2021-2022《数据结构》大作业报告

学号：　　201220120

姓名：　　单博

院系：　　计算机科学与技术

Data Structure Project

单博 201220120

一、允许使用的头文件

```cpp
#include <iostream>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <string>
#include <cctype>
#include <cerrno>
#include <iterator>
#include <bitset>
```

二、通用变量与函数

```cpp
using ll = long long;
inline ll get_max(ll a, ll b);
inline ll get_min(ll a, ll b);
inline ll input();
constexpr ll MAXN = 1e7 + 50;
```

1. `using ll = long long`: 使用`ll`作为带符号长整型的别名;
2. `get_max`: 获取两个带符号长整型数的较小值;
3. `get_min`: 获取两个带符号长整型数的较大值;
4. `input`: 利用`getchar`函数进行读入优化，降低输入过程的时间复杂度;
5. `MAXN`: 定义题目中将要使用的数组的最大规模;

三、题目解析

用户名	总分	A	B	C	D	E
201220120	490	100	100	100	100	90
用时	82:45:26	0:43:38	15:34:16	1:22:37	24:08:53	40:56:02

1. Problem A

> 题目大意：寻找n个数中第m大的数。

解题思路

利用桶对输入的数字进行记录，每个桶中保存数字出现的次数，然后从最大的数字开始向前寻找，遇见第m大的数字后输出答案并终止循环。

复杂度分析：时间复杂度$O(n)$，空间复杂度$O(n)$.

核心代码

```
// buckets[k] counts the times number k appears.
for(int k = max_num; k > 0; k--) {
    m -= buckets[k];
    if(m <= 0) { // meet the mth number
        // cout << k << endl;
        printf("%d\n", k);
        break;
    }
}
```

运行结果

详细

Test #1:	score: 10	Accepted	time: 3ms	memory: 3480kb
Test #2:	score: 10	Accepted	time: 6ms	memory: 3452kb
Test #3:	score: 10	Accepted	time: 3ms	memory: 3628kb
Test #4:	score: 10	Accepted	time: 2ms	memory: 3376kb
Test #5:	score: 10	Accepted	time: 2ms	memory: 3552kb
Test #6:	score: 10	Accepted	time: 6ms	memory: 3452kb
Test #7:	score: 10	Accepted	time: 4ms	memory: 3524kb
Test #8:	score: 10	Accepted	time: 3ms	memory: 3552kb
Test #9:	score: 10	Accepted	time: 6ms	memory: 3520kb
Test #10:	score: 10	Accepted	time: 1200ms	memory: 42564kb

2. Problem B

> 题目大意：输入n个数字，每次输入时计算当前输入数字中第$\lceil \frac{i}{m} \rceil$大的数并输出，$i$为当前输入数字个数。

解题思路

可以发现每次`i`增加1的时候，$k = \lceil \frac{i}{m} \rceil$最多增加1，而数字总数也增加1，故可以在上一次答案的基础上进行暴力求当前答案；

使用变量`now_num`和`now_ind`表示当前数字的位置，`now_num`为当前数字，`now_ind`为当前数字在相同数字中排名；

数字排列表示如下：

$< now\_num$	$= now\_num$	$= now\_num$	$= now\_num$	$> now\_num$
$\dots$	$> now\_ind$	$= now\_ind$	$< now\_ind$	$\dots$

讨论新插入一个数字时，k的变化以及上一次选中数字排名的变化：

	$\leq now\_num$	$> now\_num$
$k' = k$	(1)上一次选中数字排名不变	(3)上一次选中数字排名加1
$k' = k + 1$	(2)上一次选中数字排名不变	(4)上一次选中数字排名加1

通过观察表格可以发现只有第(2)和第(3)种情况需要对答案进行调整，具体调整过程见核心代码；

复杂度分析：期望时间复杂度$O(n)$，空间复杂度$O(n)$.

核心代码

```
// i from 0 to (n - 1)
if(i == 0) {
    now_num = num, now_ind = 1;
}
else {
    if(i % m == 0) { // last selected num is the (k - 1)th
        if(num <= now_num) { // move forward
            now_ind++;
            if(now_ind > buckets[now_num]) {
                while(buckets[--now_num] == 0);
                now_ind = 1;
            }
        }
    }
    else { // last selected num is the kth
        if(num > now_num) { // move backward
            now_ind--;
            if(now_ind == 0) {
                while(buckets[++now_num] == 0);
                now_ind = buckets[now_num];
            }
        }
    }
}
printf("%d ", now_num);
```

运行结果

详细

Test #1:	score: 10	Accepted	time: 4ms	memory: 3476kb
Test #2:	score: 10	Accepted	time: 4ms	memory: 3584kb
Test #3:	score: 10	Accepted	time: 3ms	memory: 3376kb
Test #4:	score: 10	Accepted	time: 3ms	memory: 3928kb
Test #5:	score: 10	Accepted	time: 7ms	memory: 5220kb
Test #6:	score: 10	Accepted	time: 9ms	memory: 6056kb
Test #7:	score: 10	Accepted	time: 1ms	memory: 3448kb
Test #8:	score: 10	Accepted	time: 39ms	memory: 7316kb
Test #9:	score: 10	Accepted	time: 205ms	memory: 7320kb
Test #10:	score: 10	Accepted	time: 1ms	memory: 3472kb

3. Problem C

> 题目大意：求n个点的无向图中点对(u, v)的最短路长度。

解题思路

利用邻接矩阵存图，处理完输入后利用Floyd算法求出最短路。

复杂度分析：时间复杂度$O(n^3)$，空间复杂度$O(n^2)$.

核心代码

```
// dis[i][j] means the shortest distance between i and j.
// Floyd
for(int k = 0; k < n; k++) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            dis[i][j] = get_min(dis[i][j], dis[i][k] + dis[k][j]);
        }
    }
}
```

运行结果

详细

Test #1:	score: 10	Accepted	time: 2ms	memory: 7888kb
Test #2:	score: 10	Accepted	time: 5ms	memory: 7836kb
Test #3:	score: 10	Accepted	time: 9ms	memory: 7748kb
Test #4:	score: 10	Accepted	time: 10ms	memory: 7724kb
Test #5:	score: 10	Accepted	time: 13ms	memory: 7896kb
Test #6:	score: 10	Accepted	time: 12ms	memory: 7780kb
Test #7:	score: 10	Accepted	time: 36ms	memory: 7776kb
Test #8:	score: 10	Accepted	time: 126ms	memory: 7776kb
Test #9:	score: 10	Accepted	time: 434ms	memory: 7720kb
Test #10:	score: 10	Accepted	time: 888ms	memory: 7640kb

4. Problem D

> 题目大意：一个n个点的有向图，存在m条线路，每条线路包含一些有向边且可能有环，切换线路需要代价，求点u到点v的最小代价。

解题思路

在原题条件的基础上建立分层图然后利用Dijkstra算法求最短路。

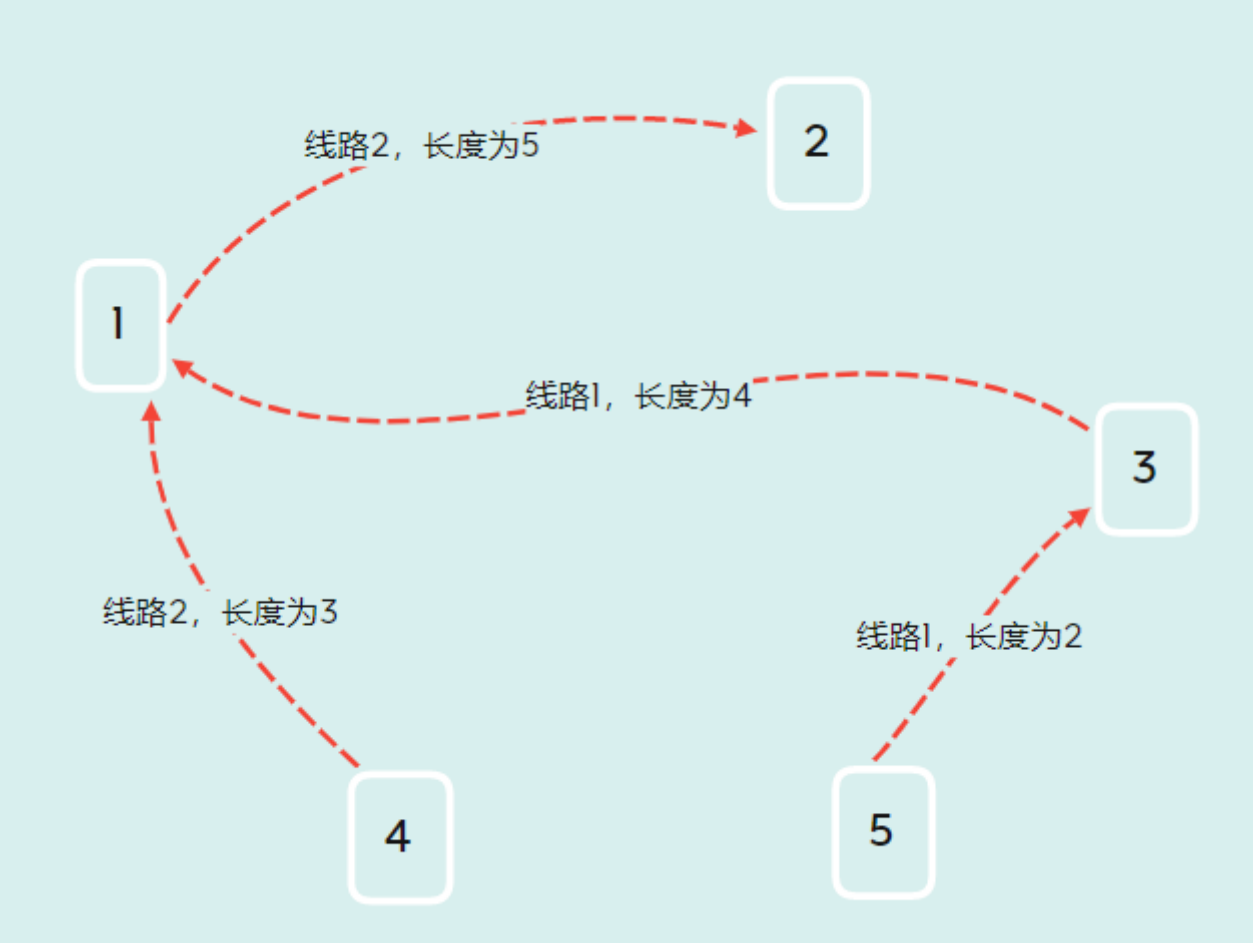
分层图的建立形式大致如下：

	1	2	...	N-1	N
第0层	1	2	...	n-1	n
第1层	n+1	n+2	...	2*n-1	2*n
...	...	...	...	...	...
第m层	m*n+1	m*n+2	...	(m+1)*n-1	(m+1)*n

第0层为汇点层，通过与其他层的点连边建立层与层之间的联系并将切换线路的代价转换为边权，使问题转化为简单的最短路问题，第0层的点之间不直接相连；

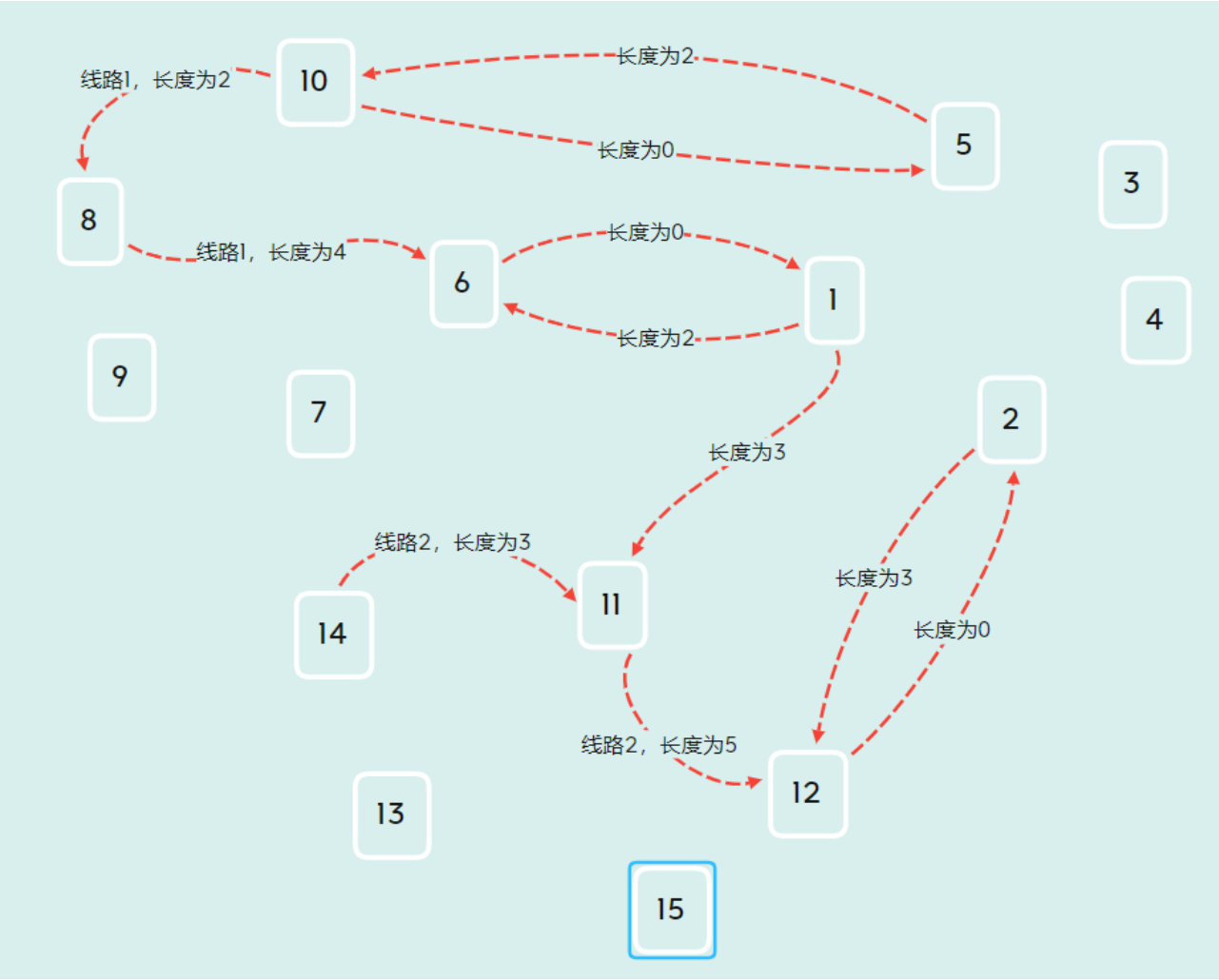
第1~m层每一层表示一条线路，层与层之间不直接相连，每一层的所有点向第0层对应汇点印一条长度为0的边，同时汇点向每一层的对应点引一条长度为“线路切换代价”的边；

下面用一个简单用例来具体解释建立图的过程：



其中切换到线路1的代价为2，切换到线路2的代价为3，求点5到点2的最短距离。

对于此图我们建立如下所示的分层图（其中部分第0层与其他层之间的边未给出）：



建立好分层图之后就可以直接利用Dijkstra求以汇点5到汇点2的最短距离了。

简要证明汇点5到汇点2的最短距离的确是原图中的最短距离：

首先题目要求初始进入线路时也需要代价，于是从汇点出发到任意非0层对应节点时路径长度都会增加对应切换线路的代价，故从汇点出发可以保证不会对结果造成损失；

由于非0层每一点前往对应汇点的代价是0，故源点到第0层目标点的距离等于源点与非0层中每一层目标点之间距离的最小值，即为答案。

另外，由于建立分层图时并没有引入负权边，所以显然图中不会出现负权环，在同一条线路上上下下（即重复走同一条环路）的情况并不会对最后答案造成影响。

复杂度分析：时间复杂度$O(nm \cdot log(nm))$，空间复杂度$O(nm)$.

核心代码

```
ll Dijkstra(int src, int dst, int num_of_pos, int num_of_layer)
{
    static ll dis[MAXN * MAXN], n = num_of_pos / num_of_layer;
    static int pre_edge[MAXN * MAXN], pre_layer[MAXN * MAXN];
    static bool vis[MAXN * MAXN];
    memset(dis, 0x3f, sizeof(dis));
    memset(vis, false, sizeof(vis));
    memset(pre_edge, -1, sizeof(pre_edge));
    memset(pre_layer, -1, sizeof(pre_layer));
    dis[src] = 0;

    static Heap <pair <ll, int>> que(MAXN * MAXN);
    for(int i = 1; i <= num_of_pos; i++) {
        que.insert({ dis[i], i });
    }
    while(!que.empty()) {
        pair <ll, int> pos = que.top();
        que.pop();
        if(vis[pos.second]) {
            continue;
        }
        vis[pos.second] = true;
        for(int i = head[pos.second]; i; i = edges[i].nxt) {
            int to = edges[i].to;
            if(vis[to]) {
                continue;
            }
            if((pos.second - 1) / n == (to - 1) / n &&
(pos.second - 1) / n == pre_layer[pos.second]
                && pre_edge[pos.second] + 1 != edge_index[i]) {
// if there is a circle in this layer, you have to travel the
graph orderly.
                continue;
            }
            if(dis[to] > dis[pos.second] + edges[i].val) {
                dis[to] = dis[pos.second] + edges[i].val;
                que.insert({ dis[to], to });

                pre_edge[to] = edge_index[i];
                pre_layer[to] = (pos.second - 1) / n;
            }
```

```
38          }
39      }
40      return dis[dst];
41  }
```

运行结果

详细

Test #1:	score: 10	Accepted	time: 4ms	memory: 38512kb
Test #2:	score: 10	Accepted	time: 12ms	memory: 38608kb
Test #3:	score: 10	Accepted	time: 7ms	memory: 38592kb
Test #4:	score: 10	Accepted	time: 7ms	memory: 38664kb
Test #5:	score: 10	Accepted	time: 11ms	memory: 38740kb
Test #6:	score: 10	Accepted	time: 17ms	memory: 38696kb
Test #7:	score: 10	Accepted	time: 11ms	memory: 39136kb
Test #8:	score: 10	Accepted	time: 18ms	memory: 39868kb
Test #9:	score: 10	Accepted	time: 45ms	memory: 40924kb
Test #10:	score: 10	Accepted	time: 146ms	memory: 46976kb

5. Problem E

> 题目大意：一个n个点的有向图，存在m条线路，每条线路包含一些有向边且可能有环，切换线路需要时间，每走一条边需要消耗一定的战力值，路径代价为消耗战力值与时间的乘积，求点u到点v的最小代价。

解题思路

处理输入建图后暴力搜索+剪枝优化。

剪枝方式：

1. 如果搜索到当前点时消耗战力与时间的乘积已然大于已经搜索得到的“答案”，那么继续搜索不会得到更优的结果，故停止搜索；
2. 记录每个节点的最小时间和最小战力消耗，如果搜索到当前节点时消耗战力与时间的乘积大于当前节点最小消耗战力与最小时间的乘积，那么停止搜索；

复杂度分析：时间复杂度$O(?)$，空间复杂度$O(nm)$.

核心代码

```
void dfs(const int src, const int dst, ll dis, ll cost, int pre_edge_ind) {
    if(src == dst) {
        ans = get_min(ans, dis * cost);
        return;
    }
    if(dis * cost >= ans) {
        return;
    }
    // if(dis * cost > min_tmp_ans[src]) {
    if(dis > min_tmp_dis[src] && cost > min_tmp_cost[src]) {
        return;
    }
    // min_tmp_ans[src] = get_min(min_tmp_ans[src], dis * cost);
    min_tmp_dis[src] = get_min(min_tmp_dis[src], dis);
    min_tmp_cost[src] = get_min(min_tmp_cost[src], cost);

    for(int i = head[src]; i; i = edges[i].nxt) {
        int to = edges[i].to, to_line = edges[i].line;
        dfs(
            to,
            dst,
            dis + edges[i].dis + ((edges[pre_edge_ind].line != to_line || /* if equal */ pre_edge_ind != pre_edge[i]) ? switch_cost[to_line] : 0),
            cost + edges[i].cost,
            i
        ); // if there is a circle in this layer, you have to travel the graph orderly.
    }
}
```

运行结果

详细

Test #1:	score: 5	Accepted	time: 2ms	memory: 3484kb
Test #2:	score: 5	Accepted	time: 2ms	memory: 3512kb
Test #3:	score: 5	Accepted	time: 5ms	memory: 3396kb
Test #4:	score: 5	Accepted	time: 4ms	memory: 3508kb
Test #5:	score: 5	Accepted	time: 5ms	memory: 3468kb
Test #6:	score: 5	Accepted	time: 2ms	memory: 3432kb
Test #7:	score: 5	Accepted	time: 5ms	memory: 3512kb
Test #8:	score: 5	Accepted	time: 5ms	memory: 3644kb
Test #9:	score: 5	Accepted	time: 9ms	memory: 3560kb
Test #10:	score: 5	Accepted	time: 15ms	memory: 3500kb
Test #11:	score: 5	Accepted	time: 17ms	memory: 3892kb
Test #12:	score: 5	Accepted	time: 57ms	memory: 4164kb
Test #13:	score: 5	Accepted	time: 126ms	memory: 4336kb
Test #14:	score: 0	Wrong Answer	time: 128ms	memory: 4744kb
Test #15:	score: 5	Accepted	time: 107ms	memory: 4868kb
Test #16:	score: 5	Accepted	time: 401ms	memory: 5148kb
Test #17:	score: 5	Accepted	time: 367ms	memory: 5264kb
Test #18:	score: 5	Accepted	time: 593ms	memory: 5696kb
Test #19:	score: 5	Accepted	time: 787ms	memory: 6220kb
Test #20:	score: 0	Time Limit Exceeded		