Travaux Dirigés 2
Complexité algorithmique

Exercice 1

Calculer la complexité de ces fragments de code :

1/ structures conditionnelles + boucles répétitives:

```
Pour i de 1 à N faire c1
        R     R + X + Y + Z ; c2
        Si T[i] + K < B alors  c3
                Pour j de 1 à N faire c4
                        R     R + T[j] ; c5
                Fin Pour
        Sinon
                R     R + T[i] ; c6
        Fin Si
Fin Pour
```

T(n) = n*(c1+c2+c3+max(n*(c4+c5),c6))
T(n) = n*(1+4+2+max(n*(1+2),2))
T(n) = n*(7+3*n)
T(n) = 3* N^2 +7*n
T(n) = O(n^2)

2/ Deux boucles imbriquées sans dépendance des indices :

```
B     0 ; c1
i     1 ; c2
Pour i de 1 à N faire c3
        B     B + 2 ; c4
        Pour j de 1 à N faire c5
                T[i,j]     (1+ T[j,i]) * B ; c6
        Fin Pour
Fin Pour
```

T(n) = c1+c2+ n*(c3+c4+n*(c5+c6))
T(n) = 2+ n*(1+2+n*(1+3))
T(n) = 2 + 2*n+ 4* N^2
T(n) = O(n^2)

3/ Deux boucles imbriquées avec dépendance des indices :

```
DEBUT
B     0 ; c1
i     1 ; c2
Pour i de 1 à N faire c3
        B     B + 2 ; c4
        Pour j de 1 à i faire c5
                T[i,j]     (1+ T[j,i]) * B ; c6
        Fin Pour
Fin Pour
```

T(n) = c1+c2+ $\sum_{i=1}^{N}$ (c3+c4+ $\sum_{j=1}^{i}$ (c5+c6))

T(n) = 2+ $\sum_{i=1}^{N}$ (3+ $\sum_{j=1}^{i}$ 4)

T(n) = 2+ 3*n +4* $\sum_{i=1}^{N}$ i

($\sum_{i=1}^{N}$ i est une suite arithmétique de raison 1, de 1er terme = 1 et de nombre d'itération n)

T(n) = 2 + 3*n + 4* $\dfrac{n(n-1)}{2}$

T(n) = 2n^2 + n + 2

T(n) = **O(**n^2)

4/ Fonctions et procédures non récursives :

T

```
Fonction
ATA (R : entier) : entier
{
        i: entier ;
        i     1 ; c1
        Pour i de 1 à N faire c2
                R     R * i ; c3
        Fin Pour
        Retourner (R) ; c4
}
```

Ttata(n) = c1+ n*(c2+c3) + 1
Ttata(n) = 2+ n*(1+2)
Ttata(n) = 3*n + 2

```
Fonction TOTO ((R : entier) : entier
{
        I, j: entier ;
        i    1 ; j    1 ; c1
        Pour i de 1 à N faire c2
                j    TATA (j) ; c3
                R    R * j ; c4
        Fin Pour
        Retourner (R) ; c5
}
```

Ttoto(n) = c1+ n*(c2+c3 + c4) + c5
Ttoto(n) = 2+ n*(1+1 + Ttata(n) + 2)
Ttoto(n) = n*(4 + 3*n + 2) + 2
Ttoto(n) = $3n^2$ + 6n + 2

```
C : bool ;
N, R, i : entier ;

Lire (C); c1
Lire(N); c2
R    0 ; c3

Si C = ‘’ # ‘’ alors c4
        Pour i de 1 à N faire c5
                R    R + TOTO(N) ; c6
        Fin Pour
Sinon
        R    TATA (N) ; c7
Fin Si
Retourner (R) c8
FIN
```

T (n) = c1 + c2 + c3 + c4 + max(n*(c5+c6),c7) + c8
T (n) = 3 + 1 + max(n*(1 + 2 + $3n^2$+ 6n + 2),1+3n+2) + 1
T (n) = 5 + n*(5 + 6n + $3n^2$)
T (n) = $3n^3 + 6n^2$ + 5n + 5
T(n) = O(n^3)

5/ Fonctions et procédures récursives :

```
Fonction factorielle (n : entier) : entier
{
        Si n <=1 alors c1
            retourner (1) ; c2
        Sinon
            Retourner (n*factorielle (n-1)) ; c3
        Fin si
}
```

T(n) = c1 + max(c2,c3)
T(n) = 1 + max(1,1+T(n-1))
T(n) = 2 + T(n-1)
T(n-1) = 2 + T(n-2)
…
T(2) = 2 + T(1)
T(1) = 2 + T(0)

T(n) = 2n + T(0)
T(n) = O(n)

```
Fonction REC(n : entier, p : entier) : entier
/*on considère que n >= 0 et p <= n*/
{
    R    0 ; c1
    Si n = 0 c2
    alors  R    p ; c3
        Sinon
            R    REC(n-1,p-1) + REC(n-1,p) + p ; c4
        Fin si
            Retourner (R) ; c5
}
```

T(n) = c1 + c2 + max(c3,c4) + c5

T(n) = 2 + max(1,3+2*T(n-1)) + 1
2^0T(n) = 2^0(6 + 2*T(n-1))
2^1T(n-1) = 2^1(6 + 2*T(n-2))
…
2^{n-2}T(2) = 2^{n-2}(6 + 2*T(1))
2^{n-1}T(1) = 2^{n-1}(6 + 2*T(0))

T(n) = ($\sum_{i=0}^{n-2} 2^i$)*6 + 2^nT(0)

$\sum_{i=0}^{n-1} 2^i$ est une suite géométrique de raison 2, de 1er terme = 1 et de nombre d'itération n)

T(n) = 6*$\frac{1-2^n}{1-2}$ + 2^n

T(n) = 7*2^n - 6
T(n) = O($\log_2 n$)

Exercice 2

Calculer la complexité des algorithmes de tri suivants :

1/ Analyse du tri par sélection

Algorithme :

```
i: entier ;
j: entier;
small : entier ;
tmp: entier;
t : tableau entier [n] ;
Début
        Pour i de 1 à n-1 faire c1
                small←i; c2
                Pour j de i+1 à n faire c3
                        Si t[j] < t[small] alors c4
                                small ←j ; c5
                                tmp←t[small]; c6
                                t[small] ←t[i]; c7
                                t[i] ← tmp; c8
                        Fin si
                Fin pour
        Fin pour
Fin
```

T(n) = $\sum_{i=1}^{n-1}$ (c1+ c2 + $\sum_{j=(i+1)}^{n}$ (c3+c4 + c5+c6+c7+c8))

T(n) = $\sum_{i=1}^{n-1}$ (2 + $\sum_{j=(i+1)}^{n}$ 5)

T(n) = 2(n-1)+ 5* $\sum_{i=1}^{n-1}$ i

($\sum_{i=1}^{n-1}$ i est une suite arithmétique de raison 1, de 1er terme = 1 et de nombre d'itération (n-1))

T(n) = 2n - 2 + 5* $\frac{(n)(n-1)}{2}$

T(n) = $\frac{5}{2}n^2$ - $\frac{1}{2}$ n - 2

T(n) = **O(**n^2)

2/ Analyse du tri par insertion

Algorithme :

```
Procédure tri_Insertion (tab : tableau entier [N])

i, k :entier ;
tmp : entier ;
    Pour i de 2 à N faire c1
        tmp ← tab[i]; c2
        k ← i; c3
        Tant que k > 1 ET tab[k - 1] > tmp faire  c4
            tab[k] ← tab[k - 1]; c5
            k ← k - 1; c6
        Fait
        tab[k] ← tmp;  c7
    Fin pour
Fin
```

T(n) = $\sum_{i=2}^{n}$ (c1+ c2 + c3 + $\sum_{j=2}^{i}$ (c4+c5 + c6) + c7)

T(n) = $\sum_{i=2}^{n}$ (4 + $\sum_{j=2}^{i}$ 4)

T(n) = 4(n-1)+ 4* $\sum_{i=1}^{n-1}$ i

($\sum_{i=1}^{n-1}$ i est une suite arithmétique de raison 1, de 1er terme = 1 et de nombre d'itération (n-1))

T(n) = 4n - 4 + 4* $\frac{(n)(n-1)}{2}$

T(n) = 2n^2 + 2n - 4

T(n) = **O(**n^2)

3/ Analyse du tri par propagation (tri bulle)

Algorithme :

Procédure **tri_Bulle** (tab : tableau entier [N])

```
i, k :entier ;
tmp : entier ;
Pour i de N à 2 faire c1
        Pour k de 1 à i-1 faire c2
          Si (tab[k] > tab[k+1]) alors c3
             tmp ← tab[k]; c4
             tab[k] ← tab[k+1];  c5
             tab[k+1] ← tmp; c6
          Fin si
        Fin pour
Fin pour
Fin
```

$T(n) = \sum_{i=n}^{2} (c1 + \sum_{j=1}^{i-1} (c1+c3+c4+c5+c6))$

$T(n) = \sum_{i=2}^{n} (1 + \sum_{j=1}^{i-1} 5)$

$T(n) = (n-1) + 5 * \sum_{i=1}^{n-1} i$

($\sum_{i=1}^{n-1}$ i est une suite arithmétique de raison 1, de 1er terme = 1 et de nombre d'itération (n-1))

$T(n) = n - 1 + 5* \frac{(n)(n-1)}{2}$

$T(n) = \frac{5}{2} n^2 - \frac{3}{2} n - 1$

$T(n) = \mathbf{O}(n^2)$

4/ Analyse du tri par fusion

Fonction **fusion** (T1, T2) : tableau entier

```
T1 : tableau entier [1...N] ;
T2 : tableau entier [1...M] ;
T: tableau entier [1...M+N] ;
i, j: entire;
i←1 ; j←1 ; c1
Tantque (i+j-1 <> N+M) faire c2
   Si (i ≤ N) alors c3
       si (j ≤ M) alors c4
           si (T1[i] < T2[j]) alors c5
                  T[i+j-1]←T1[i] ; c6
                  i←i+1 ; c7
           sinon
                  T[i+j-1]←T2[j] ; c8
                  j←j+1 ; c9
           Fin si
       sinon
           T[i+j-1]←T1[i]; c10
            i←i+1; c11
        Fin si
   sinon
         T[i+j-1]←T2[j] ; c12
         j←j+1 ; c13
   Fin si
Fait
Retourner T ; c14
FIN
```

Soit n = N+M.

$T_{fusion}(n)$ = c1 +n*(c2+c3+ max(c4+max(c5+max(c6+ c7, c8 + c9), c10+c11), c12+c13))+ c14

$T_{fusion}(n)$ = 2 + n*(2+max(1+max(1+max(2,2),2),2))+1

$T_{fusion}(n)$ = 3+ 6n

$T_{fusion}(n)$ = 6n +3

$T_{fusion}(n)$ = O(n)

Algorithme :

Fonction **tri_fusion** (T) : tableau entier
T: tableau entier [1...N] ;
T1, T2 : tableau entier [N/2] ;
Si (N=1) alors c1
Retourner T ; c2
Sinon
T1 = tri_fusion (SousTableau (T, 1, N/2)) ; c3
T2 = tri_fusion (SousTableau (T, N/2, N)); c4
Retourner fusion (T1, T2) ; c5
Fin si
FIN

T(n) = c1+max(c2,c3+c4+c5)

T(n) = 1 + max(1,T(n/2)+T(n/2) +Tfusion(n))
T(n) = 1+ 2*T(n/2)+ Tfusion(n)
T(n) = 1+ 2*T(n/2)+ 6n+3
T(n) = 2*T(n/2) + 6n+4

T(n) = 2^1 *T(n/2) + 1*(6n+4)

T(n) = 2^2 *T($\frac{n}{2^2}$) + 2*(6n+4)

…

T(n) = 2^k *T($\frac{n}{2^k}$) + k*(6n+4)

T(n) = 2^n T(1) + k*(6n+4)
= n $\log_2 n$ + n

T(n) = O(n $\log_2 n$)

5/ Analyse du tri rapide

Algorithme :

Fonction **partition** (tab : tableau entier [N], debut : entier, fin : entier, indicePivot : entier) : entier
i, k, pivot, tmp : entier ;
pivot ← tab[indicePivot]; c1
k ← debut; c2
Pour i de debut à fin faire c3
Si (tab[i] < pivot) alors c4
tmp ← tab[i]; c5
tab[i] ← tab[k]; c6
tab[k] ← tmp; c7
k ← k + 1; c8
Fin si
Fin pour
tab[k] ← pivot; c9
Retourner k; c10
Fin

n = fin - debut + 1

Tpart(n) = c1 + c2 + n*(c3+c4 + c5+c6+c7+c8) + c9+c10
Tpart(n) = 2+ 6*n + 2
Tpart(n) = 6n+4

Procédure **triRapideR** (tab : tableau entier [N], entier debut, entier fin)
indicePivot : entier ;
pivot0 : entier ;
Si (fin > debut) alors c1
indicePivot ←partition(tab,debut,fin, pivot0); c2
triRapideR(tab, debut, indicePivot - 1); c3
triRapideR(tab, indicePivot + 1, fin); c4
Fin si
Fin

Dans le meilleur des cas, le pivot est, à chaque fois, situé au milieu de la parti à trier.
T(n) = 2T(n/2) + cn
Ce développement s'arrête dès qu'on atteint T(1).
Autrement dit, dès que
$\frac{n}{2^k}$ = 1
TtriRapideR(n) = c1 + max(c2 + c3 + c4)

TtriRapideR(n) = 1*(6n + 5) + 2^1 *T($\frac{n}{2^1}$)

TtriRapideR(n) = 2*(6n+5) + 2^2 *T($\frac{n}{2^2}$)

….

TtriRapideR(n) = k*(6n+5) + 2^k *T($\frac{n}{2^k}$)

avec (T($\frac{n}{2^k}$) = T(1) = O(1))

TtriRapideR(n) = n + O(n $\log_2 n$)

Procédure **triRapide**(tab : tableau entier [N])
    triRapideR (tab, 1, N);
Fin
$T(n) = T_{triRapideR}(n)$
$T(n) = O(n \log_2 n)$