Rattrapage 2020
Analyse d'algorithmes et Programmation

Yann Rotella

23 juin 2020

La durée du rattrapage est de 2 heures. Il est possible de communiquer si nécessaire par mail ou par téléphone (yann.rotella@uvsq.fr) ou 06.16.41.00.19. Il est impératif d'envoyer un e-mail entre 9h55 et 10h10, indiquant la bonne réception du sujet. La fin de l'épreuve est à 12h00. Toute copie reçue après 12h20 ne sera pas corrigée. Le format de la copie doit être en PDF.

Toute erreur dans le sujet sera prise en compte dans la correction. Le barème est donné à titre indicatif. La ressemblance entre les copies sera examinée et sera sanctionnée. Toute ressemblance avec des informations trouvées sur le Web sera aussi examinée et sanctionnée. La qualité de la rédaction sera aussi prise en compte dans la correction.

Fibonacci

On considère la suite de Fibonacci définie par $F_0 = 0$, $F_1 = 1$ et pour tout $n \geq 2$,

$$F_n = F_{n-1} + F_{n-2}\,.$$

Questions de cours :

1 (1 point) Quel est le problème avec l'algorithme de Fibonacci vu en cours qui utilise la récursivité ? Quel est le problème de l'algorithme de Fibonacci qui utiliserait la formule de Binet (avec le nombre d'or) ?

2 (1 point) Montrez que pour tout $n \geq 2$,

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Nous utilisons donc l'algorithme itératif suivant, qui nous permet de calculer les premiers termes de la suite de Fibonacci.

Algorithm 1 fibonacci(k)

```
a = 0
b = 1
i = 1
while i < k do
    tmp = a + b
    a = b
    b = tmp
    i = i + 1
end while
return b
```

3 (1 point) Montrez l'exactitude du programme pour $k \geq 1$.

4 (1 point) Soit $n \geq 1$. Combien de bits sont nécessaires pour encoder F_n ?

5 (1 point) Soit $i \geq 1$. Dans l'algorithme, quel est le coût en opérations binaires de l'étape $tmp = a + b$ au i-ème passage dans la boucle while ?

6 (1 point) Donnez alors le représentant de la classe de complexité de l'algorithme fibonacci en opérations binaires (en fonction de k).

On remarque que la relation de récurrence peut s'écrire de la forme

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

pour tout $n \geq 1$.

7 (1 point) En supposant (pour cette question uniquement) que les additions et les multiplications se font en temps constant, proposez un algorithme qui permet de calculer F_n avec une complexité en $O(\log(n))$.

8 (2 points) Évaluez maintenant la complexité de votre algorithme en prenant en compte le fait que les multiplications et les additions que vous réalisez sont en fait en $O(n^{\log_2(3)})$ pour la multiplication et $O(n)$ pour l'addition lorsque les éléments à additionner ou à multiplier sont de taille n.

Le problème 2-SAT

Le problème 2-SAT est un problème de décision, qui prend en entrée une conjonction de m clauses ayant chacunes 2 littéraux. Les variables booléennes (au nombre de n) sont notées x_1, x_2,..., x_n. Par exemple la conjonction

$$(x_1 \vee \neg x_4) \wedge (\neg x_2 \vee x_4) \wedge (x_3 \vee x_1) \wedge (\neg x_2 \vee \neg x_4)$$

est une entrée particulière pour 2-SAT, composée de $m = 4$ clauses et possède $n = 4$ variables. Les littéraux sont $\neg x_i$ ou bien x_i pour tout i allant de 1 à n. Il y a donc au maximum $2n$ littéraux présents s'il y a n variables.

9 (1 point) Quelle est la mémoire nécessaire pour stocker une instance en entrée de 2-SAT à n variables et m clauses ?

10 (1 point) Si y et z représentent 2 littéraux, donnez deux formules booléennes équivalentes à $y \vee z$ en utilisant $\Rightarrow$ et $\neg$.

À partir d'un instance de 2-SAT, on crée donc un graphe orienté de la manière suivante. Pour chaque clause, on rajoute dans le graphe les deux arêtes définies par les relations précédemment trouvées entre littéraux. Les littéraux étant les sommets du graphe. On note le graphe $G = (S, A)$ où S désigne l'ensemble des sommets et A l'ensemble des arêtes.

11 (1 point) Quelle est la mémoire nécessaire pour stocker le graphe avec la représentation matricielle et avec la représentation sous liste d'adjacence (on part d'une instance à m clauses et n littéraux).

12 (1 point) Quelle est la complexité de l'algorithme qui transforme l'instance de 2-SAT initiale en la matrice d'adjacence ?

13 (2 points) Montrez que s'il existe une composante fortement connexe C dans le graphe pour laquelle $\exists y \in S$ tel que $y \in C$ et $\neg y \in C$, alors 2-SAT n'est pas satisfiable (il n'existe pas d'affectation aux variables, telle que toutes les clauses soient vraies).

À partir de maintenant, on suppose qu'aucune composante connexe ne comporte jamais y et $\neg y$, pour n'importe quelle variable y.

14 (1 point) Montrez que si $C = \{l_1, l_2, \ldots, l_\ell\}$ est une composante fortement connexe, alors $\neg C = \{\neg l_1, \neg l_2, \ldots, \neg l_\ell\}$ est une composante fortement connexe, où les l_i désignent des littéraux (des sommets du graphe).

15 (2 points) Une fois les composantes fortement connexes identifiées, notées C_1 à C_k et $\neg C_1$ à $\neg C_k$, on considère le graphe, dont les sommets sont étiquetés par C_i ou $\neg C_i$ pour i allant de 1 à k. On met une arête entre C_i (ou $\neg C_i$) et C_k (ou $\neg C_k$) s'il existe au moins une arête entre un sommet de la première composante fortement connexe et la deuxième. Montrez qu'il est possible d'appliquer l'algorithme de tri topologique sur ce graphe.

16 (1 point) Montrez que le graphe obtenu vérifie la propriété suivante. "Il existe une arête de C vers D (où C et D sont des sommets du graphe précedent) si et seulement si il existe une arête de $\neg D$ vers $\neg C$".

17 (1 point) Montrez qu'il existe un sommet C qui n'admet aucune arête entrante.

18 (1 point) Déduire des questions précédentes une manière récursive d'affecter les variables des composantes fortement connexes du graphe, de manière à ne jamais avoir d'arête VRAI vers FAUX.

19 (1 point) Montrez alors que 2-SAT n'est pas satisfiable si et seulement si il existe une composante fortement connexe C dans le graphe initial pour laquelle $\exists y \in S$ tel que $y \in C$ et $\neg y \in C$.

20 (1 point) Donnez alors la complexité de résolution de 2-SAT en fonction du nombre de clauses et du nombre de variables.

21 (1 point) Le problème 2-SAT est-il dans la classe P ? Le problème 2-SAT est-il dans la classe NP ? Justifiez.