浙江工业大学

算法分析与设计实验报告

(2021 级)

实验题目	实验 6
学生姓名	温家伟
学生学号	202103151422
专业班级	大数据分析 2101
所在学院	理学院
提交日期	2023-5-29

实验目的

掌握回溯法的解题步骤

第七章 回溯算法

实验内容

1.旅行售货员

7-1 旅行售货员 分数 10

全屏浏览题目 切换布局 作者 任唯 单位 河北农业大学

某售货员要到若干城市去推销商品，已知各城市之间的路程(或旅费)。他要选定一条从驻地出发，经过每个城市一遍，最后回到驻地的路线，使总的路程（或总旅费）最小。

输入格式:

第一行为城市数n

下面n行n列给出一个完全有向图，如 i 行 j 列表示第 i 个城市到第 j 个城市的距离。

输出格式:

一个数字，表示最短路程长度。

输入样例:

```
3
0 2 1
1 0 2
2 1 0
```

输出样例:

```
3
```

2.单源最短路径

7-2 单源最短路径 分数 10

全屏浏览题目 切换布局 作者 朱允刚 单位 吉林大学

请编写程序求给定正权有向图的单源最短路径长度。图中包含n个顶点，编号为0至n-1，以顶点0作为源点。

输入格式:

输入第一行为两个正整数n和e，分别表示图的顶点数和边数，其中n不超过20000，e不超过1000。接下来e行表示每条边的信息，每行为3个非负整数a、b、c，其中a和b表示该边的端点编号，c表示权值。各边并非按端点编号顺序排列。

输出格式:

输出为一行整数，为按顶点编号顺序排列的源点0到各顶点的最短路径长度（不含源点到源点），每个整数后一个空格。如源点到某顶点无最短路径，则不输出该条路径长度。

输入样例:

```
4 4
0 1 1
0 3 1
1 3 1
2 0 1
```

输出样例:

```
1 1
```

代码长度限制 16 KB

实验结果及相应代码

1.旅行售货员

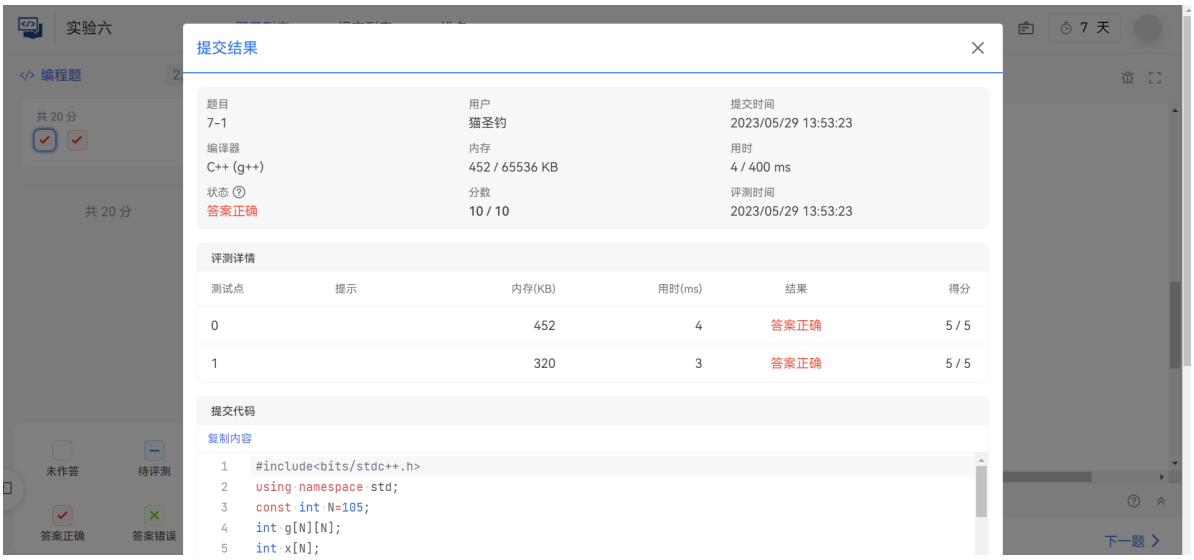
1.1 PTA提交代码截图

```
int x[N];
int n;
int ans=0x3f3f3f3f;
int now=0;
void TSP(int u)
{
    if(u>n)
    {
        if(now+g[x[n]][x[1]]<ans)
        {
            ans=now+g[x[n]][x[1]];
        }
    }
    else
    {
        for(int i=u;i<=n;i++)
        {
            if(g[x[u-1]][x[i]]!=0x3f3f3f3f&&now+g[x[u-1]]
            {
                swap(x[u],x[i]);
                now+=g[x[u-1]][x[u]];
                TSP(u+1);
                now-=g[x[u-1]][x[u]];
                swap(x[u],x[i]);
            }
        }
    }
}
int main()
{
    cin>>n;
    memset(g,0x3f,sizeof(g));
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            cin>>g[i][j];
```

```
    }
    for(int i=1;i<=n;i++) x[i]=i;

    TSP(2);
    cout<<ans;
    return 0;
}
```

1.2 PTA提交代码结果截图



1.3 算法分析

> 使用回溯法可以枚举所有情况（全排列），从而求出最优解。
>
> - 将起始点作为当前点。
> - 从当前点开始，前往没有遍历过的点。
> - 在第一次回到起始点时，判断是否遍历了所有点。
> - 如果是，则更新最优解，然后返回上一个节点（回溯到上一个分支）。
> - 如果不是，则继续前往未访问过的节点。
> - 直到所有情况都枚举完毕。

2.单源最短路径

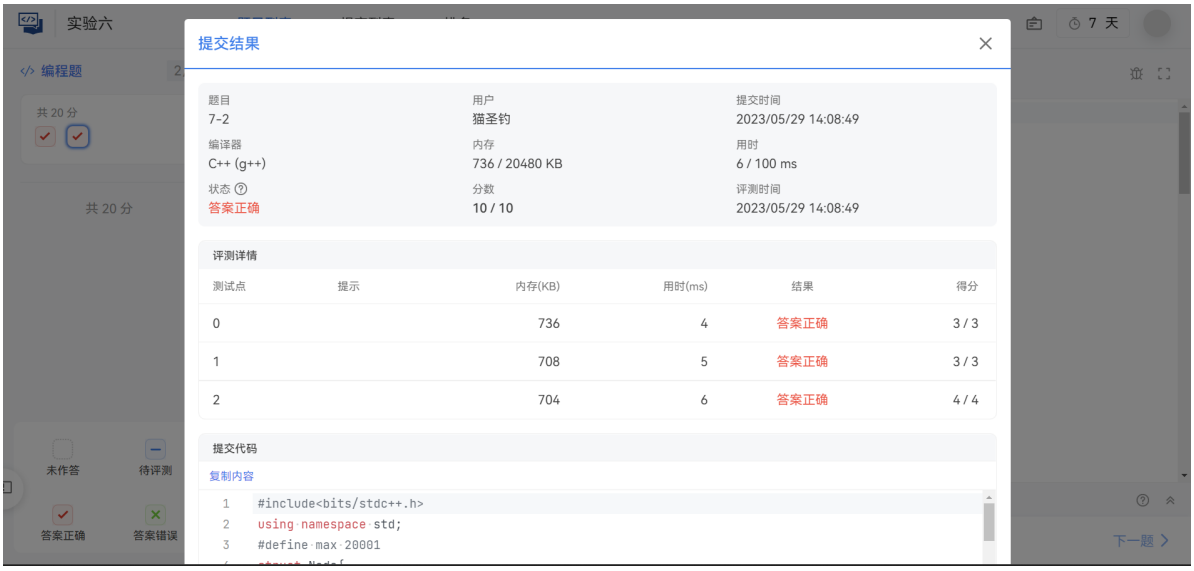
2.1 PTA提交代码截图

```cpp
#include<bits/stdc++.h>
using namespace std;
#define max 20001
struct Node{
    int to;
    int val;
    int next;
}node[max];
int head[max],n,m,num = 0;
int infinite = 99999;
void add(int from,int to,int val){
    num++;
    node[num].to = to;
    node[num].val = val;
    node[num].next = head[from];
    head[from] = num;
}
void dijkstra(){
    int dist[max];
    fill(dist, dist + 20001, infinite);
    int vis[max] = {0};
    for(int i = head[0]; i != 0; i = node[i].next){
        dist[node[i].to] = node[i].val;
    }
    vis[0] = 1;
    while(1){
        int m = -1;
        int min = infinite;
        for(int i = 0; i < n; i++){
            if(vis[i] != 1 && dist[i] < min){
                min = dist[i];
                m = i;
            }
        }
        if(m == -1){
            break;
        }
        vis[m] = 1;
```

```
            for(int i = head[m]; i != 0; i = node[i].next){

                if(vis[node[i].to] != 1 && min + node[i].val
                    dist[node[i].to] = min + node[i].val;
                }
            }
        }
        for(int i = 0; i < n; i++){
            if(dist[i] != infinite){
                cout << dist[i] << ' ';
            }
        }
    }
    int main(){
        memset(head,0,sizeof(head));
        cin >> n >> m;
        for(int i = 0; i < m; i++){
            int from,to,val;
            cin >> from >> to >> val;
            add(from,to,val);
        }
        dijkstra();
    }
```

2.2 PTA提交代码结果截图



2.3 算法分析

> 首先使用邻接表（链式前向星）存储图，每个节点维护一个存储边的数组，记录从该节点出发的所有边。然后采用 Dijkstra 算法求解单源最短路径。
>
> 1. 初始化：
>
> 用 dist[i] 记录源点 0 到 i 的最短距离，起初全部初始化为正无穷，即不存在一条从源点到达该节点的路径。
>
> 用 st[i] 记录节点 i 是否已访问过，起初全部初始化为 false。
>
> 1. 循环过程：
>
> （1）在未确定最短路的点中，找到到源点 0 距离最小的点 u。
>
> （2）更新 u 的所有相邻节点 v 的距离。如果源点 0 到 v 的距离大于源点 0 到 u 的距离加上 u 到 v 的距离，说明存在从源点到 v 更短的路径，更新 dist[v] 记录的最短距离。
>
> （3）重复（1）和（2），直到源点到所有其他点的最短路径都被确定。
>
> 最后输出按顶点编号顺序排列的源点0到各顶点的最短路径长度。