华东师范大学计算机科学技术系上机实践报告

课程名称：数据结构实践	年级：大二	上机实践成绩：
指导教师：杨静	姓名：汪子凡	
上机实践名称：数据结构实践	学号：10185102153	上机实践日期：2019.12.28
上机实践编号：无编号	组号：无分组	上机实践时间：

1.1076 Huffman树的最小外部加权值

1.1源代码

```
//Huffman树生成过程中会形成n-1个节点，最小外部路径就是这n-1个节点的值
#include<bits/stdc++.h>

using namespace std;

priority_queue<int, vector<int>, greater<int> > pq;  //定义一个小顶堆的优先队列
int n, x, ans;                 //ans用来记录最后答案
int main()
{
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>x;
        pq.push(x);
    }

    if(n == 1) ans=pq.top();
    for(int i = 0;i < n - 1;i++)
    {
        x=pq.top();  pq.pop();          //每次选取两个最小的节点
        x+=pq.top() ;pq.pop();
        ans += x;
        pq.push(x);                    //将新生成的节点push回优先队列
    }
    cout<<ans<<endl;
    return 0;
}
```

1.2 程序调试经验总结

分析题意，要求最小外部加权值就是要求建树过程中形成的n-1个新的节点的权值和，所以只需要一边建树一边记录答案即可。

建树过程中，每次都需要挑选两个最小的节点生成新的节点，可以每次都sort一遍然后取前两个值，但是这样的话复杂度大概就是 O(n*n)，改进后用了优先队列(小的元素在队首)，这样每次取两个复杂度可以控制在 O(nlogn)。

2. 1089 六度空间

2.1源代码

```
#include<bits/stdc++.h>

using namespace std;

typedef pair<int, int> P;
vector<int> v[10003];                                   //来记录边(两个人认识则构成一条边)
queue<P> q;                                             //bfs时用的队列
int mark[10003], cnt;

int main()
{
    int n, m, x, y;                                     //读入数据部分
    scanf("%d %d", &n, &m);
    for(int i = 1; i <= m; i++)
    {
        scanf("%d %d", &x, &y);                 //v[i]记录的是和节点i直接相连的点的标号
        v[x].push_back(y);
        v[y].push_back(x);
    }

    for(int i = 1; i <= n; i++)                      //对每一个人进行BFS，范围为6
    {
        memset(mark, 0, sizeof(mark));
        cnt = 0;                                    //用来记录认识的总人数
        q.push(make_pair(i, 0));
        mark[i] = 1;                                //若这个人已经认识，则mark记为1
        while(!q.empty())                           //从距离0开始扩展，相当于BFS过程
        {
            P tmp = q.front(); q.pop();
            mark[tmp.first] = 1;                    //first表示人的序号，second表示与第i个人的距离
            cnt++;
            if(tmp.second == 6) continue;       //若是距离为6，则不需要扩展下去
            for(int i : v[tmp.first])
            {
                if(!mark[i])                                //若距离小于6，并且还没有被标记
                {
                    q.push(make_pair(i, tmp.second + 1));   //扩展，距离+1
                    mark[i] = 1;
                }
            }
        }
        printf("%d: %.2f%%\n", i, 100.0 * cnt / n);
    }
}
```

2.2 程序调试经验总结

若用图来处理这题，每个结点表示一个人，若两个人之间直接认识，则有边相连。

建图的话，由于结点的最大值可能到10000，用邻接矩阵的话内存会过大，所以采用邻接表的方式来建立，与结点 i 认识的人记录在相应的 vector数组里。

对于每一个人，都要找出与其相应结点距离不超过6的结点的个数，所以对于每一个结点，可以采用广度优先的办法进行搜索，每次插入队列的是一个 pair类型，first 表示结点的下标，second表示与开始结点的距离；对于每一个搜索到的结点用mark标记，若是这个结点第一次被mark标记并且与开始结点的距离小于6，则要将该结点扩展下去，若是距离等于6，则不需要扩展；当队列为空，则搜索结束。

在输出方面，若采用 printf函数进行输出，则输出百分号应该在双引号里连续打两个%; 在数据存储方面，如果用set记录已经被标记过的人，程序执行的时间在 1.5 - 2秒之间，而如果用mark数组(每次开始前需要初始化成0)，程序执行时间不到 0.1秒, 所以采用了后者。