

2021-11-14 作业

1. 证明: 设  $E = \{1, 2, \dots, n\}$  为所给活动的集合.

活动按结束时间非递减排序.

设  $A \subseteq E$  是所给活动安排问题一个最优解.

且  $A$  中活动也按结束时间非递减排序.

$A$  中第一个活动是活动  $k$ .

① 当  $k=1$ , 则  $A$  即为一个以贪心选择开始的最优解.

② 当  $k>1$ , 设  $B = A - \{k\} \cup \{1\}$ .

由于  $t_1 < t_k$ , 其中  $t_i$  为当前集合中最大结束时间.

$A$  中活动互为相容.

$\therefore B$  中活动也互为相容.

又由于  $B$  中活动数与  $A$  中活动数, 且  $A$  是最优的, 故  $B$  也是最优的.

## 2. N皇后问题的递归回溯算法

算法实现:

```
#include <iostream>
#include <math>
#include <vector>
#include <array>
#include <algorithm>
using namespace std;

template<int N>
class NQueen
{
public:
    typedef array<int, N> ColumnType;

    NQueen()
    {
        Column.fill(-1);
    }

    /*使用回溯法, 求出一个n*n的棋盘上放置n个皇后, 使其不能互相攻击的所有可能位置。*/
    void NQueen()
    {
        int count = 0;
        int currentRow = 0; // currentRow 是当前行

        while(currentRow > -1) // 对所有的行执行以下语句
        {
            Column[currentRow] += 1; // 移动到下一列
            while((Column[currentRow] < N) && (Place(column, currentRow)))
            {
                Column[currentRow] += 1;
            }
            if (Column[currentRow] < N) // 找到一个位置
            {
                if(currentRow == (N-1)) // 是一个完整的解吗
                {
                    resVec.push_back(column);
                }
                else
                {
                    currentRow++;
                    Column[currentRow] = -1; // 转向下一行
                }
            }
            else
            {
                currentRow--; // 回溯
            }
        }
    }

    void PrintAll()
    {
        cout << "一共有" << resVec.size() << "组解。" << endl;
        for_each(resVec.begin(), resVec.end(), [](
            const ColumnType& column)
        {
            for_each(column.begin(), column.end(), [](int& value){cout << value << " ";});
            cout << endl;
        });
    }

private:
    vector<ColumnType> resVec; // 存放所有的解
    ColumnType Column; // 存放回溯过程中的一组成功解, 下标代表行, 数组值代表列

    //一个皇后是否能放在第 row 行, 和第 Column[row] 列?
    bool Place(ColumnType& columnArray, int row)
    {
        for(int oldRow = 0; oldRow < row; ++oldRow)
        {
            if (columnArray[oldRow] == columnArray[row] || //同一行有两个皇后
                abs(columnArray[oldRow]-columnArray[row]) == abs(oldRow-row)) //在同一斜线上
            {
                return false;
            }
        }
        return true;
    }
};

int main()
{
    system("chcp 65001"); // 解决window输出中文乱码问题

    /* N皇后问题, 找出在一个n*n的棋盘上防止n个皇后, 并使其不能互相攻击的所有方案。
    即, 所有的皇后不能同行或同列。*/
    NQueen<8> Queens;
    Queens.NQueen();
    Queens.PrintAll();
}
```

结果打印，以8皇后问题为例， $N=8$ ：

```
--共有92组解。
0 4 7 5 2 6 1 3
0 5 7 2 6 3 1 4
0 6 3 5 7 1 4 2
0 6 4 7 1 3 5 2
1 3 5 7 2 0 6 4
1 4 6 0 2 7 5 3
1 4 6 3 0 7 5 2
1 5 0 6 3 7 2 4
1 5 7 2 0 3 6 4
1 6 2 5 7 4 0 3
1 6 4 7 0 3 5 2
1 7 5 0 2 4 6 3
2 0 6 4 7 1 3 5
2 4 1 7 0 6 3 5
2 4 1 7 5 3 6 0
2 4 6 0 3 1 7 5
2 4 7 3 0 6 1 5
2 5 1 4 7 0 6 3
2 5 1 6 0 3 7 4
2 5 1 6 4 0 7 3
2 5 3 0 7 4 6 1
2 5 3 1 7 4 6 0
2 5 7 0 3 6 4 1
2 5 7 0 4 6 1 3
2 5 7 1 3 0 6 4
2 6 1 7 4 0 3 5
2 6 1 7 5 3 0 4
2 7 3 6 0 5 1 4
3 0 4 7 1 6 2 5
3 0 4 7 5 2 6 1
3 1 4 7 5 0 2 6
3 1 6 2 5 7 0 4
3 1 6 2 5 7 4 0
3 1 6 4 0 7 5 2
3 1 7 4 6 0 2 5
3 1 7 5 0 2 4 6
3 5 0 4 1 7 2 6
3 5 7 1 6 0 2 4
3 5 7 2 0 6 4 1
3 6 0 7 4 1 5 2
3 6 2 7 1 4 0 5
3 6 4 1 5 0 2 7
3 6 4 2 0 5 7 1
3 7 0 2 5 1 6 4
3 7 0 4 6 1 5 2
3 7 4 2 0 6 1 5
4 0 3 5 7 1 6 2
4 0 7 3 1 6 2 5
4 0 7 5 2 6 1 3
4 1 3 5 7 2 0 6
4 1 3 6 2 7 5 0
4 1 5 0 6 3 7 2
4 1 7 0 3 6 2 5
4 2 0 5 7 1 3 6
4 2 0 6 1 7 5 3
```