2021-2022《数据结构》大作业报告

学号：201830210

姓名：孙文博

院系：计算机科学与技术系

一、 小蓝鲸的奇妙冒险-第一季

1. 解题思路：

本题题意不难理解，即找到给定数组 A[n]中第 m 大的数并输出，其中包括重复的数字。分析数据规模 n<=1e7 可知时间复杂度大概是 O(n)级别，于是考虑采用桶排序的方式：对于一个元素个数为 size，数据范围是(0-max)的数组 A 我们可以创建一个大小为 max+1 的辅助数组 B[max+1]，其中每个元素 B[i]表示值为 i 的数的个数。然后从头开始遍历数组 A，读取每个 A[i]时对应的 B[A[i]]值加一；再从头遍历 B，此时每个大于 0 的元素的下标为排序后 A 数组的值，元素大小为这个值的个数，将排序后的结果赋值给 A，排序完成。使用桶排序中的时间复杂度为 O(n+m)，空间复杂度也是 O(n+m)。

2. 核心代码+注释：

```
//桶排序函数，给定数组 A 元素个数为 size，元素范围为 0~max
void BucketSort(int* A, int max, int size)
{
    int* B = new int[max + 1]; //B数组大小为max+1
    for (int i = 0; i < max + 1; i++)
    {
        B[i] = 0; //B数组初始化
    }
    int i, j, count = 0;
    for (i = 0; i < size; i++)
    {
        //遍历A数组，A数组每个元素作为下标，对应B数组中的元素+1
        B[A[i]] += 1;
    }
    for (i = 0; i <= max; i++)
    {
        //遍历B数组，此时下标即排序后A数组中的值，重新赋值到数组A中
        if (B[i] > 0)
          {
            for (j = 0; j < B[i]; j++)
              {
                A[count] = i;
```

```
                count++;
            }
        }
    }
```

3. OJ 运行结果截图

ID	题目	提交者	结果	用时	内存	语言	文件大小	提交时间	测评时间
#38926	#95. 小蓝鲸的奇妙冒险-第一季	201830210	100	1509ms	81156kb	C++	1.5kb	2021-12-08 19:47:34	2021-12-08 19:47:37

answer

```
#include<stdio.h>
using namespace std;
/*
//使用桶排序，对于一个元素个数为size，数据范围是(0-max)的数组A
//我们可以创建一个大小为max+1的辅助数组，每个元素初始化为0
//从头遍历A，读取A[i]时对应的B[A[i]]值加一(类似于hash)
//从头遍历B，每个大于0的元素的下标即排序后A数组的值，元素大小即这个值的个数
//重新赋值给A，排序完成，复杂度O(n+k)
*/
void BucketSort(int* A, int max, int size)
{
    int* B = new int[max + 1];
    for (int i = 0; i < max + 1; i++)
    {
        //B数组初始化
        B[i] = 0;
    }
```

二、 小蓝鲸的奇妙冒险-第二季

1. 解题思路：

本题由第一题改编而来，不同之处在于这次要输出 n 个值，其中第 i 个值表示给定数组的前 i 个数构成的子数组中第 k 大的值，而 k 也是个关于 i 的函数，表达式为 k=ceil(i/m)，其中 ceil 为取上整函数。分析数据规模可知 n<=1e6，算法的时间复杂度应当小于等于 O(nlogn)，故考虑到使用二叉搜索树，其中外层循环 n 次读入每个数，每次循环时先将新节点插入已有 BST 中，复杂度为 O(logn)，再找到第 k 大的节点并输出，复杂度也是 O(logn)，故总复杂度为 O(nlogn)满足条件。其中 BST 的插入和搜索采用了迭代的方式，核心代码见后。

2. 核心代码+注释：

```
//二叉搜索树节点
struct Node
{
```

```
    Node* left = nullptr;
    Node* right = nullptr;
    int val = -1;
    int left_num = 0;
    int right_num = 0;
     };

// 插入新节点p
Node* cur = root;
while (true)
{
    if (p->val > cur->val) //比当前节点值大，那么插入到节点的右边
    {
        cur->right_num++;
        if (cur->right == nullptr) //右子树为空，直接插入即可
        {
            cur->right = p;
            break;
        }
        else
            cur = cur->right;      //迭代，插入到右子树
    }
    else
    {
        cur->left_num++;
        if (cur->left == nullptr) //与插入到右边的过程类似
        {
            cur->left = p;
            break;
        }
        else
            cur = cur->left;
    }
}
//找到并输出第k大的节点
int larger = 0;
cur = root;
while (true)
{
    if (cur->right_num + 1 + larger == k) //判断当前节点是不是第k大
    {
        printf("%d ", cur->val);
        break;
    }
```

```
    else if (cur->right_num + 1 + larger > k) //当前数不够大，向右子树寻找
        cur = cur->right;
    else //当前数大了，向左子树寻找，larger记录了当前右边又多少比自己大的数
    {
        larger += cur->right_num + 1;
        cur = cur->left;
    }
}
```

3. OJ 运行结果截图：

ID	题目	提交者	结果	用时	内存	语言	文件大小	提交时间	测评时间
#47564	#96. 小蓝鲸的奇妙冒险-第二季	201830210	100	1279ms	50448kb	C++	2.5kb	2021-12-28 00:05:33	2021-12-28 00:05:35

answer

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

/*
* 使用二叉搜索树，外层为n的循环，内层先插入一个新节点，再输出当前第k大的树
*/

//
struct Node
{
    Node* left = nullptr;
    Node* right = nullptr;
    int val = -1;
    int left_num = 0;
    int right_num = 0;
};

int main()
```

三、 小蓝鲸的奇妙冒险-第三季

1. 解题思路：

本题考察的是图论中的最短路径问题，也是一道经典的板子题，我们可以用 Dijkstra 算法或者 Floyd 算法轻松解决。注意到顶点个数 n<=1e3，故可以用邻接矩阵存图，根据题意分析建好图后使用 Dij 算法即可得到答案。需要注意的是，本题中的顶点编号是 0~n-1，且图是无向图，故邻接矩阵需要满足对称性。PS：Dijkstra 算法是通过辅助数组 dis[i]和 flag[i]实现的，前者表示源点 s 到编号为 i 的顶点的当前最短路径，开始时仅标记源点，接着循环 n-1 次直到所有顶点都被标记，每次循环找到当前未被标记且距离 dis[i]最小的顶点，对其标记，之后以其为中心松弛其他的顶点到源点的距离，若满足 d[v] + a[v][j] < d[j]则更新该顶点的距离。Dij 算法的时间复杂度是

O(n2)，建图及初始化邻接矩阵的复杂度为 O(n2)，总时间复杂度也是 O(n2)。

2. 核心代码+注释：

```
void Dij(int n, int s) //Dijkstra算法计算单源最短通路问题
{
  //除源点以外所有点标记为0
  for (int i = 0; i < n; i++)
      flag[i] = 0;
  flag[s] = 1;

  for (int i = 0; i < n - 1; i++) //共执行n-1次，直到所有点都被标记
  {
      int min = INF;
      int v = 0;
      for (int j = 0; j < n; j++)
      {
          //找到未被标记的且d[j]值最小的顶点，记为v
          if (flag[j] == 0 && d[j] < min)
          {
              min = d[j];
              v = j;
          }
      }
      flag[v] = 1; //最后找到的这个点标记，之后遍历会跳过它
      for (int j = 0; j < n; j++)
      {
          //对于未被标记且满足三角不等式的顶点，更新数值
          if (flag[j] == 0 && d[v] + a[v][j] < d[j])
          {
              //d[j]表示s到j的距离，d[v]+a[v][j]表示s经v到j的距离
              d[j] = d[v] + a[v][j];
          }
      }
  }
  }
```

3. OJ 运行结果截图：

ID	题目	提交者	结果	用时	内存	语言	文件大小	提交时间	测评时间
#46878	#97. 小蓝鲸的奇妙冒险-第三季	201830210	100	61ms	7376kb	C++	2.4kb	2021-12-23 21:21:19	2021-12-23 21:21:22

answer

```
#include<iostream>
using namespace std;
const int N = 1005; //最大顶点数
const int INF = 1e9; //无穷大
int a[N][N], d[N], flag[N];

void init(int n) //初始化
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                a[i][j] = 0;
            else
                a[i][j] = INF;
        }
    }
```

四、 小蓝鲸的奇妙冒险-第四季

1. 解题思路：

本题由第三题改编而来，需要注意的是此时每条交通线路都有一个换乘值，故需要分别考虑，而不能像第三题中那样把所有线路拆开放在一个图中，此外编号为 1~n，且线路变为单向，都是需要注意的地方。于是考虑使用分层图的方法，建立一个虚层第 m+1 层，其中每个顶点与其他层对应顶点相连。使用 Dijkstra 算法，首先在每一层图上跑一遍 dij 初始化 dist 数组，再遍历不同路线之间的权值更新 dist 数组的值，得到结果。数据结构方面我们使用邻接表+矩阵的方式存图，使用结构体数组保存路线和 dist 数组的信息，算法的时间复杂度为 O(n2)，空间复杂度为 O(n3)。

2. 核心代码+注释：

```
//定义每条路线构成的结构体
struct Line
{
    int n; //经过的城市数量
    long long cost; //换乘或进入这条路线的花费
};

//定义有向边结构体
struct Edge
```

```
{
    int line_id = -1; //表示是哪一条路线上的边
    long long cost = -1; //表示这条边的花费（权值）
    Edge* next = nullptr;
};

//定义距离数组结构体，注意每条边初始值为INF
struct Distance
{
    int line_id = -1;
    long long cost = INF;
    Distance* next = nullptr;
     };
// 初始化dist数组
    bool flag[maxn];
    for (int i = 1; i <= n; i++)
    {
        flag[i] = false;
        if (edges[s][i] != nullptr)
        {
            Edge* e = edges[s][i];
            while (e != nullptr)
            {
                int t = e->cost + lines[e->line_id].cost;
                if (dist[i] == nullptr)
                {
                    Distance* d = new Distance;
                    d->cost = t;
                    d->line_id = e->line_id;
                    dist[i] = d;
                }
                else
                {
                    bool find = false;
                    Distance* tail = dist[i];
                    while (tail->next != nullptr)
                    {
                        if (tail->line_id == e->line_id)
                        {
                            if (t < tail->cost)
                            {
                                tail->cost = t;
                                find = true;
                                break;
```

```
                    }
                }
                tail = tail->next;
            }
            if (!find)
            {
                Distance* d = new Distance;
                d->cost = t;
                d->line_id = e->line_id;
                tail->next = d;
            }
        }
        e = e->next;
    }
  }
}
```

3. OJ 运行结果截图：

ID	题目	提交者	结果	用时	内存	语言	文件大小	提交时间	测评时间
#47637	#98. 小蓝鲸的奇妙冒险-第四季	201830210	100	240ms	9948kb	C++	8.2kb	2021-12-28 11:32:53	2021-12-28 11:32:55

answer

```
#include<iostream>
using namespace std;
const int maxn = 501;
const int INF = 1e9;

//定义每条路线构成的结构体
struct Line
{
    int n; //经过的城市数量
    long long cost; //换乘或进入这条路线的花费
};

//定义有向边结构体
struct Edge
{
    int line_id = -1; //表示是哪一条路线上的边
    long long cost = -1; //表示这条边的花费（权值）
    Edge* next = nullptr;
};

//定义距离数组结构体，注意每条边初始值为INF
```

五、 总结与反思

本次数据结构大作业难度相较之前的实验有所提升，每道题都有卡时间的一个或者两个样例（比如第二题原先的算法只能拿 80 被卡了很久😭），但是前四题全 AC 的时候还是非常开心的🎉。前两题题意不难，可使用的方法很多，但是真正拿到满分还是需要选择较优的算法甚至需要进一步优化；后两题（其实是后三

题）是经典图论题，从分析到建图到实现算法还是需要很多时间的，当然整个做题的过程也是对本学期学习的数据结构知识的一次实践和巩固（由于时间和精力原因第五题暂时没有好的办法，也许等考试周过后会重新再试试吧😥）。最后要感谢老师以及助教们一路以来的支持和帮助，让我们收获了一个充实而顺利的课程😻，也希望自己能再接再厉，下学期继续学习算法！

2021.12.28

End.