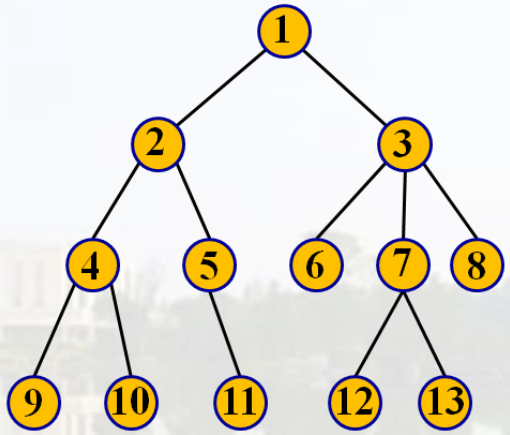
八数码问题

编号为1~8的8个正方形滑块被摆成3行3列(有一个格子留空). 每次可以把与空格相邻的滑块(有公共边才算相邻)移到空格中, 而它原来的位置就成为了新的空格. 给定初始局面和目标局面(用0表示空格), 你的任务是计算出最少的移动步数. 如果无法到达目标局面, 则输出-1.

2	6	4
1	3	7
	5	8

8	1	5
7	3	6
4		2



9!=362880

Input	Output
1 2 6 4 1 3 7 0 5 8 8 1 5 7 3 6 4 0 2	31

题目分析

2	6	4
1	3	7
	5	8

状态迁移

2	6	4
	3	7
1	5	8

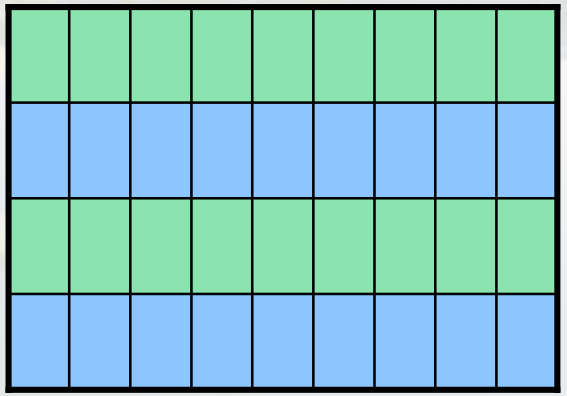
一个状态　　下一个状态

2	6	4	1	3	7	0	5	8

状态迁移

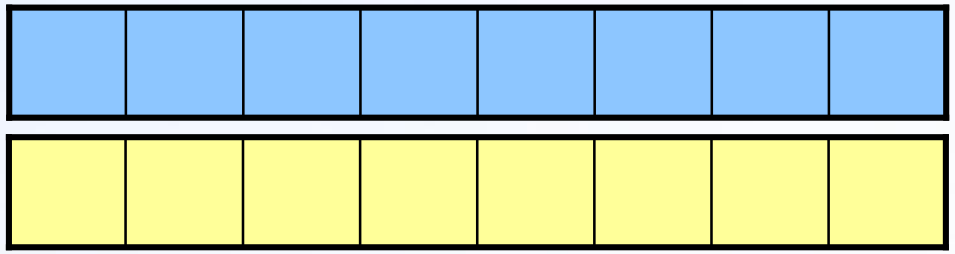
2	6	4	0	3	7	1	5	8

队列queue记录结点的入队与出队



两个同类型同长度数组之间的运算:

例如： int a[8], b[8];



- 内存拷贝memcpy: 按字节顺序复制

  a数组内容复制到b: memcpy(b, a, sizeof(a));

- 内存比较memcmp: 按字节顺序比较

  memcmp(a, b, sizeof(a));

  若a>b, 返回正值; a<b, 返回负值; a=b, 返回0.

0~8的排列1-1映射到0 ~ 9!-1

思想：利用逆序数, 变进制数, 阶乘数系

阶乘数	8!	7!	6!	5!	4!	3!	2!	1!	0!
一个排列	2	6	4	1	3	7	0	5	8
以某位开始的逆序数	2	5	3	1	1	2	0	0	0

则该排列对应的数为:

$$2\cdot 8!+5\cdot 7!+3\cdot 6!+1\cdot 5!+1\cdot 4!+2\cdot 3!$$

排列876543210对应的数为:

$$8\cdot 8!+7\cdot 7!+6\cdot 6!+5\cdot 5!+4\cdot 4!+3\cdot 3!+2\cdot 2!$$
$$+1\cdot 1!+0\cdot 0!=9!-1$$

参考程序

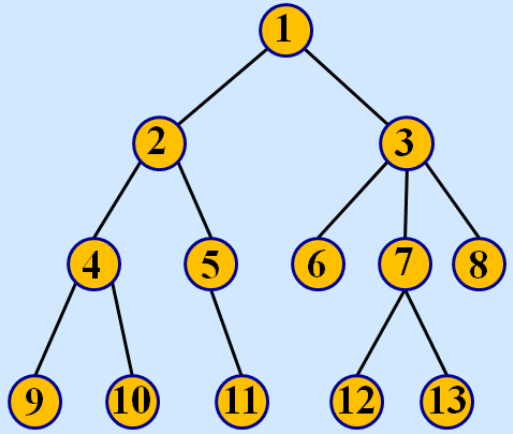
```
#include <stdio.h>
#include <string.h>
#define N1 362880
#define N2 1000000
int s[9], g[9], fact[9], queue[N2][9], vis[N1], dist[N2];
int dx[] = {-1, 1, 0, 0}, dy[] = {0, 0, -1, 1};
int encode(int cur);
int bfs(void);
int main()
{
    int i, ans, T;
    for(i = 1, fact[0] = 1; i < 9; i++)
        fact[i] = fact[i-1] * i; //阶乘表
    scanf("%d", &T);
    while(T--)
    {
        for(i = 0; i < 9; i++)
            scanf("%d", &s[i]); //source state
        for(i = 0; i < 9; i++)
            scanf("%d", &g[i]); //goal state
```

2	6	4
1	3	7
	5	8

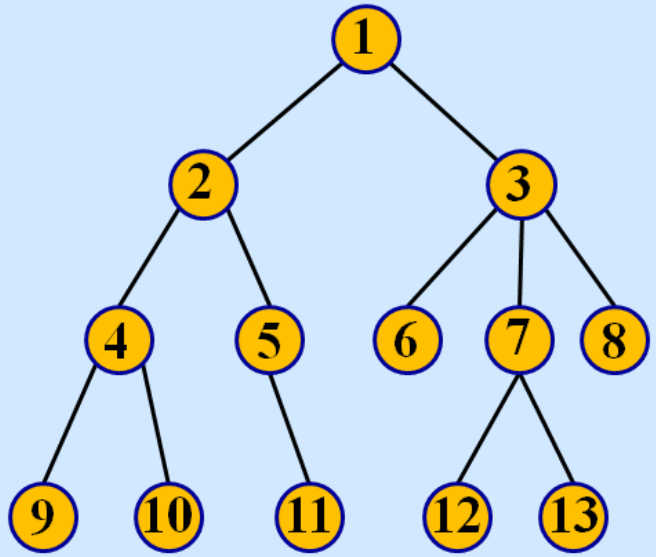
```
 memset(vis, 0, sizeof(vis));
        ans = bfs();
        if(ans > 0)   printf("%d\n", dist[ans]);
        else          printf("-1\n");
    }
    return 0;
}
int encode(int cur) //0~8的排列转换为0~9!-1之间的整数
{
    int i, j, cnt, code = 0;
    for(i = 1; i < 9; i++)
    {
        cnt = 0;
        for(j = i - 1; j >= 0; j--)
            if(queue[cur][j] < queue[cur][i])
                cnt++;
        code += fact[i]*cnt;
    }
    return code;
}
```

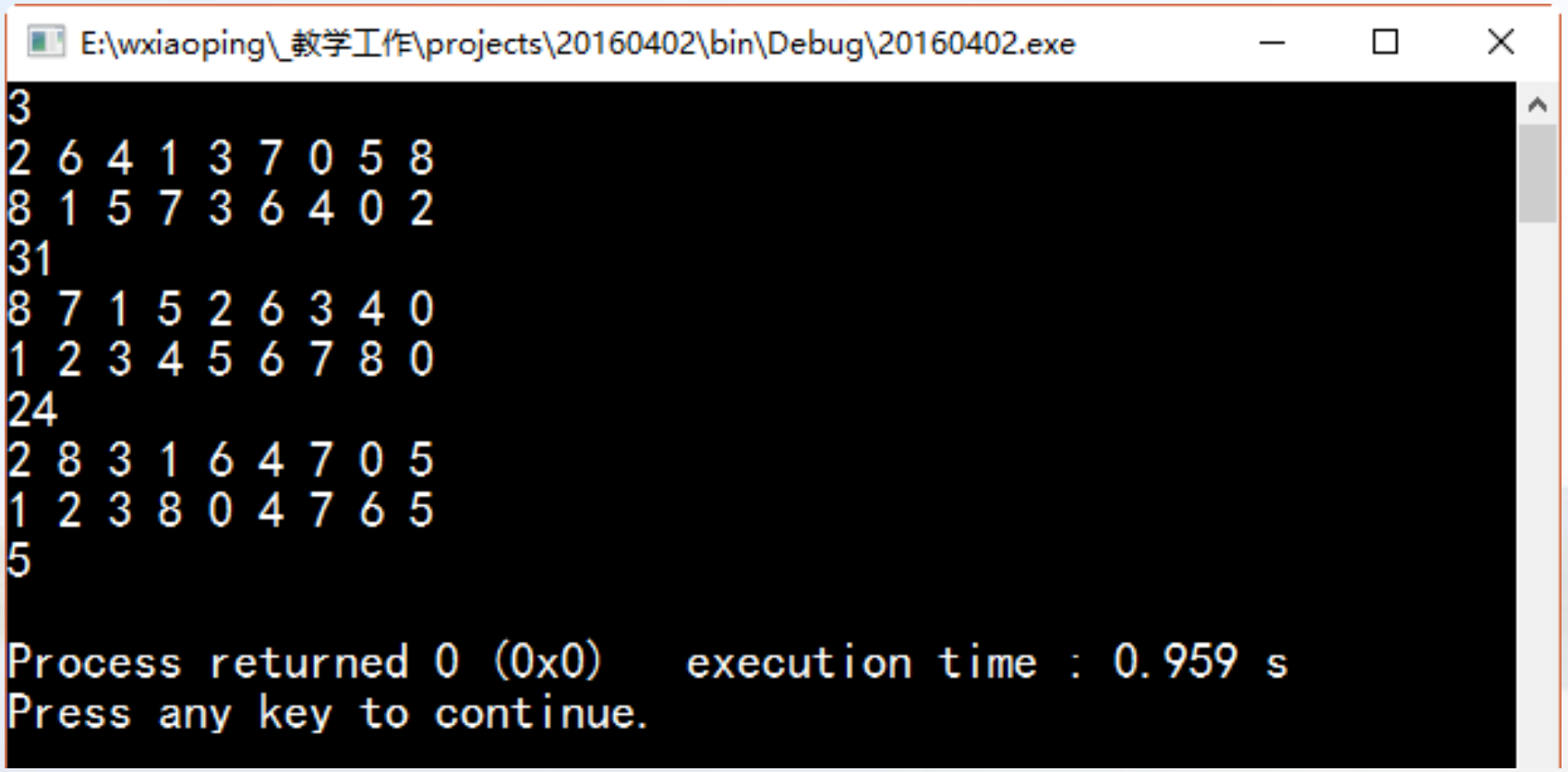
阶乘数	8!	7!	6!	5!	4!	3!	2!	1!	0!
一个排列	2	6	4	1	3	7	0	5	8
以某位开始的逆序数	2	5	3	1	1	2	0	0	0

```
int bfs(void)
{
    int front = 0, rear = 1, *pf, *pr, code;
    int i, x, y, z, newx, newy, newz;//x行,y列,0下标
    memcpy(queue[0], s, sizeof(s));
    dist[0] = 0;
    vis[encode(0)] = 1;
    while(front < rear)
    {
        pf = queue[front];
        if(memcmp(pf, g, sizeof(g)) == 0) //找到解决方案
            return front;
        for(z = 0; z < 9; z++)
            if(pf[z] == 0)
                break;
        x = z / 3;
        y = z % 3;
        for(i = 0; i < 4; i++)//开始扩展
        {
            newx = x + dx[i];
            newy = y + dy[i];
            newz = newx * 3 + newy;
```



```
if(newx>=0 && newx<3 && newy>=0 && newy<3)
            {
                pr = queue[rear];
                memcpy(pr, pf, sizeof(g));//扩展新结点
                pr[newz] = pf[z];  pr[z] = pf[newz];
                dist[rear] = dist[front] + 1;
                code = encode(rear);
                if(vis[code] == 0) //没有访问过
                {
                    vis[code] = 1;
                    rear++;
                }
            }
        }
        front++;
    }
    return 0;
}
```




E:\wxiaoping_教学工作\projects\20160402\bin\Debug\20160402.exe
3
2 6 4 1 3 7 0 5 8
8 1 5 7 3 6 4 0 2
31
8 7 1 5 2 6 3 4 0
1 2 3 4 5 6 7 8 0
24
2 8 3 1 6 4 7 0 5
1 2 3 8 0 4 7 6 5
5
Process returned 0 (0x0)   execution time : 0.959 s
Press any key to continue.