第四章　155103163　宋博宇.

参数 g(i.j) (i≥j.)

4.1 (1) ① 当划分中包含j. 即{j. x_1. x_2. x_3 … x_k} 其中{x_1. x_2. x_3 … x_k}和为i-j.

其中最大值 $x_{max} \le j$. 因此是(n-j)的j划分. 划分个数为 g(i-j,j)

② 当划分中不包含j. 则划分中所有值都小于j. 即 $x_k \le j-1$ 即 i的 j-1划分.

划分的个数为 g(i. j-1)

所求拆分个数为以上两种情况之和. 即 g(i.j) = g(i. j-1) + g(i-j. j)

(2) 输入: 正整数n. ~~[illegible]~~.

输出: 数组 M[0:n][0:n]. ~~[illegible]~~ (拆分个数) m

M[i][i]表示拆分整数i时最大加数 ≤ i的方案个数

```
for i=0 to n
    M[i][0] = 0
    M[0][i] = 0
for i=1 to n
    for j=1 to n
        if i<j
            M[i][j] = M[i.i]
        else if i=j
            M[i][j] = M[i. j-1] + 1
        else
            M[i][j] = M[i][j-1] + M[i-j][j]
m = M[n][n]
```

① 当 i<j 时 g(i.j) = g(i.i)

例: g(2.3) = g(2.2)

② 当 i=j. a. 划分中包含i 只有一种 即{i}

b. 划分中不包含i. 则划分中所有值 < i

即 ≤ i-1　g(i. i-1)

不超过i的划分总数为 g(i. j-1) + 1.

③ 当 i>j. 结果如题目(1).

4.3 (1). X = acabdc

Y = abcdabdc

Z = acbeabdc

LCS(Y. Z) = ababdc

LCS(X. LCS(Y. Z)) = LCS(X. ababdc) = aabdc

LCS(X. Y. Z) = acabdc

所以 LCS(X. Y. Z) ≠ LCS(X LCS(Y. Z))

(2) 三个字符串 X, Y, Z. L[a][b][c]代表 X_a, Y_b, Z_c 的最长公共子序列.
长度分别为 n.m.o.

当 X[a] = Y[b] = Z[c].　L(a.b.c) = ~~max{~~ L(a-1. b-1. c-1) + 1.

~~否则 L(a.b.c) = max{~~

). 当 X[a] = Y[b] 且 X[a] ≠ Z[c].　L(a.b.c) = max{L(a-1. b-1. c), L(a.b. c-1)}.

) 当 X[a] = Z[c] 且 X[a] ≠ Y[b].　L(a.b.c) = max{L(a-1. b. c-1). L(a. b-1. c)}

) 当 Y[b] = Z[c] 且 Y[b] ≠ X[a]　L(a.b.c) = max{L(a. b-1. c-1). L(a-1. b.c)}

) 否则　L(a.b.c) = max{L(a. b. c-1). L(a. b-1. c). L(a-1. b. c)}.

时间复杂度为 O(mno).

$$C[i]=\begin{cases}0, & (i=0)\\ \max_{0\le j<i}(C[j]+1) & (i>0 \text{且} A[j]\le A[i])\end{cases}$$

计算代价：

```
for i=0 to n-1
    B[i]= i
for i=1 to n-1
    C[i]=1
    for j=0 to i-1
        if (A[j] ≤ A[i] and
            C[j]+1 > C[i])
            C[i]=C[j]+1
            B[i]=j
return max_{0≤i<n} {C[i]}
```

构造最优解：

设k为max{C[i]}下标.

```
path[0]=A[k]    i=1
while (B[k] != k)
    path[i++] = A[B[k]]
for i=max-1 to 0
    输出 path[i]
```

4.5 · 与上题相似，设置 A[j] ≤ A[i] 为元素之差绝对值不超过 d.

```
for i=0 to n-1
    B[i]=i
for i=1 to n-1
    C[i]=1
    for j=0 to i-1
        if |A[i]-A[j]| ≤ d and C[j]+1 > C[i]
            C[i]=C[j]+1
            B[i]=j
return max_{0≤i<n} {C[i]}
```

---

6 · 令mij表示 $a_1 a_2 \cdots a_n$ 合并的最大代价.

$$m_{ij}=\begin{cases}0, & i=j\\ \max_{i\le k<j}\{m_{ik}+m_{k+1,j}\}+\sum_{n=i}^{j}a_n\end{cases}$$

输入：整数序列 $a_1,a_2,\cdots,a_n$，存储于数组 A[1:n]

输出：代价数组 M[1:n][1:n]，结构信息数组 S[1:n][1:n]

```
for i=1 to n
    M[i,i]=0
for L=2 to n
    for i=1 to n
        j=i+L-1
        M[i,j] = ~~0~~ 0
        for k=i to j-1
            q = M[i,k]+M[k+1,j]+sum(A[i:j])
            if q > M[i,j]
                M[i,j] = q
                S[i,j] = k
输出 M 和 S.
```

填满对角线右侧部分) 总 时间复杂度 $O(n^2)$

构造最优解：

```
i=1, j=n
BuildPath (S, i, j)
    if i=j
        输出 A[i]
    输出(
    BuildPath (S, i, S[i][j]
    BuildPath (S, S[i][j]+1, j
    输出)
```

→ O(nlo

7 (1). 令 M_{ij} 代表 $a_1 O_1 a_2 O_2 \cdots O_{n-1} a_n$ 的最大值， m_{ij} 代表最小值

$$M_{ij} = \begin{cases} a_i, & i = j \\ \max_{i \le k < j} \{ \max \{ M_{ik} O_k M_{k+1j} \}, M_{ik} O_k m_{k+1j} \} \} & \end{cases}$$

~~算法步骤与5.6类似，步骤~~

输入：整数序列 $a_1 a_2 \cdots a_n$ 存储在数组 A[1:n]，~~操作数组 O[1:n-1]~~ ✓

输出：~~操作数组~~ 最大值数组 M[1:n][1:n]，最小值数组 m[1:n][1:n]

结构信息数组 S[1:n][1:n]

```
1.  for i=1 to n
2.      M[i,i]=A[i]
3.      m[i,i]=A[i]
4.  for l=2 to n
5.      for i=1 to n
6.          j=i+l-1
7.          M[i,j]=0
8.          m[i,j]=+∞
9.          for k=i to j-1
10.             p = M[i,k] O_k M[k+1,j]
11.             q = M[i,k] O_k m[k+1,j]
~~12~~          ~~M = max{p, q}~~
12.             ~~M[i,j]~~ = max{p, q}
                if M > M[i,j], M[i,j] = M
13.             S[i,j] = k
14.             p' = m[i,k] O_k m[k+1,j]
15.             q' = m[i,k] O_k M[k+1,j]
16.             ~~m[i,j]~~ = min{p', q'}
                if m' < m[i,j], m[i,j] = m'
17. 输出 M, S
```

构造最优解过程与5.6类似

时间复杂度 T = O(n²)　~~两次~~填满矩阵对角线右侧

---

(2). 增加一个~~数组 I[1:n][1:n]~~ 记录 0 ≤ i < 1 情况

10-16 步骤改为：

~~算法步骤变化：~~

```
~~p = M[i,k] O_k M[k+1,j]~~ ✓
~~q = M[i,k] O_k m[k+1,j]~~ ✓
~~for a=1 to len(I[k+1,j])~~
~~    i = M[i,k] O_k I[k+1,j,a]~~
~~    w = max{p, q, i}~~
~~if w > M[i,j]  M[i,j] = w~~
~~p' = m[i,k] O_k m[k+1,j]~~
~~q' = m[i,k] O_k M[k+1,j]~~
~~for a=1 to len(I[k+1,j])~~
~~    i = M[i,k] O_k I[k+1,j,a]~~
~~    w' = min{p, q, i}~~
~~if w' < m[i,j]  m[i,j] = w'~~

~~for a=1 to len(P[k+1,j])~~
~~    x = M[i,k] O_k P[k+1,j,a]~~
~~    P[i,j] += x~~
~~    if~~
```

~~增加数组 P[三维] 数组~~　~~(i<j)~~

增加 ~~数组~~ P[i,j,n] 代表 A[i:j] 所有可能的值

~~且当 i=j 时，P[i,j] = {A[i]}~~

```
~~for a=1 to len(P[i,k])~~
~~    x = M[i,k] O_k P[k+1,j,a]~~
~~    for b=1 to len(P[k+1,j])~~
~~        x = M[i,k] ... O_k P[k+1,j,b]~~
~~        P[i,j] += x~~
~~        if x > M[i,j]~~
~~            M[i,j] = x~~
~~            S[i,j] = x~~
```

O(x³)

```
for a=1 to len(P[i,k])
    for b=1 to len(P[k+1,j])
        x = P[i,k,a] O_k P[k+1,j,b]
        P[i,j] += x
        if x > M[i,j]
            x[i,j] = x
```
✓

4.8 设 C[i,j] 表示从 (0,0) 到 (i,j) 的最小代价. B[i,j] 表示[illegible]

则 $C[0,j] = \sum_{k=0}^{j} b_{0k}$.

$C[i,0] = \sum_{k=0}^{i} a_{0k}$.

$C[i,j] = \min(a_{i-1j} \cdot C[i-1,j],\ b_{ij-1} C[i,j-1])$

计算解代价

```
for i=0 to m.
    $C[i,0] = \sum_{k=0}^{i} a_{0k}$.
for j=0 to n
    $C[0,j] = \sum_{k=0}^{j} b_{0k}$
for i=1 to m
    for j=1 to n
        ~~$C[i,j] = \min(a_{i-1j} \cdot C[i-1,j],\ b_{ij-1} C[i,j-1])$~~
        if $a_{i-1j} C[i-1,j] < b_{ij-1} C[i,j-1]$
            $C[i,j] = a_{i-1j} C[i-1,j]$
            B[i,j] = ↓
        else
            $C[i,j] = b_{ij-1} C[i,j-1]$
            B[i,j] = ←.
输出 C.B.
```

~~构造最优解~~: 构造~~最~~优解: i=m j=n.

```
BuildPath(B. i. j).
    if i=0 或 j=0    return
    输出(i,j)
    if B[i,j] = ↓
        BuildPath(B. i-1, j)
    else if B[i,j] = ←.
        BuildPath(B. i. j-1).    输出为反向序列.
~~输出~~
```

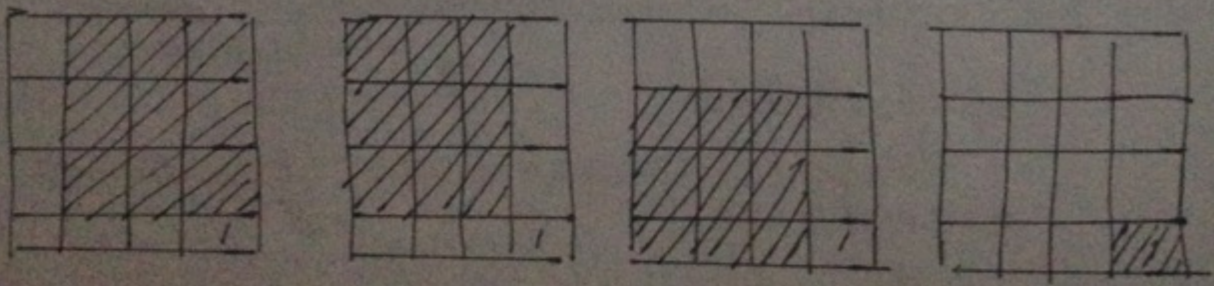
全1

4.9 任何维度的方阵都可以看做由三个低一阶维度的全1方阵与一个一维方阵组成

即方阵的~~上面~~右上、左上、左下都为 n-1 阶全1矩阵且右下角为1时.

才能构成一个n阶全1矩阵. 于是得到递推式: (V[i,j]表示当[i,j]为1时以[i,j]为右下角的全1矩阵规模)

$$\begin{cases} V[i,j] = \min\{V[i-1,j],\ V[i-1,j-1],\ V[i,j-1]\} + 1. & (i>0,\ j>0) \\ V[i,0] = 1 \\ V[0,j] = 1 \end{cases}$$



10. 此题可以转化为 假设S所有元素和为N，能否找出[illegible]

当N为奇数时，显然不存在。

当N为偶数时，可以转为类似0-1背包的算法。　　$A[0:n]$ 表示数组S。

$c[i,j]$ 表示序列中前i个数中~~某几个数能否~~是否能找出和为j的组合。

$c[0:n][0:N]$。

$c[0,j]=0$.　$(j\geq 1)$

$c[i,0]=1$.　$(i\geq 0)$

$$c[i,j]=\begin{cases} c[i-1,j], & j<A[i] \\ \max\{c[i-1,j],\ c[i-1,j-A[i]]\}, & j\geq A[i] \end{cases} \quad (i\geq 1,\ j\geq 1)$$

```
for i=0 to n
    c[i,0] = 1
for j=1 to n
    c[0,j] = 0.
for i=1 to n
    for j=1 to A[i]-1
        c[i,j] = c[i-1,j]
    for j=A[i] to n
        c[i,j] = max{ c[i-1,j], c[i-1,j-A[i]] }.
输出c.
```

若 $c[n,\frac{N}{2}]=1$ 则存在
否则不存在

4.11 (1). 首先将各点按x坐标从小到大排序 $\{P_1,P_2,P_3\cdots P_n\}$ 其中 $P_1.x<P_2.x<P_3.x<\cdots<P_n.x$. 复杂度O(nlog

(2). 优化子结构：定义从 P_i 到 P_j 路径为 $(j\geq i)$：从 P_i 开始，从右到左一直到 P_1，然后从左到右一直到 P_j，经过 P_1 到 P_j 之间所有点各一次。定义 $c[i,j]$ 为满足以上条件的最短路径。定义 $dis(i,j)$ 为点 P_i 到 P_j 的欧几里得距离。

(3). 构造递归式：

$$c[i,j]=\begin{cases} c[i,j-1]+dis(j-1,j), & 1<i<j-1 \\ \min\limits_{1\leq k<j}\{c[i,k]+dis(k,j)\}, & j\geq i\geq j-1 \end{cases} \quad (j\geq i).$$

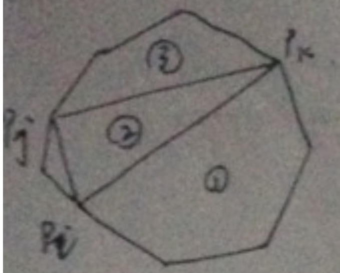
题目所求的解即为 $c[n,n]$.

```
for i=1 to n
    ~~c[i,1] = dis(i,1)~~.
~~for i=1 to n~~
    c[1,i] = dis(i,1)
for i=2 to n
    for j=i to n
        c[i,j] = +∞
        if i < j-1
            c[i,j] = c[i,j-1] + dis[j-1,j]
```

输出 $c[n,n]$.

12 凸n边形 $P_iP_{i+1}\cdots P_j$ 的最优三角形划分T包含 $P_iP_kP_j$.（$i<k<j$）.

则T的权值是由三部分权值之和



① 凸多边形 $P_iP_{i+1}\cdots P_k$ 权值

② 三角形 $P_iP_kP_j$ 周长

③ 凸多边形 $P_kP_{k+1}\cdots P_j$ 权值.

所以凸多边形三角划分问题具有优化子结构.

设 T[i,j] 表示凸多边形最优三角划分所对应的权值.（凸多边形为 $P_iP_{i+1}\cdots P_j$）的

C(i,k,j) 表示三角形 $P_iP_kP_j$

$T[i,j]=0.\qquad (i=j)$

$T[i,j]=\min_{i<k<j}\{T[i,k]+T[k,j]+C(i,k,j)\}\qquad (i<j)$

输出 T[1,n] 即为所求最优解.

4.13 首先将根节点左右划分. 左右各进行动态规划

一棵树最大权值 = max{除去子树根节点各节点最大子树权值, 包含根节点}

2 n位整数x. 各位存为A[1:n].

设c[i,j]表示第i位到第j位整数的最优解. 只考虑 i≤j 情况.

$$c[i,j]=\begin{cases} 0. & (A[i]\neq A[j],\ i<j). \\ c[i-1,j-1]+(A[i]+1) & (A[i]=A[j],\ i<j) \\ 1 & (i=j,\ 且\ i为偶). \\ 0. & (i=j,\ 且\ i为奇). \end{cases}$$

第五章 515103163 宋博宇

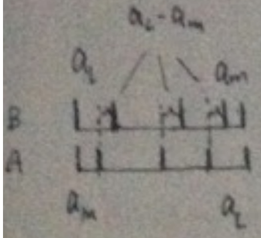
5.1 该问题可以转化为：求总等待时间最短的任务调度的贪心策略为：优先完成所需时间最短任务。

贪心选择性：设 $S=\{1,2\cdots n\}$ 是n个任务执行的序列，设 $a_m=\min_{i\le n}\{a_i\}$，证明S的最优解第一个任务为a

假设 B 是 S 的一个最优解，第一个任务为 l

若 l=m，则命题成立。

若 l≠m，令A为B中任务 l 和任务m交换位置之后的任务序列。



假设在B中任务m是第k个执行的。

因为 $a_m<a_l$，所以A中 $e_i(i\le k)$ 都比 B中 $e_i(i<k)$ 小 (a_l-a_m)。

所以B的总等待时间－A总等待时间 $=(j-1)(a_l-a_m)>0$。

这与B最优解矛盾，所以原问题具有贪心选择性。

最优子结构：设 $S=\{1,2,\ldots,n\}$ 是n个任务序列，a_i 是活动i的时间，且 $a_1\le a_2\le a_3\cdots\le a_n$。

设 A 是 S 的一个优化解且包含活动1，$A'=A-\{1\}$，证明 A' 是 $S'=\{i\in S|s_i\ge f_1\}$ 的最优解

假设S'存在一个最优解 B'，使得平均等待时间 e_b 低于A'的平均等待时间 e_a。

设 $B=B'\cup 1$，则 $B<A(A=A'\cup 1)$ 这与 A是最优解矛盾，所以原问题具有优化子结构。

```
sort(S).
for i=1 to n.
    T= T∪S[i]              T(n)= O(nlogn).
return T.
```

5.2 贪心算法：每次从面值不大于n的硬币中选择最大的硬币。

贪心选择性：设 m_i 表示面值不大于i的硬币中最大面值的硬币(S_i)个数。

设B是兑换面值n的一个最优解，且 $m_n=0$ 即最优解中不包含小于等于n的最大硬币 S_n

由于S中每种硬币都可以用面值小于它的若干硬币表示，则

设A为如下解：在B中 ~~[illegible]~~ 选择若干硬币，使其面值和为 S_n，即 $A=(B-\{S_j\})\cup S_n$。

这样A个数 < B个数 这与B是最优解矛盾，所以原问题有贪心选择性。

最优子结构：设小于i的硬币中最大面值硬币为 S_i，个数为 m_i。

当总面值为n，设A是n的一个优化解且包含 S_n，个数为K。

设 A'是 ~~[illegible]~~ 证明 A'是 $n'=n-S_i$ 的最优解。 A'硬币个数为 K−1

假设 n' 存在一个最优解B'，~~使得个数为 k_b 使得 $k_b<k_a$~~。

即其硬币个数为 m < K−1 则 m+1 < k。

令 $B=B'\cup S_i$ 则B是面值为n的问题的一个解，其硬币个数为 m+1 < k，则 B < A。

这与A是原问题最优解矛盾。

所以原问题具有优化子结构。

T(n)= O(1).

3 合并的过程类似于一个二叉树. 对任意两个序列的长度作为子节点. 合并后的新序列长度作为父节点. L(i)表示序列i的长度. 设x. y为所有序列中长度最小的两个序列. 则存在一个最优的合并树T. 使得 ~~[illegible]~~ x. y为具有最大深度的兄弟叶节点.

贪心策略: 优先选择长度最小的序列放置在二叉树最深的叶子节点.

贪心选择性: 设T为一个优化解. a. b为T最大深度兄弟叶子节点.. x. y为长度最小序列.

若 a=x. b=y. 命题成立

dep(a)表示T中a深度

若 a≠x. 交换a和x位置得到T'. 用B(T)表示树T的操作次数

$B(T) - B(T') = (L(a) - L(x))(dep(a) - dep(x)) \geq 0$.

所以 $B(T') \leq B(T)$. 所以B(T')最优解 即最优解中最大深度叶节点必有x.

类似的若 b≠y. 可得最优解中最大深度叶节点必有y.

所以 x. y为具有最大深度的兄弟叶节点. 原问题具有贪心选择性.

优化子结构: 设S'是S去掉x. y 再加上 x.y合并后序列z的序列集. L(z) = L(x)+L(y)

~~T_1'表示S'的合并树~~ ~~那么将S'中叶节点z替换为具有x. y的内部节点所得树T_1表示S上一个最优化解~~

~~$B(T_1') = B(T_1) - L(x) - L(y)$~~

T_1表示S上x. y具有最大深度的兄弟叶节点的最优化解

T_1'为T_1去掉x. y 将x. y父结点替换为z的合并树. 只需证明T_1'是S'上最优的

显然 $T_1' \subseteq S'$. 若T_1'不是最优化解 则S'必存在一棵树T_2'使得 $B(T_2') < B(T_1')$

令T_2为T_2' ~~[illegible]~~ z替换为x. y父结点. 则 $T_2 \subseteq S$.

$B(T_2) = B(T_2') + L(x) + L(y) < B(T_1') + L(x) + L(y) = B(T_1)$

即 $B(T_2) < B(T_1)$ 与T_1为最优解矛盾. 所以 T_1'为S'上的最优解

所以原问题具有优化子结构.

```
BuildTheap(S. n)
  ~~for i=1 to n~~ . Q=S
    z = node().
    x = left(z) = min(Q)
    y = left(z) = min(Q)
    L(z)=L(x)+L(y)
  ✓ ~~delete(Q, x)~~
  ✓ ~~delete~~(Q.y)   ~~[illegible]~~).
    insert(Q.z).
return Q.top.
```

构建堆过程时间复杂度为O(n)

堆操作时间复杂度 O(logn)

共需要执行n-1次.

所以总复杂度为O(nlogn)

按照排序结果，按层构建，即每次取出长度最小的 code，长度为 n，放置在第 n 层，父结点为 code-1 的

复杂性：$O(n\log n)+O(n)=O(n\log n)$.

5.5 贪心策略：每次选取集合中最大的两个整数 a、b 做映射，即可获得最大映射.

只需证明：假定 $a_n \ge a_m$，$b_n \ge b_m$，则有 $a_n^{b_n}+a_m^{b_m} \ge a_n^{b_m}+a_m^{b_n}$

$$\text{原式}=(a_n^{b_n}+a_m^{b_m})-(a_n^{b_m}+a_m^{b_n})$$
$$=(a_n^{b_n}-a_n^{b_m})-(a_m^{b_n}-a_m^{b_m})$$

~~$f(x)=x^{k}\ (k>0)$ 导数大于0 递增，则~~

$f(x)=x^{b_n}-x^{b_m}$ $(x\ge 1)$. $f'(x)=b_n x^{b_n-1}-b_m x^{b_m-1}>0$.

所以 $f(x)$ 递增 所以原式 ≥ 0.

所以 $a_n^{b_n}+a_m^{b_m}\ge a_n^{b_m}+a_m^{b_n}$.

伪代码如下：

```
sort (A.1.n)
sort (B.1.n)
r=0
for i=1 to n
    r += A[i]^B[i]
return r.
```

5.6 贪心策略：在集合 $\{(p_i,q_i)\mid p_i\ge q_{i-1}\}$ 中选择最小的 p_i

```
sort (X).
set = X[1]
for i=2 to m
    if X[i].p ≥ X[i-1].q
        set = set ∪ X[i]
return set.
```

5.7 贪心策略：计算 $k_i=\frac{v_i}{w_i}$，~~将[illegible]~~ 每次选取 k_i 最大的物品 i 装入背包 C

若~~[illegible]~~ $w_i>C$ 则装入 $x_i=\frac{C}{w_i}$.

若将 i 装入 C 后背包物品总重量 $<C$ 则选择 k 次高的物品进行以上策略

直到背包装满.

贪心选择性：按 k 排序：$k_1>k_2>k_3>\cdots>k_n$，证明第一次能尽可能多装入物品 1，即 $x_1=\min\{1,\frac{C}{w_1}\}$

① 若 $w_1\ge C$，$\frac{C}{w_1}\le 1$，设 B 为一个最优解，其中 $x_1<\frac{C}{w_1}$，则剩余 $C-x_1w_1>0$ ~~[illegible]~~.
则剩余需用 $x_2\cdots n$ 填充，则单位 $k<k_1$，则存在 A 为 $x_1=\frac{C}{w_1}$，即 $C=x_1w_1$ 全部填充1，使得 $A>B$，所以 $x_1=\frac{C}{w}$

② 若 $w_1<C$，$\frac{C}{w_1}>1$，设 B 为一个最优解，其中 $x_1<1$，则剩余 $C-x_1w_1>0$.
则 ~~[illegible]~~ $(1-x_1)w_1$ 部分的单位 $k<k_1$，则存在 A 为 $x_1=1$，使得 $A>B$ 所以 $x_1=1$

所以本问题具有贪心选择性.

最优子结构：去掉 1 后问题变为 $2\cdots n$ 个物品，背包容量为 $C-W_1$，~~[illegible]~~ A 为最优解
$A'=A-\{1\}$ ~~若 $A'=A-1$ 不是最优解~~，设 B' 是 $C-W_1$ 最优解，即 $B'>A'$，则 $B=B'\cup\{1\}>A'\cup\{1\}$

贪心策略：在P中找到 x 最大的点，在Q中找到满足 $x_p > x_q$ 且纵轴满足 $y_p > y_q$.

5.[illegible] 贪心策略：每个区间按右端点递~~增~~减排序，$b_1 > b_2 > b_3 \cdots > b_n$，

~~每次选~~ k 表示已覆盖区域右端点，初始时 $k=0$.

每次选取 $a \le k$ 中 b 最大的区间 i，并置 $k=b_i$

贪心选择性：需证明在最优化解 A 中，$1 \in A$.

设 B 是一个优化解，m 是 B 中 b 最大区间

若 $m=1$，~~显然~~成立

若 $m \neq 1$，令 $A=(B-\{m\})\cup\{1\}$ ~~第一次选择~~ 由于 $b_1 > b_m$

所以在 m 能覆盖到的区域，1 也能被覆盖到，所以 A 是一个

又因为 $|A|=|B|$ 所以 A 也是一个优化解

所以原问题具有贪心选择性

优化子结构：设 A 是一个区间集合 S 的满足 $1 \in A$ 的最小区间覆盖解.

~~假~~设 $A'=A-\{1\}$ ~~S' 为区间集~~ $S'=\{i \in S \mid a_i < b_1\}$.

显然 A' 是覆盖区间 $[b_1, n]$ 的，且 $A' \subseteq S'$

设 S' 存在一个最优解 B' 使得 $|B'|<|A'|$，且覆盖 $[b_1, n]$

令 $B=\{1\}\cup B'$ ~~对于~~ 由于 B' 中元素覆盖 $[b_1, n]$

所以 B 覆盖 $[1:n]$，由 $|A'|=|A|-1$ 可知

$|B| = |B'|+1 < |A'|+1 = |A|$. 即 $|B| < |A|$

这与 A 是 S 中最小区间矛盾

所以原问题具有优化子结构.

5.[illegible] 对 b_i 排序，选择截止时间较小的，订单数较大的