Ch5

1.(1)贪心策略

- 由于要求元素之和不小于 W，每次选择 S 中剩余的最大元素加入 S’，直到元素和恰好满足条件。

(2)引理 1(贪心选择性)

- 集合 S={1,2,…,n}是 n 个正整数元素集合，假设已经按照降序排列，即 S[1]>S[2]>···>S[n]，则优化解 S’一定包含元素 S[1]。
- 证明:设 A 是 S 的一个优化解，设其第一个元素下标为 k，第二个元素下标为 j。如果 k=1，引理成立。如果 k≠1，令 B=A-{S[k]}∪{S[1]}。因为 S[1]⩾S[k]，则|B|⩽|A|，B 也一定是 S 的优化解，与 k≠1 矛盾。

引理 2(优化子结构)

- 集合 S={1,2,…,n}是 n 个正整数元素集合，假设已经按照降序排列，即 S[1]>S[2]>···>S[n]。设 S’是 S 的一个优化解且包含元素 S[1]，则 A=S’-{S[1]}是 T={i∈S|A[i]⩽S[1]}对于原问题的优化解。
- 证明:显然，优化解 A 中元素是 T 的所有子集合中最少的。设不然，存在原问题对于 T 的一个优化解 B，满足|B|<|A|。令 B’={S[1]}∪B，则 B’是 S 的一个优化解。由于|S′|=|A|+1，|B′|=|B|+1＜|A|+1=|S′|，与 S’是优化解矛盾。

(3)算法 GreedyMinElements(S,W)

输入 :正整数集合 S [1:n]，正整数 W

输出 :S 的子集合 S’，其中元素之和不小于 W，且 S’是满足这个条件的子集合中包含元素数量最少的

```
Sort(S)
S’←{S[1]}
j←1
For i=2 TO n DO
    If S[i]⩽S[j]
        Then S’←S’∪{S[i]}, j←i
return S’
```

排序集合 S 的操作次数为 $O(n\log n)$，贪心扩充 S’的操作次数为 $O(n-1)$，所以算法总的时间复杂度为 $O(n\log n)$。

2.(1)贪心策略

- 每次选择这样的区间[a,b]：a 小于等于未覆盖区间的下界，而 b 尽可能大。

(2)引理 1(贪心选择性)

- 假设 n 个区间[a_i,b_i]已按照下界升序排序，即 $a_1<a_2<\cdots<a_n$,则优化解 S 一定包含[1,a_i]。其中，a_i 是下界为 1 的区间中最大的上界值。
- 证明:设 S’是题设问题的一个优化解，它包含[1,a_i']，$a_i'\leqslant a_i$。令 A=S’-{[1,a_i']}∪{[1,a_i]}，则|A|=|S′|。所以 A 是一个优化解，且包含[1,a_i]。

引理 2(优化子结构)

- 假设 n 个区间[a_i,b_i]已按照下界升序排序，即 $a_1<a_2<\cdots<a_n$。设优化解 S 已经包含[1,a_i]，则 S’=S-{[1,a_i]}是区间[a_i,n]对于原问题的优化解。
- 证明:显然，优化解 S’是覆盖区间[a_i,n]中个数最少的。设不然，存在原问题对于区间[a_i,n]的一个优化解 A，满足|A|<|S′|。令 B=A∪{[1,a_i]},则 B 是原问题的一个优化解。由于|S|=|S′|+1，|B|=|A|+1＜|S′|+1=|S|，与 S 是优化解矛盾。

(3)算法 GreedyMinInterval(N)

输入 :数组 N [n][2]，N[n][0]表示区间的下界，N[n][1]表示区间的上界

输出 :最少的区间，要求覆盖[1,n]

Sort(N),按照 N[:][0]升序排序
S←{[1,a_i]}, a_i 是下界为 1 的区间中最大的上界值
k←1, min←a_i, max←0
While min<n //当未覆盖区间的下界仍小于 n
For i=k **TO** n **DO**
If N[i][0]≤min //当区间的下界小于未覆盖区间的下界
Then max←N[i][1] //找到下界小于未覆盖区间的最大上界值
Else Then
$temp$←N[][1].at(max) //找到下界小于未覆盖区间，上界又最大的索引
S←S∪{[N[$temp$][0], N[$temp$][1]]}, k←i, min←max
return S

排序二维数组 N 的操作次数为 $O(n\log n)$，对 S 赋初值{[1,a_i]}的操作次数至多为 $O(n-1)$，贪心扩充 S 的操作次数为 $O(n^2)$，所以算法总的时间复杂度为 $O(n^2)$。

3.(1)贪心策略

- 将数组 A 和 B 均按照升序排序，A 中第 i 个值与 B 中第 i 个值对应进行匹配。

(2)引理 1(贪心选择性)

- A 中最小的元素一定与 B 中最小的元素匹配。
- 证明:设 A 和 B 已经排好序，A[i]是数组 A 中最小元素，B[i]是与之对应的 B 中最小元素。若 A[i]与 B[i]匹配，A[j]与 B[j]匹配，则两对元素的绝对值之差 abs=|A[i]-B[i]|+|A[j]-B[j]|。若 A[i]不与 B[i]匹配，而与 B[j]匹配，则涉及到的 2 对元素的绝对值之差 abs'=|A[i]-B[j]|+|A[j]-B[i]|。下面分情况讨论:
  ①若 A[i]<A[j]<B[i]<B[j]，则 abs=abs'
  ②若 A[i]<B[i]<A[j]<B[j]，则 abs<abs'
  ③若 A[i]<B[i]<B[j]<A[j]，则 abs<abs'
  ④若 B[i]<A[i]<B[j]<A[j]，则 abs<abs'
  ⑤若 B[i]<B[j]<A[i]<A[j]，则 abs=abs'
  ⑥若 B[i]<A[i]<A[j]<B[j]，则 abs<abs'
  综上，abs≤abs'。如果不按贪心策略选取匹配规则，则绝对值之差不是最小的。

引理 2(优化子结构)

- 已得整体的优化解为 abs，如果同时去除 A 和 B 中最小的两个元素 m 和 n，则 abs'=abs-|m-n|是 A-{m}和 B-{n}对于题设的优化解。
- 证明:假设 abs'不是 A-{m}和 B-{n}对于题设的优化解，而 C 才是。那么，优化解 C'=C+|m-n|是原问题的优化解，与 abs 是整体优化解矛盾。

(3)算法 GreedyMinAbsSum(N)

- 输入 :数组 A[1:n],数组 B[1:n]
- 输出 :A,B 的配对方案，使得总体平均绝对差值最小

A'=Sort(A),B'=Sort(B) //分别按照升序排列
For i=1 **TO** n **DO**
S←{[A.at(A'[i]),B.at(B'[i])]} //将 A 和 B 中第 i 小的元素的索引按二元组加入 S
return S

排序数组 A 和 B 的操作次数为 $O(n\log n)$，贪心扩充 S 的操作次数为 $O(n)$，所以算法总的时

间复杂度为 $O(n\log n)$。

4.(1)贪心策略

- 将$[p_1,q_1],\ldots,[p_m,q_m]$按照 p 的值升序排序，每次选择 p 在未覆盖区间中值最小的。

(2)引理 1(贪心选择性)

- 假设 n 个区间$[p_m,q_m]$已按照 p 值升序排序，即 $p_1<p_2<\cdots<p_n$,则优化解 S 一定包含$[p_1, q_1]$。
- 证明:假设优化解 S'=的第一个区间是$[p_1', q_1']$，它不包含$[p_1, q_1]$，且 $p_1'\geqslant p_1$。构造 A=S'-$\{[p_1', q_1']\}\cup\{[p_1, q_1]\}$，则|A|=|S|。所以 A 是一个优化解，且包含$[p_1, q_1]$。

引理 2(优化子结构)

- 假设 n 个区间$[p_m,q_m]$已按照 p 值升序排序，即 $p_1<p_2<\cdots< p_n$,S 是原问题的优化解。则 S'=S-$\{[p_1, q_1]\}$是 n-1 个区间$[p_2,q_2],\ldots,[p_m,q_m]$对于题设问题的优化解。
- 证明:显然优化解 S'是 n-1 个区间$[p_2,q_2],\ldots,[p_m,q_m]$中对于题设问题独立冗余度最小的。若不然，存在优化解 A，且|A|<|S|。令 A'=A$\cup\{[p_1, q_1]\}$，则|A'|=|A|+1<|S'|+1=|S|，与 S 是优化解矛盾。

(3)算法 GreedyMaxInterval(A)

- 输入 :数组 A[1:n][2] //A[:][0]和 A[:][1]分别代表了区间的上界和下界。
- 输出 :互不相交区间的最大个数，即 x 的独立冗余度。

```
Sort(A[:][0]) //按照区间下界升序排列
S←1, j←1
For i=2 TO n DO
    If A[i][0]⩾A[j][1]
        Then S←S+1, j←i
return S
```

排序数组 A 的操作次数为 $O(n\log n)$，贪心扩充 S 的操作次数为 $O(n-1)$，所以算法总的时间复杂度为 $O(n\log n)$。

5.(1)贪心策略

- 每次选择 A 和 B 中分别最大的元素组成映射。

(2)引理 1(贪心选择性)

- 优化解 S 一定包含 A 和 B 中分别最大的元素 m, n 组成的映射$\{m, n\}$。
- 证明:假设 A 中第 m 大的元素与 B 中第 n 大的对应，第 a 大的元素与 B 中第 b 大的对应，且 A[m]>A[a]，B[n]>B[b]。如果有优化解 S'不包含映射$\{m, n\}$，它的代价是 m^b+a^n，而优化解 S 的代价是 m^n+a^b。$m^n+a^b>m^b+a^n\rightarrow m^n-m^b>a^n-a^b$，而 $a=m+t$。可以通过二项式定理展开证得 $m^n-m^b>a^n-a^b$。故优化解 S'不包含映射$\{m, n\}$矛盾。

引理 2(优化子结构)

- 假设 A 和 B 已经降序排列，且 S 是原问题的优化解，则 S'=S-$\{m, n\}$是 A[2:n]和 B[2 :n]对于题设的优化解。
- 证明:若 S'不是 A[2:n]和 B[2:n]对于题设代价最大的映射集，存在优化解 T，它的代价大于 S'的代价。构造 T'=T$\cup\{m, n\}$，则 T'的代价大于 S'$\cup\{m, n\}$的代价，即 T'的代价大于 S 的代价，与 S 是原问题的优化解矛盾。

(3)算法 GreedyMaxMap(A,B)

- 输入 :正整数集 A[1:n], B[1:n]
- 输出 :A 到 B 代价最大的一一映射

```
A'=Sort(A),B'=Sort(B) //分别按照降序排列
```

```
For i=1 TO n DO
    S←{[A.at(A'[i]),B.at(B'[i])]} //将 A 和 B 中第 i 大的元素的索引按二元组加入 S
return S
```

排序数组 A 的操作次数为 $O(n\log n)$，贪心扩充 S 的操作次数为 $O(n)$，所以算法总的时间复杂度为 $O(n\log n)$。