091M4041H - Assignment 1
Algorithm Design and Analysis

Song Qige 2017E8018661044

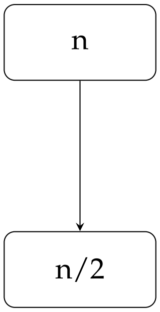
2017 年 10 月 13 日

1 determine the median of 2n values(1)

1.1 Algorithm Description

The problem is to find the median of two databases with n elements.We can separately obtaine the median of database a and the median of database b.If the median of database a is less than the median of database b, then the overall median may only appear in the right interval of a and the left interval of b.If the median of a is greater than or equal to the median of b, then the overall median may only appear in the left interval of a and the right interval of b.So that the range of seeking will continue to shrink.

1.2 subproblem reduction graph



```
function findMediaNum($databaseA, beginA, databaseB, beginB, n$)
    if $n == 1$ then
        return $min(A[beginA], B[beginB])$
    end if
    $mediumA = A[beginA + n/2 - 1]$
    $mediumB = B[beginB + n/2 - 1]$
    if $mediumA < mediumB$ then
        findMediaNum($A, beginA + n/2, B, beginB, n/2$)
    else
        findMediaNum($A, beginA, B, beginB + n/2, n/2$)
    end if
end function
```

1.3 Prove the correctness

Set up two databases are A, B, each containing n elements.If we remove k numbers smaller than the median and k numbers bigger than the median,the median of the resulting sub-array is same as the median of the origin array.Then

$$Medium(A,B) = f(n)_{(A_{0-n},B_{0-n})} = \begin{cases} f(n/2)_{(A_{n/2-n},B_{0-n/2})} & A_{n/2} < B_{n/2} \\ \\ f(n/2)_{(A_{n/2-n},B_{0-n/2})} & A_{n/2} >= B_{n/2} \end{cases}$$

1.4 time complexity

$T(n) = T(n/2) + c = O(logn)$

2 Find the kth largest element in an unsorted array(2)

2.1 Algorithm Description

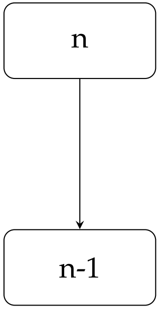
Randomly choose element Ai,Comparing the other elements with Ai,if bigger than Ai put into set S+,if smaller put into set S-.If $|S-| = k$ - 1,the result is Ai.If $|S-| > k$ - 1,the problem can be simplified as find the kth largest element in S-.If $|S-| < k$ - 1,the problem can be simplified as find the $(k - |S-| + 1)^{th}$ largest element in S+.

2.2 pseudo-code

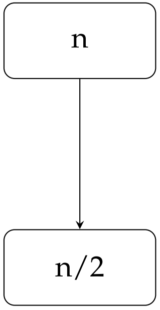
```
function FindKthElement($A, k$)
    $Randomly\ choose\ Ai$
    for $j = 0 \to n - 1$ do
        $put\ A[j]\ in\ S-\ if\ A[j] < A[i]$
        $put\ A[j]\ in\ S+\ if\ A[j] > A[i]$
    end for
    if $|S-| = k$ - 1 then
        return $Ai$
    else
        if $|S-| > k$ - 1 then
            FindKthElement($S-, k$)
        end if
        FindKthElement($S-, k - |S-| + 1$)
    end if
end function
```

2.3 subproblem reduction graph

The worst case:



The best case:



2.4 Prove the correctness

$$f(k)_A = \begin{cases} Ai & |S-|+1=k \\ f(k)_{S-} & |S-|+1>k \\ f(k-|S-|+1)_{S+} & |S-|+1<k \end{cases}$$

2.5 time complexity

The worst case:Ai is the biggest/smallest element of the array, $T(n) = T(n-1)+cn = O(n^2)$
The best case:Ai is the median element of the array, $T(n) = T(n/2)+cn = O(n)$

3 Divide convex polygon into traingles(5)

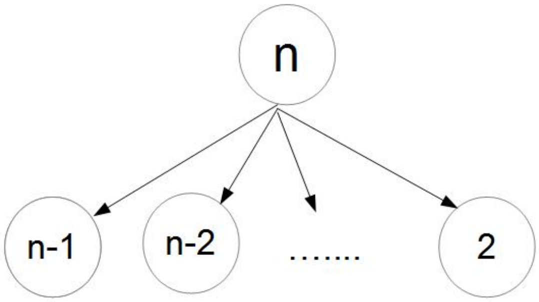
3.1 Algorithm Description

Numbering the vertices of the convex polygon from 1 to n,choose the vertice 1,n and any one of vertices 2 to n-1.The 3 vertices choosed can divide the polygon into a polygon of i edges,a traingle and a polygon of n-i+1 edges.The total ways to divide the origin polygon equal to the product of the ways to divide the polygon of i edges and the ways to divide the polygon of n-i+1 edges. To avoid repeat caculation,we can store the ways to divide a polygon of k edges into the kth element of array Walked[].

3.2 pseudo-code

```
int Walked[N] = 0
function DivideTraingle($n$)
    if $n \leq 3$ then
        return $1$
    end if
    if $Walked[n]! = 0$ then
        return $Walked[n]$
    end if
    $int\ count = 0$
    for $i = 2 \rightarrow n - 1$ do
        $count+ = DivideTraingle(i) * DivideTraingle(n - i + 1)$
    end for
    $Walked[n] = count$
end function
```

3.3 subproblem reduction graph



3.4 Prove the correctness

$$f(n) = \begin{cases} 1 & n \leq 3 \\ \sum_{n=1}^{N} f(i) * f(n+1-i) & n > 3 \end{cases}$$

3.5 time complexity

$T(n) = T(n-1) + T(n-2) + T(n-3) + .. + T(2) = O(n-2) + O(n-3) + .. + O(1) = O(n^2)$

4 Sort-and-Count algorithm(8)

4.1 8-1 Source code

```
#include <iostream>
#include <fstream>
#include <string>
#include <time.h>
```

```

using namespace std;
const int N = 100001;

int num[N];//store the elements need to be sorted
int tmp[N];//temp array after merge
long long ans;//number of invertions

void merge_sort(int l,int m,int r)
{
    int i = l;
    int j = m + 1;
    int k = l;
    while(i <= m && j <= r)
    {
        //i < j,a[i] > a[j]->invertion
        if(num[i] > num[j])
        {
            tmp[k++] = num[j++];
            //each element after a[i]->invertions
            ans += m - i + 1;
        }
        else
        {
            tmp[k++] = num[i++];
        }
    }

    while(i <= m) tmp[k++] = num[i++];
    while(j <= r) tmp[k++] = num[j++];

    for(int i=l;i<=r;i++)
        num[i] = tmp[i];
}

void sort_and_count(int l,int r)
{
    if(l < r)
    {
        int m = (l + r) / 2;
        sort_and_count(l,m);
        sort_and_count(m+1,r);
        merge_sort(l,m,r);
    }
}

int main()
{
    clock_t start,end;
    //Read the number from the file line by line
    ifstream infile;
    infile.open("8.txt");
    if(!infile) cout<<"error"<<endl;
    int t1;

    int len = 0;
    cout<<"store in the array"<<endl;
    while(infile>>t1)
    {
      num[len] = t1;
```

```
        len++;
    }

    /*print the array*/
    int i;
    for(i=0;i<len;i++)
        cout<<num[i]<<endl;
    cout<<endl;

    /*sort the array and count inversions*/
    ans = 0;
    start = clock();
    sort_and_count(0,len-1);
    end = clock();

    cout << "The number of inversions:" << ans;
    cout << "time to merge sort and count : " \
    << (double)(end-start)/CLK_TCK <<endl;
    return 0;
}
```

4.2 8-2 Souce code

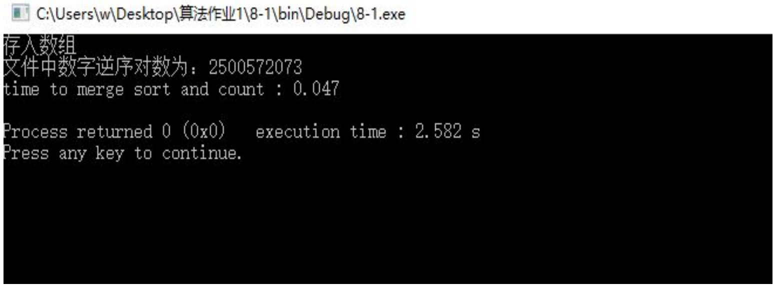
It's possible to use the Quick-Sort idea.
source code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <time.h>

```

```
using namespace std;

const int N = 100001;
long long ans;//number of invertions

void quick_sort_and_count(int *a,int l,int r)
{
    int left_num = 0;
    int right_num = 0;

        /*S- set,store the elements
    smaller than pivot*/
    int *left = (int*)calloc(r-l,sizeof(int));
        /*S+ set,store the elements
    bigger than pivot*/
    int *right = (int*)calloc(r-l,sizeof(int));

    int pivot;
    /*put smaller element into s-,
         add the number of s+ now to
         the inversion number.
     after a round,add the number
     of s- to the inversion number*/
    if(l < r)
    {
        pivot = a[l];
        int i;
        for(i = l + 1;i <= r;i++)
        {
            if(a[i] < pivot)
            {
                left[left_num++] = a[i];
                ans += right_num;
            }
            else
            {
                right[right_num++] = a[i];
            }

        }
        ans += left_num;

    quick_sort_and_count(left,0,left_num-1);
    quick_sort_and_count(right,0,right_num-1);
    }
}


int main()
{
    clock_t start,end;
    //Read the number from the file line by line
    ifstream infile;
    infile.open("8.txt");
    if(!infile) cout<<"error"<<endl;
    int t1;

    int len = 0;
    cout<<"store in the array"<<endl;
    while(infile>>t1)
    {
        num[len] = t1;
        len++;
    }

    /*print the array*/
    int i;
    for(i=0;i<len;i++)
        cout<<num[i]<<endl;
    cout<<endl;

    /*sort the array and count inversions*/
    ans = 0;
    start = clock();
    quick_sort_and_count(0,len-1);
    end = clock();

    cout << "The number of inversions:" << ans;
    cout << "time to quick sort and count : " \
    << (double)(end-start)/CLK_TCK <<endl;
    return 0;
}
```

4.3 8-2 comparing

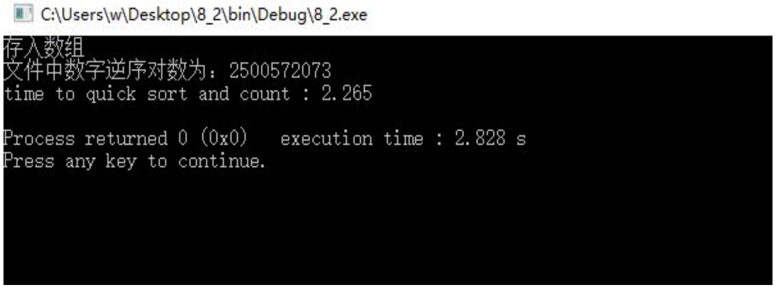
result for merge sort and count:



C:\Users\w\Desktop\算法作业1\8-1\bin\Debug\8-1.exe

```
存入数组
文件中数字逆序对数为：2500572073
time to merge sort and count : 0.047

Process returned 0 (0x0)   execution time : 2.582 s
Press any key to continue.
```

result for quick sort and count:



C:\Users\w\Desktop\8_2\bin\Debug\8_2.exe

```
存入数组
文件中数字逆序对数为：2500572073
time to quick sort and count : 2.265

Process returned 0 (0x0)   execution time : 2.828 s
Press any key to continue.
```

5 the closest pair problem(9)

5.1 Source code

```
#include <iostream>
#include <math.h>
#include <algorithm>

using namespace std;
typedef struct point//point in the plane
{
  double x;
  double y;

  //sort the points by x coordinate
  friend bool operator < \
  (const point &a, const point &b){
                if(a.x == b.x)
                        return a.y < b.y;
                return a.x < b.x;
        }
}point;

const int N = 100;
point points[N];
point a,b;//the closest point pair

//use quick sort to sort the points by y
void y_QuickSort(point y_sort_points[],\
  int l,int r)
{
    if (l < r)
    {
        int i = l, j = r;
        point x = y_sort_points[l];
        while (i < j)
        {
           while(i < j &&\
           y_sort_points[j].y>= x.y)
             j--;
           if(i < j)
           y_sort_points[i++] = y_sort_points[j];
           while(i < j && \
           y_sort_points[i].y< x.y)
              i++;
           if(i < j)
           y_sort_points[j--] = y_sort_points[i];
        }
        y_sort_points[i] = x;
        y_QuickSort(y_sort_points,l, i - 1);
        y_QuickSort(y_sort_points,i + 1, r);
    }
}

double Distance(point a,point b)
{
    return sqrt((a.x - b.x)*(a.x - b.x) \
    + (a.y - b.y) * (a.y - b.y));
}

double closest(int l,int r)
{
    if(l == r)
    {
         a.x = points[l].x;
                 a.y = points[l].y;

                 b.x = points[r].x;
                 b.y = points[r].y;
        return 0;
    }
    else if(l == r - 1)
```

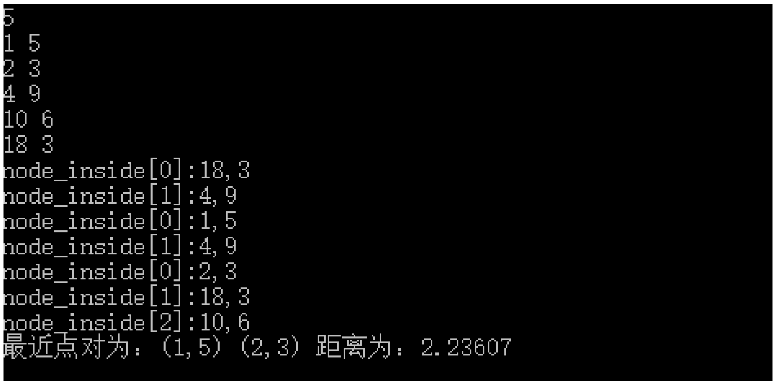
```
        {
             a.x = points[l].x;
                     a.y = points[l].y;

                     b.x = points[r].x;
                     b.y = points[r].y;

            return Distance(points[l],points[r]);
        }

        int m = (l + r)/2;
            /*caculate the closest pair
        in left and right*/
        double res = min(closest(l,m),closest(m,r));
        double tmp;
        int i,j;
        int node_inside_num = 0;
        point node_inside[N];

        /*store the points in 2d width strip
        in temp array*/
        for(i = m - 1; i >= l && \
        points[m].x - points[i].x < res; i--){
        node_inside[node_inside_num++] = points[i];
        }

        for(i = m + 1; i <= r &&\
        points[m].x - points[i].x < res; i++){
        node_inside[node_inside_num++] = points[i];
        }
```

```

    //sort the points in 2d width strip by y
    y_QuickSort(node_inside,0,node_inside_num-1);

        /*for each points in 2d width
    strip sorted by y,caculate the
    diatance with 7 points after it,
    if closer,update the result*/
    for(i = 0;i < node_inside_num;i++)
    {
        int end_inside;
        if(node_inside_num - i - 1 < 8)
        end_inside = node_inside_num;
        else end_inside = i + 8;
        for(j = i + 1;j < end_inside;j++)
        {
            tmp = Distance(node_inside[i], \
            node_inside[j]);
            if(tmp < res)
            {
                res = tmp;
                a.x = node_inside[i].x;
                a.y = node_inside[i].y;
                b.x = node_inside[j].x;
                b.y = node_inside[j].y;
            }
         }
      }
    return res;
}
```

```
int main()
{
    int n;
    //read the count of points in the plain
    cin>>n;
    int i;
    for(i=0; i<n ; i++){
        //read the coordinate of the n points
        cin>>points[i].x>>points[i].y;
    }

    //sort the points by x coordinate
    sort(points,points+n);
    double result = closest(0,n - 1);
    cout<<"The␣closest␣point␣pair␣is:(";
    cout<<a.x<<","<<a.y<<")␣("<<b.x<<","<<b.y;
    cout<<")␣the␣closest␣distance␣is:"<<result;

    return 0;
}
```

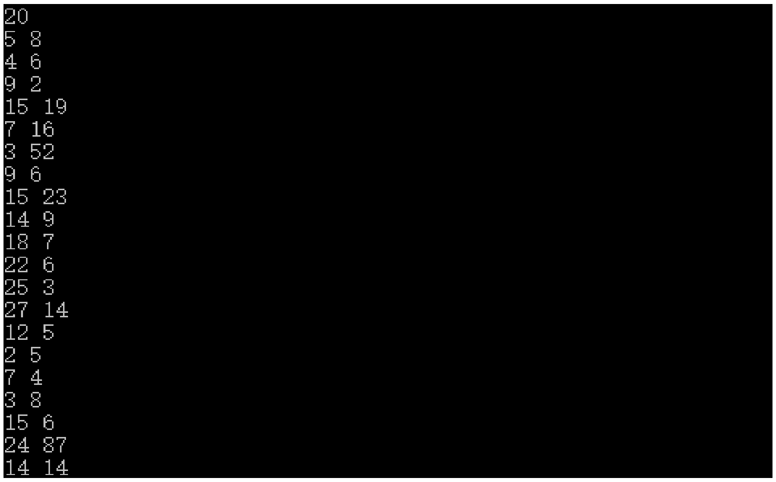
5.2 result

test case for 5 points:

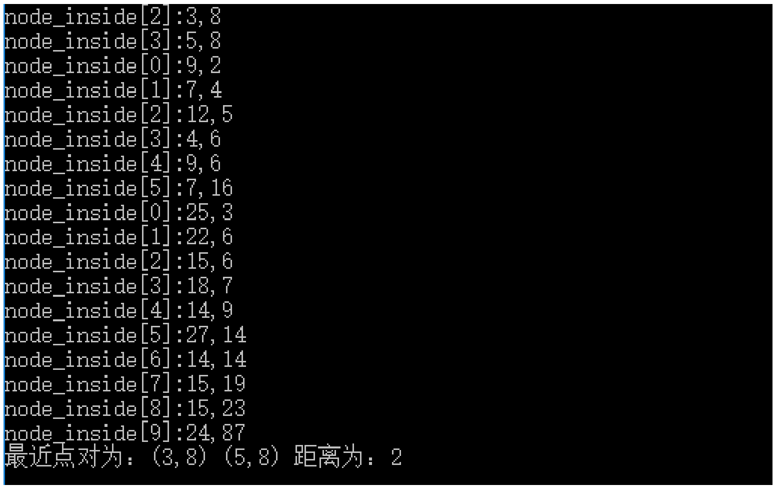


```
5
1 5
2 3
4 9
10 6
18 3
node_inside[0]:18,3
node_inside[1]:4,9
node_inside[0]:1,5
node_inside[1]:4,9
node_inside[0]:2,3
node_inside[1]:18,3
node_inside[2]:10,6
最近点对为：（1,5）（2,3）距离为：2.23607
```

test case for 20 points:



```
20
5 8
4 6
9 2
15 19
7 16
3 52
9 6
15 23
14 9
18 7
22 6
25 3
27 14
12 5
2 5
7 4
3 8
15 6
24 87
14 14
```



```
node_inside[2]:3,8
node_inside[3]:5,8
node_inside[0]:9,2
node_inside[1]:7,4
node_inside[2]:12,5
node_inside[3]:4,6
node_inside[4]:9,6
node_inside[5]:7,16
node_inside[0]:25,3
node_inside[1]:22,6
node_inside[2]:15,6
node_inside[3]:18,7
node_inside[4]:14,9
node_inside[5]:27,14
node_inside[6]:14,14
node_inside[7]:15,19
node_inside[8]:15,23
node_inside[9]:24,87
最近点对为：（3,8）（5,8）距离为：2
```