Carnet de Travaux Dirigés
Algorithmique et structures de données
Licence 2 Informatique

Julien Bernard

Table des matières

Travaux Dirigés d'Algorithmique n°1

Exercice 1 : Ordres de grandeur

Question 1.1 Qu'est-ce qui est le plus grand : 10^{100} ou 100^{10} ?

Question 1.2 Combien de chiffres comporte 2^n en écriture décimale ?

Exercice 2 : Échelle de fontion

Question 2.1 Classer ces fonctions par ordre croissant asymptotique :
$n \log n$, $2^n \log^3 n$, $n^2 \log n$, $3^n \log n$, $n \log \log n$, $n \log^3 n$, n, $2^n n^2$

Question 2.2 Les assertions suivantes sont elles vraies ou fausses, pourquoi ?

1. $3^n = O(2^n)$
2. $\log 3^n = O(\log 2^n)$

Question 2.3 Calculer $O(\log(n!))$. On pourra utiliser le résultat suivant : si $f \sim g$ alors $O(\log f) = O(\log g)$.

Exercice 3 : Comptage d'opérations

Dans cet exercice, on note $\mathcal{N}$ le nombre d'opérations `+` effectuées par chaque fonction.

Question 3.1 On considère la fonction suivante :

```
int f(int n) {}
  int res = 0;

  for (int i = 0; i < n; ++i) {
    res += i;
  }

  return res;
}
```

Calculer $\mathcal{N}$ en fonction de n ?

Question 3.2 On considère la fonction suivante :

```
int g(int n) {
  int res = 0;

  for (int i = 0; i < n; ++i) {
    res += f(i);
  }

  return res;
}
```

Calculer $\mathcal{N}$ en fonction de n ?

Question 3.3 Dans la fonction précédente, on remplace :

```
res += f(i);
```

par :

```
res += f(n)
```

Que vaut alors $\mathcal{N}$ en fonction de n ?

Question 3.4 Dans ce dernier cas, proposez une modification pour améliorer $\mathcal{N}$. Que vaut alors $\mathcal{N}$?

Exercice 4 : Taille de problème

On considère la fonction `h` dont la signature est la suivante :

```
int h(int n);
```

Sur votre ordinateur, en 1 minute, la fonction h peut renvoyer un résultat jusqu'à $n = M$. Une entreprise vous propose son nouveau microprocesseur qui est 100 fois plus rapide que le vôtre ! Si vous achetez ce microprocesseur, en 1 minute votre fonction pourra sans doute atteindre des n plus grands.

Question 4.1 Donner une fourchette du nouveau n maximum atteint si le nombre d'opérations $\mathcal{N}$ effectuées par la fonction h est : n, $2n$, $3n$, n^2, n^4, 2^n

Travaux Dirigés d'Algorithmique n°2

Exercice 5 : Complexité et boucles

On considère la fonction suivante :

```
int f(int n) {
  int sum = 0;

  for (int i = 0; i <= n; ++i) {
    for (int j = 0; j <= (n-i)/2; ++j) {
      for (int k = 0; k <= (n-i-2*j)/5; ++k) {
        if (i + 2*j + 5*k == n) {
          sum++;
        }
      }
    }
  }

  return sum;
}
```

Question 5.1 Tester cette fonction pour $n = 5$. Que calcule f ?

Question 5.2 On prend comme opération fondamentale l'addition. Calculer la complexité de cette fonction ?

Question 5.3 Réécrire cette fonction en inversant l'ordre des boucles. Éliminer, en le justifiant, la boucle la plus interne. Quelle est la complexité de cette nouvelle fonction ?

Question 5.4 Peut-on encore améliorer la complexité ?

Exercice 6 : La fonction puissance

Dans cet exercice, on considère le problème du calcul de la puissance $n^{\text{ième}}$ d'un nombre flottant x. L'opération fondamentale est évidemment la multiplication.

On considère tout d'abord la fonction suivante :

```
double power_v1(double x, unsigned n) {
  if (n == 0) {
    return 1;
  }

  return x * power_v1(x, n - 1);
}
```

Question 6.1 Quelle est la complexité de `power_v1` ?

On considère maintenant la fonction suivante :

```
double power_v2(double x, unsigned n) {
  if (n == 0) {
    return 1;
  }

  double p = power_v2(x, n / 2);

  if (n % 2 == 0) {
    return p * p;
  }

  return x * p * p;
}
```

Question 6.2 Quelle est la complexité de `power_v2` ? Est-elle meilleure que celle de `power_v1` ?

On considère enfin la fonction suivante :

```
double power_v3(double x, unsigned n) {
  if (n == 0) {
    return 1;
  }

  if (n % 2 == 0) {
    return power_v3(x, n/2) * power_v3(x, n/2);
  }

  return x * power_v3(x, n/2) * power_v3(x, n/2);
}
```

Question 6.3 Quelle est la complexité de `power_v3` ? Quelle conclusion peut-on en tirer ?

Exercice 7 : Algorithme de Karatsuba

Pour multiplier deux polynômes $A(X)$ et $B(X)$ de degré n, une technique consiste à couper les polynômes en deux de la manière suivante :

$$\begin{cases} A(X) = A_1(X) \times X^{\frac{n}{2}} + A_2(X) \\ B(X) = B_1(X) \times X^{\frac{n}{2}} + B_2(X) \end{cases}$$

avec $A_1(X)$, $A_2(X)$, $B_1(X)$ et $B_2(X)$ de degré inférieur à $\frac{n}{2}$. Ensuite, on effectue la multiplication de la manière suivante :

$$A(X)B(X) = C_1(X) \times X^n + C_2(X) \times X^{\frac{n}{2}} + C_3(X)$$

avec :

$$\begin{cases} C_1(X) = A_1(X) \times B_1(X) \\ C_2(X) = A_1(X) \times B_2(X) + A_2(X) \times B_2(X) \\ C_3(X) = A_2(X) \times B_2(X) \end{cases}$$

On peut appliquer récursivement l'algorithme de multiplication pour le calcul de $C_1(X)$, $C_2(X)$, $C_3(X)$. Quand les polynômes sont de degré 0, il s'agit d'une multiplication sur les réels. La multiplication par X^k consiste simplement à décaler les coefficients du polynômes et est donc une opération constante.

Question 7.1 Quelle est la complexité de l'addition de deux polynômes de degré n en nombre d'additions et de multiplications sur les réels.

Question 7.2 Donner la formule pour calculer la complexité $C(n)$ de la multiplication de deux polynômes de degré n en nombre d'additions et de multiplications sur les réels, en fonction de $C(\frac{n}{2})$. Justifier précisément.

Question 7.3 Résoudre cette formule grâce au Théorème Diviser pour Régner.

On calcule désormais $C_2(X)$ de la manière suivante :

$$C_2(X) = (A_1(X) + A_2(X)) \times (B_1(X) + B_2(X)) - C_1(X) - C_3(X)$$

Question 7.4 Vérifier que ce calcul est bien exact

Question 7.5 Donner la formule pour calculer la complexité $C(n)$ de la multiplication de deux polynômes de degré n en nombre d'additions et de multiplications sur les réels, en fonction de $C(\frac{n}{2})$. Justifier précisément.

Question 7.6 Résoudre cette formule grâce au Théorème Diviser pour Régner.

Exercice 8 : Calcul de complexité

On considère les fonctions suivantes :

```
int f(int n) {
  int x = 0;

  for (int i = 1; i < n; i++) {
    for (int j = i; j < i + 5; j++) {
      x += j;
    }
  }

  return x;
}
```

```
int g(int n) {
  int y = 0;

  for (int i = 0; i < f(n); i++) {
    y += i;
  }

  return y;
}
```

Question 8.1 Quelle est l'opération fondamentale dans ces fonctions ?

Question 8.2 Quelle est la complexité de `f` ? Justifier précisément.

Question 8.3 Donner en fonction de `n` un ordre de grandeur du résultat de ce qui est calculé par `f` ?

Question 8.4 Quelle est la complexité de `g` ? Justifier précisément.

Question 8.5 Donner en fonction de `n` un ordre de grandeur du résultat de ce qui est calculé par `g` ?

Question 8.6 Réécrire la fonction `g` pour diminuer sa complexité ? Quelle est alors sa complexité ?

Exercice 9 : Application du théorème Diviser pour régner

Pour chacune des relations de récurrence suivante, dire si le théorème peut s'appliquer et donner le résultat le cas échéant en précisant lequel des trois cas est utilisé.

1. $T(n) = 3T\left(\frac{n}{2}\right) + n^2$
2. $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
3. $T(n) = T\left(\frac{n}{2}\right) + 2^n$
4. $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$
5. $T(n) = 16T\left(\frac{n}{4}\right) + n$
6. $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$
7. $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$
8. $T(n) = 2T\left(\frac{n}{4}\right) + n^{0.51}$
9. $T(n) = 0.5T\left(\frac{n}{2}\right) + \frac{1}{n}$
10. $T(n) = 16T\left(\frac{n}{4}\right) + n!$
11. $T(n) = \sqrt{2}T\left(\frac{n}{2}\right) + \log n$
12. $T(n) = 3T\left(\frac{n}{2}\right) + n$
13. $T(n) = 3T\left(\frac{n}{3}\right) + \sqrt{n}$
14. $T(n) = 4T\left(\frac{n}{2}\right) + cn$
15. $T(n) = 3T\left(\frac{n}{4}\right) + n\log n$
16. $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{2}$
17. $T(n) = 6T\left(\frac{n}{3}\right) + n^2\log n$
18. $T(n) = 4T\left(\frac{n}{2}\right) + \frac{n}{\log n}$
19. $T(n) = 64T(n/8) - n^2 logn$
20. $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

21. $T(n) = 4T\left(\frac{n}{2}\right) + \log n$

22. $T(n) = T\left(\frac{n}{2}\right) + n(2 - \cos n)$