**Nom:** Rakotondrabe Miora Fanomezana

**Code Permanent:** RAKM11100201

**Cours:** 8INF259 – Structures des données

**Devoir:** 03 Efficacité

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fonction | n=10 | n=100 | n=1000 | n=10000 | O(n) |
| F1(X) | 0,0001 millisecondes | 0,0003 millisecondes | 0,0020 millisecondes | 0,0195 millisecondes | Cn |
| F2(X) | 0,0004millisecondes | 0,0218 millisecondes | 2,1902 millisecondes | 213,8728 millisecondes | Cn2 |
| F3(X) | 0,0023 millisecondes | 2,0156 millisecondes | 2008,5560 millisecondes | 20085560 millisecondes | Cn3 |
| F4(X) | 0,0003 millisecondes | 0,0114 millisecondes | 1,1116 millisecondes | 109,5785 millisecondes | Cn2 |
| F5(X) | 0,0179  millisecondes | 2024,1410 millisecondes | 81916986,27 millisecondes | 3.315\*1012 millisecondes | Cn6 |
| F6(X) | 0,0040 millisecondes | 33,8550 millisecondes | 264828,7664  millisecondes | 1.07\*1010 millisecondes | Cn6 |

1+2+3+...+n = n(n+1) / 2

12+22+32+...+n2 = n(n+1)(2n+1)/6

**Preuve :**

void F1(int n) { // C1 + nC2 = Cn

int somme = 0; // C1

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

somme = somme+ 1; // C2

}

}

void F2(int n) { // C1 + (n\*n\*C2) = C1 + C2n^2 = Cn^2

int somme = 0; //C1

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

for (int j = 0; j < n; j++) { //n

somme = somme+1; // C2

}

}

}

void F3(int n) { // C1 + n(n\*n\*C2) = C1 + C2n^3 = Cn^3

int somme = 0;// C1

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

for (int j = 0; j < (n \* n); j++) { // n\*n

somme = somme + 1;//C2

}

}

}

void F4(int n) { // C1 + C2/2\*(n(n-1)) = C1 + C2n^2/2 - C2n/2 = Cn^2

int somme = 0;//C1

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

for (int j = 0; j < i; j++) { // (n-1) , 0+1+2+3+...+n-1 = (n(n-1))/2

somme = somme + 1; //C2

}

}

}

void F5(int n) { // C1 + (((n(n-1)(2n-1))/6)\*(((n(n-1)(2n-1))/6)-1)) /2 \* C2 = C1 + C2((n^6-12n^5+5n^4+4n^3+4n^2+n)/72) = Cn^6

int somme = 0;//C1

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

for (int j = 0; j < i \* i; j++) { // (n-1)(n-1) = n^2 - 2n + 1, 0 + 1^2 +2^2 +3^2 +....+(n-1)^2 = (n(n-1)(2n-1))/6

for (int k = 0; k < j; k++) { // (n-1)(n-1) - 1 = n^2 - 2n +1 -1 = n^2 -2n , (((n(n-1)(2n-1))/6)\*(((n(n-1)(2n-1))/6)-1)) /2

somme = somme + 1; // C2

}

}

}

}

void F6(int n) { // C1 + C2((n(n-1)(2n-1))/6) + C3(((n(n-1)(2n-1))/6)\*(((n(n-1)(2n-1))/6)-1)) /4n = C1 + C2(2n^3-n^2-2n+1) + C3((n^6-12n^5+5n^4+4n^3+4n^2+n)/72) - 2C3n = Cn^6

int somme = 0; // C1

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

for (int j = 0; j < i \* i; j++) { //(n-1)(n-1) = n^2 - 2n +1,0 + 1^2 +2^2 +3^2 +....+(n-1)^2 = (n(n-1)(2n-1))/6

if (j % i == 0) { // C2

for (int k = 0; k < j; k++) { // (n-1)(n-1) - 1 = n^2 - 2n +1 -1 = n^2 -2n,(((n(n-1)(2n-1))/6)\*(((n(n-1)(2n-1))/6)-1)) /2 - 2n

somme = somme + 1; // C3

}

}

}

}

}

Analyse de résultat:

Pour faire l’analyse algorithmique, on a pris le temps d’exécution de n = 10 et de prédire

le temps d’exécution de n = 10000. Soit 1000 fois plus grand que n.

* Pour la 1ère fonction, on a 0,0001 millisecondes pour n=10 donc pour n=10000, on doit avoir à peu près 1000\*0,0001 millisecondes = 1 millisecondes ≈ 0,0195 millisecondes
* Pour la 2ème fonction , on a 0,0004 millisecondes pour n=10 donc pour n=10000, on doit avoir à peu près 10002 \*0,0001 millisecondes = 400 millisecondes ≈ 213,87828
* Pour la 3ème fonction, on a 0,0023 millisecondes pour n=10 donc pour n=10000, on doit avoir à peu près 10003 \* 0,0023 millisecondes = 2300000 millisecondes(2300 secondes) ≈ 20085560 millisecondes
* Pour la 4ème fonction, on a 0,0003 millisecondes pour n=10 donc pour n=10000, on doit avoir à peu près 10002 \* 0,0003 millisecondes = 300 millisecondes. On peut constater ici que dans notre tableau on a 109,5785 millisecondes alors notre fonction n’ est pas vraiment d’ordre de n2
* Pour la 5ème fonction, on a 0,0179 millisecondes pour n=10 donc pour n=10000, on doit avoir à peu près 10006 \* 0,0179 millisecondes = 1,79 \* 10 6 millisecondes. Dans notre tableau, on a 3,315 \* 1012. Donc la fonction n’est pas vraiment d’ordre n6
* Pour la 6ème fonction, on a 0,0040 millisecondes pour n=10 donc pour n=10000, on doit avoir à peu près 10006 \* 0,004 millisecondes = 4\*1015 millisecondes. La fonction n’est pas aussi vraiment d’ ordre n6 car dans le tableau on a 1,07\*1010