SOFT500327: 算法设计与分析 2022-Fall

Homework-**2** Solutions

软件 2101 杨豪 学号: 2206213297

2022 年 10 月 23 日

Honor Code: I promise that I finished the homework solutions on my own without copying other people's work.

pdf 中所示源码均附于 src/文件夹中

3-2

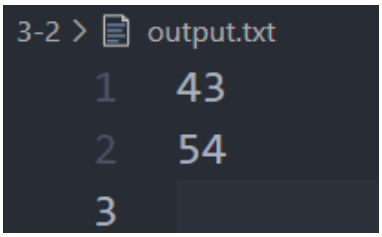
代码

```cpp
#include <iostream>

using namespace std;
const int N = 110;
int dp[N][N], num[N][N];
// DP(i,j) is score from i to j.
// DPmin(i,j) = min{DP(i,k) + DP(k+1,j) + num[i][i + j]}
int DPmin(int n)
{
    for (int j = 2; j < n; j++){
        for (int i = 0; i + j < n; i++){
            for (int k = 0; k < j; k++){
                if (k == 0){
                    num[i][i + j] = num[i][i + k] + num[i + k + 1][i + j];
                    dp[i][i + j] = dp[i][i + k] + dp[i + k + 1][i + j] + num[i][i + j];
                }
                else dp[i][i + j] = min(dp[i][i + j], dp[i][i + k] + dp[i + k + 1][i + j] + num[i][i + j]);
            }
        }
    }
    return dp[0][n - 1];
}

// DPmax(i,j) = max{DP(i,k) + DP(k+1,j) + num[i][i + j]}
int DPmax(int n)
{
    for (int j = 2; j < n; j++){
        for (int i = 0; i + j < n; i++){
            for (int k = 0; k < j; k++){
                if (k == 0){
                    num[i][i + j] = num[i][i + k] + num[i + k + 1][i + j];
                    dp[i][i + j] = dp[i][i + k] + dp[i + k + 1][i + j] + num[i][i + j];
                }
                else dp[i][i + j] = max(dp[i][i + j], dp[i][i + k] + dp[i + k + 1][i + j] + num[i][i + j]);
            }
        }
    }
    return dp[0][n - 1];
}

int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    int n;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> num[i][i];
        if (i > 0) num[i - 1][i] = dp[i - 1][i] = num[i - 1][i - 1] + num[i][i];
    }
    cout << DPmin(n) << endl << DPmax(n) << endl;
    return 0;
}
```

输入

```
3-2 > input.txt
1   4
2   4 4 5 9
```

输出

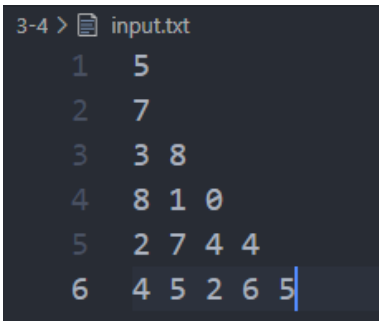


3-4

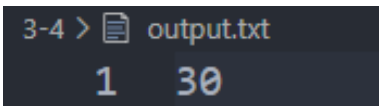
代码

```cpp
3-4 > 3-4.cpp > main()
#include <iostream>

using namespace std;

const int N = 110;
// dp[i][j] represents max path from top to i th floor and j th number.
// dp[i][j] = max{dp[i-1][j],dp[i-1][j-1]}+num[i,j];
int dp[N][N];

int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    int n,m; cin >> n;
    cin >> dp[0][0];
    for(int i = 1; i < n; i ++){
        cin >> m; dp[i][0] = dp[i-1][0] + m;
        for(int j = 1; j <= i; j++){
            cin >> m;
            dp[i][j] = max(dp[i-1][j], dp[i-1][j-1]) + m;
        }
    }
    int res = 0;
    for(int i = 0; i < n; i ++) res = max(res, dp[n-1][i]);
    cout << res;
    return 0;
}
```

输入

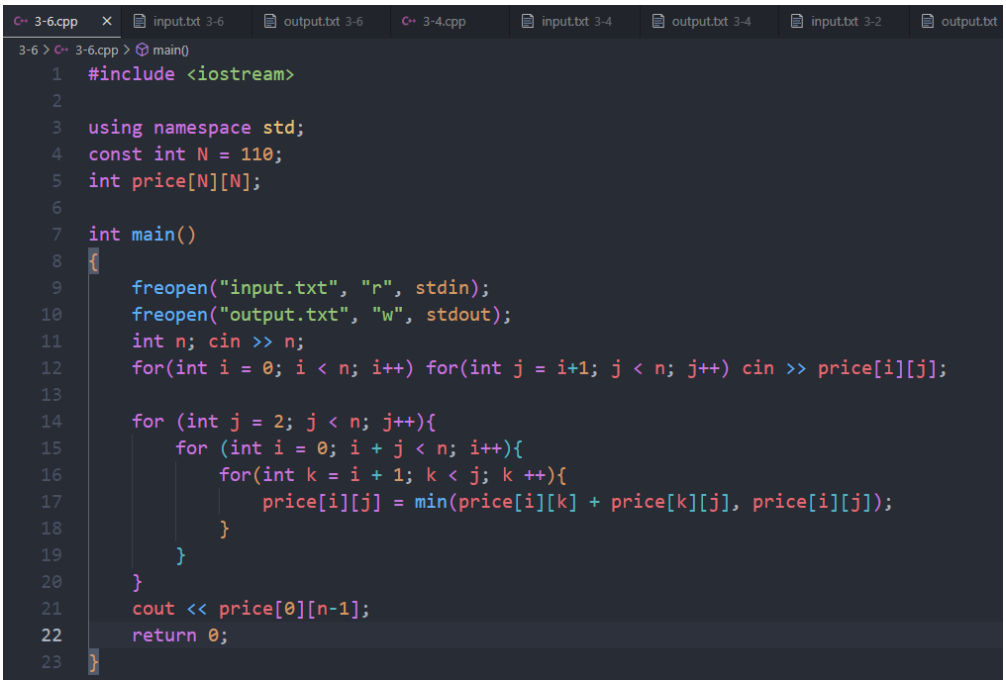

3-4 > input.txt
1 5
2 7
3 3 8
4 8 1 0
5 2 7 4 4
6 4 5 2 6 5

输出

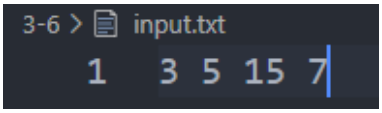

3-4 > output.txt
1 30

3-6

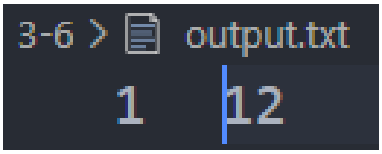
代码


3-6.cpp
input.txt 3-6
output.txt 3-6
3-4.cpp
input.txt 3-4
output.txt 3-4
input.txt 3-2
output.txt
3-6 > 3-6.cpp > main()
#include <iostream>
using namespace std;
const int N = 110;
int price[N][N];
int main()
{
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
int n; cin >> n;
for(int i = 0; i < n; i++) for(int j = i+1; j < n; j++) cin >> price[i][j];
for (int j = 2; j < n; j++){
for (int i = 0; i + j < n; i++){
for(int k = i + 1; k < j; k ++){
price[i][j] = min(price[i][k] + price[k][j], price[i][j]);
}
}
}
cout << price[0][n-1];
return 0;
}

输入

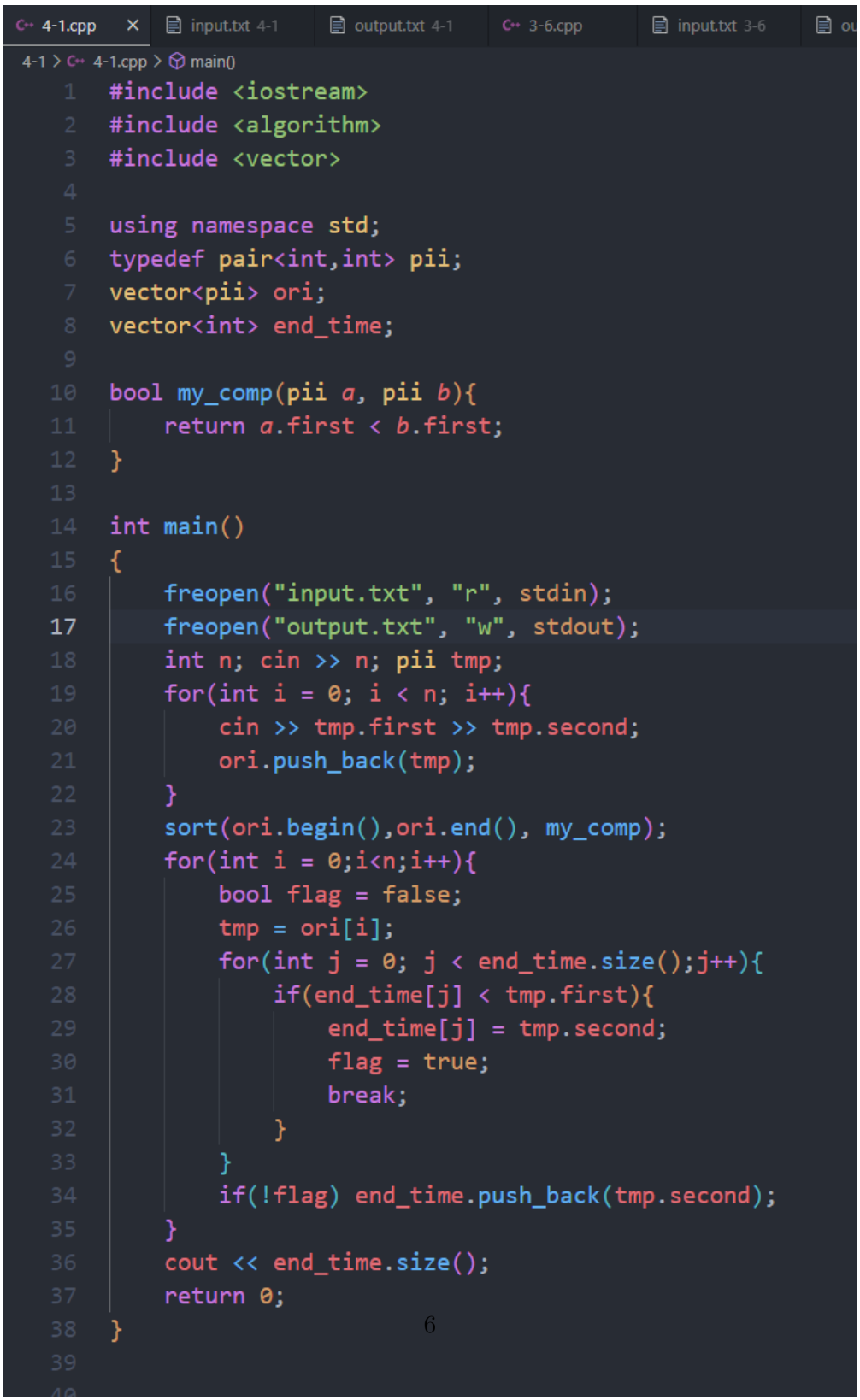


输出

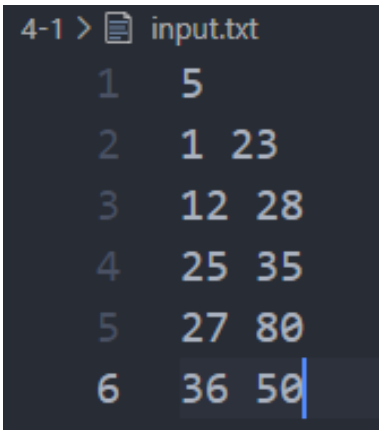


4-1

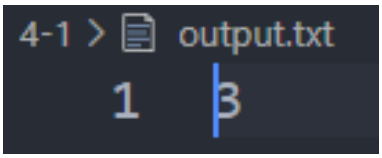
代码


4-1.cpp
input.txt 4-1
output.txt 4-1
3-6.cpp
input.txt 3-6
4-1 > 4-1.cpp > main()
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
typedef pair<int,int> pii;
vector<pii> ori;
vector<int> end_time;
bool my_comp(pii a, pii b){
return a.first < b.first;
}
int main()
{
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
int n; cin >> n; pii tmp;
for(int i = 0; i < n; i++){
cin >> tmp.first >> tmp.second;
ori.push_back(tmp);
}
sort(ori.begin(),ori.end(), my_comp);
for(int i = 0;i<n;i++){
bool flag = false;
tmp = ori[i];
for(int j = 0; j < end_time.size();j++){
if(end_time[j] < tmp.first){
end_time[j] = tmp.second;
flag = true;
break;
}
}
if(!flag) end_time.push_back(tmp.second);
}
cout << end_time.size();
return 0;
}

输入



输出



4-9

证明

对于选好的分割如果还有更小值，则需要将其中 n 组相邻合为 n_1 个。将每相邻分割的分界线看做一个边。这里看做是对已选定的边进行“平移”，直接去掉某条边看做该边平移至下一条边。显然，每次平移至少需要一条边向后平移。

假设可以平移，则对于向后平移的边，其平移所得到的新分割之和　原分割之和 + 原边所在的下一个数且要求新分割之和 <n，但如果新分割之和 <n，原边就并不符合我们贪心的算法。

代码

4-9.cpp | output.txt 4-9 | 4-1.cpp | input.txt 4-1 | output.txt 4-1

4-9 > 4-9.cpp > main()

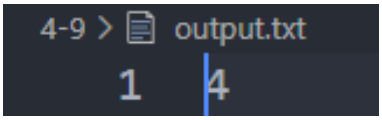
```cpp
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    int n,k; cin >> n >> k;
    int tmp_n = n, num = 0;
    for(int i = 0;i < k+1;i++){
        int tmp; cin >> tmp;
        if(tmp > n){
            cout << "No Solution";
            return 0;
        }
        if(tmp_n > tmp) tmp_n -= tmp;
        else tmp_n = n - tmp, num++;
    }
    cout << num;
    return 0;
}
```

输入

4-9 > input.txt

```
7 7
1 2 3 4 5 1 6 6
```

输出

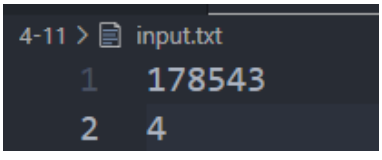

4-9 > output.txt
1 4

4-11

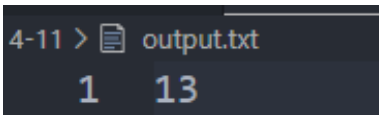
代码


4-11.cpp
4-11 > 4-11.cpp > main()
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main()
{
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
string ori; int n; cin >> ori >> n;
for(int i = 0; i < n; i ++)
ori.erase(max_element(ori.begin(),ori.end()));
cout << ori;
return 0;
}

输入


4-11 > input.txt
1 178543
2 4

输出



Other things

LaTeX code refer to these things and was complied on texlive2020.

- UCB-CS70's given homework template.
- A free website useful to edit LaTeX formula code.

Thanks for your correcting and grading :).