IFT-3001 (Hiver 2025) Travail 1

Enseignant : Claude-Guy Quimper
Date de remise : 23 février 2025, 23h59.

- Les travaux remis en retard ne seront pas corrigés et se verront attribuer la note de zéro.
- À moins de spécifications contraires, vous devez toujours pleinement justifier vos réponses.
- Il s'agit d'un travail individuel. Toute communication ou entraide avec une autre personne au sujet de ce travail est interdite.
- Vous devez présenter un travail original : aucun emprunt provenant de l'Internet ou de toute autre source n'est permis.
- Le code de programmation doit pouvoir compiler avec g++. À partir d'une connexion VPN, vous pouvez compiler votre code sur cette page web : http://132.203.114.30:8080/.

Format de remise :

- Vous devez remettre un fichier zip contenant un fichier PDF par question. La structure du fichier zip ne doit pas contenir de sous-répertoire et doit seulement contenir les fichiers : Question1.pdf, pgcd.cpp et Question2.pdf.
- Votre nom, prénom et matricule doivent apparaître dans l'en-tête de chaque page.

Question # 1: [25 points] Dans le chapitre 1, nous avons vu deux algorithmes pour calculer le plus grand commun diviseur entre deux entiers : l'algorithme de force brute pgcdFB et l'algorithme diminuer pour régner d'Euclide. Nous avons démontré que le temps d'exécution de l'algorithme d'Euclide est $C(m) \in O(\log m)$ pour toute instance. Cependant, nous n'avons pas démontré que le pire cas s'exécute en $C_{WORST}(m) \in \Theta(\log m)$. Nous allons procéder à une analyse empirique de l'efficacité de ces deux algorithmes. Nous allons également analyser, en pire cas, l'algorithme d'Euclide.

a) Programmez les algorithmes pgcdFB et Euclide en C/C++ dans un fichier nommé `pgcd.cpp`. Modifiez les deux algorithmes pour qu'ils retournent, en plus du plus grand commun diviseur, le nombre d'itérations effectuées par la boucle principale. À partir des données générées par votre programme, présentez dans votre document PDF un tableau avec ces six colonnes [1] :

1. i : Cette colonne contient les entiers de 1 à 40 (le tableau a donc 40 lignes) ;
2. $F(i+1)$ où F est la fonction de Fibonacci (voir diapositive 65 du Chapitre 2) ;
3. $F(i)$;
4. $pgcd(m, n)$ est le plus grand commun diviseur entre $m = F(i+1)$ et $n = F(i)$;
5. pgcdFB : Le nombre d'itérations qu'effectue l'algorithme pgcdFB pour calculer $pgcd(F(i+1), F(i))$;
6. Euclide : Le nombre d'itérations qu'effectue l'algorithme d'Euclide pour calculer $pgcd(F(i+1), F(i))$.

Vous trouverez un algorithme efficace pour calculer les nombres de la suite de Fibonnaci à la diapositive 78 du chapitre 2.

b) Démontrez, pour tout $i \geq 4$, que $\left\lfloor \frac{F(i)}{F(i-1)} \right\rfloor = 1$. De façon équivalente, vous pouvez démontrer que $1 \leq \frac{F(i)}{F(i-1)} < 2$.

c) Démontrez, pour $i \geq 4$, que $F(i) \bmod F(i-1) = F(i-2)$ pour tout $i \geq 4$. Vous pouvez réutiliser le résultat démontré en (b).

d) Justifiez pourquoi $Euclide(F(i+1), F(i))$ requiert $i-1$ itérations. Vous pouvez réutiliser le résultat démontré en (c).

e) Démontrez que $\log_b(F(i)) \in \Theta(i)$ pour n'importe quelle base $b > 1$. Vous pouvez utiliser cette forme non-récursive de la suite de Fibonacci.

$$F(i) = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^i - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^i \tag{1}$$

1. Le tableau doit être lisible. Évitez les saisies d'écran.

Fin de la démonstration : Vous avez démontré en (d) que l'algorithme d'Euclide requiert $i-1$ itérations pour calculer le plus grand commun diviseur entre $m = F(i+1)$ et $n = F(i)$. Nous avons donc la relation $C_{WORST}(m) = i - 1$. Il faut à présent exprimer C_{WORST} en fonction de m.

$$
\begin{aligned}
C_{WORST}(m) &= i - 1 && \text{Démontré en (d)} \\
&\in \Theta(i+1) \\
&= \Theta(\log F(i+1)) && \text{Puisque } \log(F(i+1)) \in \Theta(i+1) \\
&= \Theta(\log m) && \text{Puisque } m = F(i+1)
\end{aligned}
$$

Nous avons donc identifié le pire cas de l'algorithme d'Euclide : lorsque m et n sont deux nombres consécutifs de la suite de Fibonacci. Ce cas s'exécute en temps $\Theta(\log m)$. C'est le pire cas car nous avons démontré au chapitre 1 que toute instance s'exécute en temps $C(m) \in O(\log m)$. Le pire cas atteint donc cette borne.

C.Q.F.D

Question # 2: [25 points] Analysez les algorithmes suivants. Si l'efficacité de l'algorithme ne dépend pas uniquement de la taille de l'instance, procédez à une analyse en pire cas, en meilleur cas et en cas moyen. Donnez l'ordre de croissance de l'efficacité de l'algorithme en utilisant la notation asymptotique Θ. N'utilisez que les relations de l'aide-mémoire pour simplifier les sommations dans vos calculs. Attention à la façon dont vous traitez les fonctions $\lfloor x \rfloor$ et $\lceil x \rceil$. Vous ne pouvez pas simplement les ignorer. Toutefois, vous pouvez les borner grâce aux inégalités $\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$ et $\lceil x \rceil - 1 < x \leq \lceil x \rceil$.

Pour chaque algorithme, le vecteur A est un vecteur de bits. La probabilité d'avoir x bits allumés est de $\frac{1}{n+1}$. Évidemment, cette distribution peut être ignorée si vous ne procédez pas à une analyse en cas moyen.

Pour le deuxième algorithme, il est plus facile d'analyser chaque phase de l'algorithme séparément pour ensuite trouver le temps de calcul total à l'aide de la règle du maximum.

Algorithme 1 : PremierAlgorithme($A[0..n-1]$)

```
c ← 0;
for i = 1 to n do
    for j = 0 to ⌈√i⌉ do
        c ← c + A[j mod n];
return c;
```

Algorithme 2 : DeuxièmeAlgorithme($A[0..n-1]$)

```
// Première phase
c ← 0;
for i = 0..n − 1 do
    c ← c + A[i];
// Deuxième phase
t ← 0;
if c < n then
    for i = 0 to c do
        for j = i to 2i do
            t ← t + 1;
return t;
```

Question # 3: [25 points]

Soit la récurrence suivante.

$$C(n) = \begin{cases} 1 & \text{si } n = 1 \\ 2C(\lfloor \frac{n}{2} \rfloor) + \log_2(n) + 1 & \text{si } n > 1 \end{cases}$$

1. Résolvez la récurrence en utilisant la substitution à rebours. Faites au moins trois substitutions. Vous pouvez supposer que n est une puissance d'un entier. Donnez votre solution dans sa forme la plus simple.
2. Donnez l'ordre de croissance de C en utilisant la notation Θ. Cet ordre doit être démontré pour tout $n \in \mathbb{N}$.

Revenez télécharger les autres questions un peu plus tard.