算法分析与设计

实验 03

实验目的

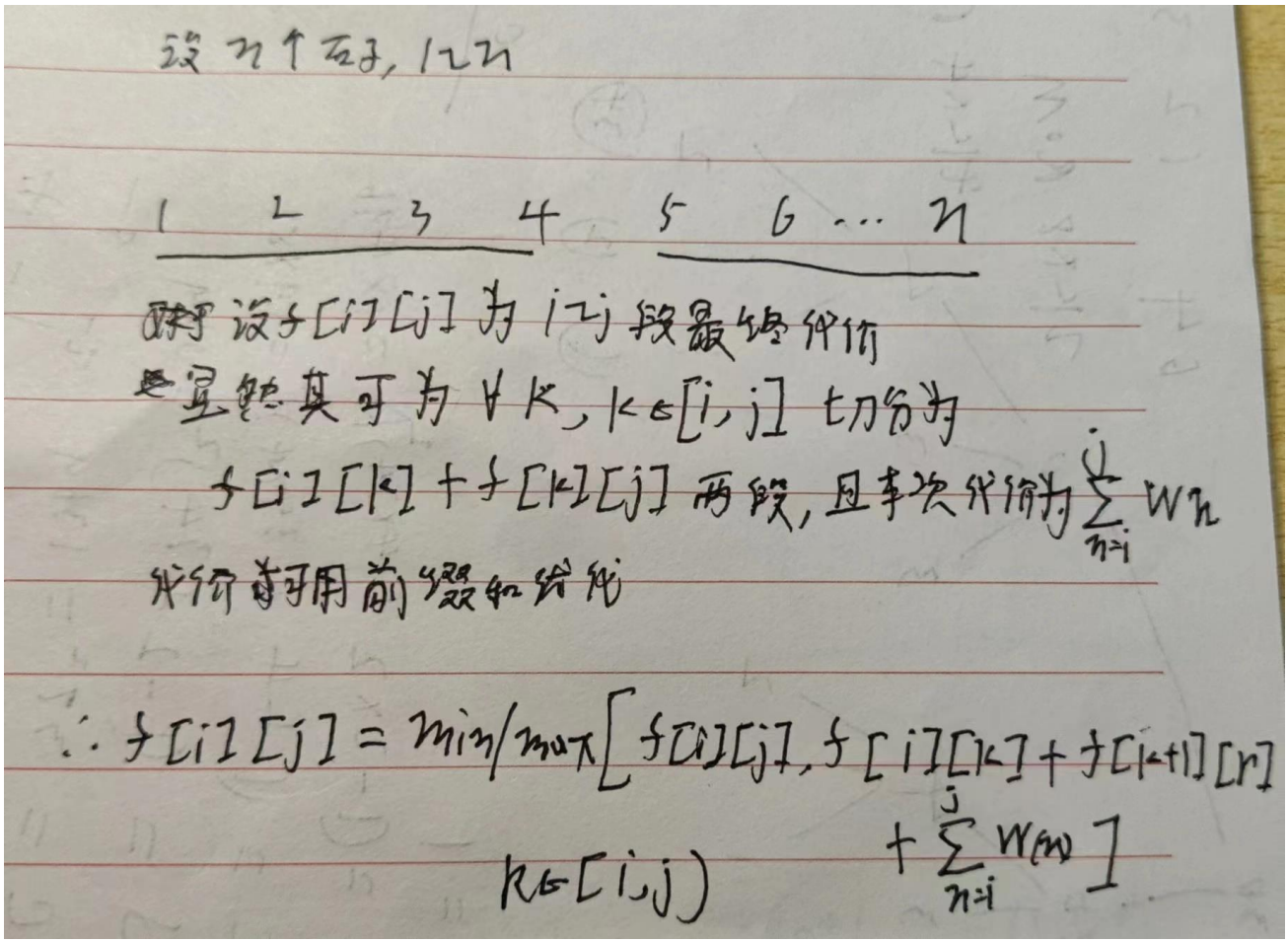
- 掌握动态规划的概念：最优子结构和子问题重叠性质。
- 掌握动态规划的解题步骤。
- 掌握采用动态规划解决矩阵连乘、最长公共子序列、凸多边形的最优三角剖分问题、简单 0-1 背包问题和最优二叉搜索树问题。

实训内容

1、 熟练掌握动态规划的最优子结构性质和子问题重叠性质，并使用算法模拟软件辅助理解矩阵连乘的求解过程。

2、 算法实现题：3-3 石子合并问题（P83）。编程实现，然后分析该问题的子问题重叠性质，并证明该问题的最优子结构性质。

- 证明



- 区间 DP 实现：

StoneMaramerge.cpp | Test.cpp | output_1.txt

算法分析与设计 3 > 实验 > D_More_Wrong > Test.cpp > main()

```cpp
#include<iostream>
#include<cstring>
#define ll long long
using namespace std;
const int N=310;
int n;
ll s[N],f[N][N],g[N][N];
int main(){
    cin>>n;
    //前缀和
    for(int i=1;i<=n;i++){
        cin>>s[i];
        s[i]+=s[i-1];
    }
    memset(f,0x3f,sizeof f);
    for(int len=1;len<=n;len++){
        for(int l=1;l+len-1<=n;l++){
            int r=l+len-1;
            if(len==1){
                f[l][r]=g[l][r]=0;
                continue;
            }
            for(int i=l;i<r;i++){
                f[l][r]=min(f[l][r],f[l][i]+f[i+1][r]+s[r]-s[l-1]);
                g[l][r]=max(g[l][r],g[l][i]+g[i+1][r]+s[r]-s[l-1]);
            }
        }
    }
    cout<<"最大分数: "<<g[1][n]<<endl;
    cout<<"最小分数: "<<f[1][n];
    return 0;
}
```

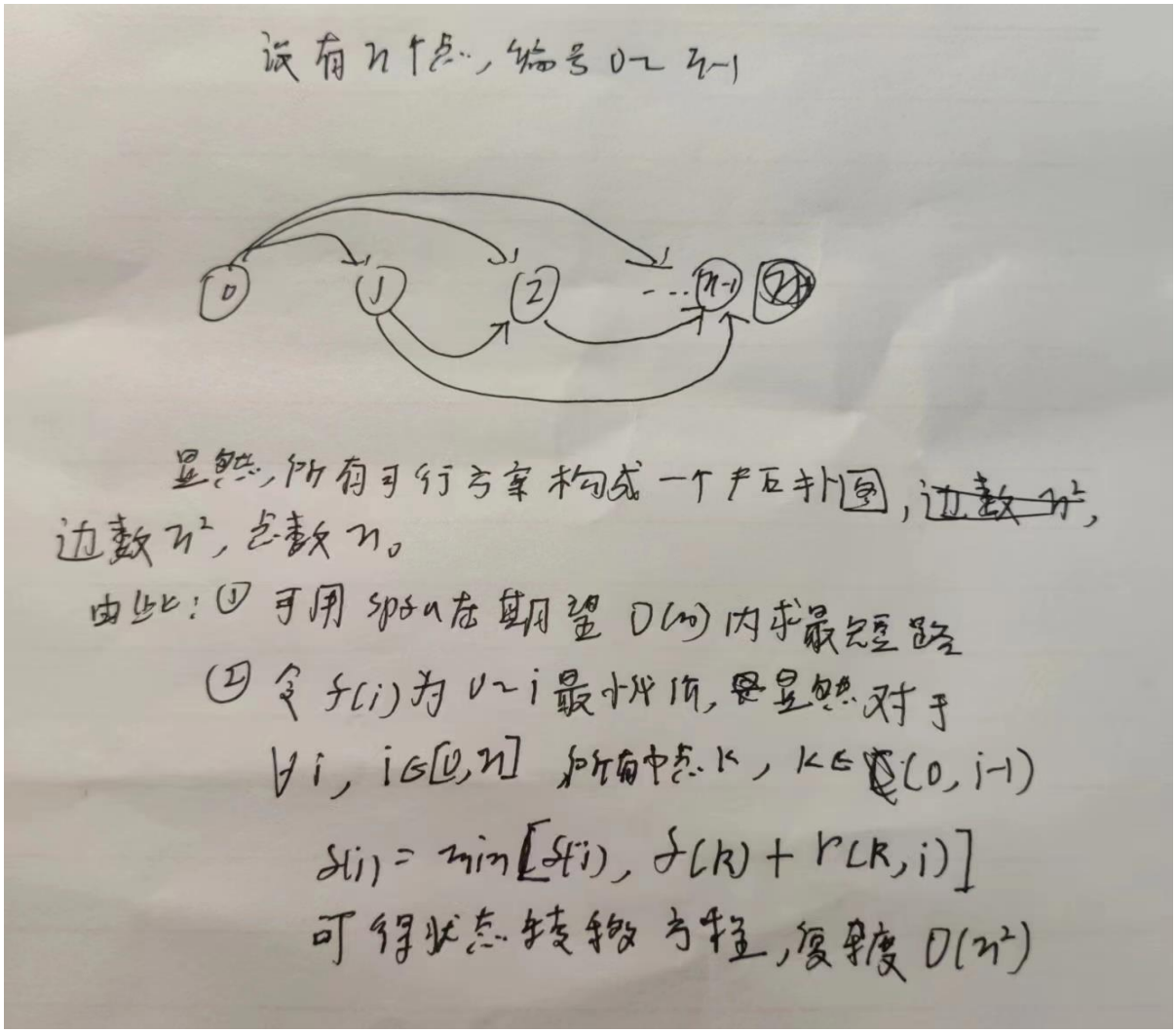
result.txt

算法分析与设计 3 > 实验 > D_More_Wrong > Tests > result.txt

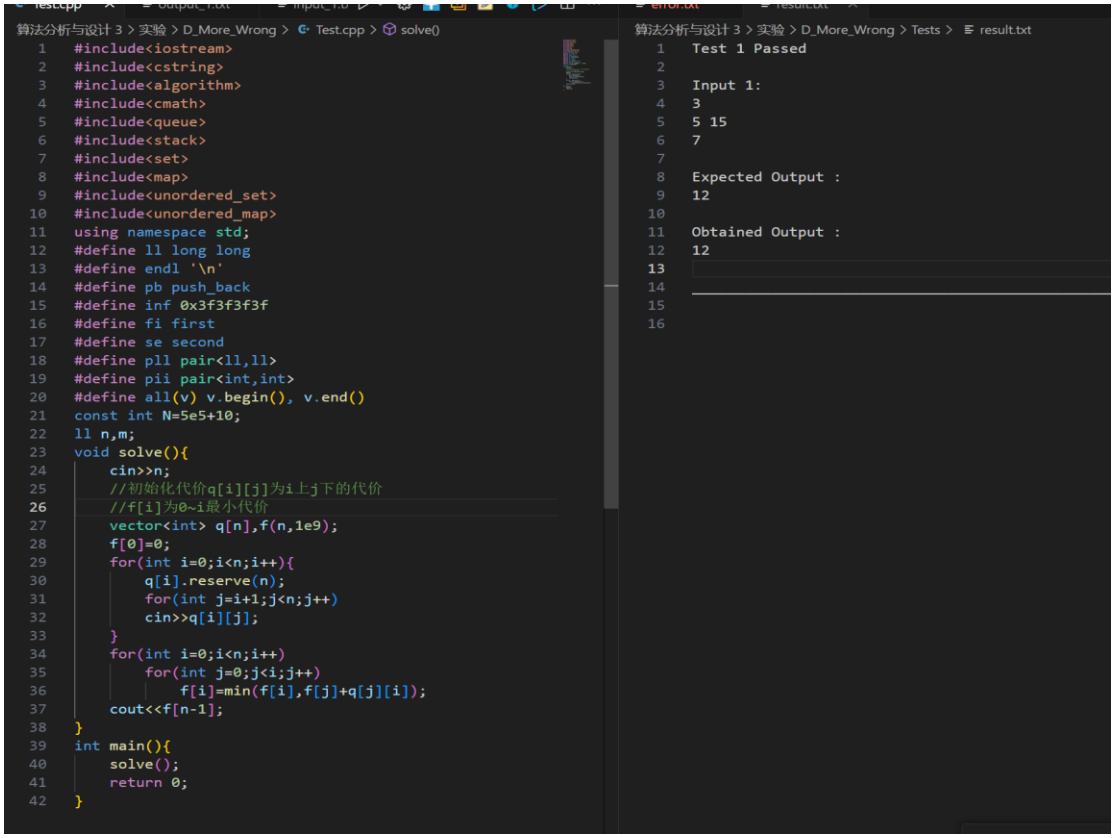
```
Test 1 Passed

Input 1:
4
1 3 5 2

Expected Output :
最大分数: 29
最小分数: 22

Obtained Output :
最大分数: 29
最小分数: 22
```

3、 算法实现题：3-6 租用游艇问题（P84）。

- 证明



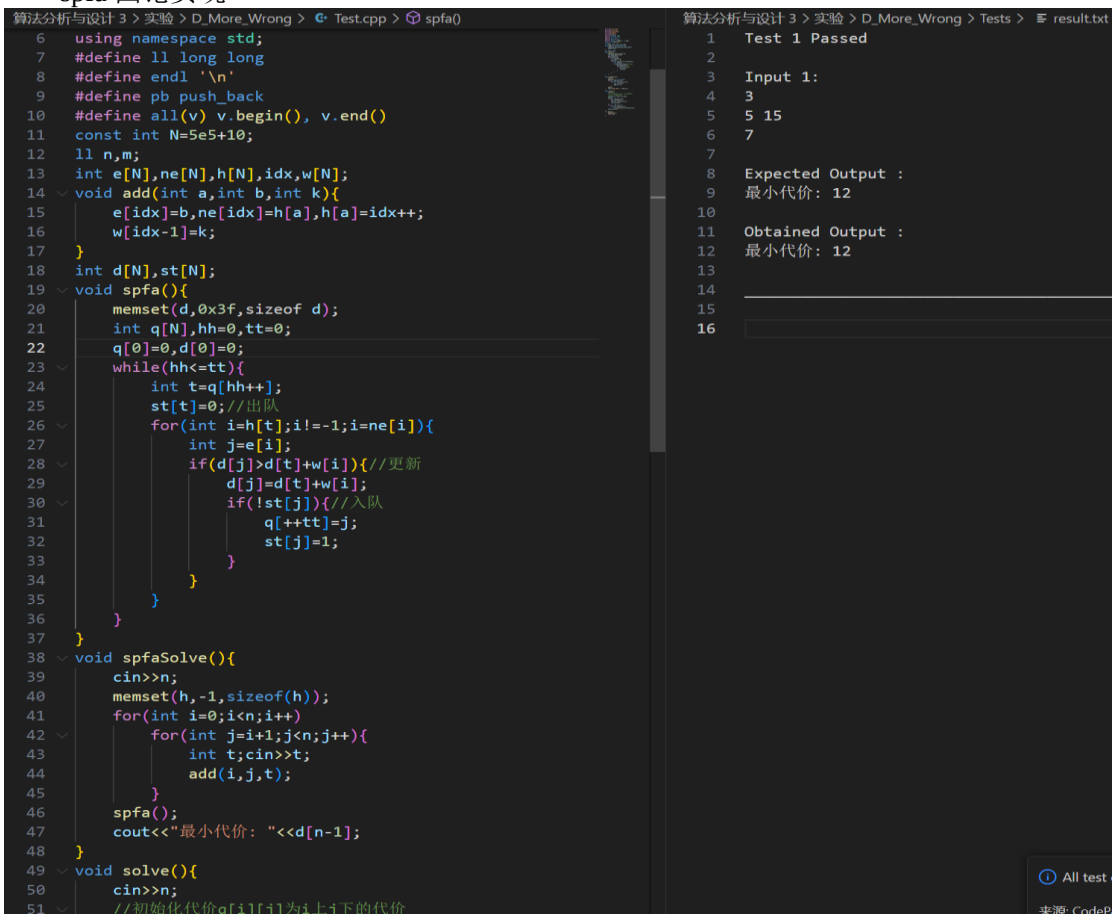
- 线性 DP 实现



```cpp
#include<iostream>
#include<cstring>
#include<algorithm>
#include<cmath>
#include<queue>
#include<stack>
#include<set>
#include<map>
#include<unordered_set>
#include<unordered_map>
using namespace std;
#define ll long long
#define endl '\n'
#define pb push_back
#define inf 0x3f3f3f3f
#define fi first
#define se second
#define pll pair<ll,ll>
#define pii pair<int,int>
#define all(v) v.begin(), v.end()
const int N=5e5+10;
ll n,m;
void solve(){
    cin>>n;
    //初始化代价q[i][j]为i上j下的代价
    //f[i]为0~i最小代价
    vector<int> q[n],f(n,1e9);
    f[0]=0;
    for(int i=0;i<n;i++){
        q[i].reserve(n);
        for(int j=i+1;j<n;j++)
        cin>>q[i][j];
    }
    for(int i=0;i<n;i++)
        for(int j=0;j<i;j++)
            f[i]=min(f[i],f[j]+q[j][i]);
    cout<<f[n-1];
}
int main(){
    solve();
    return 0;
}
```

```
Test 1 Passed

Input 1:
3
5 15
7

Expected Output :
12

Obtained Output :
12
```

- spfa 图论实现



```cpp
using namespace std;
#define ll long long
#define endl '\n'
#define pb push_back
#define all(v) v.begin(), v.end()
const int N=5e5+10;
ll n,m;
int e[N],ne[N],h[N],idx,w[N];
void add(int a,int b,int k){
    e[idx]=b,ne[idx]=h[a],h[a]=idx++;
    w[idx-1]=k;
}
int d[N],st[N];
void spfa(){
    memset(d,0x3f,sizeof d);
    int q[N],hh=0,tt=0;
    q[0]=0,d[0]=0;
    while(hh<=tt){
        int t=q[hh++];
        st[t]=0;//出队
        for(int i=h[t];i!=-1;i=ne[i]){
            int j=e[i];
            if(d[j]>d[t]+w[i]){//更新
                d[j]=d[t]+w[i];
                if(!st[j]){//入队
                    q[++tt]=j;
                    st[j]=1;
                }
            }
        }
    }
}
void spfaSolve(){
    cin>>n;
    memset(h,-1,sizeof(h));
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++){
            int t;cin>>t;
            add(i,j,t);
        }
    spfa();
    cout<<"最小代价: "<<d[n-1];
}
void solve(){
    cin>>n;
    //初始化代价q[i][j]为i上j下的代价
```

```
Test 1 Passed

Input 1:
3
5 15
7

Expected Output :
最小代价: 12

Obtained Output :
最小代价: 12
```