Ex 1:

L’équation de récurrence est : C(n)= (½)\*C(n-1)+(½)\*C(n-2)+1

Ex2 :

Nombre de comparaisons : 26

Ex4 :

1. Nombre d’affectations : 46
2. Nombre d'opérations : 35
3. Ex3:
4. Pour n une puissance de 2 : ex(n) = log2(n)
5. Pour n quelconque : ex(n) = log2(n)+1
6. O(n\*m)
7. O(n+m)
8. O(1)
9. O(max(m;n)+1)

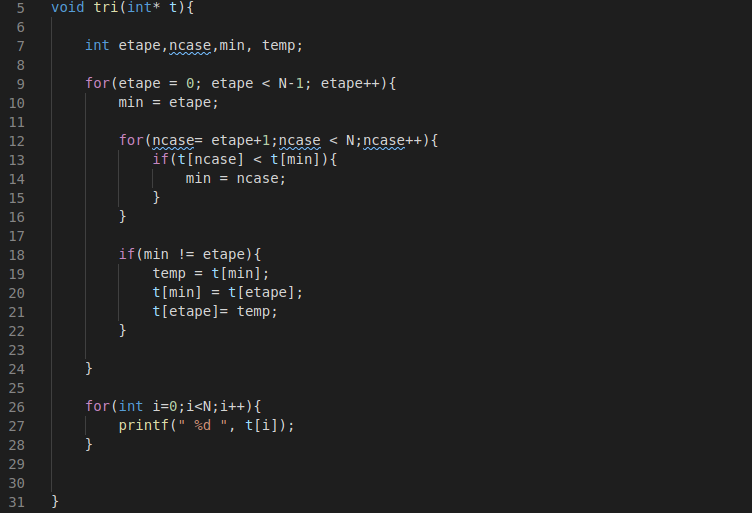
Ex5:

1. Total : n additions.
2. Total : 2n additions.
3. Total :

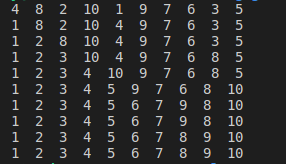
TD6 Tableaux et tris :

Ex 1:

Tri lent utilisé : Tri par selection



Processus du tri :



EX2:

#include <stdio.h>

#define N 5

//simplification pour définir la taille du tableau

int TriVerification(int tab[N]) //fonction insertion classique

{

int i,j,tmp;

for(i = 2;i < N;i++)

{

tmp = tab[i];

j = i-1;

while (j > 0 && tab[j] > tmp)

{

printf("erreur de trie à la position %d\n",i);

//phrase pour prévenir l’utilisateur de la position de l’erreur de trie

tab[j+1] = tab[j];

j = j-1;

}

tab[j+1] = tmp;

}

printf("-------- FIN VÉRIFICATION DE TABLEAU --------\n");

}

int main()

{

printf("Initialisation:\n");

int tab1[N],tab2[N]; //tableau de test

tab1[0] = 1;

tab1[1] = 9;

tab1[2] = 3;

tab1[3] = 4;

tab1[4] = 5;

tab2[0] = 2;

tab2[1] = 3;

tab2[2] = 4;

tab2[3] = 7;

tab2[4] = 6;

printf("------ DEBUT DE VÉRIFICATION DE TABLEAU -----\n");

TriVerification(tab1); //appelle de la fonction

printf("------ DEBUT DE VÉRIFICATION DE TABLEAU -----\n");

TriVerification(tab2);

} //résultat tout chaud sortie du four

EX3 :

Dans le pire des cas, on a :

n affections ligne 8;

1 affectation ligne 10;

1 affectation ligne 11;

n-j affectation ligne 12;

n-j affectation ligne 13;

n-j affectation ligne 14;

n-j affectation ligne 15;

TOTAL : n+4(n-j)+2 affectations

Complexité :

C(n)=

EX4 :

#include <stdio.h>

void insertion(int tableau[n]) {

int i;j

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

for(j = i; j >= 0; j--){

if (tableau[j] > tableau[j-1]) {

permutter(tableau[j] ; tableau[j-1]) ;

}

}

}

Printf(« Tableau trié : \n ») ;

For (i = 0; i < n; i++) {

Printf(“%d”, tableau[i]);

}

}

EX 5 : (test)

void tribool(int tab[n]) {

int i; j; B[n]

j=0;

for (i=0, i<n, i++) {

if (tab[i]==0) {

B[j]==tab[i];

j++;

}

}

for (i=0, i<n, i++) {

if (tab[i]==1) {

B[j]==tab[i];

j++;

}

}

return B[n];

}