Tri rapide

Le programme C de la procédure ‘partitionner’

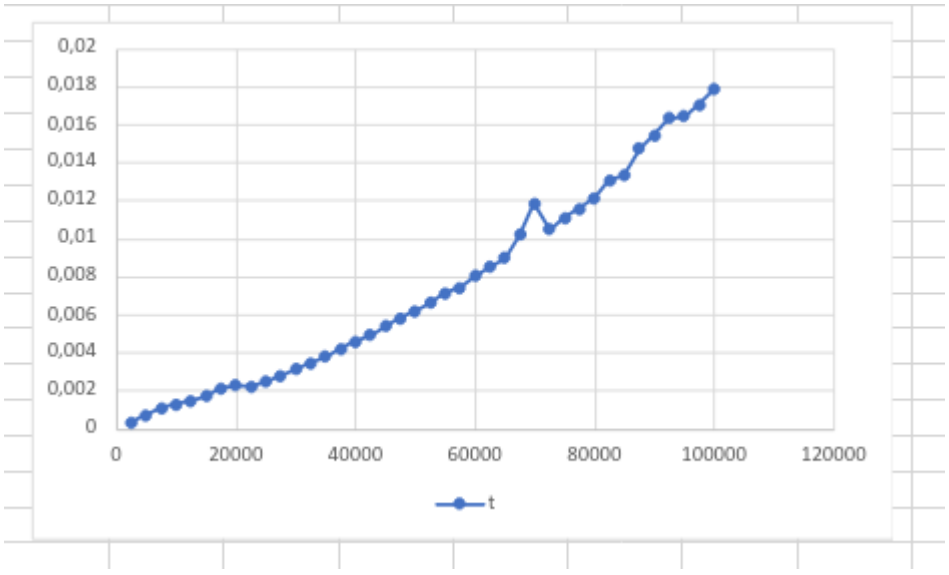
```c
int partitionner(int* T, int d, int f){
  int x;
  int eltPivot = d;
  int i = d+1;
  int j = f;
  while (i <= j) {
    while (i <= f && T[i] <= T[eltPivot]) {
      i++;
    }
    while (j >= d && T[j] > T[eltPivot]) {
      j--;
    }
    if (i <= j) {
      x = T[j];
      T[j] = T[i];
      T[i] = x;
      j--;
      i++;
    }
  }
  x = T[j];
  T[j] = T[eltPivot];
  T[eltPivot] = x;
  return j;
}
```

Le programme C de la procédure ‘triRapide’

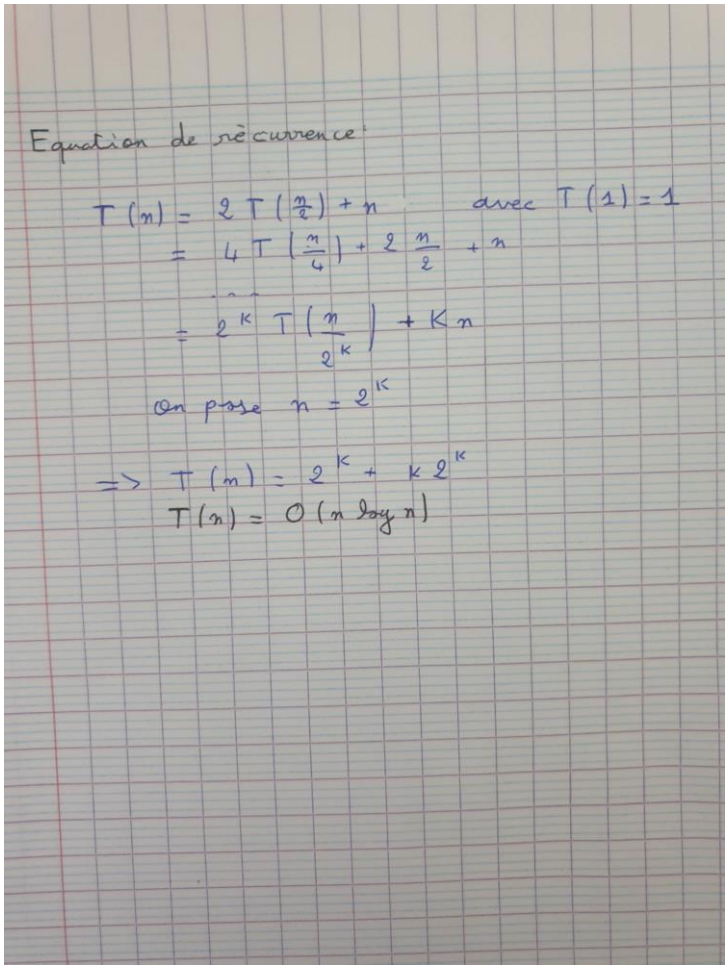
```c
void triRapide(int T[], int p, int r)
{
  int q;
  if (p < r) {
    q = partitionner(T, p, r);
    triRapide(T, p, q-1);
    triRapide(T, q+1, r);
  }
}
```

Le temps d’exécution dans le cas moyen

n	t1	t2	t3	t4	t5	t
2500	0,00033	0,000314	0,000318	0,00031	0,000324	0,000319
5000	0,000634	0,000626	0,000633	0,000739	0,000732	0,000673
7500	0,001025	0,001192	0,001181	0,000967	0,000908	0,001055
10000	0,001245	0,001251	0,001234	0,001283	0,001202	0,001243
12500	0,001544	0,001494	0,001443	0,001388	0,001403	0,001454
15000	0,001695	0,001766	0,001698	0,001617	0,001645	0,001684
17500	0,001893	0,00235	0,00232	0,001987	0,001917	0,002093
20000	0,002151	0,002178	0,002513	0,002476	0,002032	0,00227
22500	0,002338	0,002212	0,002219	0,002118	0,002206	0,002219
25000	0,002443	0,002417	0,0024	0,002406	0,002435	0,00242
27500	0,002806	0,002752	0,002706	0,002699	0,002787	0,00275
30000	0,003147	0,003093	0,003053	0,003158	0,003101	0,00311
32500	0,00351	0,003374	0,003338	0,003519	0,003487	0,003446
35000	0,003714	0,003688	0,00386	0,003829	0,00373	0,003764
37500	0,004127	0,004283	0,004287	0,004142	0,004141	0,004196
40000	0,004629	0,004522	0,004468	0,00462	0,004553	0,004558
42500	0,004819	0,005025	0,00487	0,004826	0,004881	0,004884
45000	0,005396	0,005252	0,005449	0,005292	0,005423	0,005362
47500	0,005855	0,00568	0,005881	0,005842	0,005677	0,005787
50000	0,006301	0,006093	0,006139	0,006202	0,006098	0,006167
52500	0,006707	0,006497	0,006681	0,006527	0,006583	0,006599
55000	0,00741	0,007043	0,007062	0,00701	0,007064	0,007118
57500	0,007355	0,007444	0,007361	0,00746	0,007363	0,007397
60000	0,00799	0,008109	0,007959	0,008081	0,007936	0,008015
62500	0,008548	0,008366	0,008603	0,008317	0,008477	0,008462
65000	0,008961	0,008983	0,008978	0,008899	0,009009	0,008966
67500	0,009381	0,009437	0,009532	0,009419	0,013068	0,010167
70000	0,014629	0,014345	0,010335	0,009922	0,009893	0,011825
72500	0,010486	0,010404	0,010411	0,010509	0,010415	0,010445
75000	0,010971	0,011154	0,011144	0,01097	0,011042	0,011056
77500	0,011615	0,011644	0,011465	0,011481	0,011613	0,011564
80000	0,012175	0,012143	0,01204	0,012105	0,012203	0,012133
82500	0,012792	0,013061	0,013992	0,012699	0,012738	0,013056
85000	0,013347	0,013315	0,013296	0,013262	0,013253	0,013295
87500	0,014921	0,014907	0,014701	0,014518	0,014371	0,014684
90000	0,015276	0,015583	0,015365	0,015233	0,01557	0,015405
92500	0,016312	0,016277	0,016318	0,016268	0,016269	0,016289
95000	0,017428	0,016015	0,016182	0,016153	0,016196	0,016395
97500	0,016829	0,017238	0,017013	0,017173	0,01691	0,017033
100000	0,017713	0,017864	0,017519	0,017622	0,018447	0,017833



Equation de récurrence



Conclusion

Le tri rapide est un algorithme de tri par comparaison, il utilise le principe de diviser pour régner, on sélectionne un élément du tableau qu'on appelle le pivot, et on reconstruit le tableau en deux sous tableaux de façon qu'un des tableaux contient tous les éléments inferieurs au pivot et l'autre contient tous les éléments supérieurs au pivot, on répète ce processus jusqu'à ce que tous les éléments du tableau soient triées.

La complexité temporelle dans le cas moyen est de O(nlogn).

La complexité temporelle dans le pire cas est de $O(n^2)$, Cela arrive lorsque les éléments du tableau sont triés (ou triés dans le sens inverse) et qu'on choisit le premier ou le dernier élément comme pivot

Tri par tas

Le programme C de la procédure ‘creer_tas’

```c
void creer_tas(int T[], int n, int i)
{
    int p = i;
    int gauche = 2 * i + 1; // fils gauche
    int droite = 2 * i + 2; // fils droit
    int x;

    if (gauche < n && T[gauche] < T[p])
    {
        p = gauche;
    }

    if (droite < n && T[droite] < T[p])
    {
        p = droite;
    }

    if (p != i) {
        x = T[i];
        T[i] = T[p];
        T[p] = x;
        creer_tas(T, n, p);
    }
}
```

Le programme C de la procédure ‘tri_tas’

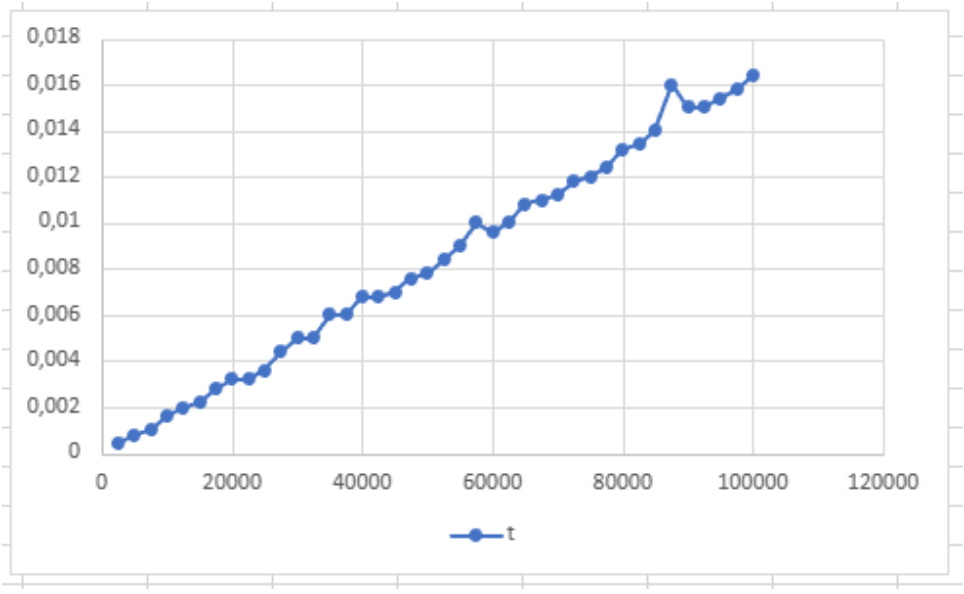
```c
void tri_tas(int T[], int n)
{
    int x;
    int i;
    i = (n/2) - 1;
    while(i>=0)
    {
        creer_tas(T, n, i);
        i--;
    }
    i = n-1;
    while(i>=0)
    {
        supprimer_min(T,i);
        creer_tas(T, i, 0);
        i--;
    }


    //inverser les lements du tableau
    for(i=0;i<n/2;i++)
    {
        x = T[i];
        T[i] = T[n-i-1];
        T[n-i-1] = x;
    }
}
```

Le programme C de la procédure ‘supprimer_min’

```c
void supprimer_min(int T[],int a)
{
    int x;
    x = T[a];
    T[a] = T[0];
    T[0] = x;
}
```

Le temps d’exécution dans le cas moyen

n	t1	t2	t3	t4	t5	t
2500	0	0,001	0	0,001	0	0,0004
5000	0,001	0,001	0	0,001	0,001	0,0008
7500	0,001	0,001	0,001	0,001	0,001	0,001
10000	0,002	0,002	0,001	0,002	0,001	0,0016
12500	0,002	0,002	0,002	0,002	0,002	0,002
15000	0,002	0,003	0,002	0,002	0,002	0,0022
17500	0,002	0,003	0,003	0,003	0,003	0,0028
20000	0,004	0,003	0,003	0,003	0,003	0,0032
22500	0,003	0,004	0,003	0,003	0,003	0,0032
25000	0,004	0,003	0,003	0,004	0,004	0,0036
27500	0,004	0,005	0,004	0,004	0,005	0,0044
30000	0,005	0,005	0,005	0,005	0,005	0,005
32500	0,005	0,005	0,005	0,005	0,005	0,005
35000	0,006	0,006	0,006	0,006	0,006	0,006
37500	0,006	0,006	0,006	0,006	0,006	0,006
40000	0,007	0,007	0,007	0,007	0,006	0,0068
42500	0,007	0,007	0,006	0,007	0,007	0,0068
45000	0,007	0,007	0,007	0,007	0,007	0,007
47500	0,008	0,007	0,007	0,008	0,008	0,0076
50000	0,008	0,008	0,008	0,008	0,007	0,0078
52500	0,008	0,008	0,009	0,009	0,008	0,0084
55000	0,009	0,009	0,009	0,009	0,009	0,009
57500	0,01	0,01	0,01	0,01	0,01	0,01
60000	0,01	0,009	0,01	0,009	0,01	0,0096
62500	0,01	0,01	0,01	0,01	0,01	0,01
65000	0,011	0,011	0,011	0,01	0,011	0,0108
67500	0,011	0,011	0,011	0,011	0,011	0,011
70000	0,011	0,011	0,012	0,011	0,011	0,0112
72500	0,012	0,011	0,012	0,012	0,012	0,0118
75000	0,012	0,012	0,012	0,012	0,012	0,012
77500	0,013	0,012	0,012	0,012	0,013	0,0124
80000	0,013	0,013	0,013	0,013	0,014	0,0132
82500	0,013	0,013	0,014	0,014	0,013	0,0134
85000	0,014	0,014	0,014	0,014	0,014	0,014
87500	0,016	0,016	0,015	0,017	0,016	0,016
90000	0,015	0,015	0,015	0,015	0,015	0,015
92500	0,015	0,015	0,015	0,015	0,015	0,015
95000	0,016	0,015	0,015	0,015	0,016	0,0154
97500	0,016	0,015	0,016	0,016	0,016	0,0158
100000	0,016	0,017	0,016	0,016	0,017	0,0164



Conclusion

Le tri par tas est un algorithme de tri par comparaisons non stable, son principe est de transformer le tableau en un tas qui est une structure de données de type arbre ou la valeur d'un parent est toujours inferieur aux valeurs de ses fils et ou la racine a la plus petite valeur (dans le cas d'un min heap).

On supprime ensuite la racine du tableau (on permute la racine avec le dernier élément du tableau) et on crée le tas à nouveau (sans prendre en considération le dernier élément supprimé du tas), on répète ce processus jusqu'à ce que tous les éléments du tableau soient triées.

La complexité temporelle dans le cas moyen est de O(nlogn).

Cet algorithme a l'avantage d'avoir une complexité temporelle de O(nlogn) dans les pires des cas.