

华中科技大学

2023

## 算法设计与分析实践报告

专    业：    计算机科学与技术

班    级：    CS2202

学    号：    U202215378

姓    名：    冯瑞琦

完成日期：    2023. 1. 7



# 华中科技大学课程设计报告

---

## 目 录

1. 完成情况.....	1
2. 3714 解题报告 .....	2
2.1 题目分析 .....	2
2.2 算法设计 .....	2
2.3 性能分析 .....	3
2.4 运行测试 .....	3
3. 1088 解题报告 .....	4
3.1 题目分析 .....	4
3.2 算法设计 .....	4
3.3 性能分析 .....	5
3.4 运行测试 .....	6
4. 1700 解题报告 .....	7
4.1 题目分析 .....	7
4.2 算法设计 .....	7
4.3 性能分析 .....	8
4.4 运行测试 .....	8

# 华中科技大学课程设计报告

---

<b>5. 1860 解题报告 .....</b>	<b>8</b>
5.1 题目分析 .....	8
5.2 算法设计 .....	9
5.3 性能分析 .....	9
5.4 运行测试 .....	60
<b>6. 总结 .....</b>	<b>11</b>
6.1 实验总结 .....	11
6.2 心得体会和建议 .....	11
<b>附录 .....</b>	<b>13</b>

# 华中科技大学课程设计报告

## 1. 完成情况

本次实验中总共完成 25 道题目，完成题目编号如下图。

U202215378--Feng RuiQi			
Last Logged Time:2024-01-08 20:37:21.0			
Compare <input type="text" value="U202215378"/> and <input type="text" value="U202215378"/> <input type="button" value="GO"/>			
Rank:	36200	Solved Problems List 1000 1005 1042 1050 1062 1088 1185 1201 1328 1636 1700 1753 1860 2228 2366 2387 2503 2506 2586 3040 3169 3233 3295 3660 3714	
Solved:	25		
Submissions:	37		
School:	华中科技大学		
Email:	u202215378@hust.edu.cn		

共测评 37 次，测评记录如下图。

Run ID	User	Problem	Result	Memory	Time	Language	Code Length
24407688	U202215378	2506	Accepted	444K	16MS	C++	1427B
24407686	U202215378	1324	Time Limit Exceeded			C++	2461B
24407683	U202215378	1324	Wrong Answer			C++	2876B
24407682	U202215378	1324	Compile Error			C++	1595B
24407674	U202215378	3169	Accepted	684K	32MS	C++	1538B
24407670	U202215378	1201	Accepted	11580K	375MS	C++	2409B
24407668	U202215378	3660	Accepted	200K	16MS	C++	844B
24407664	U202215378	1062	Accepted	288K	47MS	C++	1434B
24407663	U202215378	2387	Accepted	324K	110MS	C++	1119B
24407654	U202215378	1860	Accepted	192K	0MS	C++	1235B
24407630	U202215378	2586	Accepted	240K	16MS	C++	543B
24407618	U202215378	1700	Accepted	248K	16MS	C++	1000B
24407615	U202215378	3040	Accepted	160K	32MS	C++	5622B
24407611	U202215378	1328	Accepted	192K	32MS	C++	1312B
24407608	U202215378	1328	Compile Error			C++	1269B
24407604	U202215378	1328	Compile Error			C++	1210B
24407601	U202215378	1042	Accepted	168K	219MS	C++	1814B
24407599	U202215378	2228	Accepted	316K	79MS	C++	1019B
24407598	U202215378	2228	Compile Error			G++	754B
24407594	U202215378	2228	Compile Error			C++	754B

Run ID	User	Problem	Result	Memory	Time	Language	Code Length
24407688	U202215378	2506	Accepted	444K	16MS	C++	1427B
24407686	U202215378	1324	Time Limit Exceeded			C++	2461B
24407683	U202215378	1324	Wrong Answer			C++	2876B
24407682	U202215378	1324	Compile Error			C++	1595B
24407674	U202215378	3169	Accepted	684K	32MS	C++	1538B
24407670	U202215378	1201	Accepted	11580K	375MS	C++	2409B
24407668	U202215378	3660	Accepted	200K	16MS	C++	844B
24407664	U202215378	1062	Accepted	288K	47MS	C++	1434B
24407663	U202215378	2387	Accepted	324K	110MS	C++	1119B
24407654	U202215378	1860	Accepted	192K	0MS	C++	1235B
24407630	U202215378	2586	Accepted	240K	16MS	C++	543B
24407618	U202215378	1700	Accepted	248K	16MS	C++	1000B
24407615	U202215378	3040	Accepted	160K	32MS	C++	5622B
24407611	U202215378	1328	Accepted	192K	32MS	C++	1312B
24407608	U202215378	1328	Compile Error			C++	1269B
24407604	U202215378	1328	Compile Error			C++	1210B
24407601	U202215378	1042	Accepted	168K	219MS	C++	1814B
24407599	U202215378	2228	Accepted	316K	79MS	C++	1019B
24407598	U202215378	2228	Compile Error			G++	754B
24407594	U202215378	2228	Compile Error			C++	754B

## 2. POJ3714 Raid 解题报告

### 2.1 题目分析

有两组坐标，一组是  $n$  个发电站的坐标，一组是  $n$  个士兵的坐标。求哪个士兵离发电站的距离最短。本题可以看作是是课上讲的求最近点对问题的变形，是典型的用分治法求解的题目。

### 2.2 算法设计

`getint` 函数用来从输入流中读取一个长整数，通过循环读取字符，直到遇到数字字符，然后将字符转换为整数。支持处理负数。结构体 `point` 示平面上的点，包括  $x$  和  $y$  坐标以及一个标记用来表示点所属的集合。函数 `cmpx` 和 `cnpy` 作为排序函数，分别按照  $x$  坐标从小到大排序，若  $x$  坐标相同则按照  $y$  坐标从小到大排序，和按照  $y$  坐标从小到大排序，若  $y$  坐标相同则按照  $x$  坐标从小到大排序。函数 `dis` 计算两点之间的欧几里得距离。

`solve` 函数是算法的核心，采用分治法的思想。首先递归地计算左右两个子问题的最小距离，然后计算横跨中间的最小距离。在计算横跨中间的最小距离时，先将中间的  $x$  坐标与左右两边的点进行比较，筛选出距离中间  $x$  坐标不超过当前最小距离的点，再按照  $y$  坐标进行排序，最后在有限的候选点中计算最小距离。同时，通过剪枝操作，可以有效减少计算量。

最后在主函数中首先读入测试样例的数量  $T$ ，然后对每个测试样例进行处理。对于每个样例，先读入点的数量  $n$ ，接着按照  $x$  坐标

从小到大的顺序读入  $2n$  个点，并初始化它们的标记。最后，调用 `solve` 函数计算最小距离并输出结果。

## 2.3 性能分析

排序的时间复杂度为  $O(n \log n)$ ，其中  $n$  是点的个数。分治法的递归过程，每一层都要对点进行排序和计算距离，时间复杂度为  $O(n \log n)$ 。由于有  $\log n$  层，总的时间复杂度为  $O(n \log^2 n)$ 。剪枝操作和其他常数项的影响相对较小。

总体来说，该算法的时间复杂度为  $O(n \log^2 n)$ ，其中  $n$  是点的个数。

## 2.4 运行测试

```
2
4
0 0
0 1
1 0
1 1
2 2
2 3
3 2
3 3
1.414
4
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0.000
```

```
Sample Input
2
4
0 0
0 1
1 0
1 1
2 2
2 3
3 2
3 3
4
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0

Sample Output
1.414
0.000
```

上图中左边为运行测试结果，白色字为输入，被标记为蓝色的字为输出，与 POJ 上给出的例子相符，功能正确。

## 3. POJ1088 滑雪解题报告

### 3.1 题目分析

此题的最长的滑坡指的是滑坡的个数，每个格子代表一个滑坡，即从 10 开始出发，最长可以经过 9 个格子。整个思路可以类比最长下降子序列，与最长下降子序列相比，这个是二维的，且边界值并非  $dp[0][0]$  或  $dp[1][1]$ 。

### 3.2 算法设计

用动态规划解决该问题。采取结构体存储各节点值和节点坐标，并对结构体升序排序，则排序后的第一个节点即为整个  $dp$  数组的边界；在求解过程中是根据排序的顺序进行求解的。 $dp[i][j]$  表示坐标为  $(i, j)$  的最长滑坡为  $dp[i][j]$ 。

自顶向下分析：

求最优解：求最长的滑坡

最优子结构：当前位置的最长滑坡，依赖于前面已经求得的最长滑坡

重叠子问题：想要求中间第  $i$  步 (坐标是  $i, j$ ) 的最大值  $dp[i][j]$ ，

需要从边界开始计算，直到求到当前位置；想要求终点的最大值

$dp[m-1][n-1]$ ，仍要从边界位置开始计算。

自下而上解决问题：

、 边界：经过排序后，值最小的即为边界

状态转移：

探索上下左右四个位置，若当前节点数值小，才符合题意，此时在当前节点  $dp[i][j]$  和周围节点  $dp[x][y]+1$  中取最大值，即  $dp[i][j] = \max(dp[i][j], dp[x][y] + 1)$ ；

设计算法时注意数组在探索时不能越界访问，ans 的初始值为 1，当矩阵中所有数值均相等时，答案为 1。

### 3.3 性能分析

时间复杂度分析：

排序操作的时间复杂度为  $O(m * n * \log(m * n))$ ，其中  $m$  和  $n$  分别为矩阵的行数和列数。对排序后的节点进行遍历，对每个节点的四个方向进行比较，更新  $dp$  数组的时间复杂度为  $O(m * n)$ 。

总体时间复杂度为  $O(m * n * \log(m * n))$ 。

空间复杂度分析：

使用了二维数组  $dp$  来保存每个位置的最长滑坡长度，空间复杂度为  $O(m * n)$ 。使用了结构体数组  $nodes$  来保存每个元素的值及其坐标，空间复杂度为  $O(m * n)$ 。

总体空间复杂度为  $O(m * n)$ 。



## 3.4 运行测试

下图中为运行测试结果，上方为输入，最后一行为输出，与 POJ 上给出的例子相符，功能正确。

```
5 5
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
25

-----
Process exited after 297.2 seconds with return value 0
请按任意键继续. . . |
```

```
10 10
1 2 300 4 5 6 7 8 9 10
20 19 18 17 16 15 14 13 12 11
21 22 23 24 25 26 27 28 29 30
40 39 38 37 36 35 34 33 32 31
41 42 43 44 45 46 47 48 49 50
60 59 58 57 56 55 54 53 52 51
61 62 63 64 65 66 67 68 69 70
80 79 78 77 76 75 74 73 72 71
81 82 83 84 85 86 87 88 89 90
100 99 98 97 96 95 94 93 92 91
97

-----
Process exited after 2.031 seconds with return value 0
请按任意键继续. . . |
```

## 4. POJ1700 Crossing River 解题报告

### 4.1 题目分析

只有一艘船，能乘 2 人，船的运行时间为 2 人中较多一人的时间，过去后还需一个人把船划回来，问把  $n$  个人运到对岸，最少需要多久。本题适合用贪心算法求解。

### 4.2 算法设计

`Init()` 函数用于初始化数组 `t[]`。`quick_Sort()` 是快速排序算法，对输入数组 `t[]` 进行排序。`solve()` 函数根据排序后的数组 `t[]` 计算出一个 `sum` 值。

总人数  $n \leq 3$  时：

$n=1$   $t=t[1]$

$n=2$   $t=\max(t[1], t[2])$                       1 和 2 中耗时长的那个

$n=3$   $t=t[1]+t[2]+t[3]$                       1 和 3 先去, 1 回来, 再和 2 去

否则：考虑每次送两人过河（其中时间已经排好序了）

1. 1 和  $n$  去, 1 回来, 1 和  $n-1$  去, 1 回来

$n-=2$  用时  $2*t[1]+t[n]+t[n-1]$

2. 1 和 2 去, 1 回来,  $n$  和  $n-1$  去, 2 回来

$n-=2$  用时  $t[1]+2*t[2]+t[n]$

每次取两种方法的最小耗时，直到  $n \leq 3$ 。

## 4.3 性能分析

`Init()` 函数的时间复杂度为  $O(n)$ ，其中  $n$  是数组的长度。快速排序 `quick_Sort()` 的时间复杂度平均为  $O(n \log n)$ ，最坏情况下为  $O(n^2)$ ，取决于数据的分布情况。`solve()` 函数中的循环是根据特定规则累加 `sum` 值的，时间复杂度为  $O(n)$ 。

整体时间复杂度最坏情况下为  $O(n^2)$ 。

## 4.4 运行测试

下图中为运行测试结果，上方为输入，最后一行为输出，与 POJ 上给出的例子相符，功能正确。

```
1
4
1 2 5 10
17
-----
Process exited after 683.8 seconds with return value 0
请按任意键继续. . . |
```

## 5. POJ1860 Currency Exchange 解题报告

### 5.1 题目分析

给定  $N$  种货币，某些货币之间可以相互兑换，现在给定一些兑换规则，问能否从某一种货币开始兑换，经过一些中间货币之后，最后兑换回这种货币，并且得到的钱比之前的多。

利用 Bellman-Ford 算法的思想，但你要求的是没有负权，只需要对 Bellman-Ford 算法做一些改变即可。

## 5.2 算法设计

node 结构体表示货币兑换关系，包括起点 a、终点 b、汇率 rate 和手续费 c。

bellman\_ford 函数使用了 Bellman-Ford 算法来检测是否存在负权回路，即是否存在一条路径使得本金能够无限增加。利用一个数组 d 记录每个节点的本金，进行松弛操作。循环进行  $n-1$  次松弛操作，其中  $n$  为节点数。如果在第  $n$  轮仍然存在松弛操作，则说明存在负权回路。

主函数：

读入货币总数、兑换点数量、初始货币种类和初始本金。通过循环读入兑换关系，构建图。调用 bellman\_ford 函数判断是否存在负权回路，输出结果。

## 5.3 性能分析

Bellman-Ford 算法的时间复杂度为  $O(V * E)$ ，其中  $V$  为节点数， $E$  为边数。在这段代码中，循环进行了  $n-1$  次 Bellman-Ford 松弛操作，每次操作都遍历了所有的边。所以总体的时间复杂度为  $O((n-1) * E)$ ，其中  $E$  为边数。

## 5.4 运行测试

下图中为运行测试结果，上方为输入，最后一行为输出，与 POJ 上给出的例子相符，功能正确。

```
3 2 1 20.0
1 2 1.00 1.00 1.00 1.00
2 3 1.10 1.00 1.10 1.00
YES
|
```

## 6. 总结

### 6.1 实验总结

通过本次实验，我对算法设计与分析课程上讲解的分治法、动态规划、贪心算法、图论中求解单源最短路径的 Bellman-ford 算法、Dijkstra 算法、图树周游中 BFS、DFS 等搜索算法和求解有向图最大流的 Ford-Fulkerson 算法有了更深刻的理解。

通过解决具体的题目，我不仅在熟练度方面有所提升，而且学会了根据题目判断选择合适的算法，尽量降低时间复杂度和空间复杂度，也能根据题目建立合适的数学模型，如生成树、带权的有向图等。此外，通过实际编程，我对 C 语言和 C++ 的掌握程度也有所提升。

### 6.2 心得体会和建议

虽然课上学习了相关算法的原理和解题步骤，但是应为课上一般使用自然语言和伪代码描述算法，我在编程实操时还是会遇到不少问题，说明我对编程语言的熟悉度还不够。

有些题目乍一看不是很明晰，但是仔细分析就会发现是某种算法基本问题的变形，只要简单修改变形的部分，再应用相应的算法，问题就迎刃而解了。有些特殊情况也要周密地考虑到，否则就会出现部分答案错误。经过上网学习和反复修改调试，才能终于通过测评。

在 POJ 网站上测评时，有时候代码功能正确却不能一次通过，但是不用修改代码，多次提交后就能够通过。此外，有时候在 Dev C++

里不会报错的代码在网站中出现 Compile error 的情况，经分析发现网站不支持类似 `int i=0,j=0;` 这种中间带有逗号的语句，必须修改为不带逗号才能通过，说明网站的功能也许还有可与继续完善之处。

## 附录

### 3714 Raid

```
#include<algorithm>
#include<iostream>
#include<cstring>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<vector>
#define inf 2147483640
#define LL long long
#define free(a) freopen(a".in","r",stdin);freopen(a".out","w",stdout);
using namespace std;
inline LL getint() {
    LL x=0,f=1;char ch=getchar();
    while (ch>'9' || ch<'0') {if (ch=='-') f=-1;ch=getchar();}
    while (ch>='0' && ch<='9') {x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
const int maxn=1000010;
struct point {double x,y;int flag;}p[maxn];
int n,tmp[maxn];
bool cmpx(point a,point b) {
    return a.x==b.x ? a.y<b.y : a.x<b.x;
}
bool cmpy(int a,int b) {
```



```
    return p[a].y==p[b].y ? p[a].x<p[b].x : p[a].y<p[b].y;
}

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

double solve(int l,int r) {
    double res=1e60;
    if (l==r) return res;
    if (l+1==r) {
        if (p[l].flag==p[r].flag) return res;
        return dis(p[l],p[r]);
    }
    int mid=(l+r)>>1;
    res=solve(l,mid);
    res=min(res,solve(mid+1,r));
    int num=0;
    for (int i=l;i<=r;i++)
        if (fabs(p[i].x-p[mid].x)<=res) tmp[++num]=i;
    sort(tmp+1,tmp+num+1,cmpy);
    for (int i=1;i<=num;i++)
        for (int j=i+1;j<=num;j++) {
            if (fabs(p[tmp[i]].y-p[tmp[j]].y)>=res) break; //剪枝
            if (p[tmp[i]].flag!=p[tmp[j]].flag)
                res=min(res,dis(p[tmp[i]],p[tmp[j]]));
        }
    return res;
}
```

# 华中科技大学课程设计报告

---

```
int main() {
    int T;
    scanf("%d",&T);
    while (T--) {
        scanf("%d",&n);
        for (int i=1;i<=n;i++) scanf("%lf%lf",&p[i].x,&p[i].y),p[i].flag=0;
        for (int i=1;i<=n;i++)
            scanf("%lf%lf",&p[i+n].x,&p[i+n].y),p[i+n].flag=1;
        n<<=1;
        sort(p+1,p+1+n,cmpx);
        printf("%.3f\n",solve(1,n));
    }
    return 0;
}
```

## 1088 滑雪

```
#include<iostream>
#include<stdio.h>
#include<algorithm>
#include<vector>
using namespace std;

int dp[505][505]; //dp[i][j]表示: 坐标为 i,j 的最长滑坡长度为 dp[i][j]

int matrix[500][500];
int dir[4][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
int m, n;
```

# 华中科技大学课程设计报告

```
int ans; //记录答案
```

```
struct Node{
    int val;
    int x;
    int y;
    Node(){}
    Node(int vv, int xx, int yy){
        val = vv;
        x = xx;
        y = yy;
    }
};

vector<Node> nodes;

bool cmp(Node n1, Node n2){
    return n1.val < n2.val;
}

int main(){
    scanf("%d%d", &m, &n);
    for(int i = 0; i < m; ++i){
        for(int j = 0; j < n; ++j){
            scanf("%d", &matrix[i][j]);
            dp[i][j] = 1;
            nodes.push_back(Node(matrix[i][j], i, j));
        }
    }

    sort(nodes.begin(), nodes.end(), cmp);

    ans = dp[nodes[0].x][nodes[0].y] = 1;
```

```
for(int i = 1; i < nodes.size(); ++i){
    int xx = nodes[i].x;
    int yy = nodes[i].y;
    int vv = nodes[i].val;
    for(int k = 0; k < 4; ++k){
        int tx = xx + dir[k][0];
        int ty = yy + dir[k][1];
        if(tx >= 0 && tx < m && ty >= 0 && ty < n){
            if(vv > matrix[tx][ty]){
                dp[xx][yy] = max(dp[xx][yy], dp[tx][ty] + 1);
                if(dp[xx][yy] > ans){
                    ans = dp[xx][yy];
                }
            }
        }
    }
}

cout << ans << endl;

return 0;
}
```

## 1700 Crossing River

```
#include <stdio.h>
```

```
#define N 1001
```

```
int t[N];
```

```
void Init(int n)
{
    for(int i=1;i<=n;i++){
        scanf("%d",&t[i]);
    }
    return ;
}

int max(int a,int b)
{
    return a<b?b:a;
}

int min(int a,int b)
{
    return a>b?b:a;
}

void quick_Sort(int array[],int left,int right)
{
    int i=left,j=right,temp;
    int pivot=array[(i+j)/2];
    while (i<=j)
    {
        while(array[i]<pivot) i++;
        while(array[j]>pivot) j--;
        if (i<=j)
        {
            temp=array[i];
```

```
        array[i]=array[j];

        array[j]=temp;

        i++;

        j--;

    }

}

if(i<right) quick_Sort(array,i,right);

if(left<j)  quick_Sort(array,left,j);

}

int solve(int n)
{
    int sum=0;

    while (n>=4){

        sum+=min(t[n]+2*t[2]+t[1],t[n]+t[n-1]+2*t[1]);

        n-=2;

    }

    if(n==3)  sum+=t[1]+t[2]+t[3];

    if(n==2)  sum+=t[2];

    return sum;

}

int main()
{

    int T,n;

    scanf("%d",&T);

    while (T--){

        scanf("%d",&n);
```

```
Init(n);  
if(n==1){  
    printf("%d\n",t[1]);  
}  
else if(n==2){  
    printf("%d\n",max(t[1],t[2]));  
}  
else if(n==3){  
    printf("%d\n",t[1]+t[2]+t[3]);  
}  
else{  
    quick_Sort(t,1,n);  
    printf("%d\n",solve(n));  
}  
}  
return 0;  
}
```

## 1860 Currency Exchange

```
#include<iostream>  
#include<algorithm>  
#include<string>  
#include<cstring>  
#include<set>  
#include<map>
```

# 华中科技大学课程设计报告

---

```
using namespace std;
```

```
const int maxn = 200 + 5;
```

```
int n, m, s;    //货币总数、兑换点数量、有第 s 种货币
```

```
double v;      //持有的 s 货币本金
```

```
int cnt;
```

```
double d[maxn];
```

```
struct node
```

```
{
```

```
    int a;
```

```
    int b;
```

```
    double rate;
```

```
    double c;
```

```
}edge[maxn];
```

```
bool bellman_ford()
```

```
{
```

```
    memset(d, 0, sizeof(d));
```

```
    d[s] = v;
```

```
    bool flag;
```

```
    for (int i = 1; i <= n - 1; i++)
```



```
{
    flag = false;
    for (int j = 0; j < cnt; j++)
    {
        if (d[edge[j].b] < (d[edge[j].a] - edge[j].c)*edge[j].rate)
        {
            d[edge[j].b] = (d[edge[j].a] - edge[j].c)*edge[j].rate;
            flag = true;
        }
    }
    if (!flag) break;
}

for (int j = 0; j < cnt; j++)
if (d[edge[j].b] < (d[edge[j].a] - edge[j].c)*edge[j].rate)
    return true;
return false;
}

int main()
{
    //freopen("D:\\txt.txt", "r", stdin);
    int a, b;
    double rab, cab, rba, cba;
    while (~scanf("%d%d%d%lf", &n, &m, &s, &v))
    {
```

```
cnt = 0;

for (int i = 0; i < m; i++)
{
    scanf("%d%d%lf%lf%lf%lf", &a, &b, &rab, &cab, &rba, &cba);

    edge[cnt].a = a;
    edge[cnt].b = b;
    edge[cnt].rate = rab;
    edge[cnt++].c = cab;
    edge[cnt].a = b;
    edge[cnt].b = a;
    edge[cnt].rate = rba;
    edge[cnt++].c = cba;
}

if (bellman_ford())
    printf("YES\n");
else
    printf("NO\n");
}

return 0;
}
```