POJ 报告

09018316 黄开鸿 第一次

POJ 1064

- 题意概述：

  按一定长度切割若干根长短不一的网线，求出最长的切割长度。

- 题意分析

  如果直接暴力搜索，那么切割长度的可能值是0.00到10000.00，也就是达到了10的六次方，而每次验证该长度可不可行又需要遍历最多10000次，明显无法在1000MS内完成。

  经分析，每次验证切割长度的可行性必须遍历所有网线，无法优化，故只能优化尝试切割长度的次数。因为切割长度的选取是有明确的大小关系，所以可以采用二分法，将原本 O(n) 复杂度的尝试过程缩短到 O(log(n))。

- 实际代码

```
#include <string>
#include <iomanip>
#include <iostream>
#include <cmath>
using namespace std;

const int maxN = 10000;//n最大值
const int maxK = 10000;//k最大值

int n = 0;
  int k = 0;

  double arr[10000] ; //储存所有网线


bool isSatisfied(double num){
    int total =0;
    for(int i = 0; i <n;i++){
        total+=arr[i]/num;
    }
    return total>=k;
}

int main(){
    cin>>n>>k;
    int i = 0;
    double maxLength = 0;
    while(n>i){
        cin>>arr[i];
```

```
            maxLength = arr[i]>maxLength?arr[i]:maxLength;
            i++;

        }
        double l = 0;double r = maxLength;
        for(int cut = 0;cut<100;cut++){
            double m = (l+r)/2.0;
            if(isSatisfied(m)){
                    l = m;
            }
            else{
                r= m;
            }
        if(r-l<0.001){
            break;
        }
        }
        cout<<fixed<<setprecision(2)<<floor(r*100)/100;
        return 0;
    }
```

POJ 3714

- 题意概述

  在给定的两个点集中找到距离最近的一组点对，两点来自不同点集

- 题意分析

  直接暴力搜索的复杂度最大为 10 的十次方，在5000ms内明显无法执行完。

  利用分治思想将问题分解，并在小问题中加以优化。 实现算法的关键在于利用子问题的返回值（即自身一部分中的最小距离点对），排除部分不可能的解，使得自身问题简化，防止部分无效比较。

  但是该算法的某一子问题解决时，仍使用暴力搜索比较所有可能的可行解（遍历所有与分界点 x 坐标之差不超过左右子问题返回的最小值 d 的点），此处其实也能优化，但是还未能完全实现，现阶段只是多排除了横坐标距离以超过 d 的点对。该题给出 5000ms 时限使得此步不需要完美优化也能在限时内完成，如果限时为标准按 1000ms 则还需优化。

  具体实现详见以下代码注释。

- 实际代码

```
#include <iostream>
#include <cstdio>
#include <iomanip>
#include <cstring>
#include <algorithm>
```

```
#include <cmath>
using namespace std;


const long long INF = 1000000000000;
const int N = 100010;//比100000略大，保证稳定性

struct Node
{
    long long x, y;//储存坐标
    int id;//标记信息，用于判定点集
    Node(long long x = 0, long long y = 0, int id = 0) : x(x), y(y), id(id) {}
    const bool operator<(const Node A) const //重载比较符用于直接调用sort函数
    {
        return x == A.x ? y < A.y : x < A.x;
    }
} node[2 * N];//用以储存所有点

int n;
//计算两点距离
double dis(int a, int b)
{
    return sqrt((double)((node[a].x - node[b].x) * (node[a].x - node[b].x) +
(node[a].y - node[b].y) * (node[a].y - node[b].y)));
}
//递归调用函数
double solve(int l, int r)
{
    //如果递归到只有一个点，返回无穷大
    if (l == r)
        return INF;
    //位运算取中点
    int mid = (l + r) >> 1;
    double a = solve(l, mid);
    double b = solve(mid + 1, r);
    //求下层递归返回的最小距离
    double d = min(a, b);
    //归并两边的递归结果
    for (int i = mid; i >= l; --i)
    {
        //如果左边当前点已经和mid距离大于下层递归结果，break
        if (node[mid].x - node[i].x > d)
            break;
        for (int j = mid + 1; j <= r; ++j)
        {
             //如果右边当前点已经和左边当前点距离大于下层递归结果，break
            if (node[j].x - node[i].x > d)
                break;
            double tmp = dis(i, j);
            //比较两不同点集距离
            if (node[i].id != node[j].id && tmp < d)
                d = tmp;
        }
    }
```

```
    return d;
}

int main()
{
    int time;
    cin>>time;
    while (time--)
    {

        cin>>n;
        for (int i = 0; i < n; ++i)
        {
            cin>>node[i].x>>node[i].y;
            node[i].id = 1;
        }
        for (int i = 0; i < n; ++i)
        {
            cin>>node[i + n].x>>node[i + n].y;
            node[i + n].id = 2;
        }
        sort(node, node + 2 * n);
        double res = solve(0, 2 * n - 1);
        //不知道为啥一定要这样取才对，取注释段代码则 PE
        printf("%.3lf\n", res);
        //cout<<fixed<<setprecision(3)<<res;
    }
}
```