算法分析与设计

实验 04

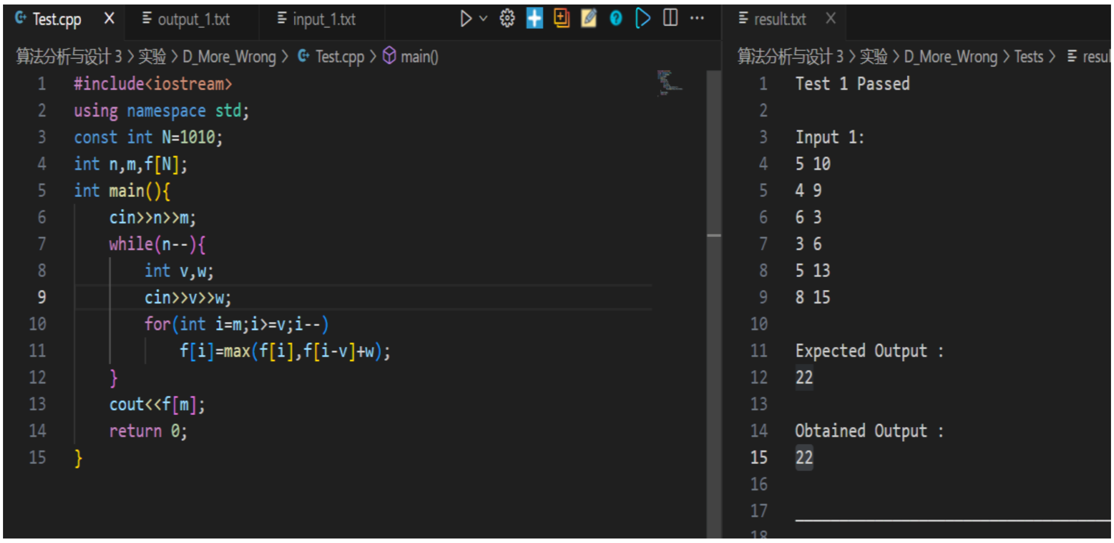
实验目的

- 掌握贪心算法的贪心选择性质和最优子结构性质；
- 掌握活动安排问题、最优装载问题和哈夫曼编码问题；
- 掌握单源最短路径、最小生成树和多机调度问题。

实训内容

1、 **简单 0-1 背包问题：**给定一个背包，其容量是 10 千克，5 件物品，其重量分别是 4，6，3，5，8 公斤；其价值分别是：9，3，6，13，15 元，试根据动态规划算法给出 0-1 背包问题的最优解以及相应的求解过程。

- **实现**



● 推导过程

设体最大容量 m

0-1 背包:

~~概念~~ $f(i,j)$ 为在选前 i 个物品条件下,选背包(总)容量 $=j$ 的最优解,

显然对于物品 i: 体积为 v, 价值为 w

$$f(i,j) = \max[f(i,j),\ f(i-1, j-v) + w]$$ 且

$j \in [v, 最大容量)$, $v \le 最大容量$

由于只使用 $f(i-1, t)$ $t \in [v, 最大容量)$

∴ 且 $j-v \le j$

∴ 可以优化第一维,故体积从大到小枚举

∴ 令 $f(i)$ 为体积为 i 的背包的最大价值

$$f(i) = \max[f(i),\ f(i - x_j v) + x_j w]$$

$j \in [1, m]$, 且 $x_j v \le i$

对于样例 $n=5$, $m=10$

① $(5,4)$ $f(10) = \max(0, f(5)+4) = 4$

$f(9) = \max(0, f(5)+4) = 4$

$f(8) = \vdots$

$f(4) = \max(0, f(0)+4) = 4$

② $(6,3)$ $f(10) = \max(f(10), f(4)+3) = 12$

$\vdots$

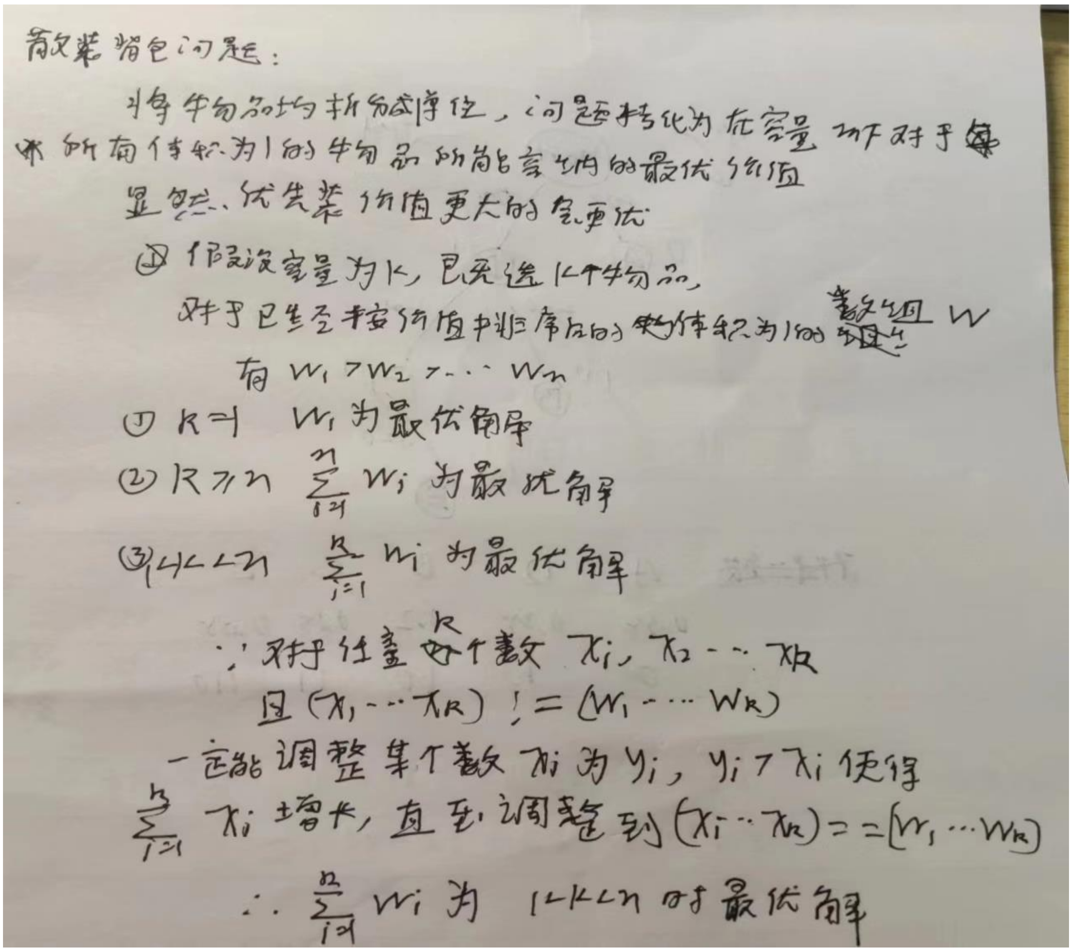
$f(6) = \max(f(6), f(0)+3) = 4$

以此类推,考虑最后一组物品后有:

$f(10) = 22$ 即在容量不超过 10 下,最大价值

2、 **散装背包问题：**给定一个背包，其容量是 10 千克，5 件物品，其重量分别是 4，6，3，5，8 公斤；其价值分别是：9，3，6，13，15 元，试根据贪心算法给出散装背包问题的最优解以及相应的求解过程。

- 过程



散装背包问题：

将物品的折分成单位，问题转化为在容量为k时对于所有体积为1的物品所能容纳的最优价值

显然，优先装价值更大的会更优

① 假设容量为k，已选k个物品，

对于已经按价值排序后的体积为1的数组 w

有 $w_1 > w_2 > \cdots w_n$

(1) $k=1$ w_1 为最优解

(2) $k>n$ $\sum_{i=1}^{n} w_i$ 为最优解

(3) $1<k<n$ $\sum_{i=1}^{k} w_i$ 为最优解

∵ 对于任意k个数 $x_1, x_2 \cdots x_k$

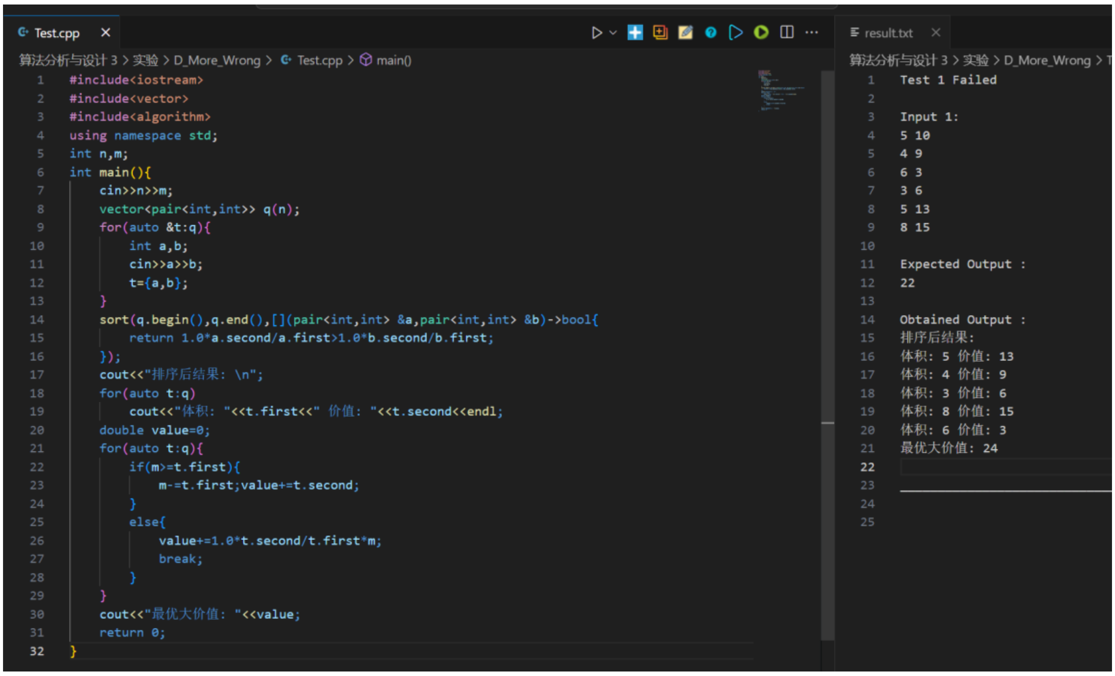
且 $(x_1 \cdots x_k) != (w_1 \cdots w_k)$

一定能调整某个数 x_i 为 y_i，$y_i > x_i$ 使得

$\sum_{i=1}^{k} x_i$ 增大，直到调整到 $(x_1 \cdots x_k) == (w_1 \cdots w_k)$

∴ $\sum_{i=1}^{k} w_i$ 为 $1<k<n$ 时最优解

- 实现



Test.cpp

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

```cpp
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int n,m;
int main(){
    cin>>n>>m;
    vector<pair<int,int>> q(n);
    for(auto &t:q){
        int a,b;
        cin>>a>>b;
        t={a,b};
    }
    sort(q.begin(),q.end(),[](pair<int,int> &a,pair<int,int> &b)->bool{
        return 1.0*a.second/a.first>1.0*b.second/b.first;
    });
    cout<<"排序后结果: \n";
    for(auto t:q)
        cout<<"体积: "<<t.first<<" 价值: "<<t.second<<endl;
    double value=0;
    for(auto t:q){
        if(m>=t.first){
            m-=t.first;value+=t.second;
        }
        else{
            value+=1.0*t.second/t.first*m;
            break;
        }
    }
    cout<<"最优大价值: "<<value;
    return 0;
}
```

result.txt

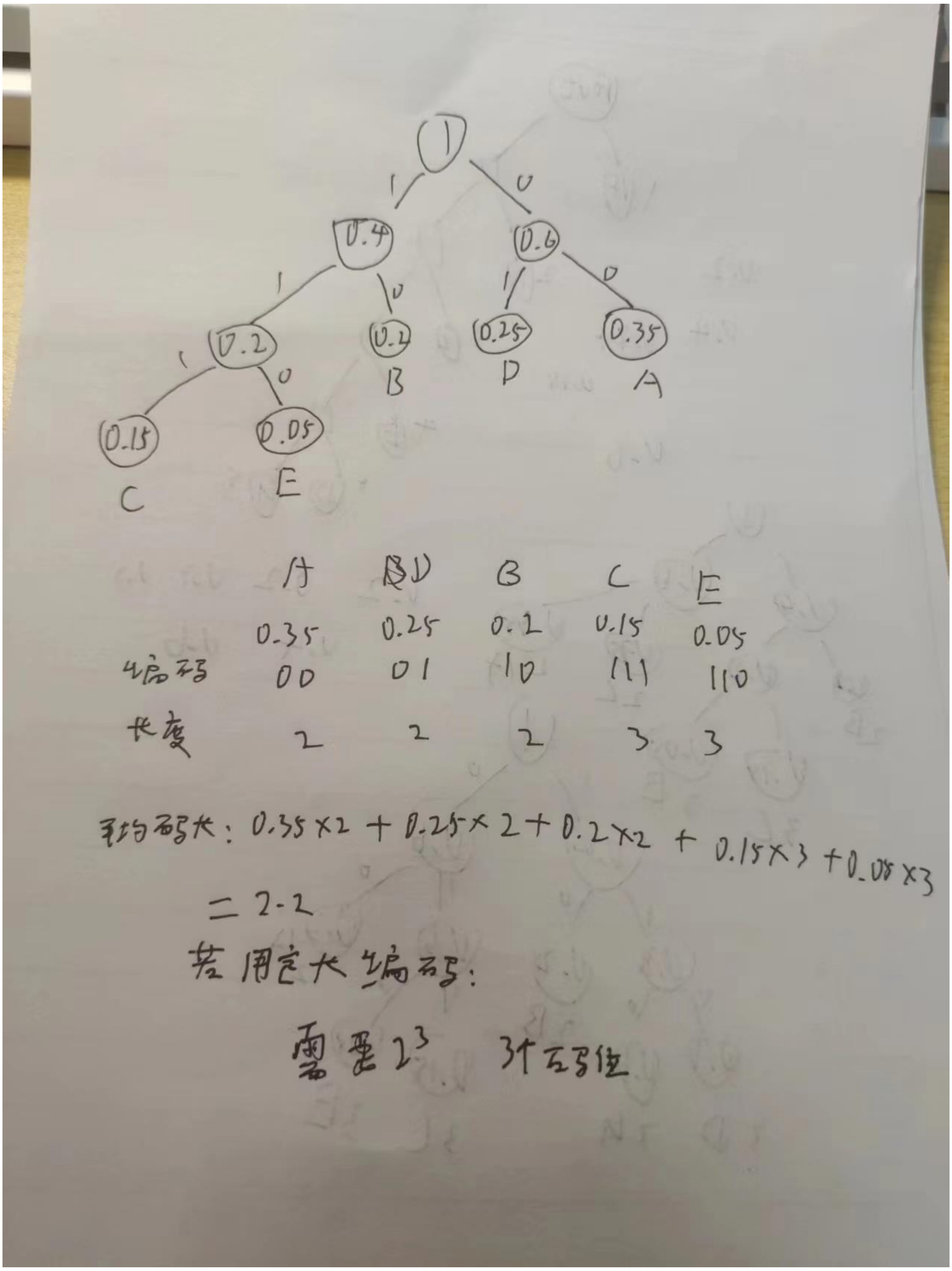
算法分析与设计 3 > 实验 > D_More_Wrong >

```
Test 1 Failed

Input 1:
5 10
4 9
6 3
3 6
5 13
8 15

Expected Output :
22

Obtained Output :
排序后结果:
体积: 5 价值: 13
体积: 4 价值: 9
体积: 3 价值: 6
体积: 8 价值: 15
体积: 6 价值: 3
最优大价值: 24
```

3、 对于一个包含五个字符 A，B，C，D，E 的字符文件，各字符出现的概率如下：

字符	A	B	C	D	E
概率	0.35	0.2	0.15	0.25	0.05

请使用哈夫曼编码构造各个字符的编码（要求给出相应的哈夫曼树），并计算出所有字符编码的平均码长。



算法设计题：请编程实现 4-7 多处最优服务次序问题（P116），并给出贪心选择性质和最优子结构性质的证明。

- 代码

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

```cpp
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
const int N=5e5+10;
int n,m;
void solve(){
    cin>>n>>m;
    vector<int> q(n),t(m,0);
    for(auto &t:q)cin>>t;
    sort(q.begin(),q.end());
    double ans=0;
    for(int i=0;i<n;i++){
        t[i%m]+=q[i];
        ans+=t[i%m];
    }
    cout<<ans/n;
}
int main(){
   int T=1;
   while(T--) solve();
   return 0;
}
```

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

```
Test 1 Passed

Input 1:

10 2
56 12 1 99 1000 234 33 55 99 812

Expected Output :
336

Obtained Output :
336
```

- 证明

最优服务：设 n 个人，m 个服务点。

① 将 n 个人按需要时间从小到大排，

② 对于 最后 $n-\lfloor\frac{n}{m}\rfloor\cdot m$ 个人无论什么顺序不影响等待

设 x_i 为排序后第 x_i 人所需时间

行 ①	x_1	x_{m+1}	x_{n-1}
②	x_2	x_{m+2}	x_n
⋮	⋮	⋮	
⑩	x_m	$x_{2\cdot m}$	
	列		

$\forall\, x_{i+km}$ 人其等待时间为 $\sum_{t=0}^{k-1} x_{i+tm} + x_{i+km}$

设 $s=\lfloor\frac{n}{m}\rfloor\cdot m$，忽略最后 $n-s$ 人时间

则有对于 $\forall\, x_{i+km}$

其对总时间贡献大为 $\underbrace{\frac{s}{m}}_{\text{列数}}-\underbrace{(k-1)}_{\text{其所在列-1}})\cdot\underbrace{(x_{i+km})}_{\text{其用时}}$

总时间：$\sum_{k=1}^{s/m}\sum_{i=1}^{m}\left[\left(\frac{s}{m}-(k-1)\right)x_{i+km}\right]$

为使其最小 需要令 $(k-1)\,x_{i+km}$ 尽可能小，

故 应令时间长的人在较靠后的列

4、算法设计题：请编程实现 4-11 删数问题（P117），并给出贪心选择性质和最优子结构性质的证明。

- 代码

```cpp
#include<iostream>
#include<string>
#include<queue>
using namespace std;
int n,m;
int main(){
    string num;
    cin>>num>>m;
    n=num.length();
    vector<int> q(n+1);
    for(int i=0;i<n;i++)
        q[i+1]=num[i]-'0';
    //需要保留的总数 当前坐标
    int rest=n-m,t=1;
    bool flag=0;
    while(rest--){
        int p=t;
        //保证剩余的数还够
        for(int i=t;i<=t+m;++i)
            if(q[p]>q[i]) p=i;
        //已经不用考虑前导0
        if(q[p]) flag=1;
        if(flag) cout<<q[p];
        m-=p-t; t=p+1;
    }
    if(!flag)cout<<0;
    return 0;
}
```

```
Test 1 Passed

Input 1:
175438 4

Expected Output :
13

Obtained Output :
13
```

- 代码

删数：设字符串为s，长度len，删m个

① $len \le m$ 0

② $len > m$ 等价于保留 $rest = len - m$ 个数，
使得保留数构成的数最小

$s_1\ s_2 \cdots s_{len}$

考虑：每次挑一个数保留，一共挑 rest 次，
每次在当前未删除下标区 $[t, t+m]$ 之间挑
最小数保留，因为位数最高的保留
保留数为k
~~对于~~ 对于第x次保留操作 ~~得到~~
设最优数为ans
则 $ans_0\ ans_1 \cdots ans_x \cdots ans_{rest-1} = Ans$
$ans_0\ ans_1 \cdots k \cdots ans_{rest-1} = Now$
显然：$ans_0 \sim ans_{x-1}$ 与 $ans_{x+1} \cdots ans_{rest-1}$
不影响最优性，将k替换为 $y \in [S_t, S_{t+m}]$
且 $y \ne k$ 则得到新串 $Now' > Now$
∴ 证：Now为当前可选的最优结果
对于每个位置均成立