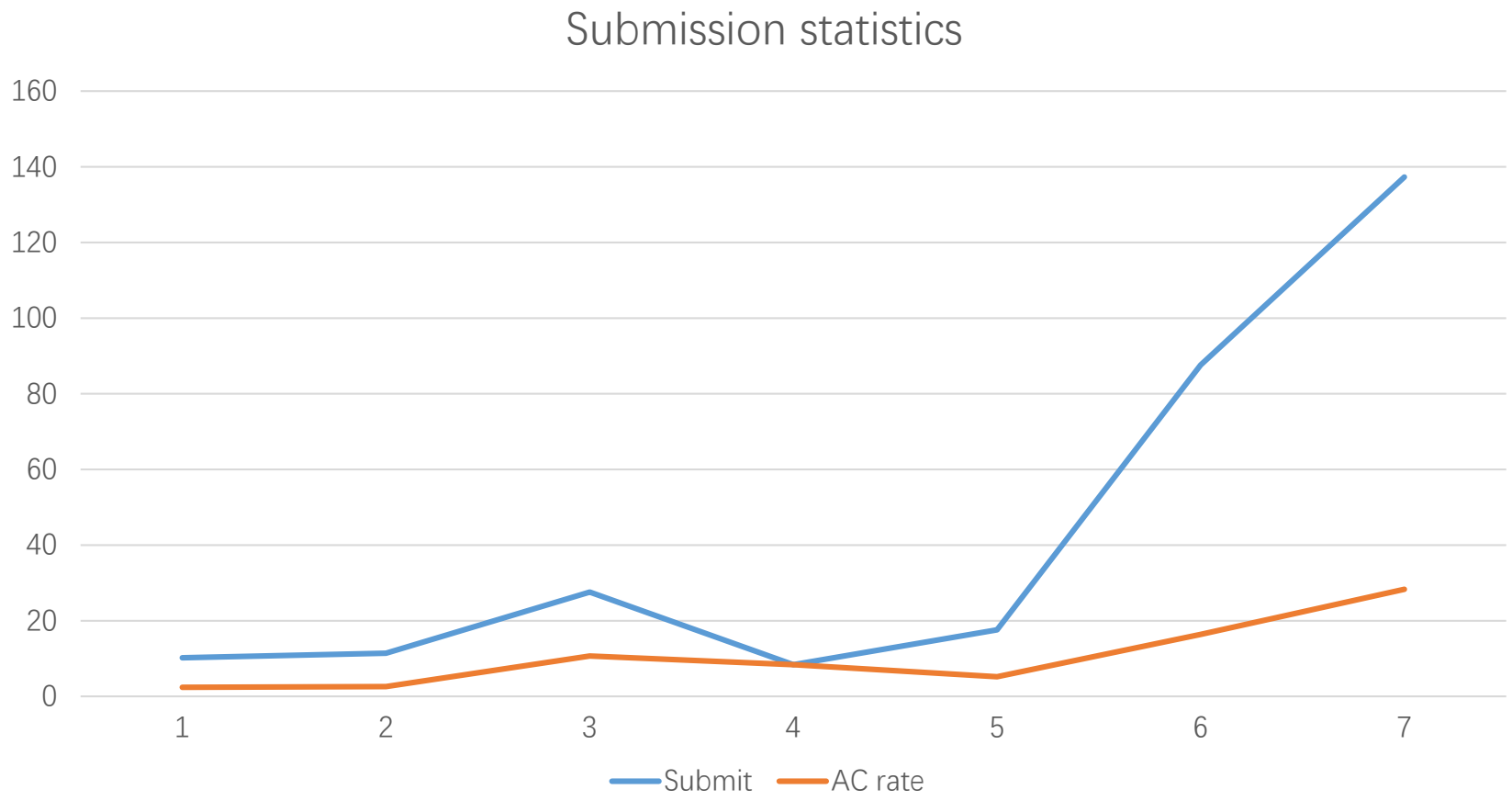
DS OJ Review

By Huangjie ZHeng

2017/06/05

Content

Overview of this semester



Content

1. C++ Implementation
   a. Big Integer Class (Class)
   b. Colorful Lecture Note (I/O)
2. Stack
   a. Rail station
3. Binary Tree
   a. Max Queue
   b. Post-order enumeration
   c. K-largest Number
   d. Hoffman Coding Tree
4. Sorting
   a. Valid Anagram
   b. Longest Word
5. Hashing
   Geohash encoding & decoding
6. Graph
   Topology Sorting
7. Dynamic
   a. Robotruck
   b. 0-1 Backpack

Encode and Decode with GeoHash

Stanislas

2017.6.5

Principle

bit	min	mid	max	val	err
1	-90.000	0.000	90.000	45.000	45.000
0	0.000	45.000	90.000	22.500	22.500
1	0.000	22.500	45.000	33.750	11.250
1	22.500	33.750	45.000	39.375	5.625
1	33.750	39.375	45.000	42.188	2.813
1	39.375	42.188	45.000	43.594	1.406
0	42.188	43.594	45.000	42.891	0.703
0	42.188	42.891	43.594	42.539	0.352
1	42.188	42.539	42.891	42.715	0.176
0	42.539	42.715	42.891	42.627	0.088
0	42.539	42.627	42.715	42.583	0.044
1	42.539	42.583	42.627	42.605	0.022

1. Encode coordination to binary form
2. Combination of code, longtitude first

lat: 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1

11100111010010001101101101110010011000101111110001

lon: 1 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 1 1 0 0

3. Encode binary form with hashmap

```
11100 11101 00100 01101 10110 11100 10011 00010 11111 10001
   28    29     4    13    22    28    19     2    31    17
    w     x     4     e     q     w     m     2     z     j
```

4. Reverse steps above for decoding

Programing

1. Using recusive function.

```
void encoder(double coor, double  lb, double  ub, vector <int> &seq)
{
    if (counter >= 25)
        return;
    if (coor > (lb + ub) / 2)
    {
        seq.push_back(1);
        encoder(coor, (lb + ub) / 2, ub, seq);
    }else
    {
        seq.push_back(0);
        encoder(coor, lb, (lb + ub) / 2, seq);
    }
    counter++;
}
```

2. Methode to keep precision during caculation

```
double precicion = 0.0000001;
lb = (int)(lb / precicion + 0.5) * precicion;
ub = (int)(ub / precicion + 0.5) * precicion;
```

Topology Sorting

胡敏浩

2017.6.5

Problem A: Topology Sorting

Time Limit: 1 Sec Memory Limit: 128 MB

Submit: 71 Solved: 43

[Submit][Status][Web Board]

Description

An ascending sorted sequence of distinct values is one in which some form of a less-than operator is used to order the elements from smallest to largest. For example, the sorted sequence A, B, C, D implies that A < B, B < C and C < D. in this problem, we will give you a set of relations of the form A < B and ask you to determine whether a sorted order has been specified or not.

Input

Input consists of multiple problem instances. Each instance starts with a line containing two positive integers n and m. the first value indicated the number of objects to sort, where 2 <= n <= 26. The objects to be sorted will be the first n characters of the uppercase alphabet. The second value m indicates the number of relations of the form A < B which will be given in this problem instance. Next will be m lines, each containing one such relation consisting of three characters: an uppercase letter, the character "<" and a second uppercase letter. No letter will be outside the range of the first n letters of the alphabet. Values of n = m = 0 indicate end of input.

Output

For each problem instance, output consists of one line. This line should be one of the following three:

Sorted sequence determined after xxx relations: yyy...y.

Sorted sequence cannot be determined.

Inconsistency found after xxx relations.

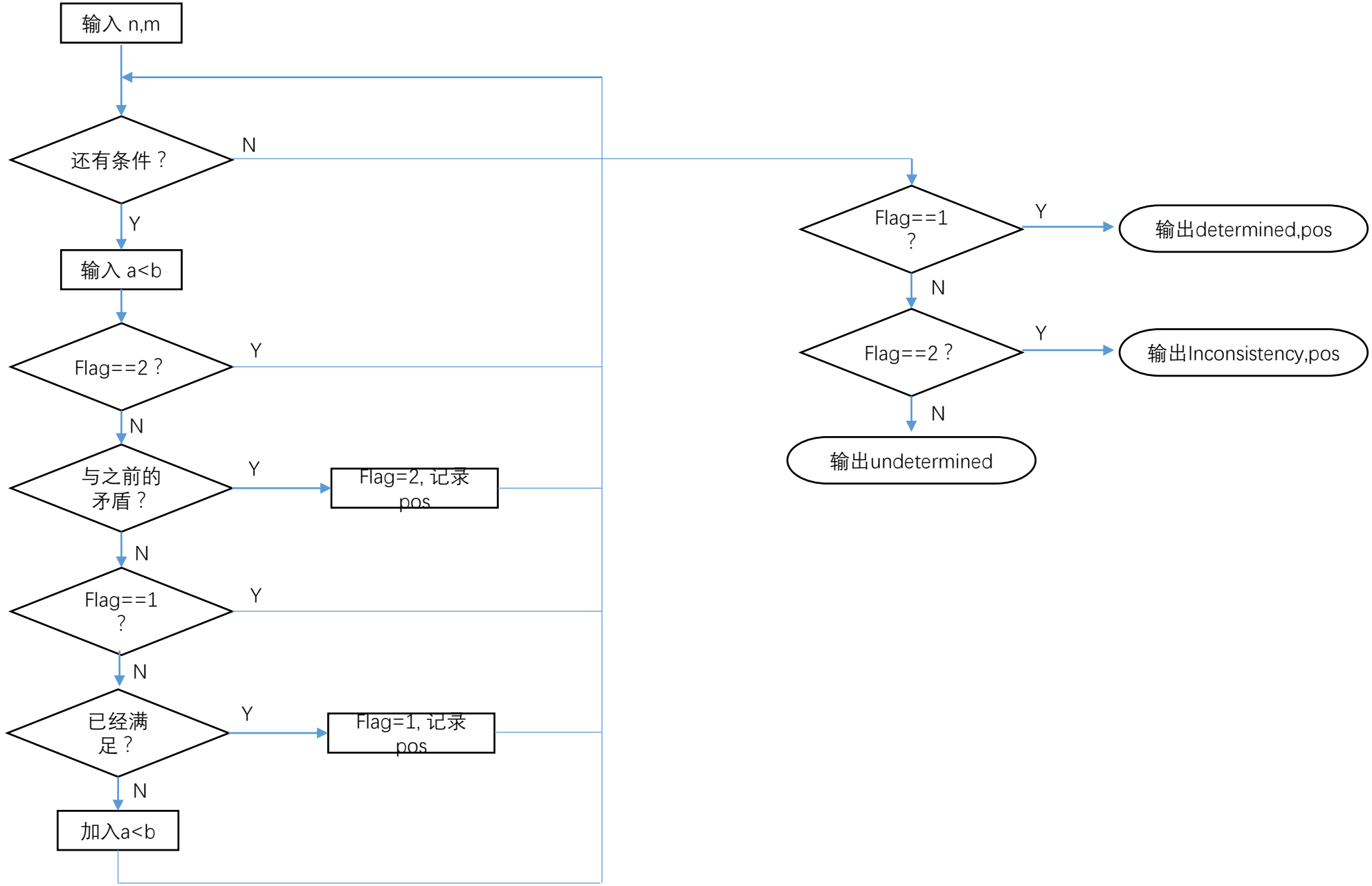
where xxx is the number of relations processed at the time either a sorted sequence is determined or an inconsistency is found, whichever comes first, and yyy...y is the sorted, ascending sequence.

一、根据题意初步流程

二、建立抽象模型——拓扑排序

三、DFS函数

四、数据结构 class

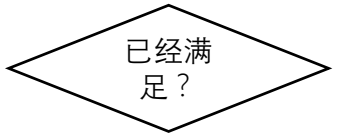

输入 n,m
还有条件？
N
Y
输入 a<b
Flag==2？
Y
N
与之前的矛盾？
Y
Flag=2, 记录pos
N
Flag==1？
Y
N
已经满足？
Y
Flag=1, 记录pos
N
加入a<b
Flag==1？
Y
输出determined,pos
N
Flag==2？
Y
输出Inconsistency,pos
N
输出undetermined

一、根据题意初步流程

二、建立抽象模型——拓扑排序

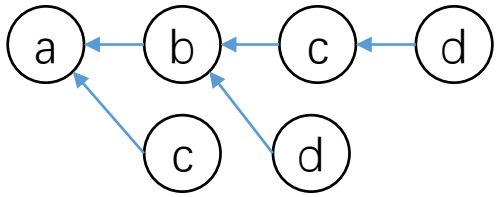
三、DFS函数

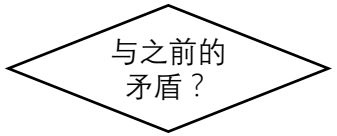
四、数据结构 class



n=4, m=5

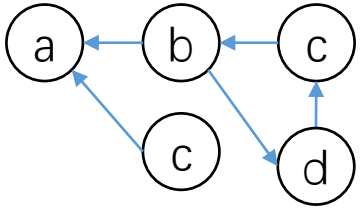
a< a<c b< b< c<
b c d d
a<-b a<-c b<-c b<-d c<-d

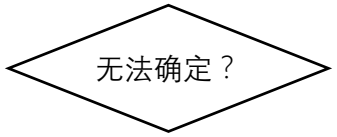




n=4, m=5

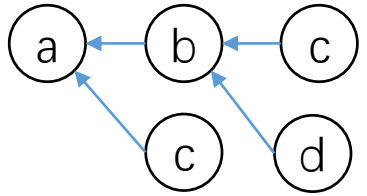
a< a<c b< d< c<
b c b d
a<-b a<-c b<-c d<-b c<-d





n=4, m=5

a< a<c b< b< a<
b c d d
a<-b a<-c b<-c d<-b a<-d



一、根据题意初步流程

二、建立抽象模型——拓扑排序

三、DFS函数

四、数据结构 class

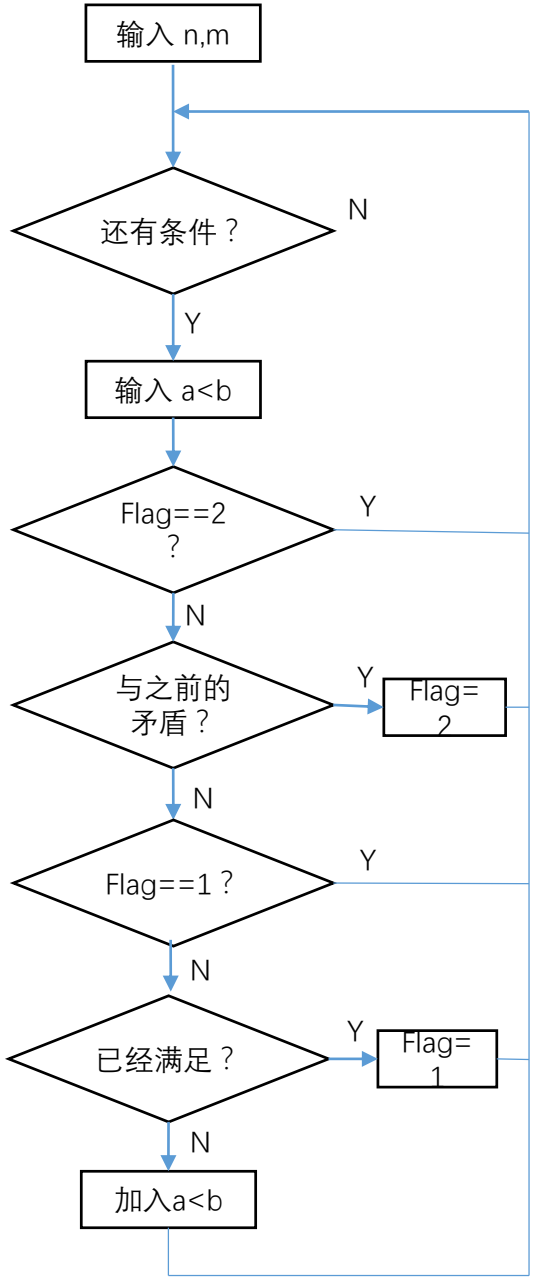
```
DFS (状态参数)
{
    if (状态==目标状态)
    {
        do something;
        return;
    }
    else
    {
        for (每个新状态)
        {
            if (新状态合法)
            {
                DFS (新状态);
            }
        }
        return;
    }
}
```

```
int  main ()
{
    do something;
    DFS (初始状态);
    do something;
    return 0;
}
```

```
DFS (状态参数)
{
    if (状态==目标状态)
    {
        do something;
        return;
    }
    else
    {
        for (每个新状态)
        {
            if (新状态合法)
            {
                DFS (新状态);
            }
        }
        return;
    }
}
```

```
int  main ()
{
    do something;
    DFS (初始状态);
    do something;
    return 0;
}
```

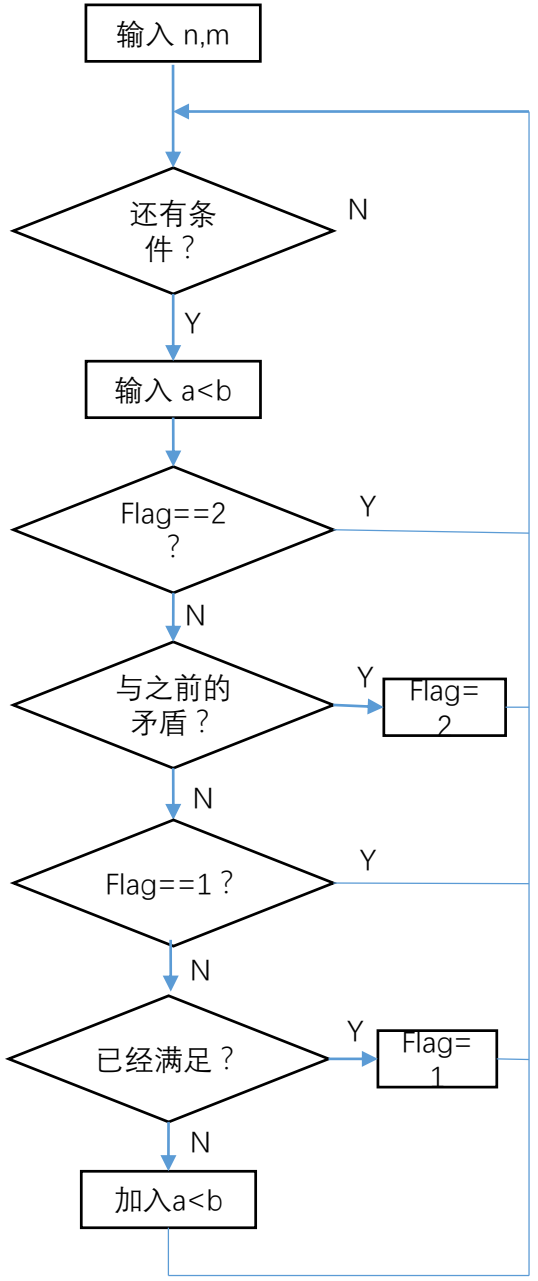
```
DFS (int d, Record record)
{
    if (flag==2)
    {
        return;
    }
    else
    {
        for (每个新状态)
        {
            if (新状态合法)
            {
                DFS (新状态);
            }
        }
        return;
    }
}
```



```
DFS (状态参数)
{
    if (状态==目标状态)
    {
        do something;
        return;
    }
    else
    {
        for (每个新状态)
        {
            if (新状态合法)
            {
                DFS (新状态);
            }
        }
        return;
    }
}
```

```
int  main ()
{
    do something;
    DFS (初始状态);
    do something;
    return 0;
}
```

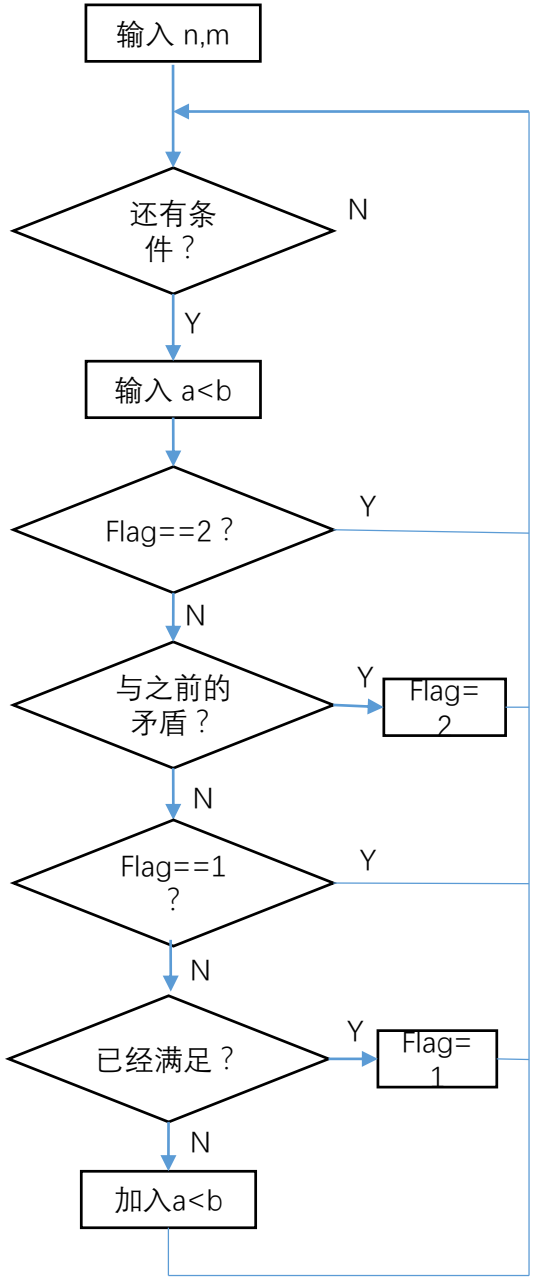
```
DFS (int d, Record record)
{
    if (flag==2)
    {
        return;
    }
    else
    {
        for ( int i = 0; i < SIZE; i ++ )
        {
            if ( map[d][i] == 1 )
            {
                DFS (i,record);
            }
        }
        return;
    }
}
```



```
DFS (状态参数)
{
    if (状态==目标状态)
    {
        do something;
        return;
    }
    else
    {
        for (每个新状态)
        {
            if (新状态合法)
            {
                DFS (新状态);
            }
        }
        return;
    }
}
```

```
int  main ()
{
    do something;
    DFS (初始状态);
    do something;
    return 0;
}
```

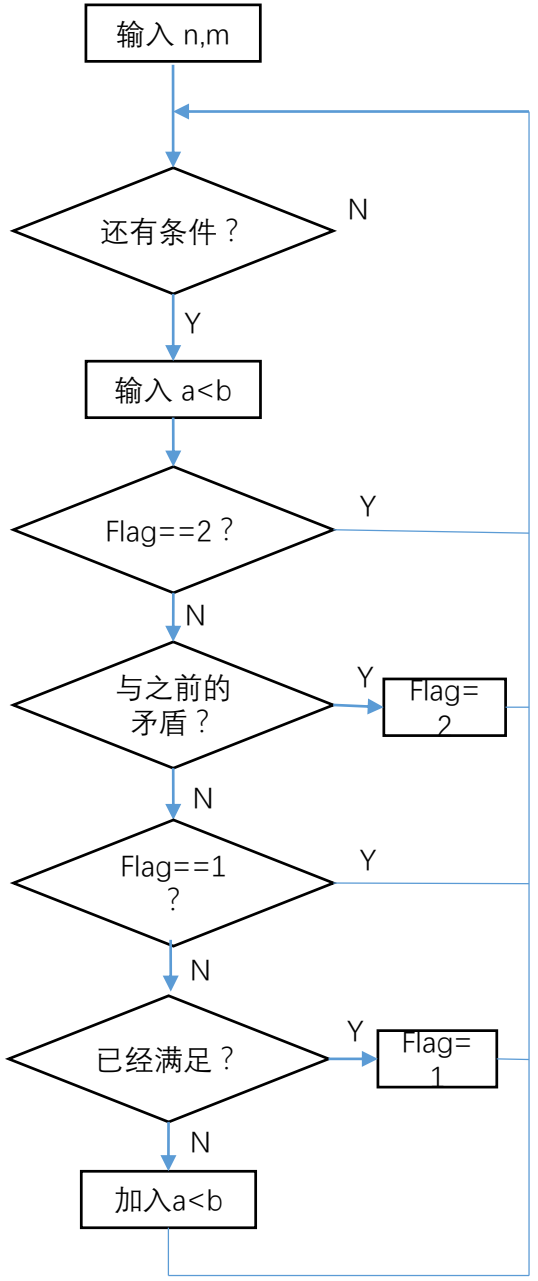
```
DFS (int d, Record record)
{
    if (flag==2)
    {
        return;
    }
    else
    {
        for ( int i = 0; i < SIZE; i ++ )
        {
            if ( map[d][i] == 1 )
            {
                if ( record.have(i) == 1 )
flag=2;
                record.add(i);
                if ( record.len == SIZE )
flag=1;
                DFS (i,record);
                record.del(i);
            }
        }
        return;
    }
}
```



```
DFS (状态参数)
{
    if (状态==目标状态)
    {
        do something;
        return;
    }
    else
    {
        for (每个新状态)
        {
            if (新状态合法)
            {
                DFS (新状态);
            }
        }
        return;
    }
}
```

```
int  main ()
{
    do something;
    DFS (初始状态);
    do something;
    return 0;
}
```

```
DFS (int d, Record record)
{
    if (flag==2)
    {
        return;
    }
    else
    {
        for ( int i = 0; i < SIZE; i ++ )
        {
            if ( map[d][i] == 1 )
            {
                if ( record.have(i) == 1 )
flag=2;
                record.add(i);
                if ( record.len == SIZE )
                {
                    flag=1;
                    ans=record.copy();
                }
                DFS (i,record);
                record.del(i);
            }
        }
        return;
    }
}
```



一、根据题意初步流程

二、建立抽象模型——拓扑排序

三、DFS函数

四、数据结构 class

```
01  class Record
02  {
03  public:
04      int index[26];
05      int len;
06
07      record  (int k)      { len=1; index[0]=k;  for (int i=0;i<26;i++)  index[i]=-1;}
08      ~record() {}
09
10      void add(int k)     { index[len++]=k; }
11
12      void del(int k)     { index[--len]=-1; }
13
14      bool have(int k)
15      {
16          for (int i=0;i<len;i++)
17              if (index[i]==k)   return true;
18          return false;
19      }
20  };
```

Robotruck

潘宇航

2017.6.5

robotruck

- 设第i个点到原点的距离为norm(i)，norm(i)=x(i)+y(i)
- 设从原点开始依次经过所有点后，到第i个点的总路程为sum(i), 则

$$sum(i)=\sum_{k=1}^{i}\left(|x(i)-x(i-1)|+|y(i)-y(i-1)|\right)$$

- 设第i个点的最优路径长度为opt(i)
- 对于第i个点，若它上一次回原点的点为j, j<=i-1，j需要满足 $\sum_{k=j+1}^{i} w(k)\le W$
- 则可以得到 $L(i,j)=opt(j)+norm(j+1)+sum(i)-sum(j+1)+norm(i)$
- 整理后 $L(i,j)=opt(j)+norm(j+1)-sum(j+1)+norm(i)+sum(i)$

robotruck

- 观察这个式子, $L(i,j)=opt(j)+norm(j+1)-sum(j+1)+norm(i)+sum(i)$
- 可以发现对于某个确定的i, $norm(i)+sum(i)$ 是一个定值
- 而 $opt(j)+norm(j+1)-sum(j+1)$ 是关于j的一个函数

- 若 $\sum_{k=0}^{i} w(k) \le W$ 则 $opt(i)=sum(i)+norm(i)$
- 若 $\sum_{k=0}^{i} w(k) > W$ 则 $opt(i)=\min\ L(i,j),\ \ j\in[0,i-1], \sum_{k=j+1}^{i} w(k) \le W$

Exam

- 15/06/2017
- EE Building 4-202, 4-204
- 3 Exercises
- How to review:
  - Oj is open until 15/06
  - Graph, Dynamic programming, BST, Heap
- Other OJ platform:
  - Leetcode: https://leetcode.com/
  - ACM: https://icpcarchive.ecs.baylor.edu/