood per day, at the start of the day, no matter how many logs it contains. Each time you pick up wood, you can choose the bag you want from the remaining bags available. What's more, you can fetch a new bag of wood only when the last bag taken has been completely emptied at the end of the previous day. To take full advantage of the wood-burning stove's heat, you need to decide how many (constant, not variable) wood logs B you and your group can burn per day. If you burn your wood too slowly, there will be wood left over at the end of your stay, but if you burn your wood too quickly, you'll run out of wood in the last few days. The aim is to find the best constant rate B (wood logs/day) $\in [1, max\{W\}]$ that will burn all the wood, as slowly as possible. Obviously, on some days, you won't burn B wood logs if there are less than B wood logs left. We can assume that all the numbers considered in the problem are integers ≥ 1.

Your algorithm must run in $\mathcal{O}(S \times \log(max\{W\}))$. Your algorithm must not run in $\mathcal{O}(S \times max\{W\})$ with a simple loop that tests all possible B from 1 to $max\{W\}$.

Data :

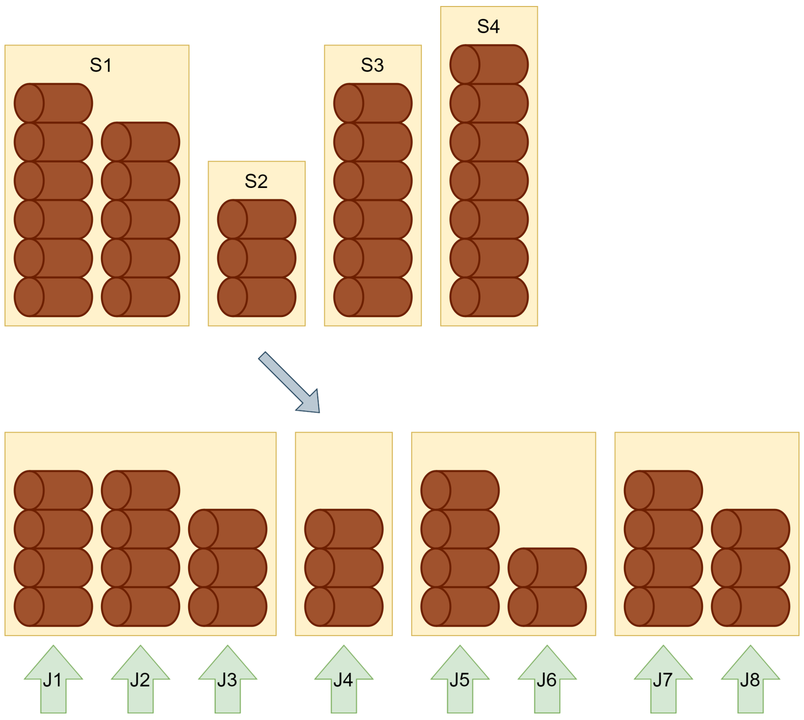
```
<total number of days : J>
```

```
<nb of wood logs in bag 0 : W[0]> <nb of wood logs in bag 1 : W[1]> ... <nb of wood logs in bag S-1 : W[S-1]>
```

Examples:

```
Example 1 :
8
11 3 6 7
--> must give B = 4 wood logs per day

Example 2:
5
23 11 30 4 20
--> must give B = 30 wood logs per day
```



Code

Your code must read a file specified in the first argument that contains a problem and write the answer to a file specified in the second argument.

Example call:

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

An *test_firewood_bags.py* file and input files are provided to help you test your code.

Example call :

```
python3 test_firewood_bags.py
```

Submission
Complete the file *firewood_bags.py* provided, and submit this file **only**. Do not submit the *test_firewood_bags.py* file, or any other test file.