TD N°3 - Complexité

Exercice 1

Donnez l'ordre de grandeur des expressions suivantes :

a. $T(n) = 4n^4 + 3n^2 + nlog(n)$
b. $T(n) = 100n^5 + 2^n$
c. $T(n) = n^3 + 2n^3log(n)$
d. $T(n) = \frac{n^2-3n-1}{n+1}$
e. $T(n) = n^2 2^n + n3^n$
f. $T(n) = 125 + \frac{2}{n^2}$

Exercice 2

Montrez en utilisant les définitions vues en cours que $f \in O(h)$ quand :

a. $f(n) = n^2 - 4n + 2$ et $h(n) = n^2$.
b. $f(n) = 2^n + 14n^5$ et $h(n) = 2^n$.
c. $f(n) = 3n^3 2^n$ et $h(n) = 3^n$.
d. $f(n) = 4nlog^2(n) + n^2$ et $h(n) = n^2$.

Exercice 3

Déterminez pour chacune des affirmations suivantes si elles sont vraies ou fausses. Justifiez.

a. $2^n \in O(1)$
b. $n \in O(n!)$
c. $n! \in O(n^n)$
d. $2^n \in O(n!)$
e. Pour tout $i, j \in \mathbb{N}$, $in \in O(jn)$
f. Pour tout $k \in \mathbb{N}$, $2^k \in O(1)$
g. $log(n) \in O(n)$
h. $n! \in O(2n)$
i. $2^n n^2 \in O(3^n)$
j. Pour toute fonction $f : \mathbb{N} \to \mathbb{N}$, $f \in O(f)$

Exercice 4

Calculez la complexité temporelle (si possible, expliquez le pire des cas) de chaque algorithme en fonction de n. Indiquez le résultat avec la notation grand $O(\ldots)$. Détaillez votre calcul, ne donnez pas juste le résultat. RECOPIEZ le code dans votre feuille pour le calcul.

```
P(): est une fonction de complexité O(1)
Q(i): est une fonction de complexité O(i)
```

(a)
```
for(int i=0;i<n;i++){
    P();
}
```

(b)
```
for(int i=0;i<n;i++){
    for(int j=0;j<i;j++){
        P();
    }
}
```

(c)
```
for(int i=0;i<n;i+=2){
    Q(i);
}
```

(d)
```
int i=1;
while(i<n){
    i=i*2;
}
```

(e)
```
int j;
for(int i=0;i<n;i++){
    j=1;
    while(j<n){
        j=j*2;
    }
}
```

(f)
```
int j=1;
for(int i=0;i<n;i++){
    while(j<n){
        j=j*2;
    }
}
```

(g)
```
int i=1;
while(i<n){
    i=i*2;
    for(int j=0;j<i;j++){
        P();
    }
}
```

(h)
```
int i=1;
for(int j=0;j<n;j++){
    i=i*2;
}
for(int j=0;j<i;j++){
    P();
}
```

(i)
```
int i=1;
for(int k=0;k<n;k++){
    i=i*2;
    for(int l=0;l<i;l++){
        for(int m=0;m<k;m++){
            P();
        }
    }
}
```

(j)
```
int i;
for(int k=0;k<n;k++){
    i=1;
    for(int j=0;j<k;j++){
        i=i*2;
    }
    for(int j=0;j<i;j++){
        Q(j);
    }
}
```

(k)
```
int i=2;
while(i<n){
    i=i*i;
}
```

(l)
```
int i;
for(int j=0;j<n;j++){
    i=1;
    while(i*i<n){
        i=i+1;
    }
}
```

(m)
```
int sumOfN(int n){
    if(n==1){
        return 1;
    }
    else{
        return n+sumOfN(n-1);
    }
}
```

(n)
```
int fact(int n){
    if(n==0){
        return 1;
    }
    else{
        return n*fact(n-1);
    }
}
```

(o)
```
int func1(int n){
    if(n==1){
        return 1;
    }
    else{
        int c = 0;
        for(int i=0;i<n;i++){
            c = c+i;
        }
        return c+func1(n-1);
    }
}
```

(p)
```
int func2(int n){
    if(n==1){
        return 1;
    }
    else{
        int c = 0;
        for(int i=0;i<n;i++){
            c = c+i+func2(n-1);
        }
        return c;
    }
}
```

(q)
```
bool prime1(int n){
    if(n<2){
        return false;
    }
    for(int i=2;i<=sqrt(n);i++){
        if(n%i==0){
            return false;
        }
    }
    return true;
}
```

(r)
```
// first called with i=n-1
bool prime2(int n, int i){
    if(n<2){
        return false;
    }
    if(i==1){
        return true;
    }
    if(n%i==0){
        return false;
    }
    return prime2(n,i-1);
}
```

(s) Expliquez aussi ce qu'il fait cet algorithme.

```
int algo(int tab[],int n,int elem){
    int start = 0;
    int end = n-1;
    int half;
    while(start<=end){
        half = (start+end)/2;
        if(tab[half]==elem){
            return half;
        }
        if(tab[half]<elem){
            start=half+1;
        }
        else{
            end=half-1;
        }
    }
    return -1;
}
```

Exercice 5

(a) Soient 2 programmes A et B qui utilisent 2 algorithmes différents pour résoudre le même problème, à savoir trier une liste de n objets.
- Le programme A a besoin d'un temps $t = 10000n$ pour trier la liste.
- Le programme B a besoin d'un temps $t = 2n^2$ pour trier la même liste.

(i) Quel programme est plus rapide pour trier une liste de 5 objets ?

(ii) Quel est la longueur critique n_c telle que pour tout $n > n_c$, le programme A est plus rapide que le B ?

(iii) Comparer les performances des programmes A et B avec un troisième programme C qui trie la liste en un temps $t = 10 + 0.5n^3$.

(b) Soient 2 programmes E et F qui utilisent 2 algorithmes différents pour trier une liste de n objets.
- Le programme E a besoin d'un temps $t = 8^n$.
- Le programme F a besoin d'un temps $t = n!$.

Quel programme est plus rapide ? Discuter les cas où $n = 2, 4, 8, 10, 12$.

Exercice 6

Calculez la complexité temporelle des exercices marqués avec $^{(*)}$ dans les TPs 3(IV), 4, 6 et 6(III). Pour les exercices suivants, on connaît les meilleurs complexités. Réfléchissez aux nouveaux algorithmes si les vôtres sont moins performants.

- Exercice 4, TP4, le meilleur algorithme qu'on connaît est en $O(n)$ temps et $O(1)$ mémoire.
- Exercices 3 et 4, TP6(III), les meilleurs algorithmes qu'on connaît sont en $O(m + n)$ temps.
- Exercices 5.2 et 5.3, TP6(III), les meilleurs algorithmes qu'on connaît sont en $O(n)$ temps.