

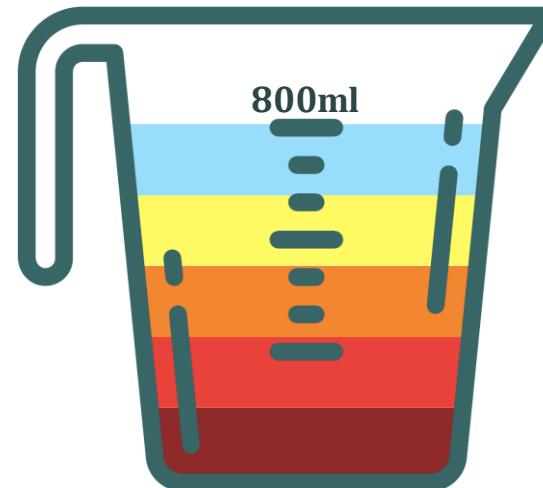
# 贪心策略篇：部分背包问题

# 问题背景

## ● 调制饮品比赛

- 参赛者拥有容量为800ml的杯子，可任选不超过体积上限的饮料进行混合
- 调制饮品价格为各所使用饮料的价格之和，所得饮品价格之和最高者获胜

| 饮料  | 价格(元) | 体积(ml) |
|-----|-------|--------|
| 苏打水 | 60    | 600    |
| 汽水  | 10    | 250    |
| 橙汁  | 36    | 200    |
| 苹果汁 | 16    | 100    |
| 西瓜汁 | 45    | 300    |



问题：如何使调制的饮品价格最高？



## 部分背包问题

### Fractional Knapsack Problem

输入

- $n$ 个物品组成的集合 $O$ , 每个物品有两个属性 $v_i$ 和 $p_i$  , 分别表示体积和价格
- 背包容量为 $C$

输出

- 求解一个解决方案 $S = \{x_i | 1 \leq i \leq n, 0 \leq x_i \leq 1\}$ , 使得:

选取物品的比例

$$\max \sum_{x_i \in S} x_i \cdot p_i \quad \text{优化目标}$$

$$s.t. \sum_{x_i \in S} x_i \cdot v_i \leq C \quad \text{约束条件}$$



## 部分背包问题

### Fractional Knapsack Problem

输入

- $n$ 个物品组成的集合 $O$ , 每个物品有两个属性 $v_i$ 和 $p_i$ , 分别表示体积和价格
- 背包容量为 $C$

输出

- 求解一个解决方案 $S = \{x_i | 1 \leq i \leq n, 0 \leq x_i \leq 1\}$ , 使得:

选取物品的比例

$x_i$ 只能取0或1时  
变为0-1背包问题

$$\max \sum_{x_i \in S} x_i \cdot p_i$$

优化目标

$$s.t. \sum_{x_i \in S} x_i \cdot v_i \leq C$$

约束条件

- **最高性价比优先**

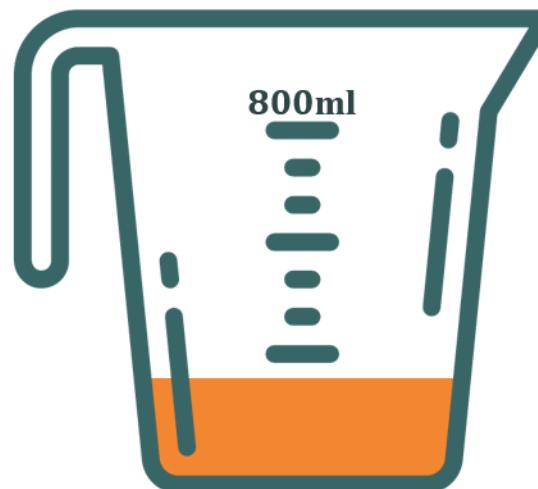
- 性价比 = 价格/体积
- 优先选择**高性价比**饮料全部装入，尽可能装满杯子

| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|---|-------|--------|-----------|
|  苏打水   | 60    | 600    | 0.10      |
|  汽水    | 10    | 250    | 0.04      |
|  橙汁   | 36    | 200    | 0.18      |
|  苹果汁 | 16    | 100    | 0.16      |
|  西瓜汁 | 45    | 300    | 0.15      |

# 算法实例

- 最高性价比优先
- 解决方案

| 饮料 | 价格(元) | 体积(ml) | 总价格(元) |
|----|-------|--------|--------|
| 橙汁 | 36    | 200    |        |



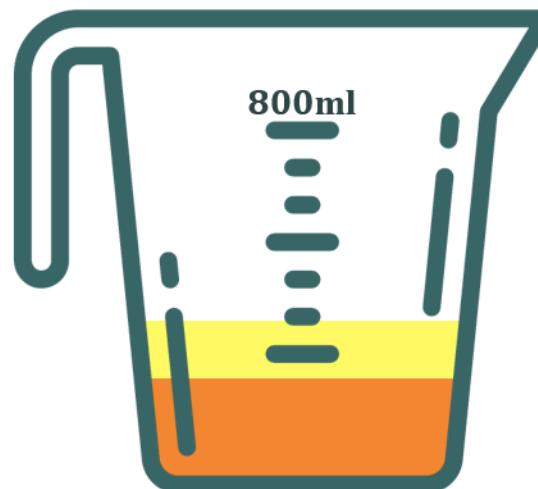
| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|-----|-------|--------|-----------|
| 橙汁  | 36    | 200    | 0.18      |
| 苹果汁 | 16    | 100    | 0.16      |
| 西瓜汁 | 45    | 300    | 0.15      |
| 苏打水 | 60    | 600    | 0.10      |
| 汽水  | 10    | 250    | 0.04      |

# 算法实例

## ● 最高性价比优先

### • 解决方案

| 饮料  | 价格(元) | 体积(ml) | 总价格(元) |
|-----|-------|--------|--------|
| 橙汁  | 36    | 200    |        |
| 苹果汁 | 16    | 100    |        |



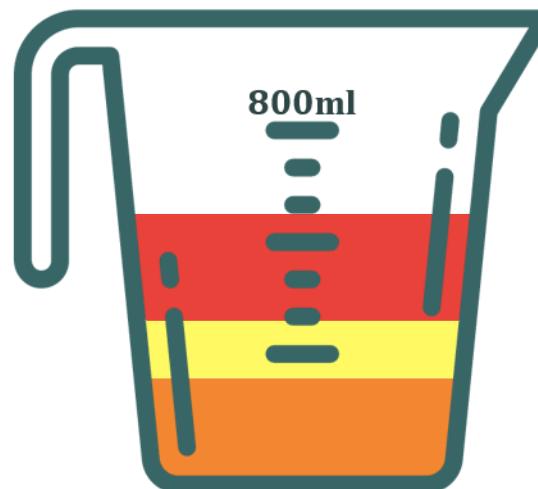
| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|-----|-------|--------|-----------|
| 橙汁  | 36    | 200    | 0.18      |
| 苹果汁 | 16    | 100    | 0.16      |
| 西瓜汁 | 45    | 300    | 0.15      |
| 苏打水 | 60    | 600    | 0.10      |
| 汽水  | 10    | 250    | 0.04      |

# 算法实例

## ● 最高性价比优先

### • 解决方案

| 饮料  | 价格(元) | 体积(ml) | 总价格(元) |
|-----|-------|--------|--------|
| 橙汁  | 36    | 200    |        |
| 苹果汁 | 16    | 100    |        |
| 西瓜汁 | 45    | 300    |        |



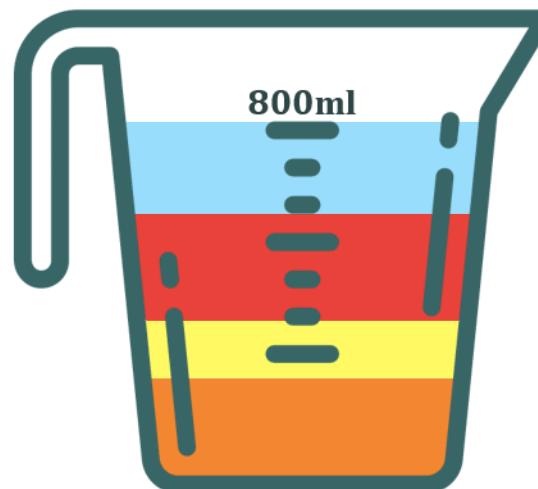
| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|-----|-------|--------|-----------|
| 橙汁  | 36    | 200    | 0.18      |
| 苹果汁 | 16    | 100    | 0.16      |
| 西瓜汁 | 45    | 300    | 0.15      |
| 苏打水 | 60    | 600    | 0.10      |
| 汽水  | 10    | 250    | 0.04      |

# 算法实例

- 最高性价比优先

- 解决方案

| 饮料  | 价格(元) | 体积(ml) | 总价格(元) |
|-----|-------|--------|--------|
| 橙汁  | 36    | 200    |        |
| 苹果汁 | 16    | 100    |        |
| 西瓜汁 | 45    | 300    |        |
| 苏打水 | 20    | 200    |        |



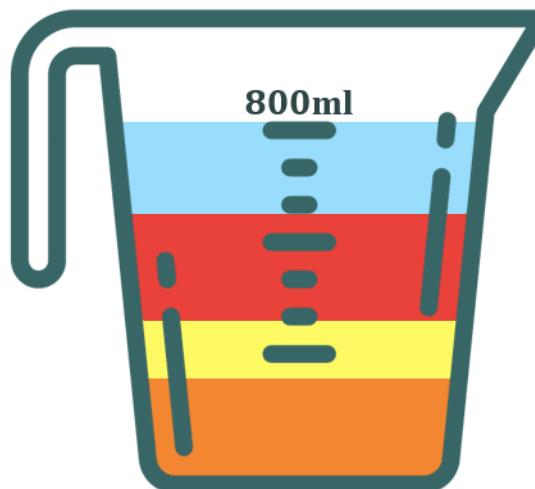
| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|-----|-------|--------|-----------|
| 橙汁  | 36    | 200    | 0.18      |
| 苹果汁 | 16    | 100    | 0.16      |
| 西瓜汁 | 45    | 300    | 0.15      |
| 苏打水 | 20    | 600    | 0.10      |
| 汽水  | 10    | 250    | 0.04      |

# 算法实例

- 最高性价比优先

- 解决方案

| 饮料  | 价格(元) | 体积(ml) | 总价格(元) |
|-----|-------|--------|--------|
| 橙汁  | 36    | 200    | 117    |
| 苹果汁 | 16    | 100    |        |
| 西瓜汁 | 45    | 300    |        |
| 苏打水 | 20    | 200    |        |

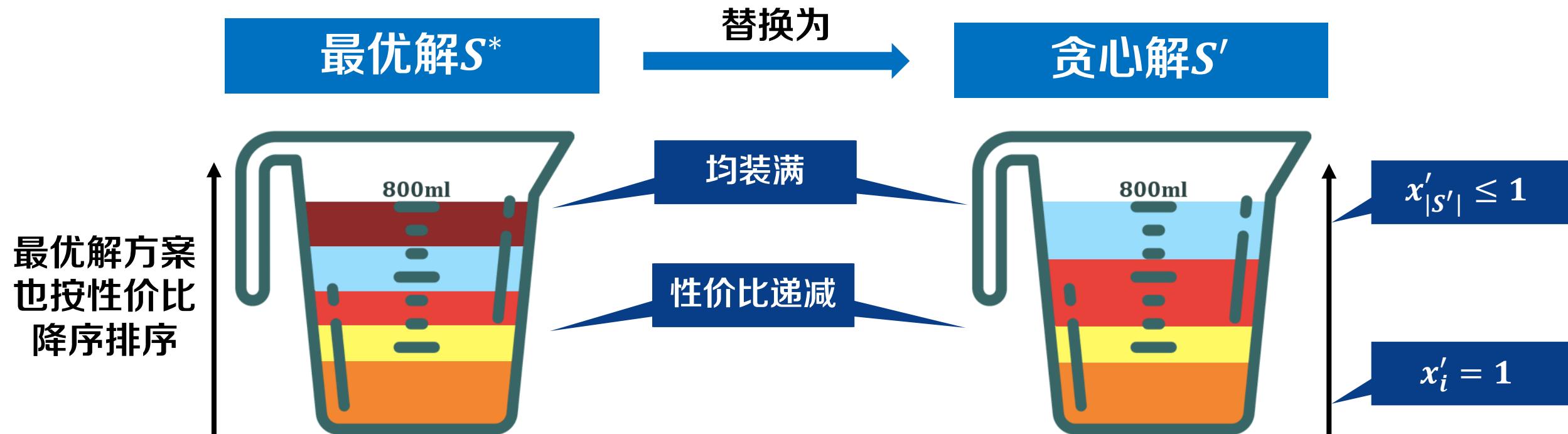


| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|-----|-------|--------|-----------|
| 橙汁  | 36    | 200    | 0.18      |
| 苹果汁 | 16    | 100    | 0.16      |
| 西瓜汁 | 45    | 300    | 0.15      |
| 苏打水 | 60    | 600    | 0.10      |
| 汽水  | 10    | 250    | 0.04      |

# 正确性证明



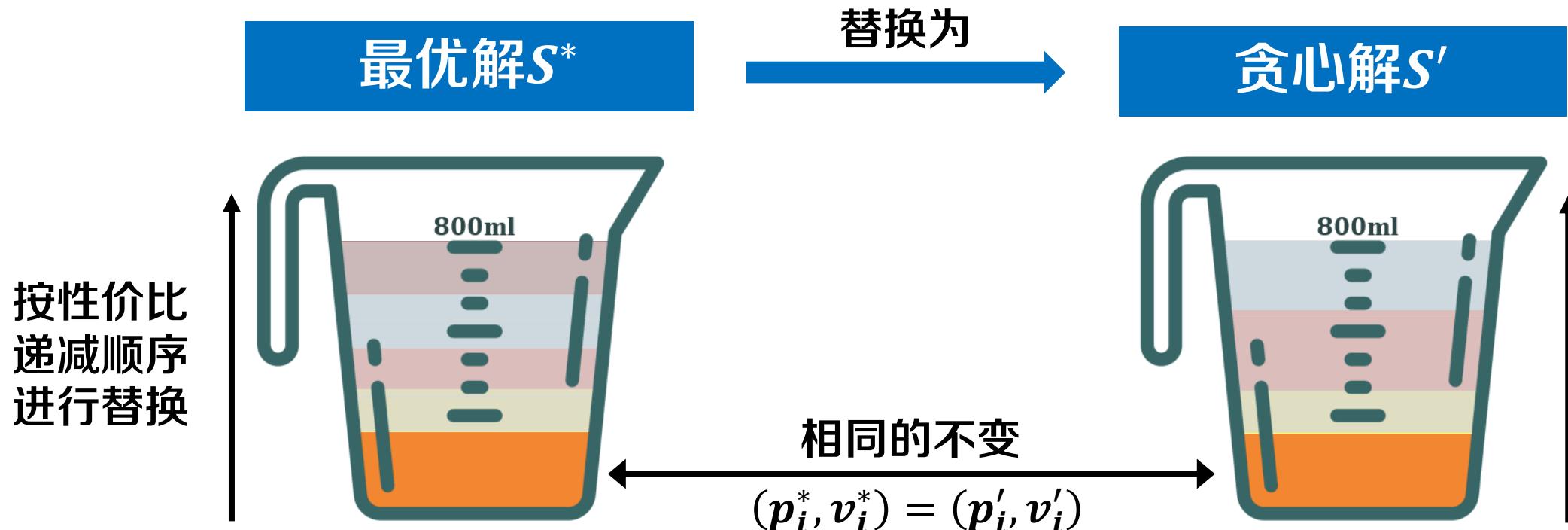
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



# 正确性证明

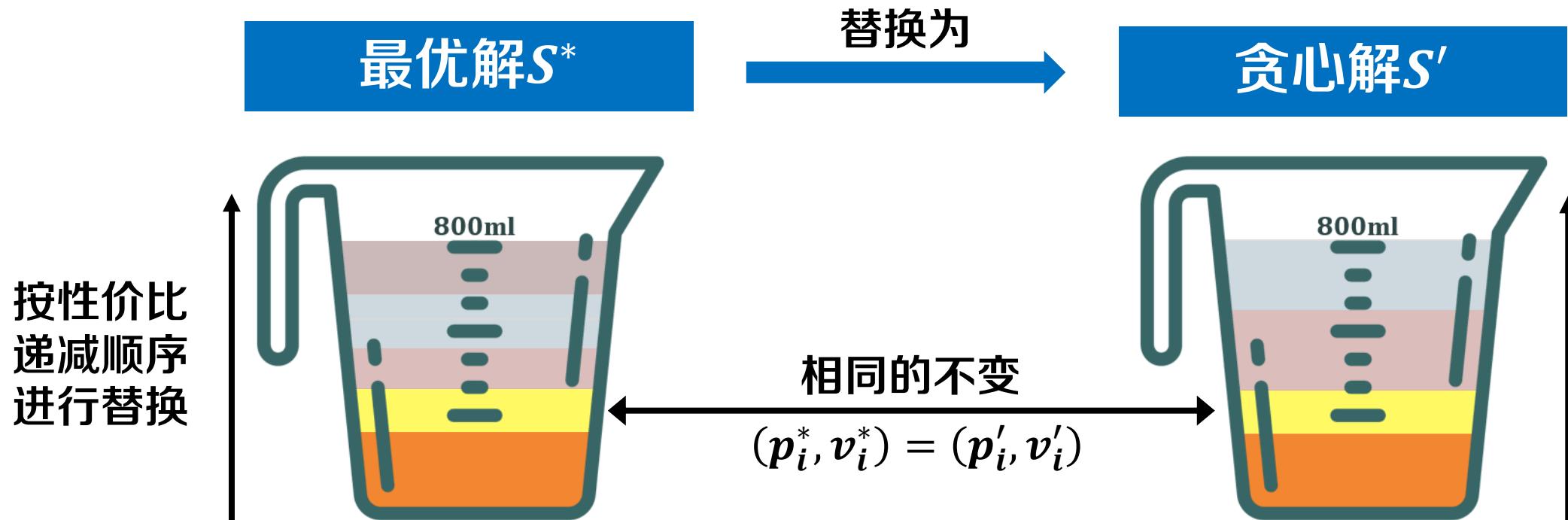


- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



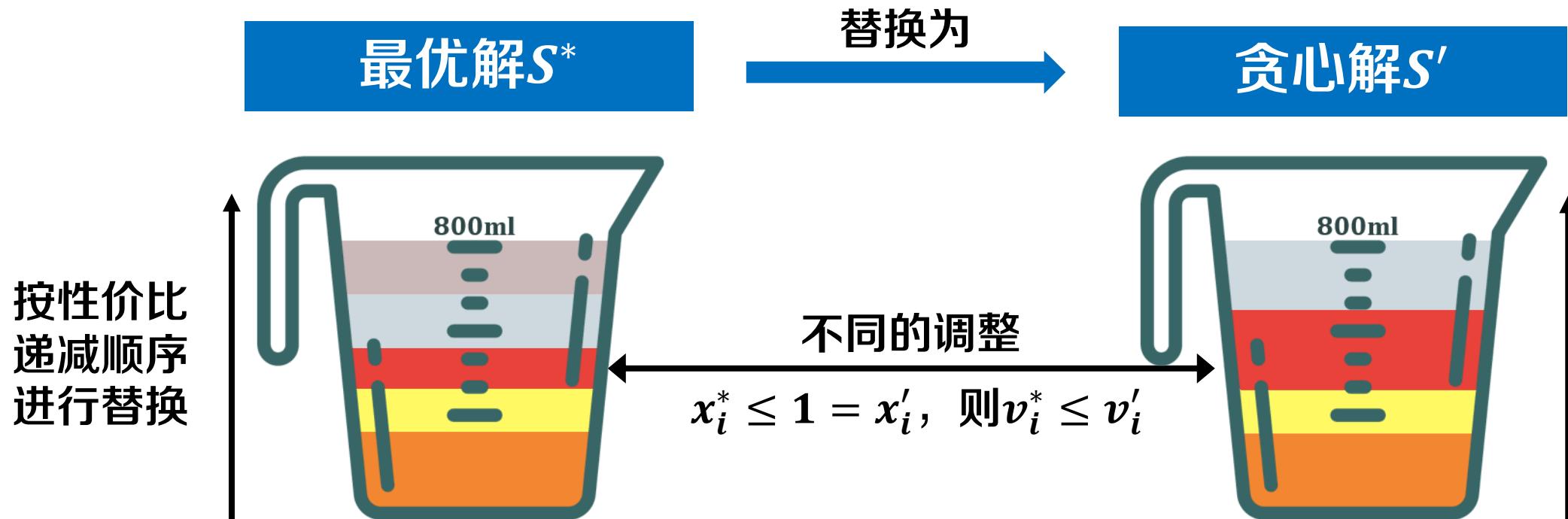
# 正确性证明

- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



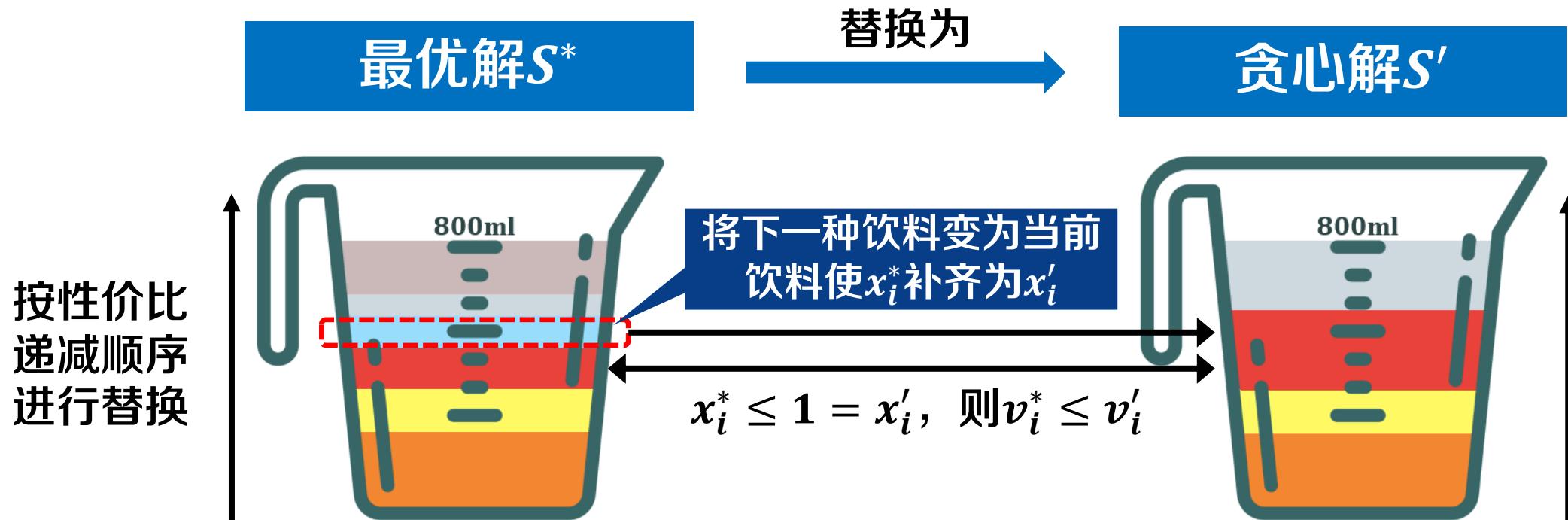
# 正确性证明

- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



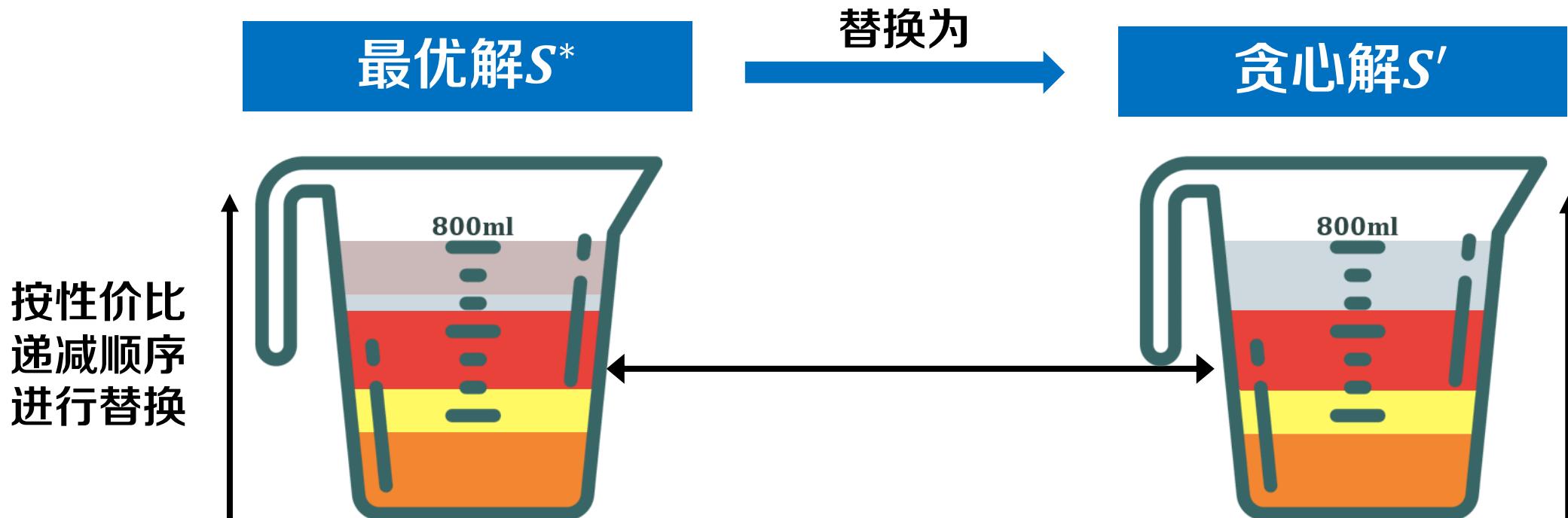
# 正确性证明

- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



# 正确性证明

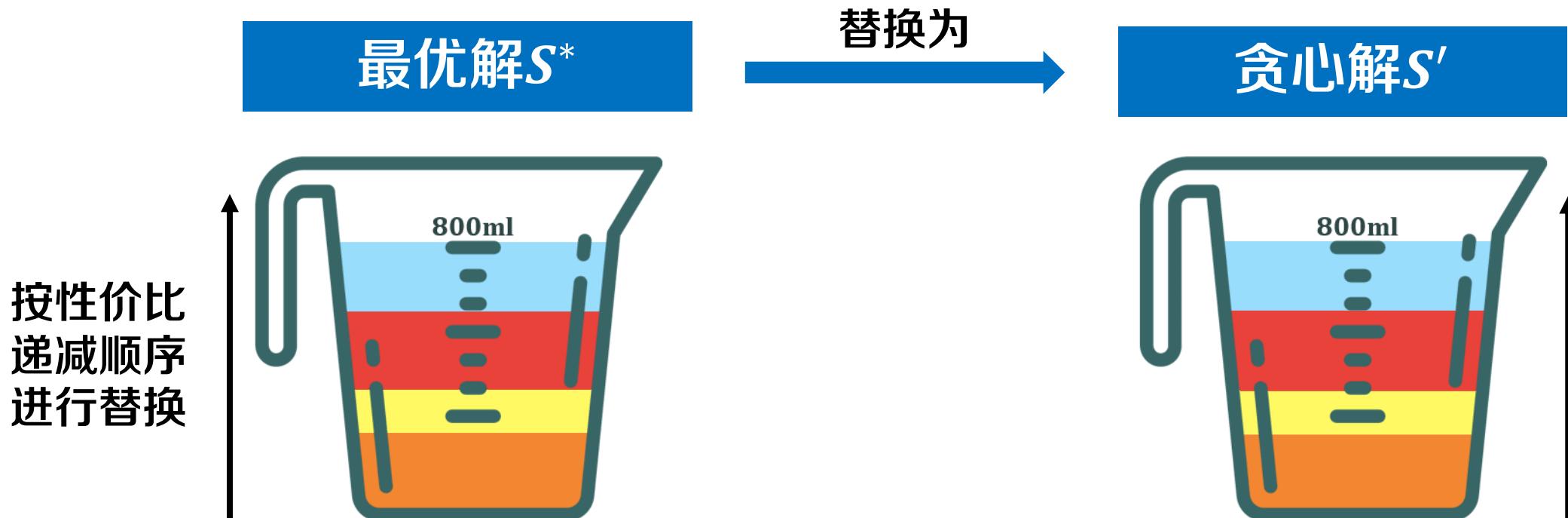
- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



性价比递减： $p'_i/v'_i \geq p^*_{i+1}/v^*_{i+1}$   
则替换后总价值不会减少

# 正确性证明

- 贪心策略：最高性价比优先
- 证明：贪心解不劣于最优解



替换后单位体积价值均不减少  
故贪心解不劣于最优解



# 伪代码

## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

    if  $v[i] \leq C$  then

        选择商品 $i$

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

    end

    else

        选择 $C$ 体积的商品 $i$

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

    end

$i \leftarrow i + 1$

end

return  $ans$

按性价比排序，并初始化



# 伪代码

## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

  if  $v[i] \leq C$  then

    选择商品 $i$

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

  end

  else

    选择 $C$ 体积的商品 $i$

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

  end

$i \leftarrow i + 1$

end

return  $ans$

当背包未装满且商品未装完时



# 伪代码

## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

if  $v[i] \leq C$  then

    选择商品*i*

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

end

else

    选择 $C$ 体积的商品*i*

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

end

$i \leftarrow i + 1$

end

return  $ans$

商品体积不大于容量则全部装入



# 伪代码

## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

    if  $v[i] \leq C$  then

        选择商品 $i$

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

    end

    else

        选择 $C$ 体积的商品 $i$

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

    end

$i \leftarrow i + 1$

end

return  $ans$

否则装入部分商品填满背包



# 复杂度分析

## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

    if  $v[i] \leq C$  then

        选择商品 $i$

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

    end

    else

        选择 $C$ 体积的商品 $i$

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

    end

$i \leftarrow i + 1$

end

return  $ans$

}

$O(n \log n)$

# 复杂度分析



## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

  if  $v[i] \leq C$  then

    选择商品 $i$

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

  end

  else

    选择 $C$ 体积的商品 $i$

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

  end

$i \leftarrow i + 1$

end

return  $ans$

$O(n \log n)$

$O(n)$



# 复杂度分析

## ● FractionalKnapsack( $n, p, v, C$ )

输入: 商品数量 $n$ ,各商品的价值 $p$ , 各商品的体积 $v$ , 背包容量 $C$

输出: 商品价格的最大值,最优解方案

计算商品性价比 $Ratio[1..n]$ 并按降序排序

// $Ratio[i], p[i], v[i]$ 分别表示性价比第 $i$ 大的商品的性价比、价格和体积

$i \leftarrow 1$

$ans \leftarrow 0$

//根据贪心策略求解

while  $C > 0$  and  $i \leq n$  do

  if  $v[i] \leq C$  then

    选择商品 $i$

$ans \leftarrow ans + p[i]$

$C \leftarrow C - v[i]$

  end

  else

    选择 $C$ 体积的商品 $i$

$ans \leftarrow ans + p[i] \cdot \frac{C}{v[i]}$

$C \leftarrow 0$

  end

$i \leftarrow i + 1$

end

return  $ans$

$O(n \log n)$

$O(n)$

时间复杂度:  $O(n \log n)$

# 贪心策略：一般步骤



提出贪心策略

观察问题特征，构造贪心选择



证明策略正确

假设最优方案，通过替换证明

# 对比0-1背包问题

- 0-1背包问题

- 贪心算法结果：



总价值24

- 动态规划算法结果：



总价值28

| 商品 | 价格 | 体积 | 性价比  |
|----|----|----|------|
| 啤酒 | 24 | 10 | 2.40 |
| 牛奶 | 9  | 4  | 2.25 |
| 饼干 | 9  | 4  | 2.25 |
| 面包 | 10 | 5  | 2.00 |

背包体积为13

0-1背包问题不能使用贪心算法

# 对比0-1背包问题

- 问题定义：

## 物品不可分割



## 物品可分割



- 解决方法：

## 动态规划

| $P[i, c]$ | $c = 0$ | 1 | 2 | 3 | ... | 10 | 11 | 12 | 13 |
|-----------|---------|---|---|---|-----|----|----|----|----|
| $i = 0$   | 0       | 0 | 0 | 0 | ... | 0  | 0  | 0  | 0  |
| 1         | 0       |   |   |   |     |    |    |    |    |
| 2         | 0       |   |   |   |     |    |    |    |    |
| 3         | 0       |   |   |   |     |    |    |    |    |
| 4         | 0       |   |   |   |     |    |    |    |    |
| 5         | 0       |   |   |   |     |    |    |    |    |



## 贪心策略

| 饮料  | 价格(元) | 体积(ml) | 性价比(元/ml) |
|-----|-------|--------|-----------|
| 橙汁  | 36    | 200    | 0.18      |
| 苹果汁 | 16    | 100    | 0.16      |
| 西瓜汁 | 45    | 300    | 0.15      |
| 苏打水 | 60    | 600    | 0.10      |
| 汽水  | 10    | 250    | 0.04      |

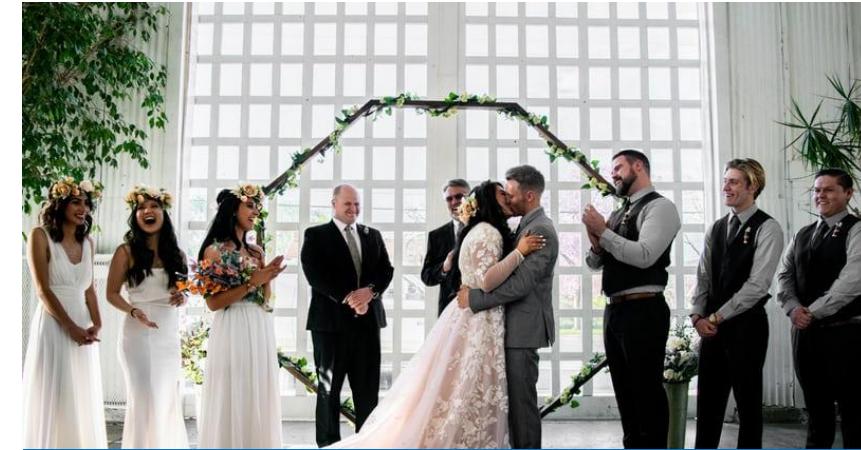
# 贪心策略篇：活动选择问题

# 问题背景

- 会场出租



公司年会：10:00 ~ 19:00



婚礼宴请：11:00 ~ 14:00



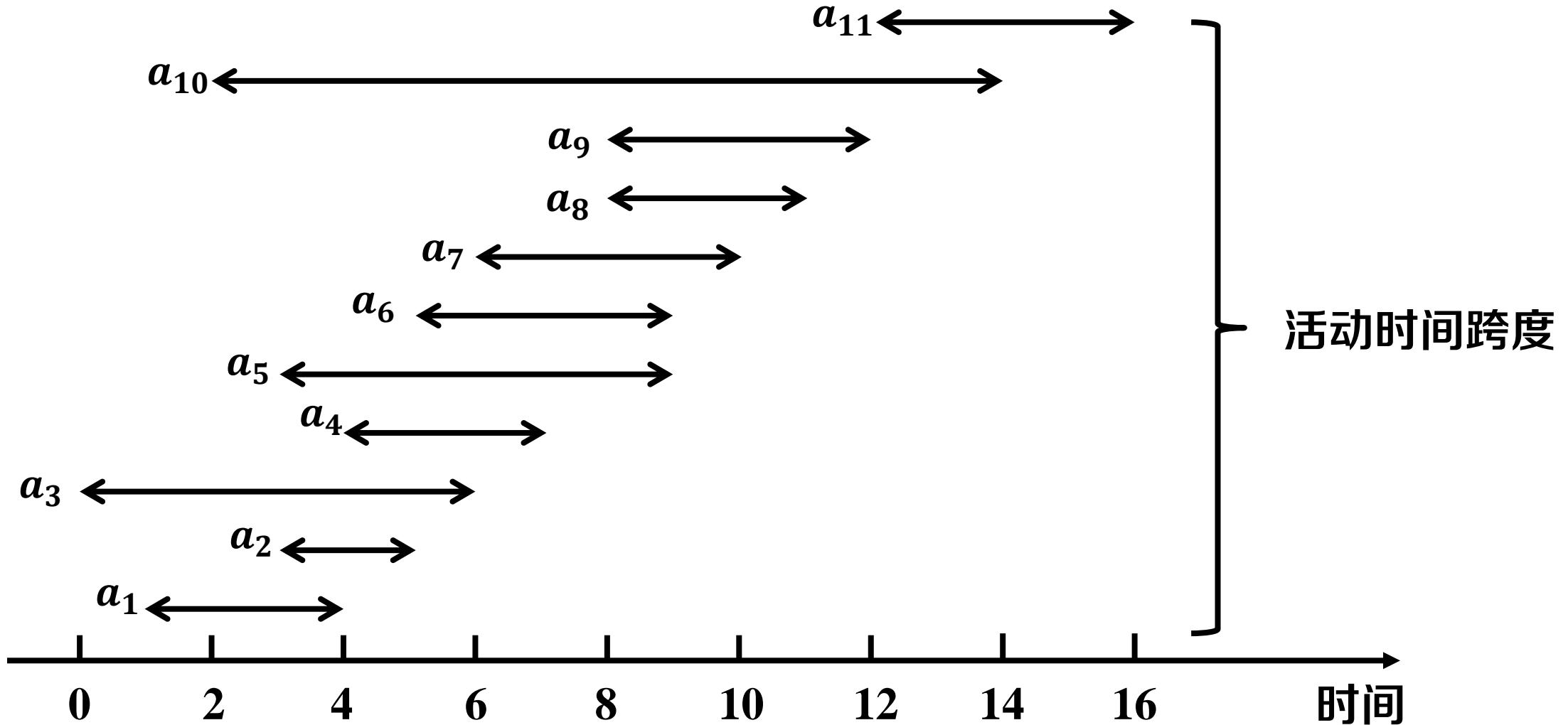
生日聚会：12:00 ~ 17:00



学术研讨：14:00 ~ 16:00

# 问题背景

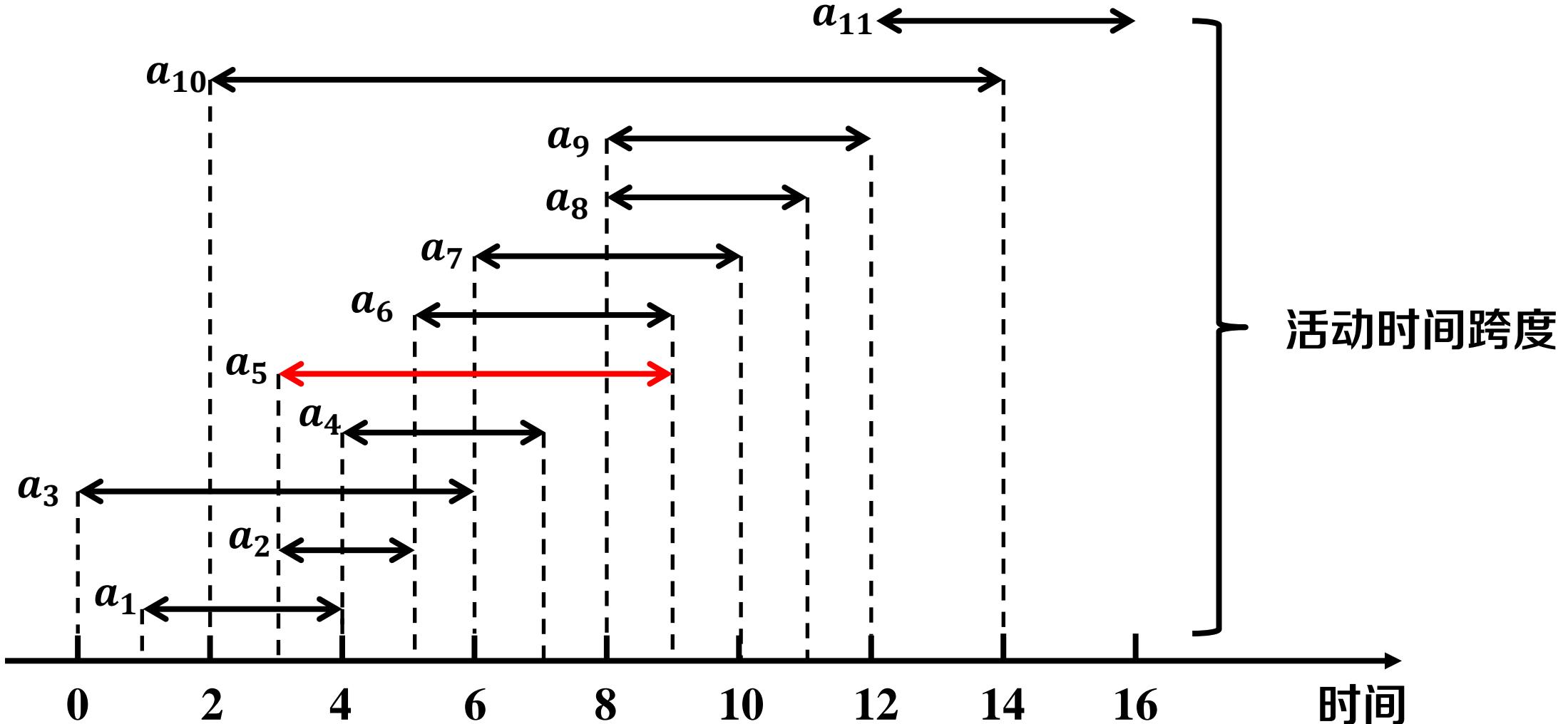
- 会场出租



# 问题背景



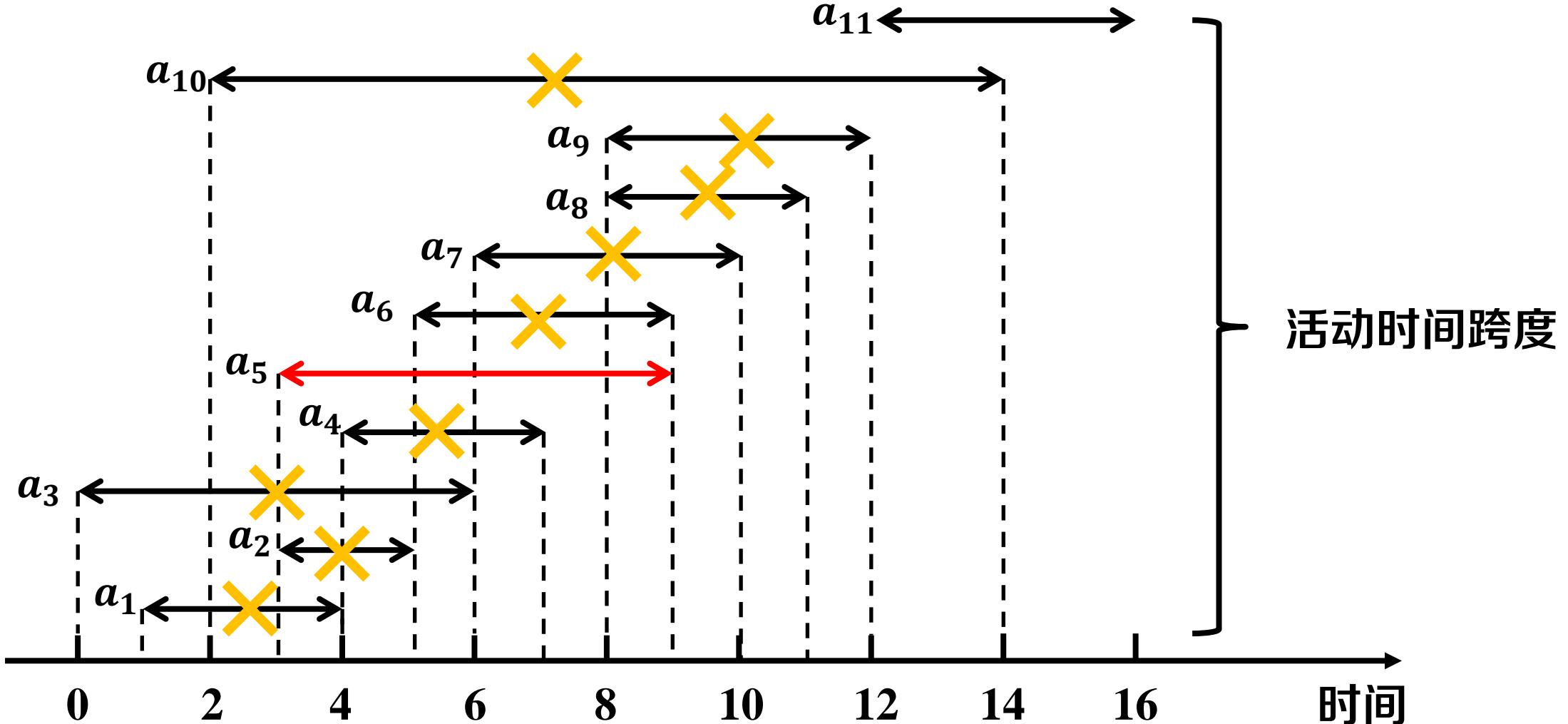
- 会场出租
  - 选择出租的活动时间不能冲突



# 问题背景

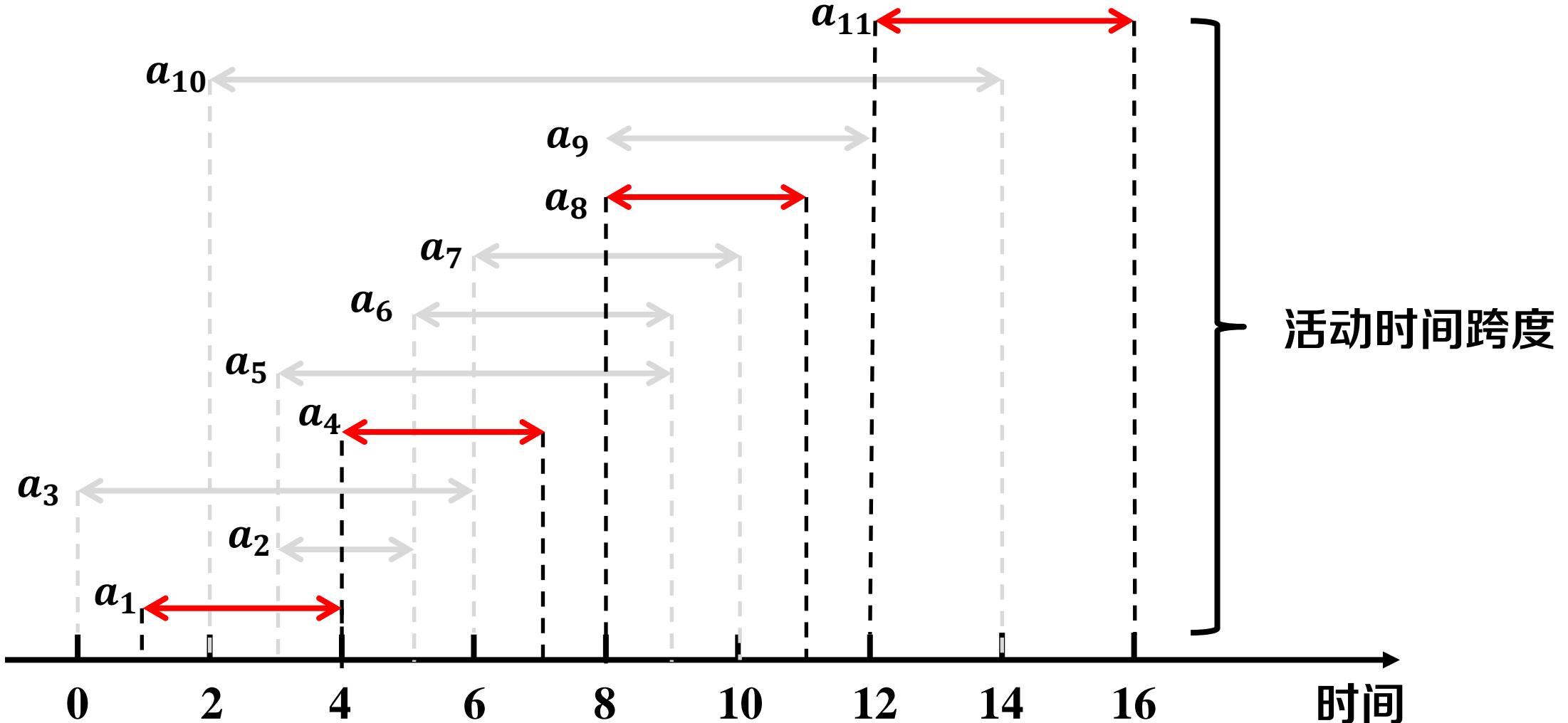


- 会场出租
  - 选择出租的活动时间不能冲突



# 问题背景

- 会场出租
  - 选择出租的活动时间不能冲突，怎样选择才能选更多的活动？





## 活动选择问题

### Activity Selection Problem

#### 输入

- $n$ 个活动组成的集合 $S = \{a_1, a_2, \dots, a_n\}$
- 每个活动 $a_i$ 的开始时间 $s_i$ 和结束时间 $f_i$

#### 输出

- 找出活动集合 $S$ 的子集 $S'$ , 令

$$\max |S'|$$

优化目标：最大化选择活动个数

$$s.t. \forall a_i, a_j \in S', s_i \geq f_j \text{ 或 } s_j \geq f_i$$

## 活动选择问题

### Activity Selection Problem

#### 输入

- $n$ 个活动组成的集合  $S = \{a_1, a_2, \dots, a_n\}$
- 每个活动  $a_i$  的开始时间  $s_i$  和结束时间  $f_i$

#### 输出

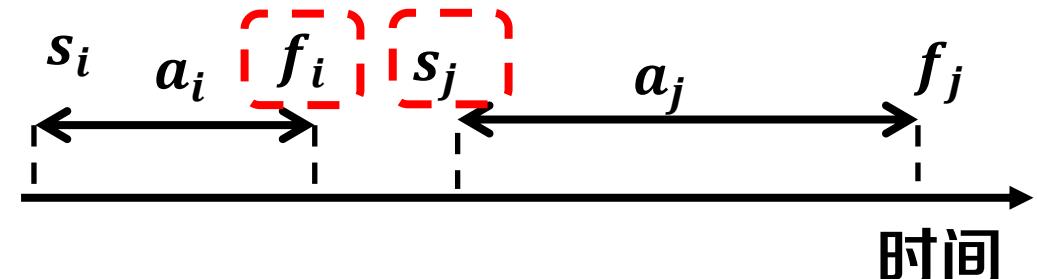
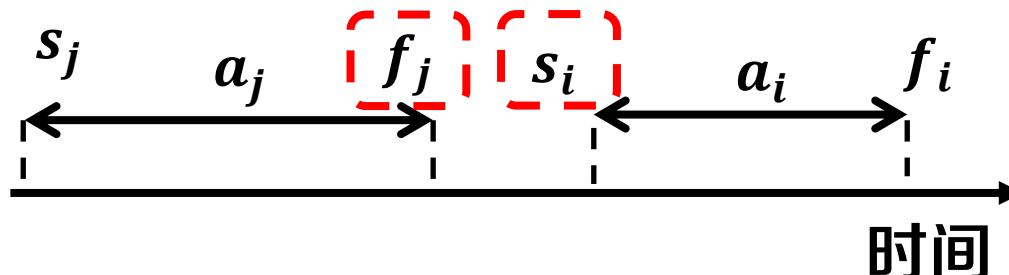
- 找出活动集合  $S$  的子集  $S'$ , 令

优化目标：最大化选择活动个数

约束条件

$$\max |S'|$$

$$s.t. \forall a_i, a_j \in S', s_i \geq f_j \text{ 或 } s_j \geq f_i$$

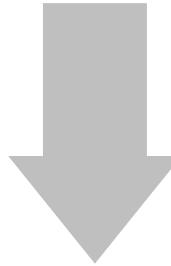


# 贪心策略：一般步骤



提出贪心策略

观察问题特征，构造贪心选择



证明策略正确

假设最优方案，通过替换证明

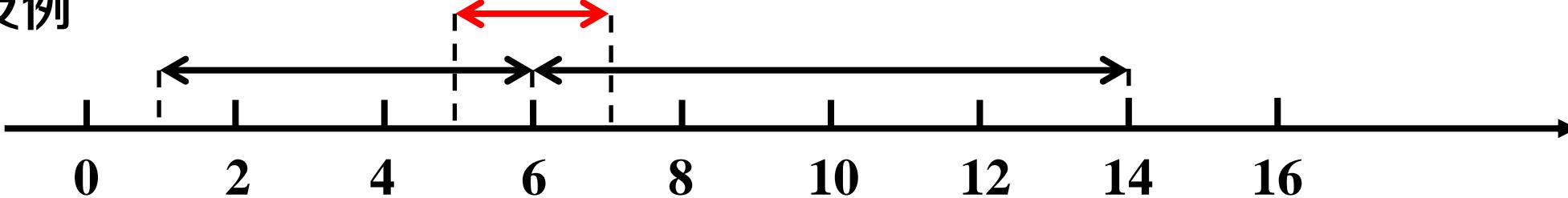


- 策略1：**最短活动优先**
- 策略2：**最早开始活动优先**
- 策略3：**最早结束活动优先**

# 贪心策略

- 策略1：最短活动优先

- 反例



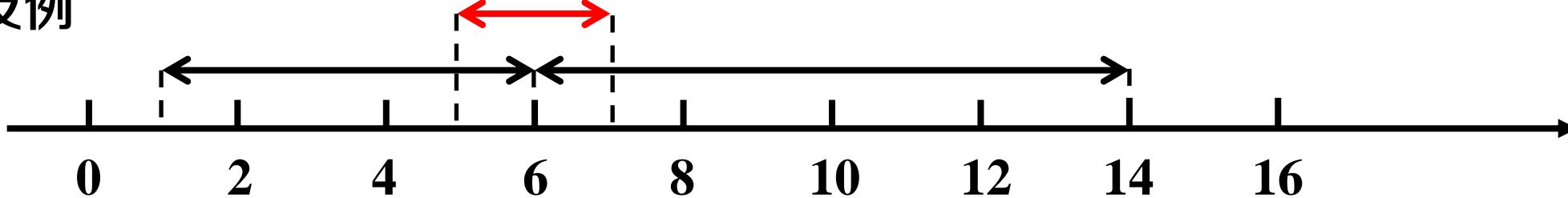
- 策略2：最早开始活动优先

- 策略3：最早结束活动优先

# 贪心策略

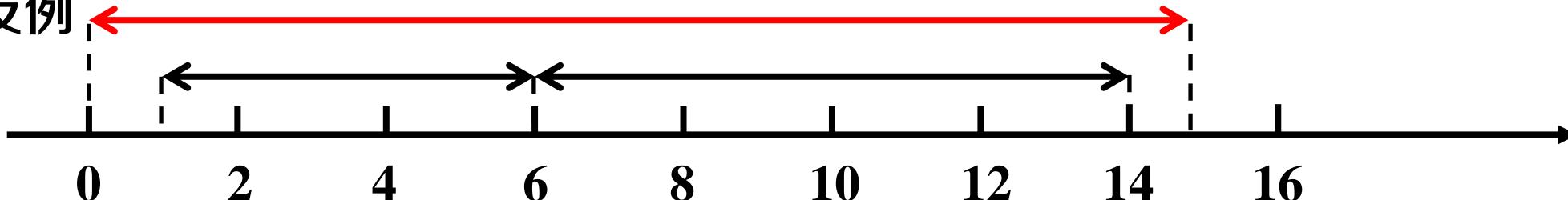
- 策略1：最短活动优先

- 反例



- 策略2：最早开始活动优先

- 反例

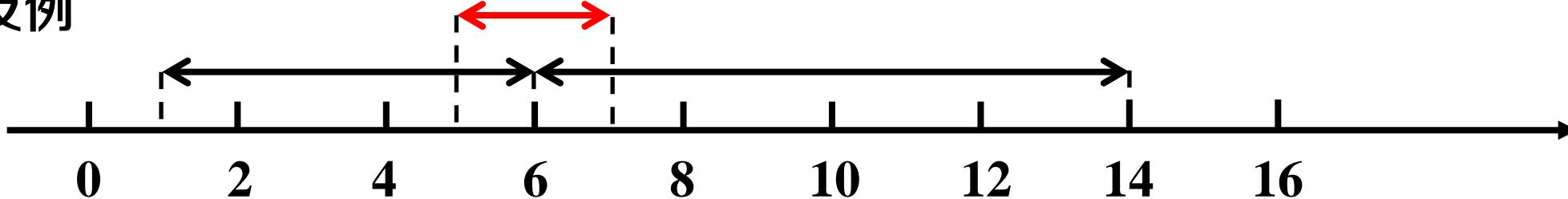


- 策略3：最早结束活动优先

# 贪心策略

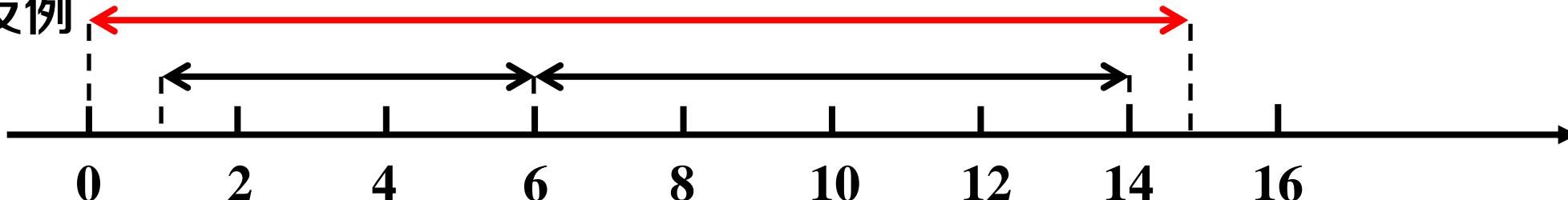
- 策略1：最短活动优先

- 反例



- 策略2：最早开始活动优先

- 反例

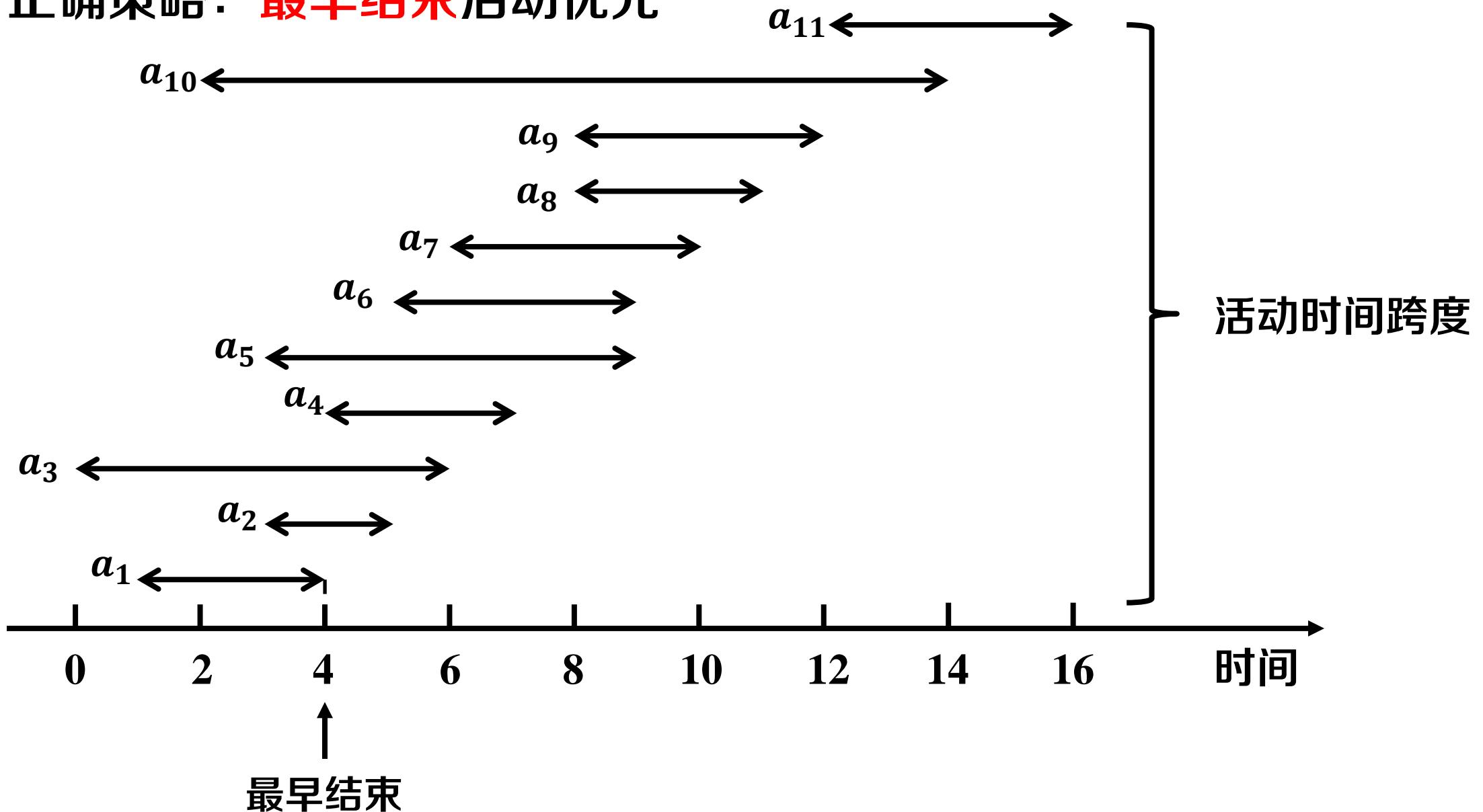


- 策略3：最早结束活动优先

- 选择最早结束的活动，可以给后面的活动留更大的选择空间

# 算法实例

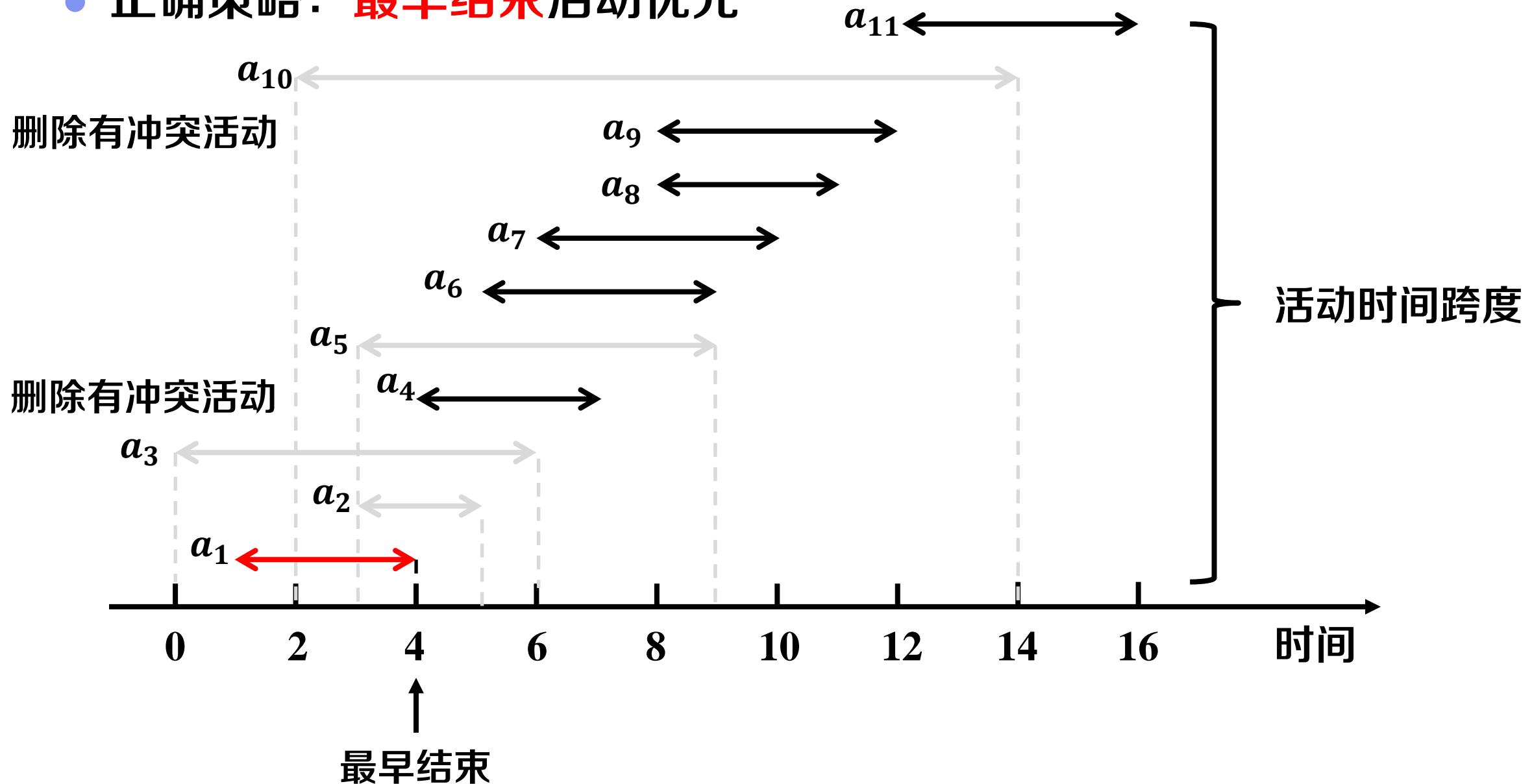
- 正确策略：最早结束活动优先



# 算法实例

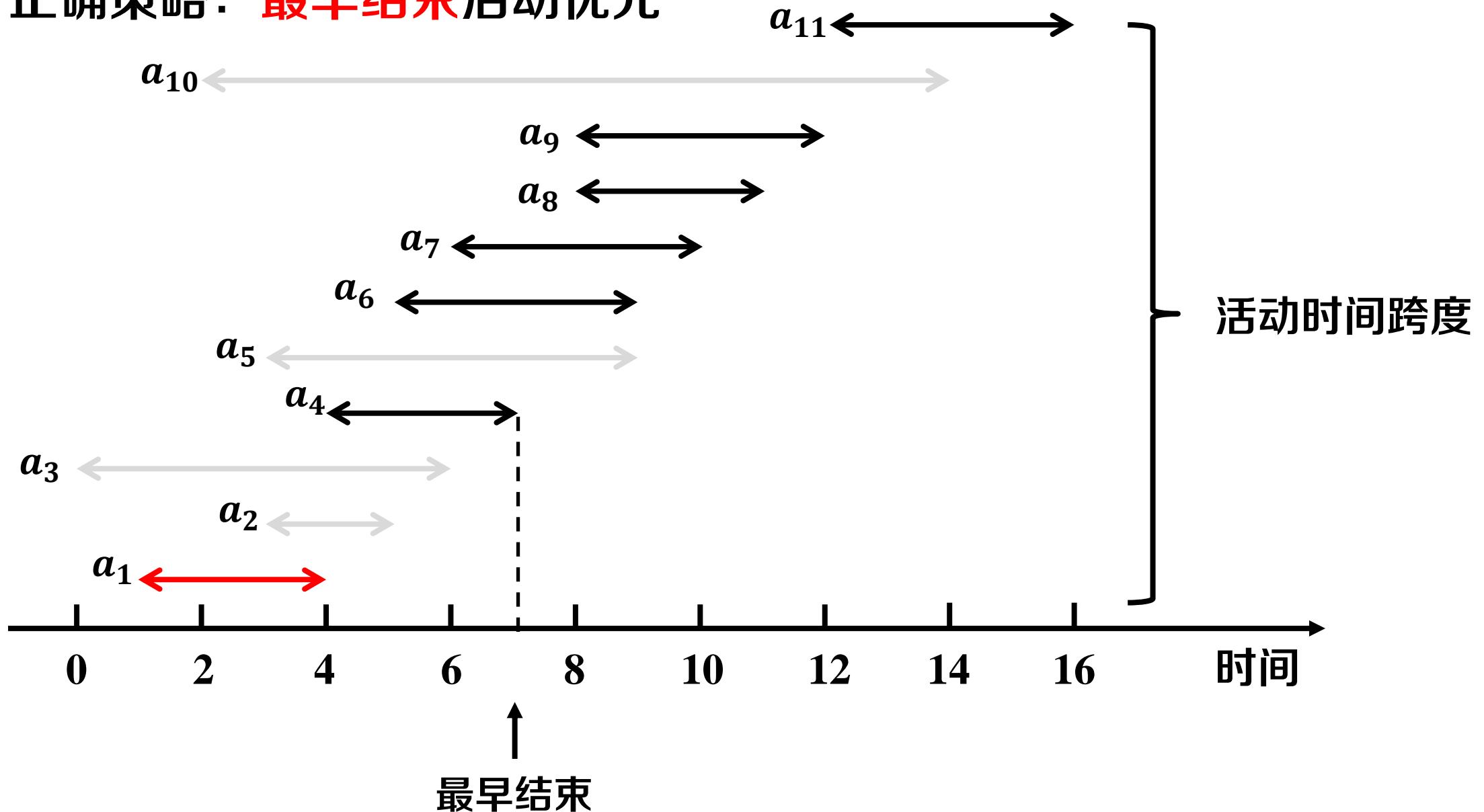


- 正确策略：最早结束活动优先



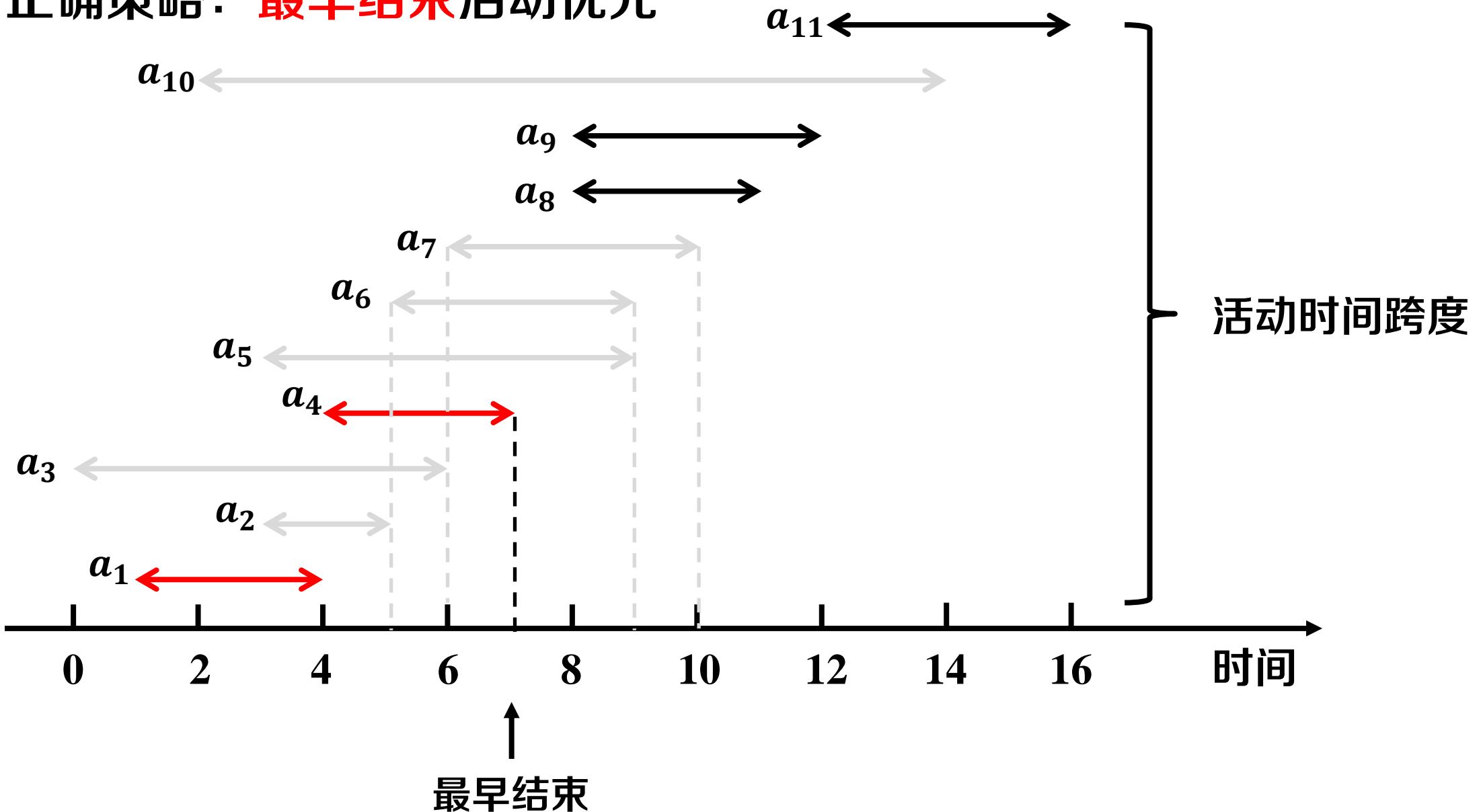
# 算法实例

- 正确策略：最早结束活动优先



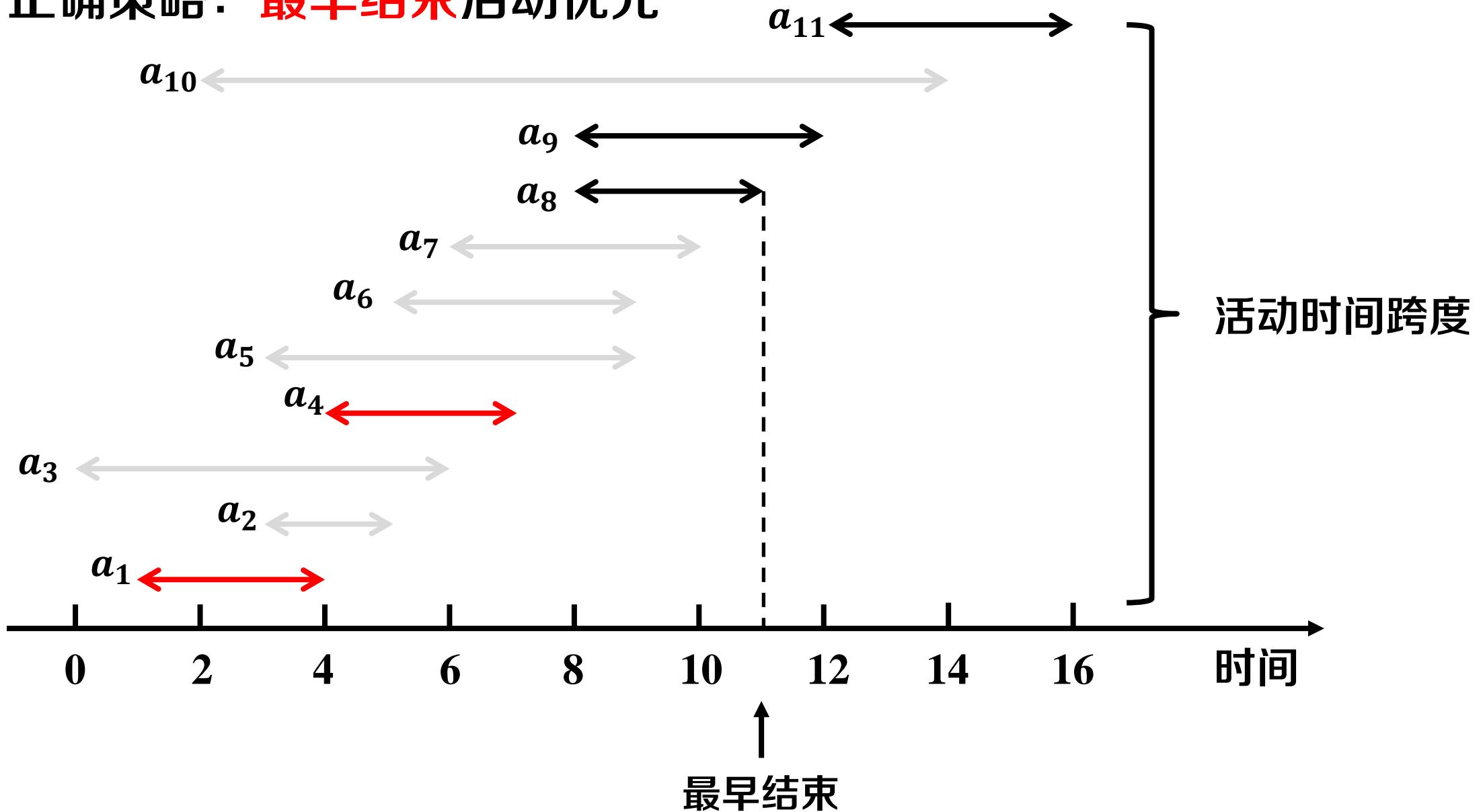
# 算法实例

- 正确策略：最早结束活动优先



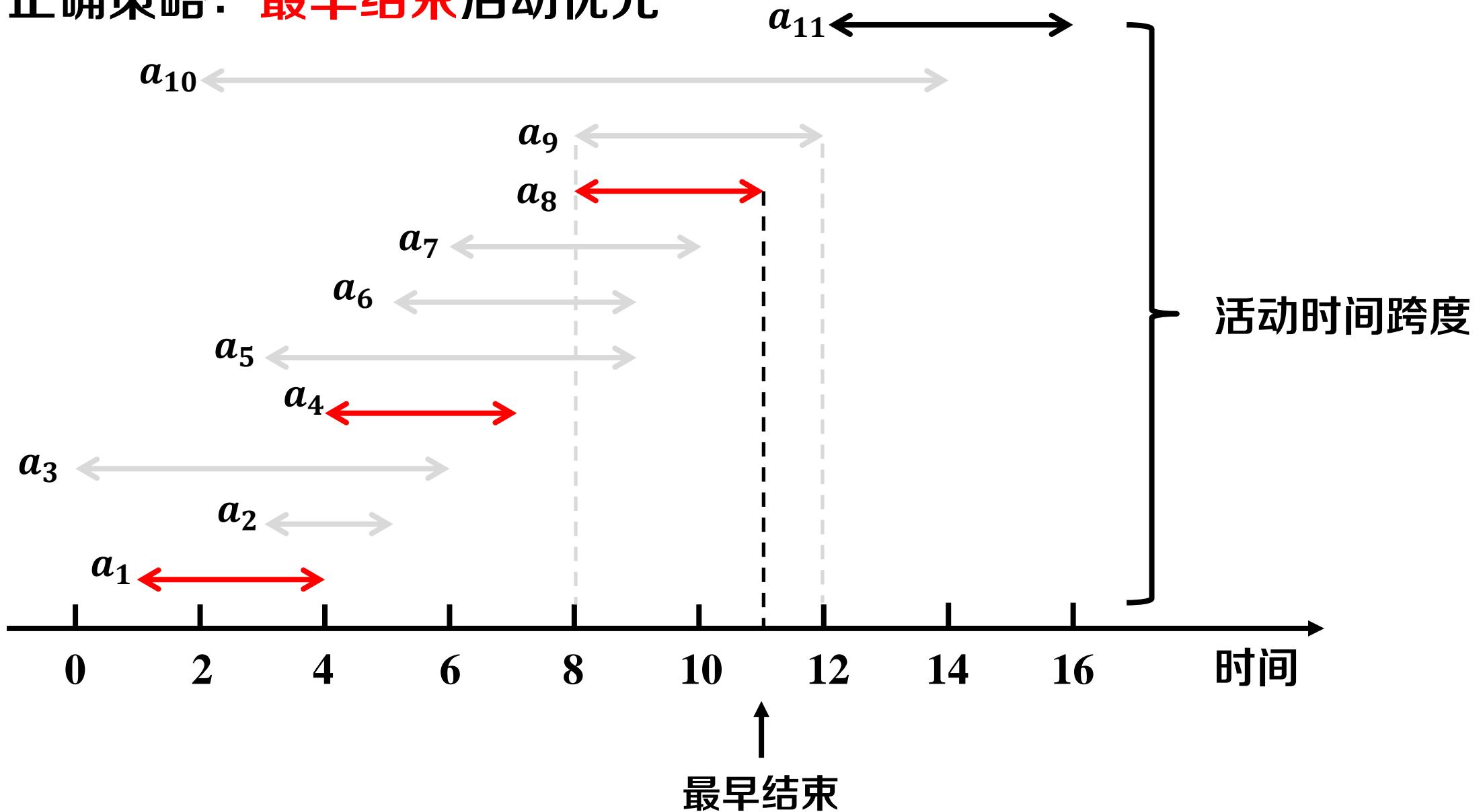
# 算法实例

- 正确策略：最早结束活动优先



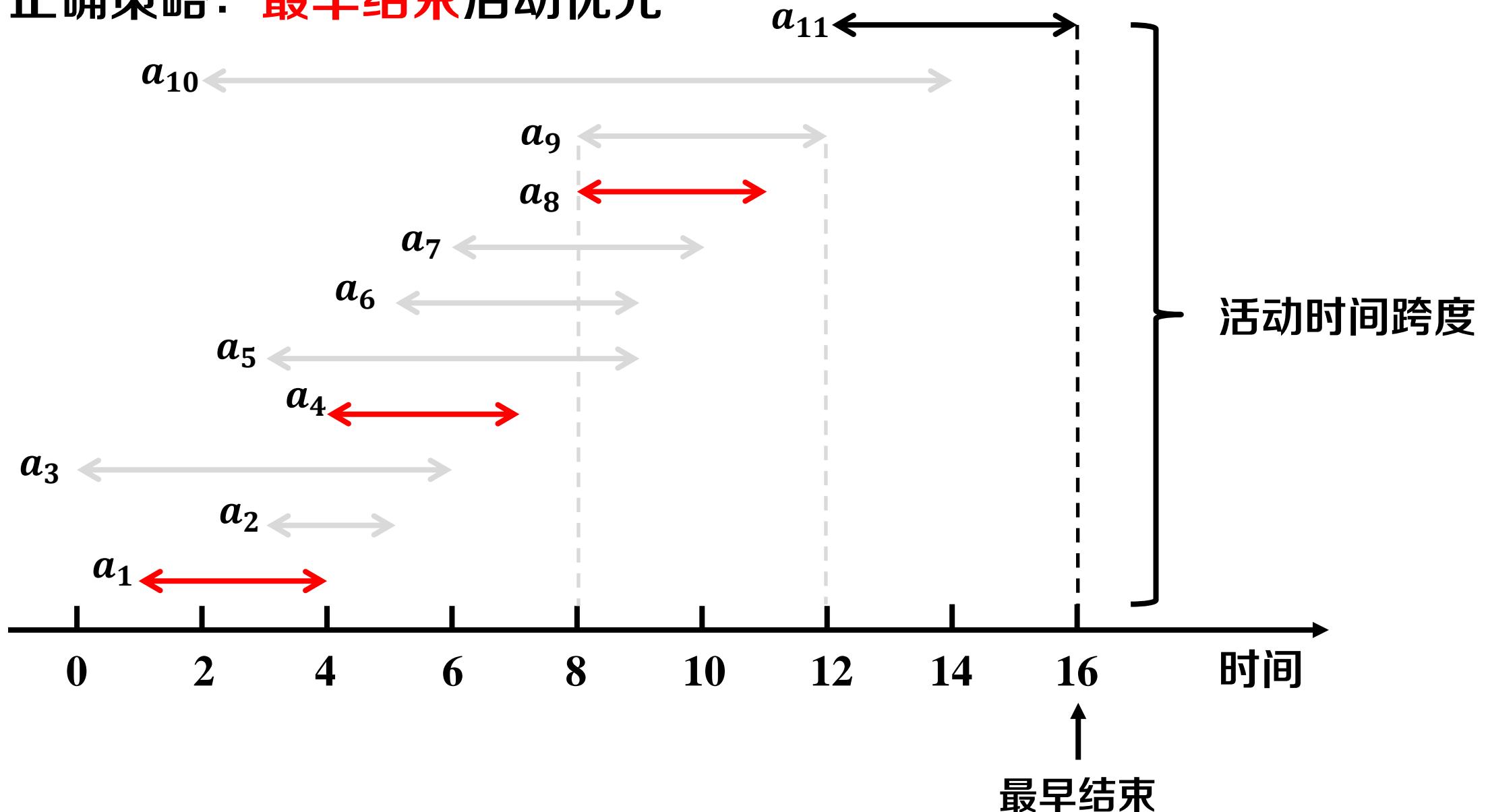
# 算法实例

- 正确策略：最早结束活动优先



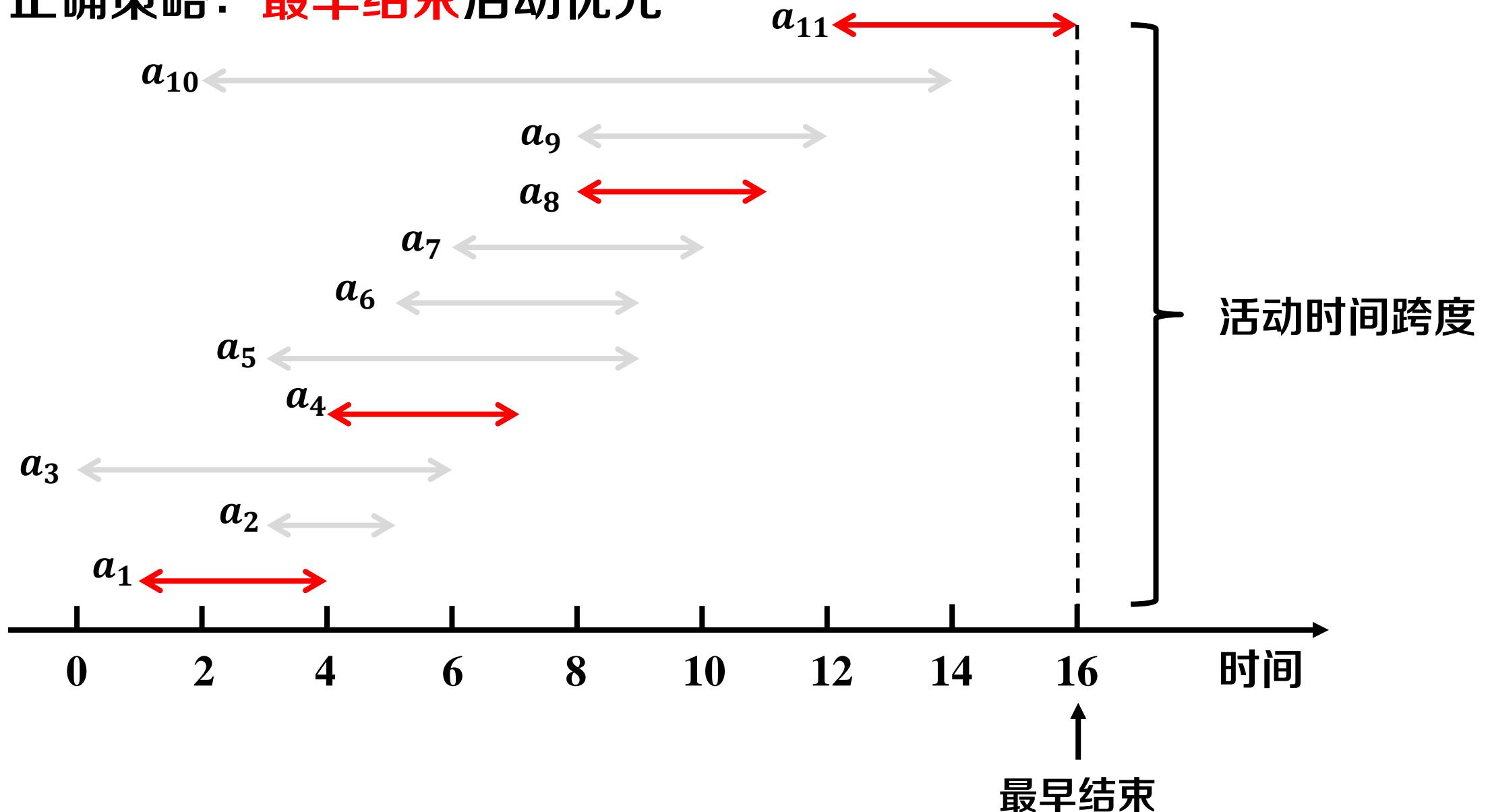
# 算法实例

- 正确策略：最早结束活动优先



# 算法实例

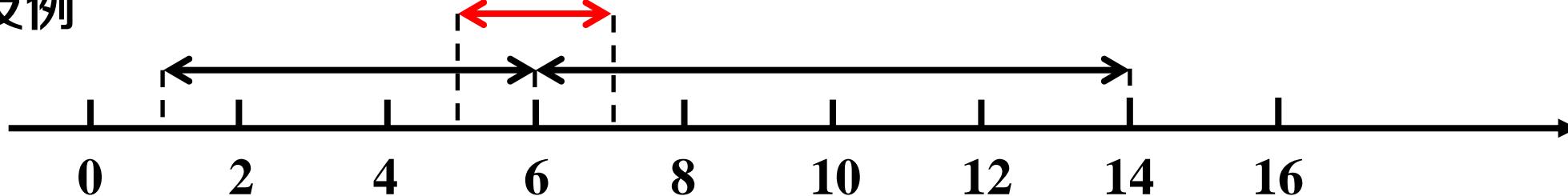
- 正确策略：最早结束活动优先



# 贪心策略

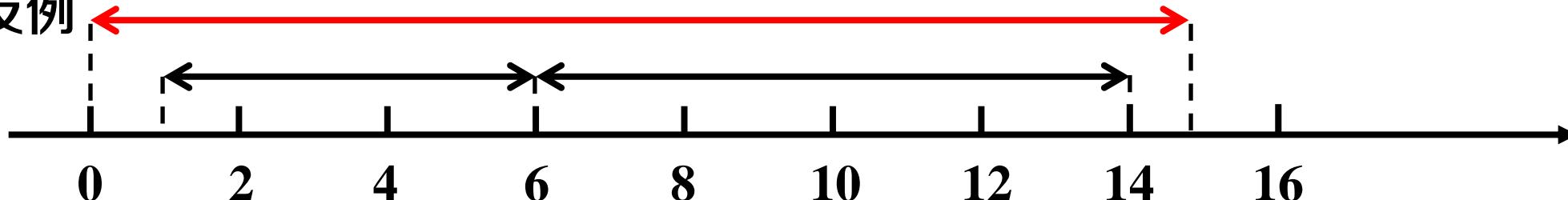
- 策略1：最短活动优先

- 反例



- 策略2：最早开始活动优先

- 反例



- 策略3：最早结束活动优先

- 选择最早结束的活动，可以给后面的活动留更大的选择空间

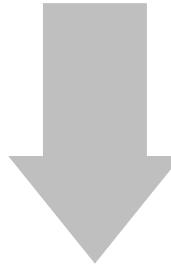
问题：策略3是否可以保证最优解？

# 贪心策略：一般步骤



提出贪心策略

观察问题特征，构造贪心选择



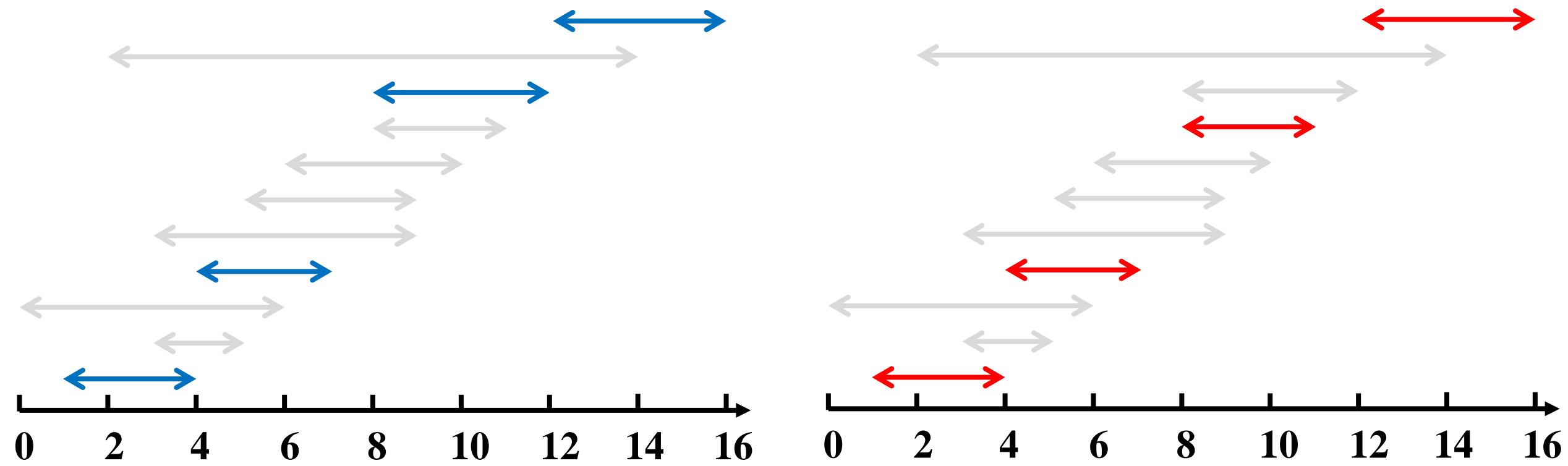
证明策略正确

假设最优方案，通过替换证明

# 正确性证明



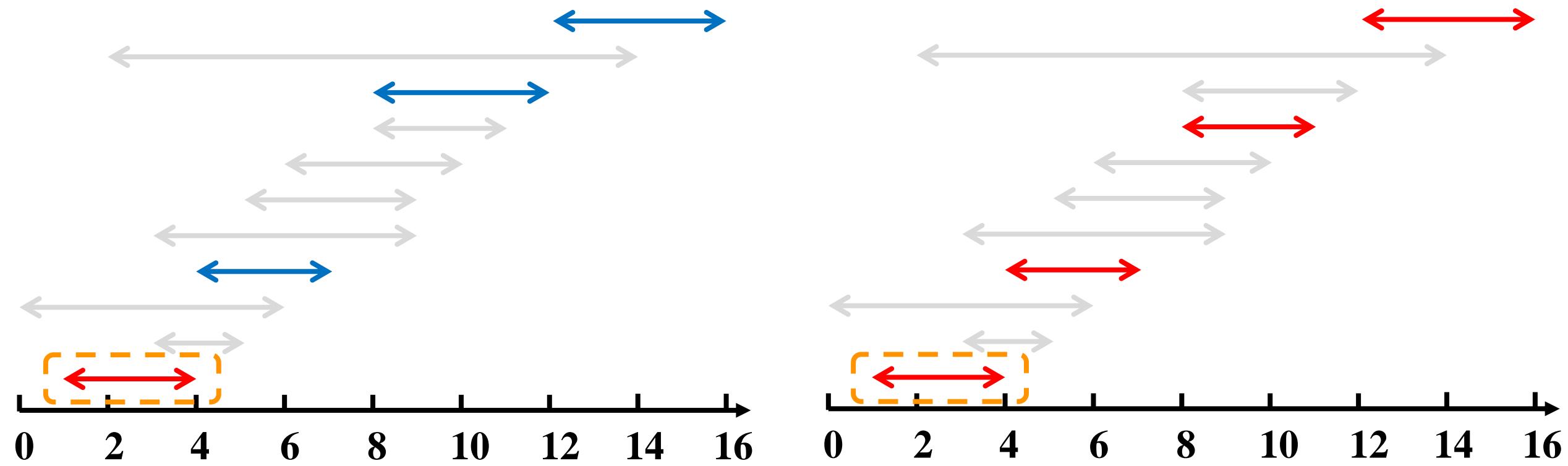
- 正确策略：最早结束活动优先
- 证明：贪心解不劣于最优解



# 正确性证明



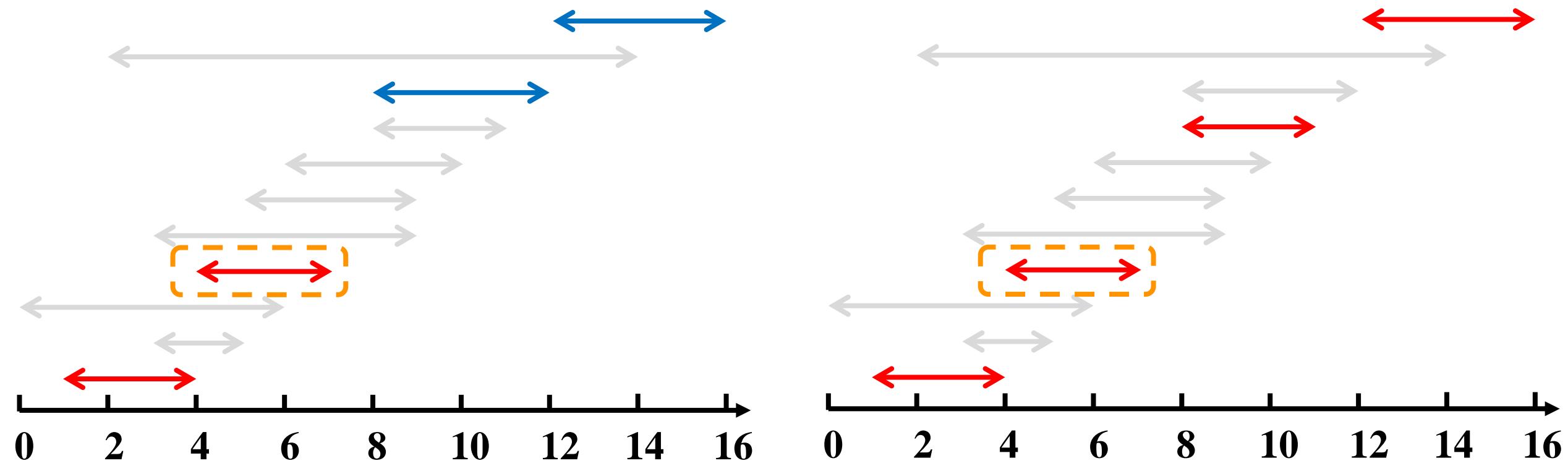
- 正确策略：最早结束活动优先
- 证明：贪心解不劣于最优解



# 正确性证明



- 正确策略：最早结束活动优先
- 证明：贪心解不劣于最优解



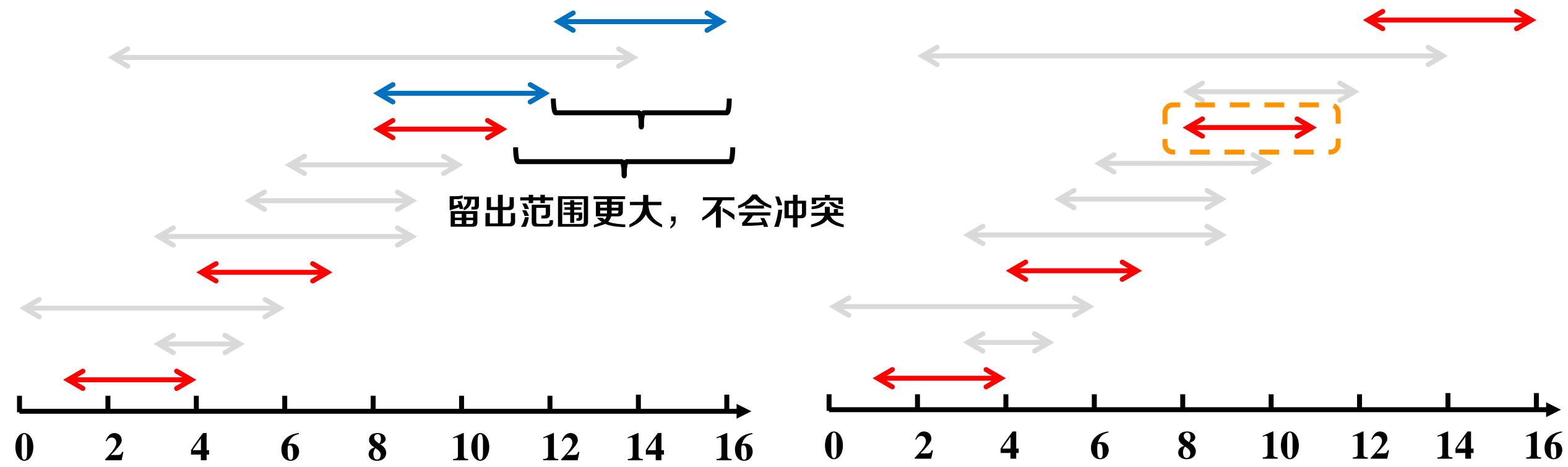
# 正确性证明



- 正确策略：最早结束活动优先
- 证明：贪心解不劣于最优解



活动相同，无需替换  
活动不同，必能替换



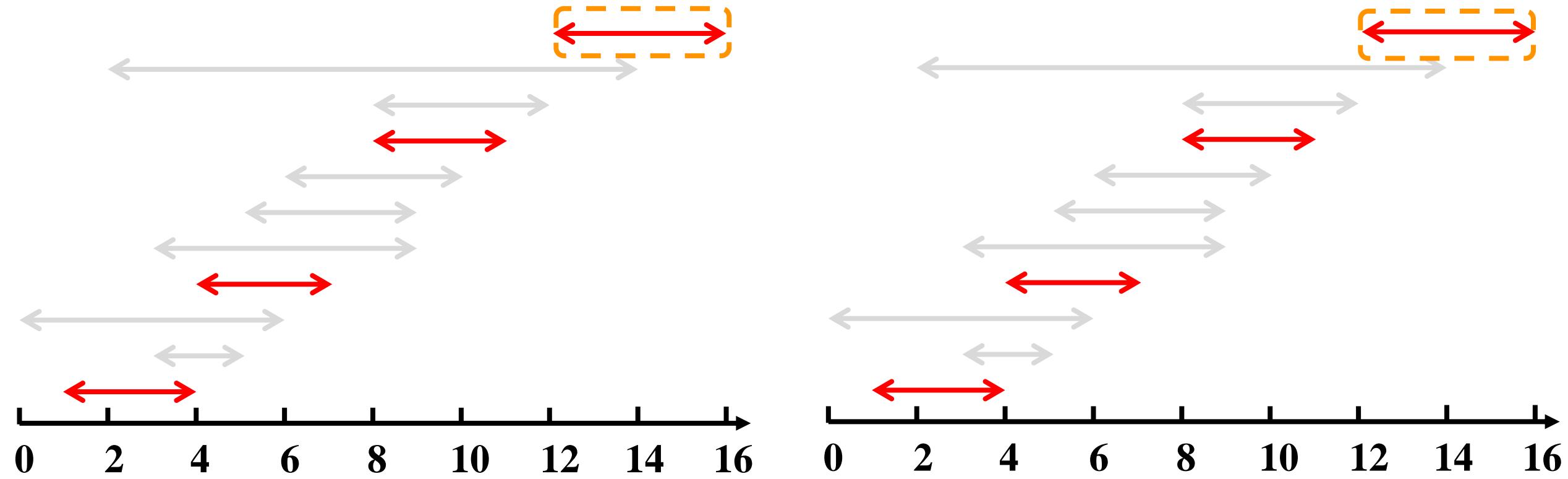
# 正确性证明



- 正确策略：最早结束活动优先
- 证明：贪心解不劣于最优解

任意最优活动集合 → 依次检查并替换 → 贪心所得活动集合

活动相同，无需替换  
活动不同，必能替换



# 贪心算法：伪代码



输入: 活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出: 不冲突活动的最大子集  $S'$

**把活动按照结束时间升序排序**

```
 $S' \leftarrow \{a_1\}$ 
 $k \leftarrow 1$ 
for  $i \leftarrow 2$  to  $n$  do
    if  $s_i \geq f_k$  then
         $S' \leftarrow S' \cup \{a_i\}$ 
         $k \leftarrow i$ 
    end
end
return  $S'$ 
```

为使用贪心策略作准备



# 贪心算法：伪代码

输入：活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出：不冲突活动的最大子集  $S'$

把活动按照结束时间升序排序

$S' \leftarrow \{a_1\}$

$k \leftarrow 1$

**for**  $i \leftarrow 2$  to  $n$  **do**

**if**  $s_i \geq f_k$  **then**

$S' \leftarrow S' \cup \{a_i\}$

$k \leftarrow i$

**end**

**end**

**return**  $S'$

**把最早结束活动加入到集合**



# 贪心算法：伪代码

输入：活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出：不冲突活动的最大子集  $S'$

把活动按照结束时间升序排序

```
 $S' \leftarrow \{a_1\}$ 
 $k \leftarrow 1$ 
for  $i \leftarrow 2$  to  $n$  do
    if  $s_i \geq f_k$  then
         $S' \leftarrow S' \cup \{a_i\}$ 
         $k \leftarrow i$ 
    end
end
return  $S'$ 
```

记录当前选择的活动



# 贪心算法：伪代码

输入：活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出：不冲突活动的最大子集  $S'$

把活动按照结束时间升序排序

```
 $S' \leftarrow \{a_1\}$ 
 $k \leftarrow 1$ 
for  $i \leftarrow 2$  to  $n$  do
    if  $s_i \geq f_k$  then
         $S' \leftarrow S' \cup \{a_i\}$ 
         $k \leftarrow i$ 
    end
end
return  $S'$ 
```

检查每个活动



# 贪心算法：伪代码

输入：活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出：不冲突活动的最大子集  $S'$

把活动按照结束时间升序排序

$S' \leftarrow \{a_1\}$

$k \leftarrow 1$

for  $i \leftarrow 2$  to  $n$  do

if  $s_i \geq f_k$  then  
     $S' \leftarrow S' \cup \{a_i\}$   
     $k \leftarrow i$

end

end

return  $S'$

没有冲突，则加入子集



# 贪心算法：伪代码

输入：活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出：不冲突活动的最大子集  $S'$

把活动按照结束时间升序排序

$S' \leftarrow \{a_1\}$

$k \leftarrow 1$

**for**  $i \leftarrow 2$  to  $n$  **do**

**if**  $s_i \geq f_k$  **then**

$S' \leftarrow S' \cup \{a_i\}$

$k \leftarrow i$

**end**

**end**

**return**  $S'$

更新当前选择的活动

# 贪心算法：复杂度分析



输入: 活动集合  $S = \{a_1, a_2, \dots, a_n\}$ , 每个活动  $a_i$  的起止时间  $s_i, f_i$

输出: 不冲突活动的最大子集  $S'$

把活动按照结束时间升序排序 - - - - -  $O(n \log n)$

```
 $S' \leftarrow \{a_1\}$ 
 $k \leftarrow 1$ 
for  $i \leftarrow 2$  to  $n$  do
    if  $s_i \geq f_k$  then
         $S' \leftarrow S' \cup \{a_i\}$ 
         $k \leftarrow i$ 
    end
end
return  $S'$ 
```

$O(n)$

时间复杂度:  $O(n \log n)$

# 问题拓展

## ● 会场出租

收益很大



公司年会：10:00 ~ 19:00

收益良好



婚礼宴请：11:00 ~ 14:00

收益较多



生日聚会：12:00 ~ 17:00

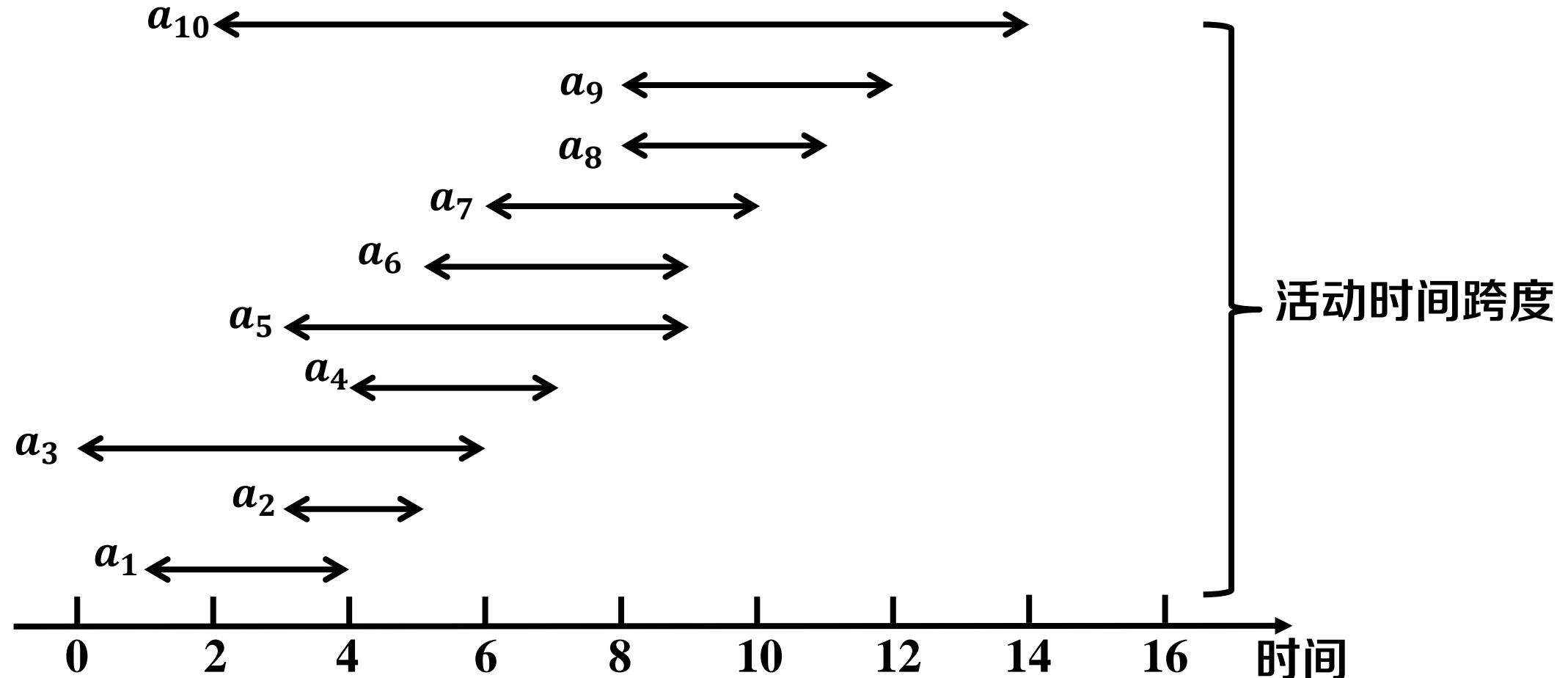
收益较少



学术研讨：14:00 ~ 16:00

# 问题拓展

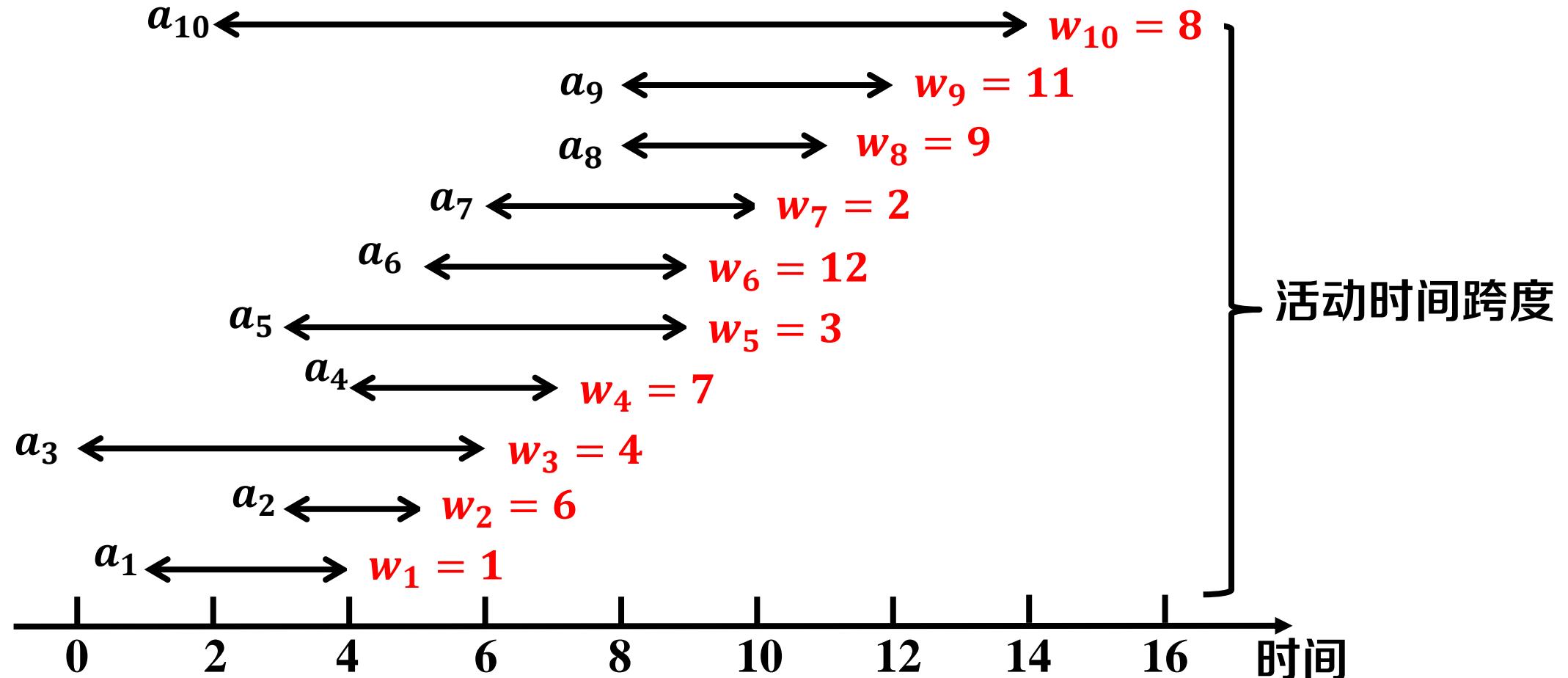
- 会场出租
  - 选择出租的活动时间不能冲突



# 问题拓展

## 会场出租

- 选择出租的活动时间不能冲突，**活动出租收益各不相同**
- 怎样选让收益总和最大？





## 带权活动选择问题

### Weighted Activity Selection Problem

输入

- $n$ 个活动组成的集合 $S = \{a_1, a_2, \dots, a_n\}$
- 每个活动 $a_i$ 的开始时间 $s_i$ , 结束时间 $f_i$ 和权重 $w_i$

输出

- 找出活动集合 $S$ 的子集 $S'$ , 令

优化目标: 最大化权重之和

$$\max \sum_{a_i \in S'} w_i$$

$$s.t. \forall a_i, a_j \in S', s_i \geq f_j \text{ 或 } s_j \geq f_i$$

约束条件



## 带权活动选择问题

$$\max \sum_{a_i \in S'} w_i$$

权重均为1

## 活动选择问题

$$\max |S'|$$



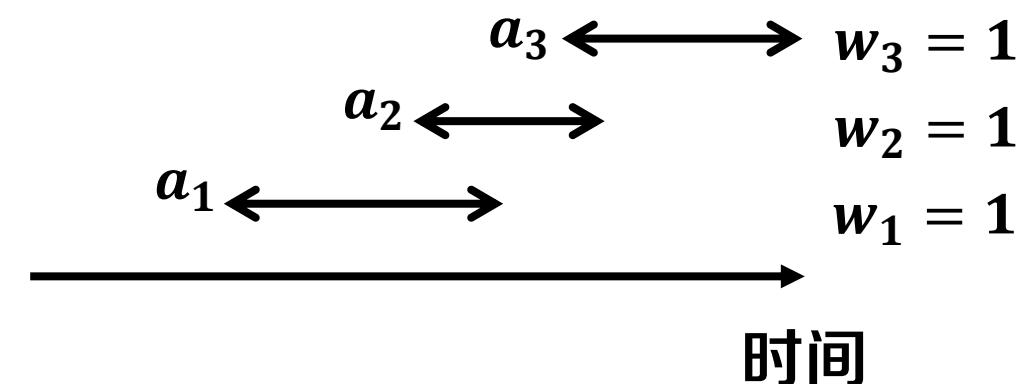
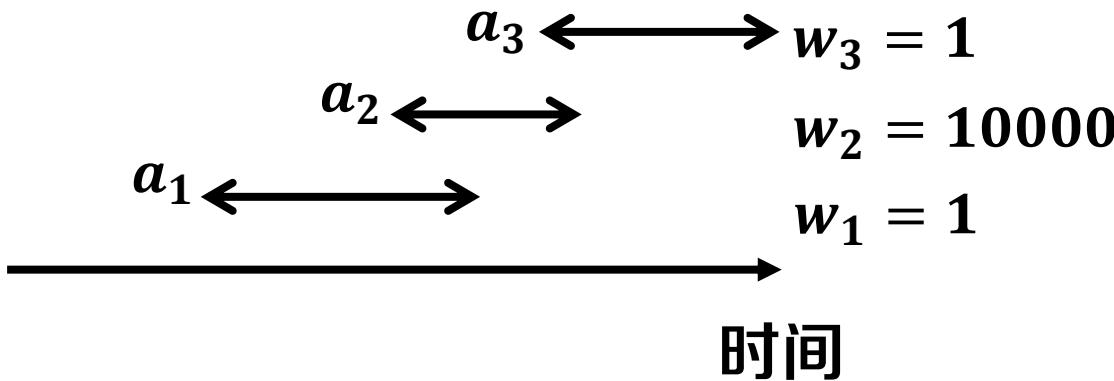
## 带权活动选择问题

$$\max \sum_{a_i \in S'} w_i$$

权重均为1

## 活动选择问题

$$\max |S'|$$



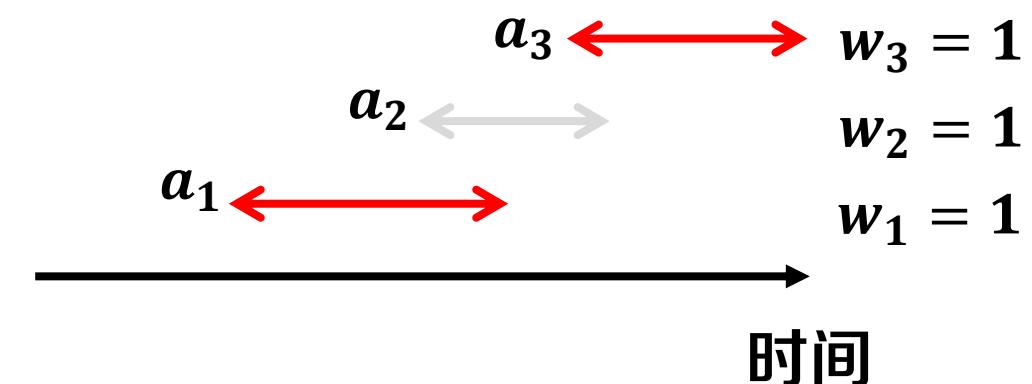
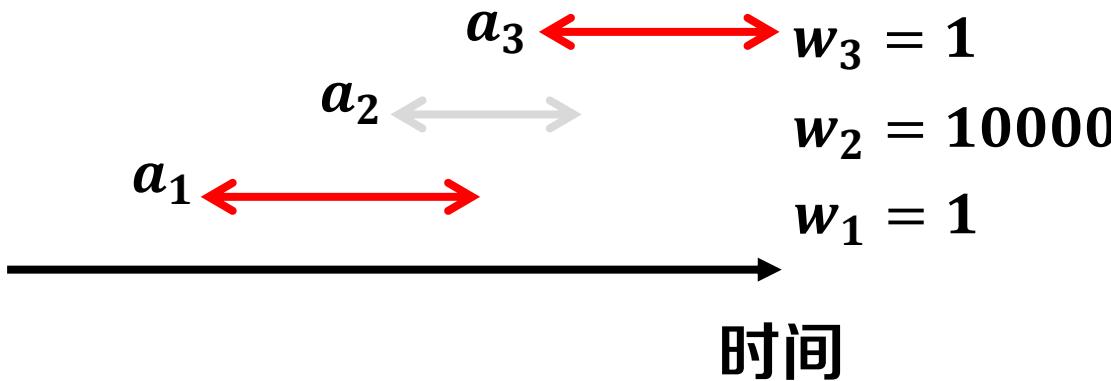
## 带权活动选择问题

$$\max \sum_{a_i \in S'} w_i$$

权重均为1

## 活动选择问题

$$\max |S'|$$





## 带权活动选择问题

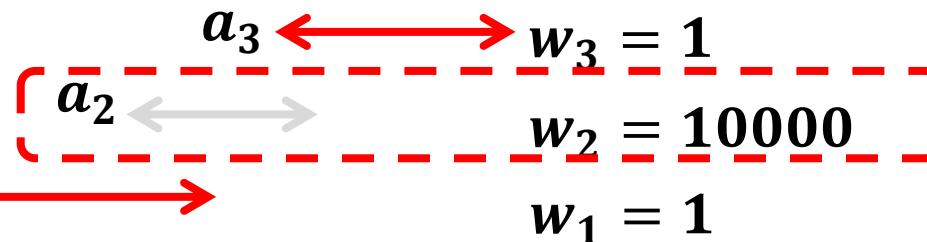
$$\max \sum_{a_i \in S'} w_i$$

权重均为1

## 活动选择问题

$$\max |S'|$$

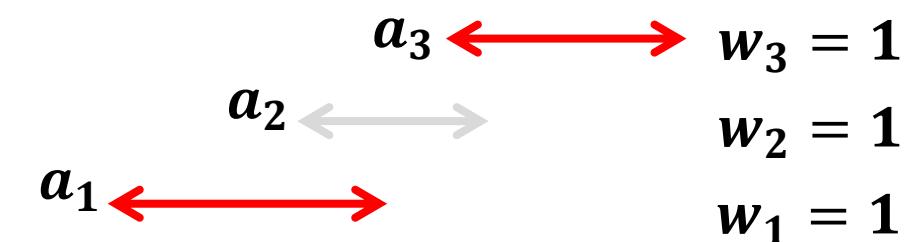
最优解



时间

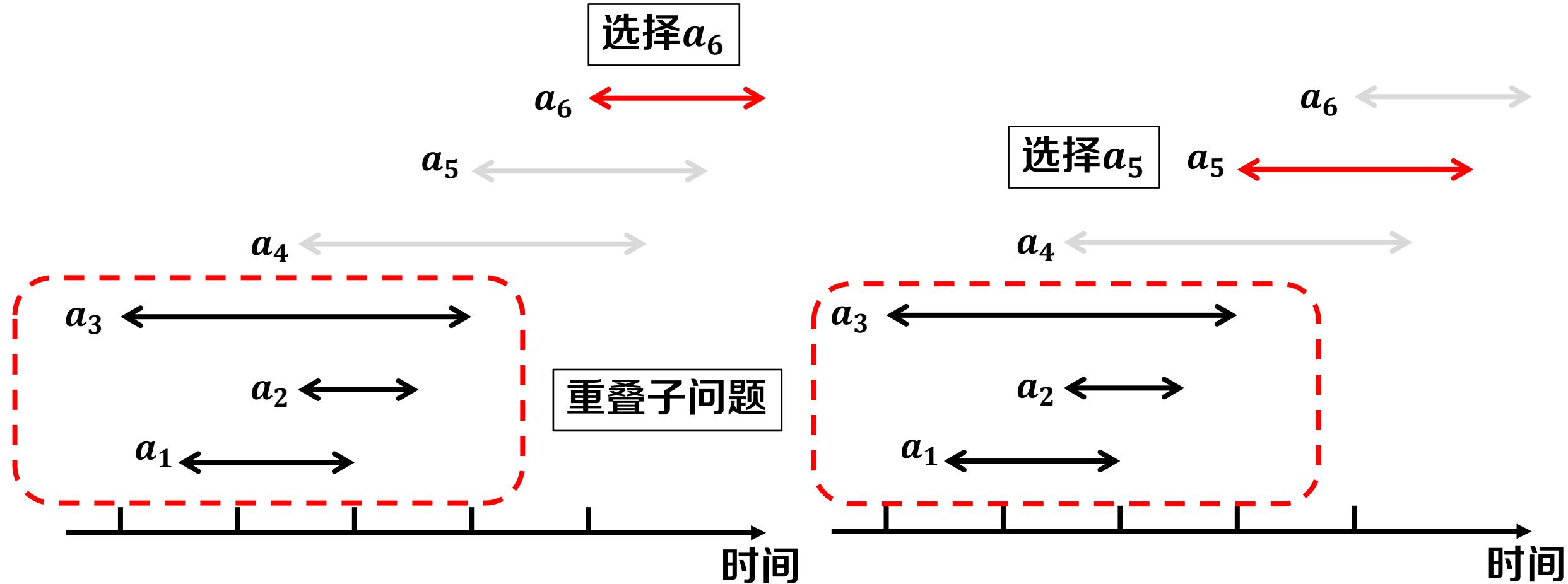
贪心策略不正确

贪心策略正确



时间

# 从贪心策略到动态规划

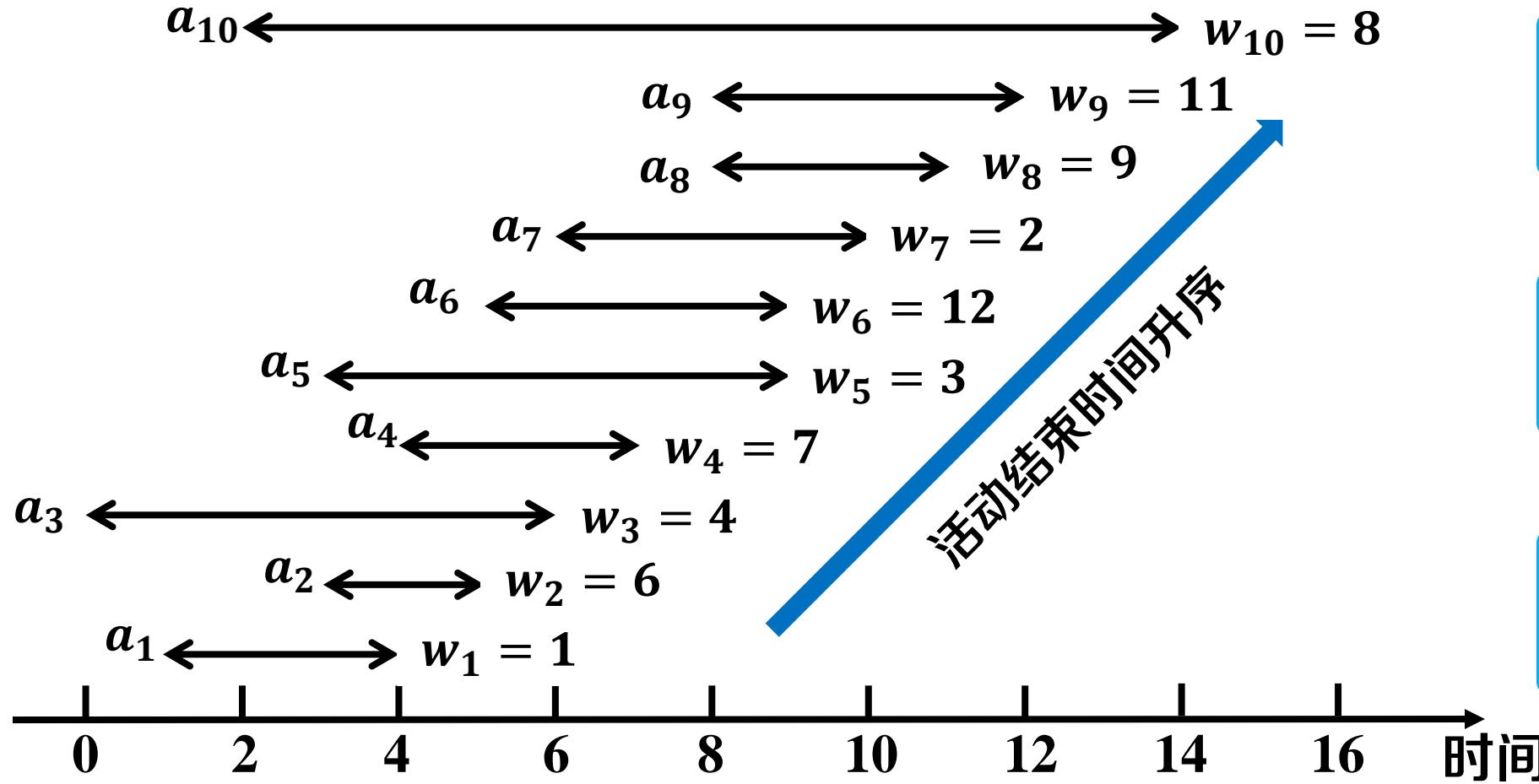


存在重叠子问题，使用动态规划求解

# 问题结构分析

## ● 预处理

- 排序：按活动结束时间升序



问题结构分析

递推关系建立

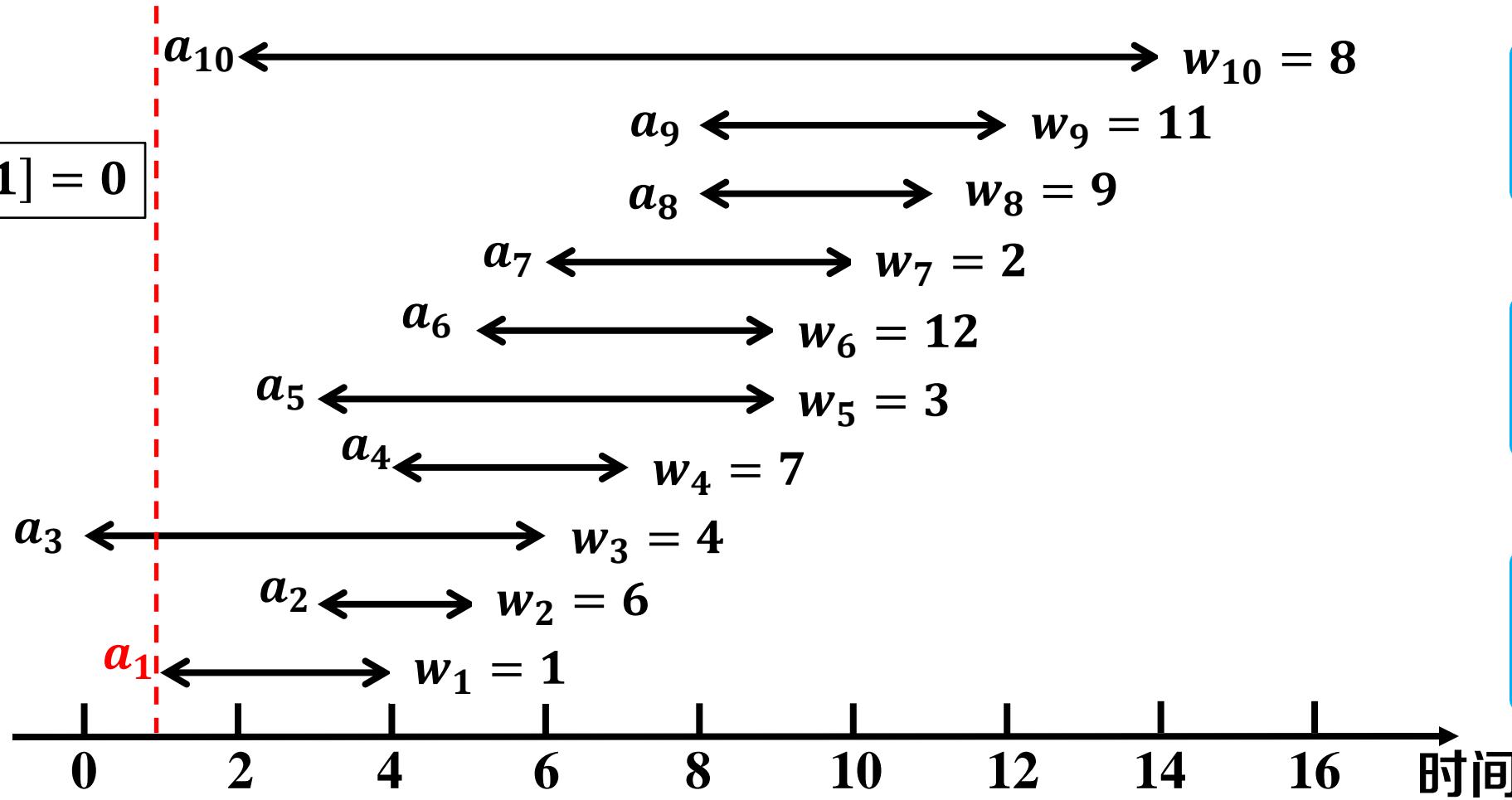
自底向上计算

最优方案追踪

# 问题结构分析

## ● 预处理

- 排序: 按活动结束时间升序
- 求 $p[i]$ : 在 $a_i$ 开始前最后结束的活动



问题结构分析

递推关系建立

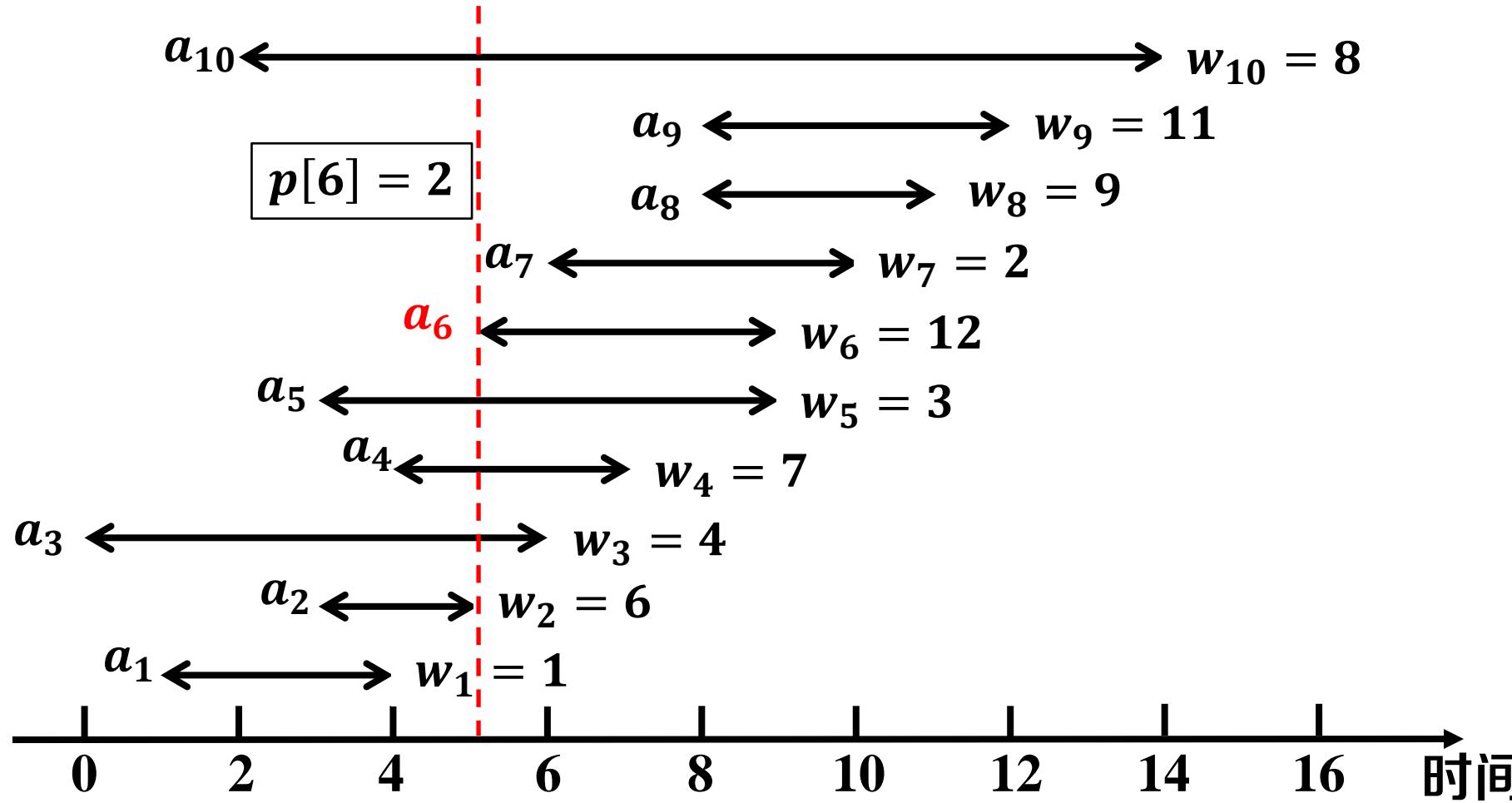
自底向上计算

最优方案追踪

# 问题结构分析

## ● 预处理

- 排序: 按活动结束时间升序
- 求 $p[i]$ : 在 $a_i$ 开始前最后结束的活动



问题结构分析

递推关系建立

自底向上计算

最优方案追踪



# 问题结构分析

## ● 预处理

- 排序: 按活动结束时间升序
- 求 $p[i]$ : 在 $a_i$ 开始前最后结束的活动
  - 如何求解 $p[i]$ ?
  - 排序后使用二分查找





# 问题结构分析

- 预处理

- 排序：按活动结束时间升序
- 求 $p[i]$ ：在 $a_i$ 开始前最后结束的活动

- 给出问题表示

- $D[i]$ ：集合 $\{a_1, a_2, a_3, \dots, a_i\}$ 中不冲突活动最大权重和

- 明确原始问题

- $D[n]$ ：集合 $\{a_1, a_2, a_3, \dots, a_n\}$ 中不冲突活动最大权重和

问题结构分析

递推关系建立

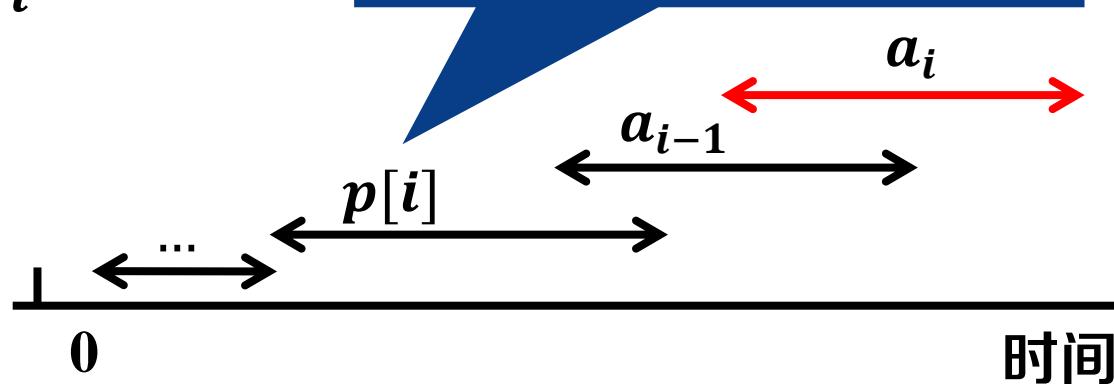
自底向上计算

最优方案追踪

# 递推关系建立：分析最优（子）结构

- 考察活动 $a_i$ 
  - 选择 $a_i$

在 $a_i$ 开始前最后结束的活动



问题结构分析

递推关系建立

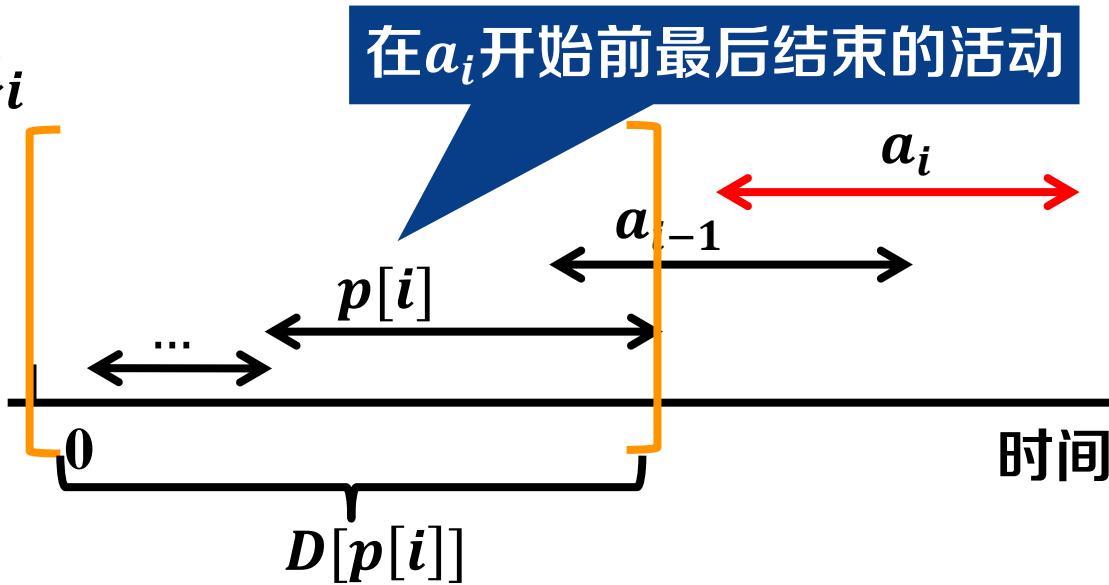
自底向上计算

最优方案追踪

# 递推关系建立：分析最优（子）结构

- 考察活动 $a_i$

- 选择 $a_i$



问题结构分析

递推关系建立

自底向上计算

最优方案追踪

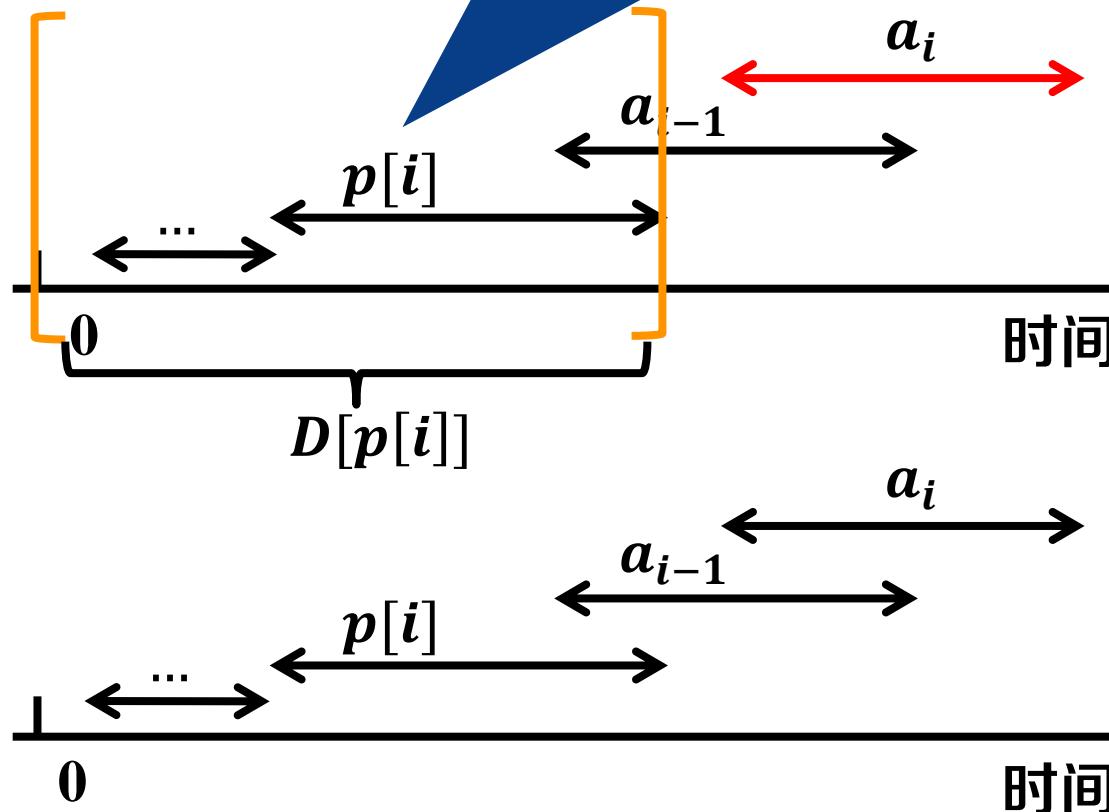
# 递推关系建立：分析最优（子）结构

- 考察活动  $a_i$

- 选择  $a_i$

- 不选  $a_i$

在  $a_i$  开始前最后结束的活动



问题结构分析

递推关系建立

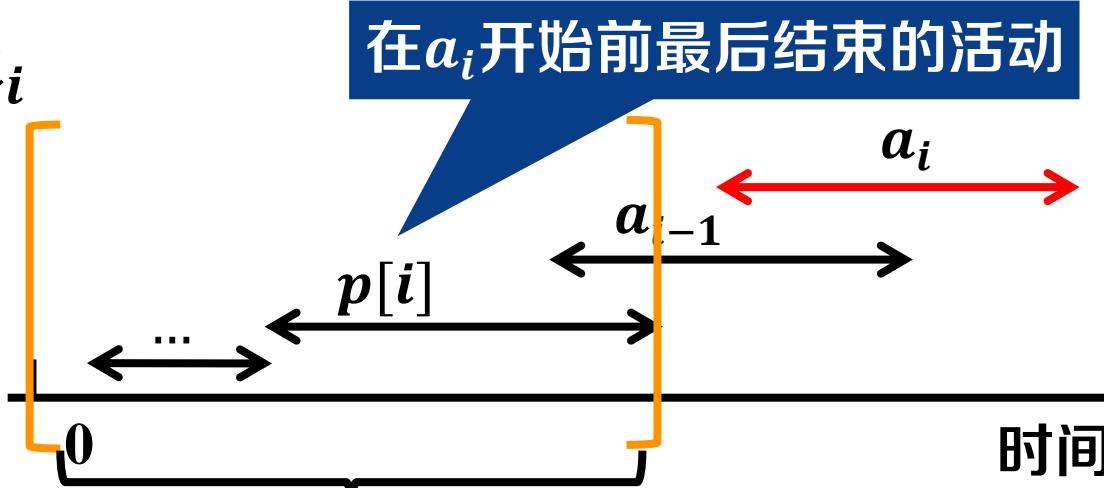
自底向上计算

最优方案追踪

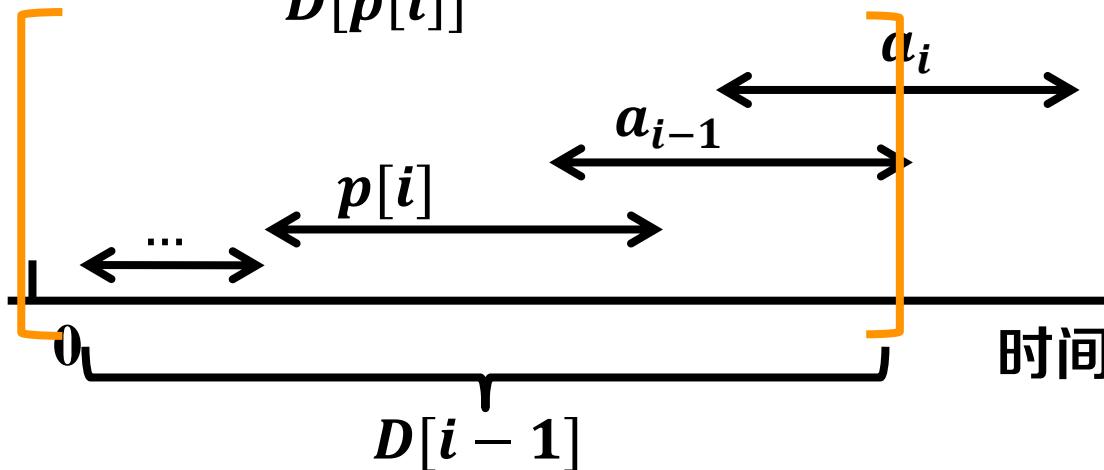
# 递推关系建立：分析最优（子）结构

- 考察活动 $a_i$

- 选择 $a_i$



- 不选 $a_i$



问题结构分析

递推关系建立

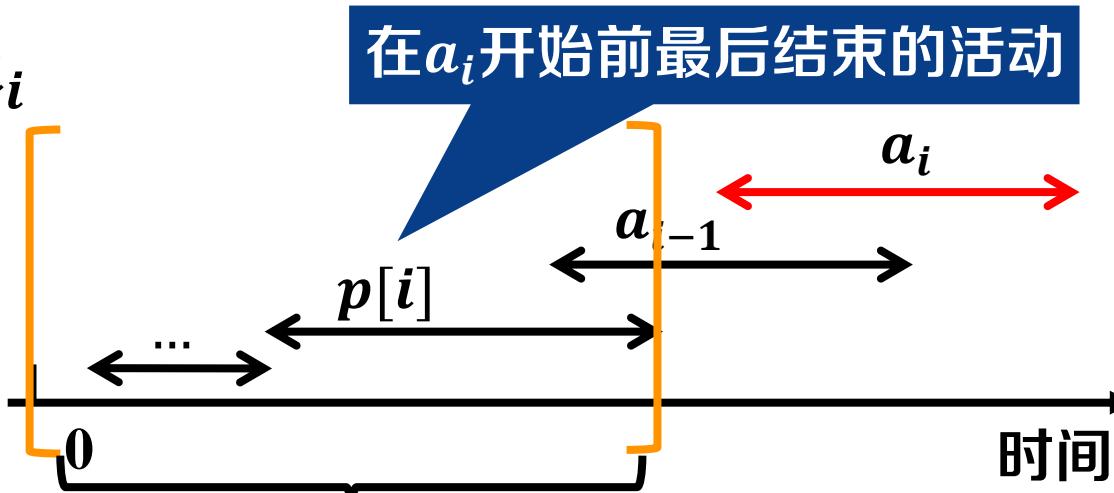
自底向上计算

最优方案追踪

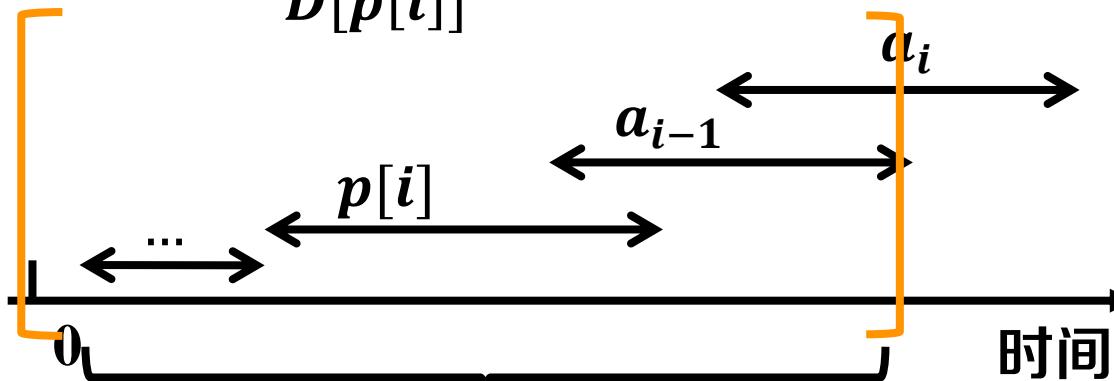
# 递推关系建立：构造递推公式

- 考察活动  $a_i$

- 选择  $a_i$



- 不选  $a_i$



- 递推公式

- $$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

问题结构分析

递推关系建立

自底向上计算

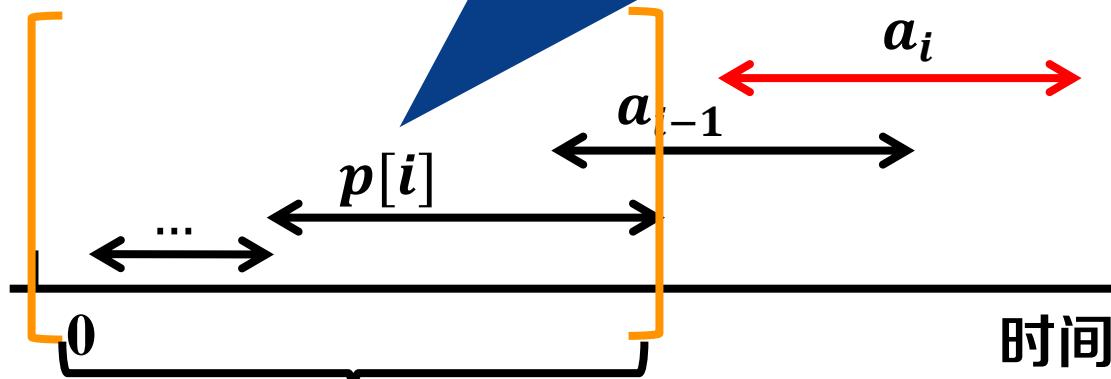
最优方案追踪

# 递推关系建立：构造递推公式

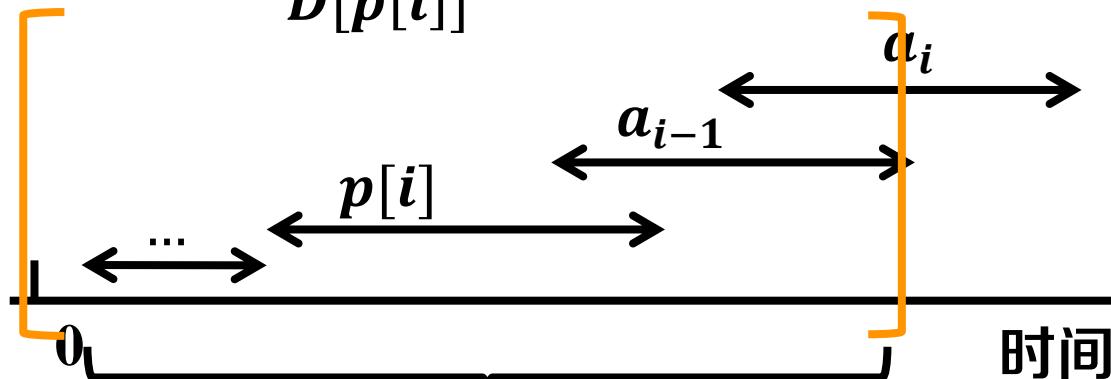
- 考察活动  $a_i$

- 选择  $a_i$

在  $a_i$  开始前最后结束的活动



- 不选  $a_i$



- 递推公式

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

最优子结构



# 自底向上计算：确定计算顺序

## ● 初始化

- $D[0] = 0$ : 空活动集最大权重和为0



# 自底向上计算：确定计算顺序

- 初始话
  - $D[0] = 0$ : 空活动集最大权重和为0
- 递推公式
  - $D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$

已知

|     | 1 | 2 | ... |  | $i - 1$ | $i$   | ... | $n$ |
|-----|---|---|-----|--|---------|-------|-----|-----|
| $w$ |   |   |     |  |         | $w_i$ |     |     |

$p[i]$

|     | 1 | 2 | ... |  | $i - 1$ | $i$    | ... | $n$ |
|-----|---|---|-----|--|---------|--------|-----|-----|
| $p$ |   |   |     |  |         | $p[i]$ |     |     |

$D[i]$

|     | 0 | ... | $p[i]$    | ... | $i - 1$    | $i$ | ... | $n$ |
|-----|---|-----|-----------|-----|------------|-----|-----|-----|
| $D$ | 0 |     | $D[p[i]]$ |     | $D[i - 1]$ |     |     |     |

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

# 自底向上计算：确定计算顺序

- 初始话

- $D[0] = 0$ : 空活动集最大权重和为0

- 递推公式

- $D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$

已知

|     | 1 | 2 | ... |  | $i - 1$ | $i$   | ... | $n$ |
|-----|---|---|-----|--|---------|-------|-----|-----|
| $w$ |   |   |     |  |         | $w_i$ |     |     |

$p[i]$

|     | 1 | 2 | ... |  | $i - 1$ | $i$    | ... | $n$ |
|-----|---|---|-----|--|---------|--------|-----|-----|
| $p$ |   |   |     |  |         | $p[i]$ |     |     |

$D[i]$

|     | 0 | ... | $p[i]$    | ... | $i - 1$    | $i$    | ... | $n$ |
|-----|---|-----|-----------|-----|------------|--------|-----|-----|
| $D$ | 0 |     | $D[p[i]]$ |     | $D[i - 1]$ | $D[i]$ |     |     |

问题结构分析

递推关系建立

自底向上计算

最优方案追踪



# 自底向上计算：依次求解问题

## ● 初始化

- $D[0] = 0$ : 空活动集最大权重和为0

## ● 递推公式

- $D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$

已知

|     | 1 | 2 | ... |  | $i - 1$ | $i$   | ... | $n$ |
|-----|---|---|-----|--|---------|-------|-----|-----|
| $w$ |   |   |     |  |         | $w_i$ |     |     |

$p[i]$

|     | 1 | 2 | ... |  | $i - 1$ | $i$    | ... | $n$ |
|-----|---|---|-----|--|---------|--------|-----|-----|
| $p$ |   |   |     |  |         | $p[i]$ |     |     |

自底向上计算

$D[i]$

|     | 0 | ... | $p[i]$ | ... | $i - 1$ | $i$ | ... | $n$ |
|-----|---|-----|--------|-----|---------|-----|-----|-----|
| $D$ | 0 | —   |        |     |         |     |     | ★   |

问题结构分析

递推关系建立

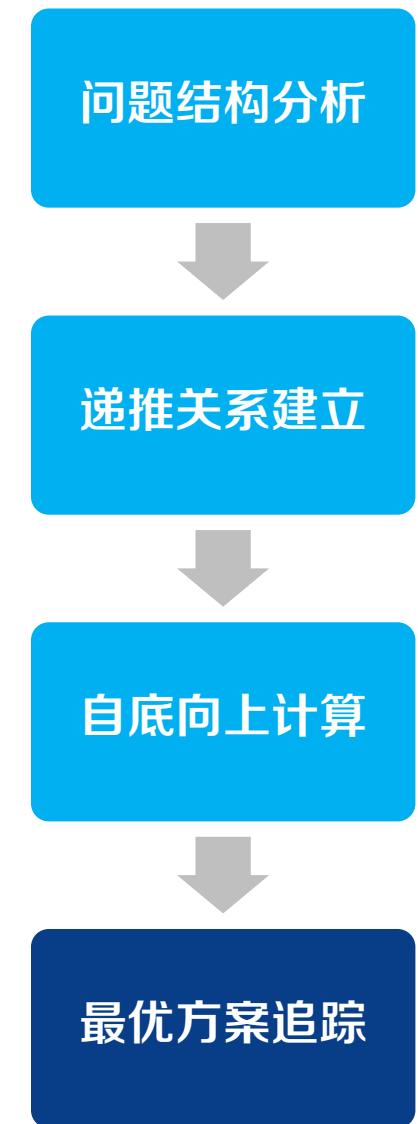
自底向上计算

最优方案追踪



## ● 记录决策过程

- $Rec[i] = \begin{cases} 1, & \text{选择活动 } a_i \\ 0, & \text{不选活动 } a_i \end{cases}$



# 最优方案追踪



## ● 记录决策过程

- $Rec[i] = \begin{cases} 1, & \text{选择活动 } a_i \\ 0, & \text{不选活动 } a_i \end{cases}$
- **输出最优方案**
  - $Rec[i] = 1$  时，选择活动  $a_i$ ，考察子问题  $D[p[i]]$
  - $Rec[i] = 0$  时，不选活动  $a_i$ ，考察子问题  $D[i - 1]$

已求

|     |   |   |     |         |        |     |         |     |
|-----|---|---|-----|---------|--------|-----|---------|-----|
|     | 1 | 2 | ... | $i - 1$ | $i$    | ... | $n - 1$ | $n$ |
| $p$ |   |   |     |         | $p[i]$ |     |         | $i$ |

|       |   |     |        |     |     |     |         |     |
|-------|---|-----|--------|-----|-----|-----|---------|-----|
|       | 1 | ... | $p[i]$ | ... | $i$ | ... | $n - 1$ | $n$ |
| $Rec$ | 1 | 0   | 0      | 0   | 1   | 0   | 1       | 0   |

问题结构分析

递推关系建立

自底向上计算

最优方案追踪

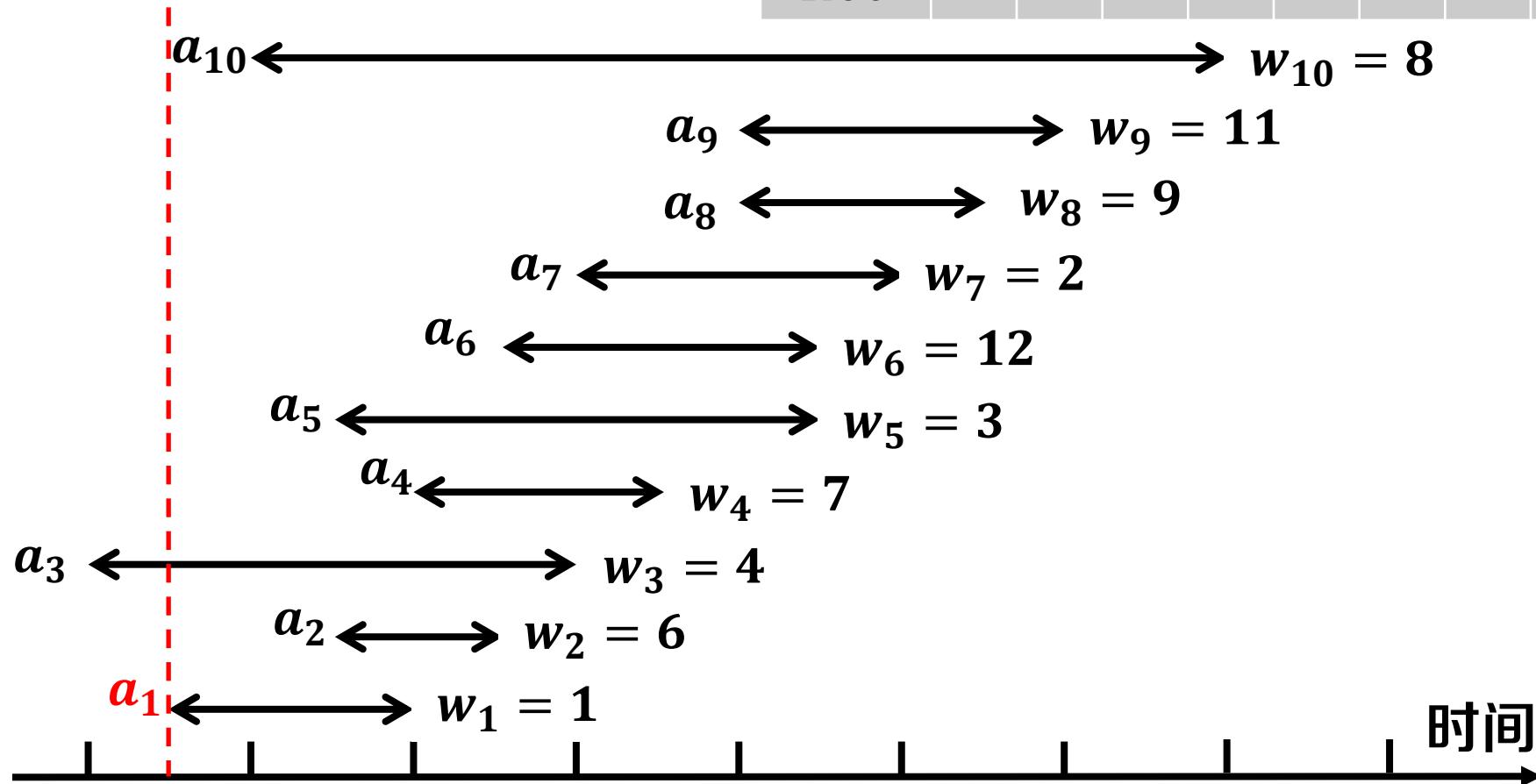
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 |   |   |   |   |   |   |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



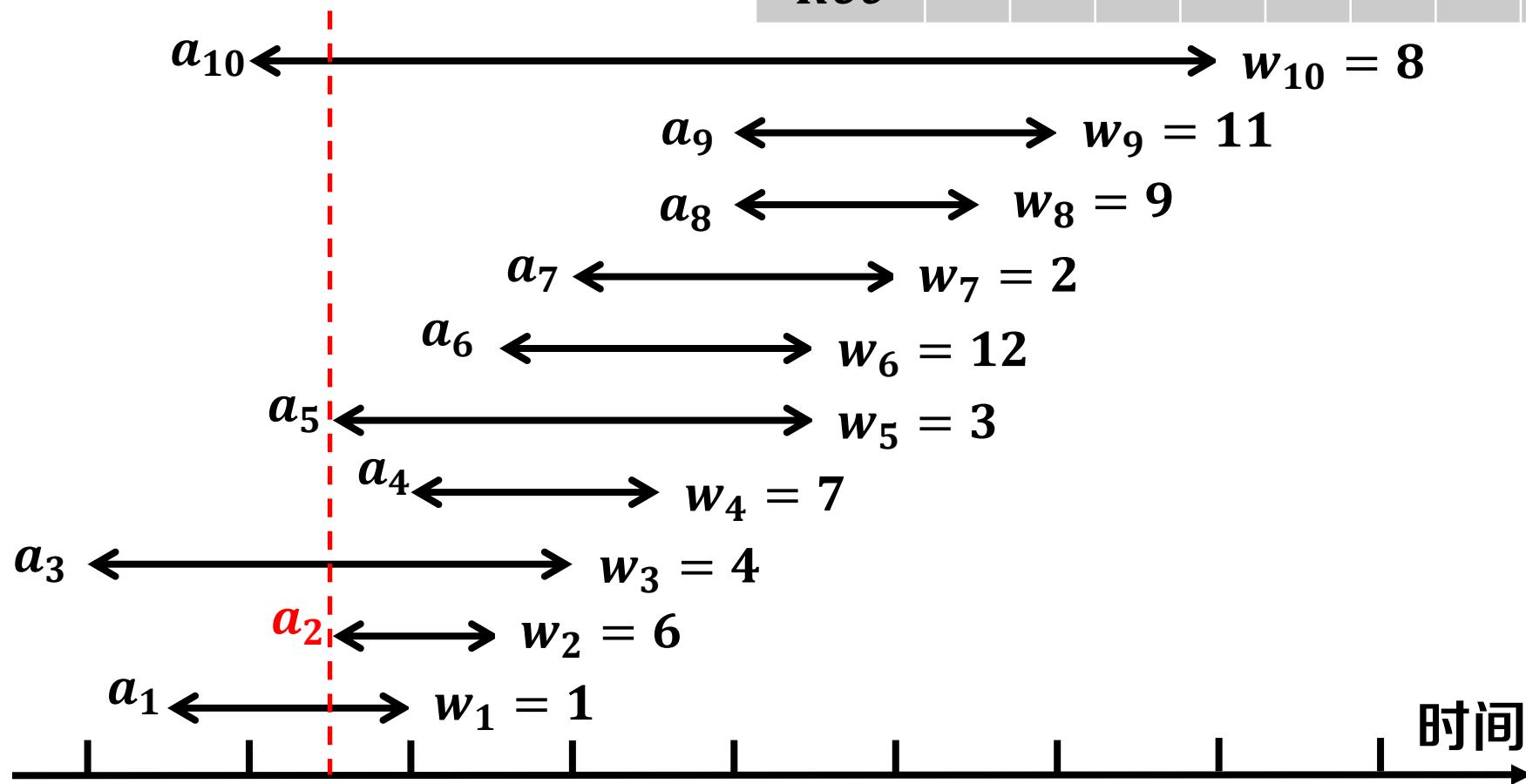
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 |   |   |   |   |   |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



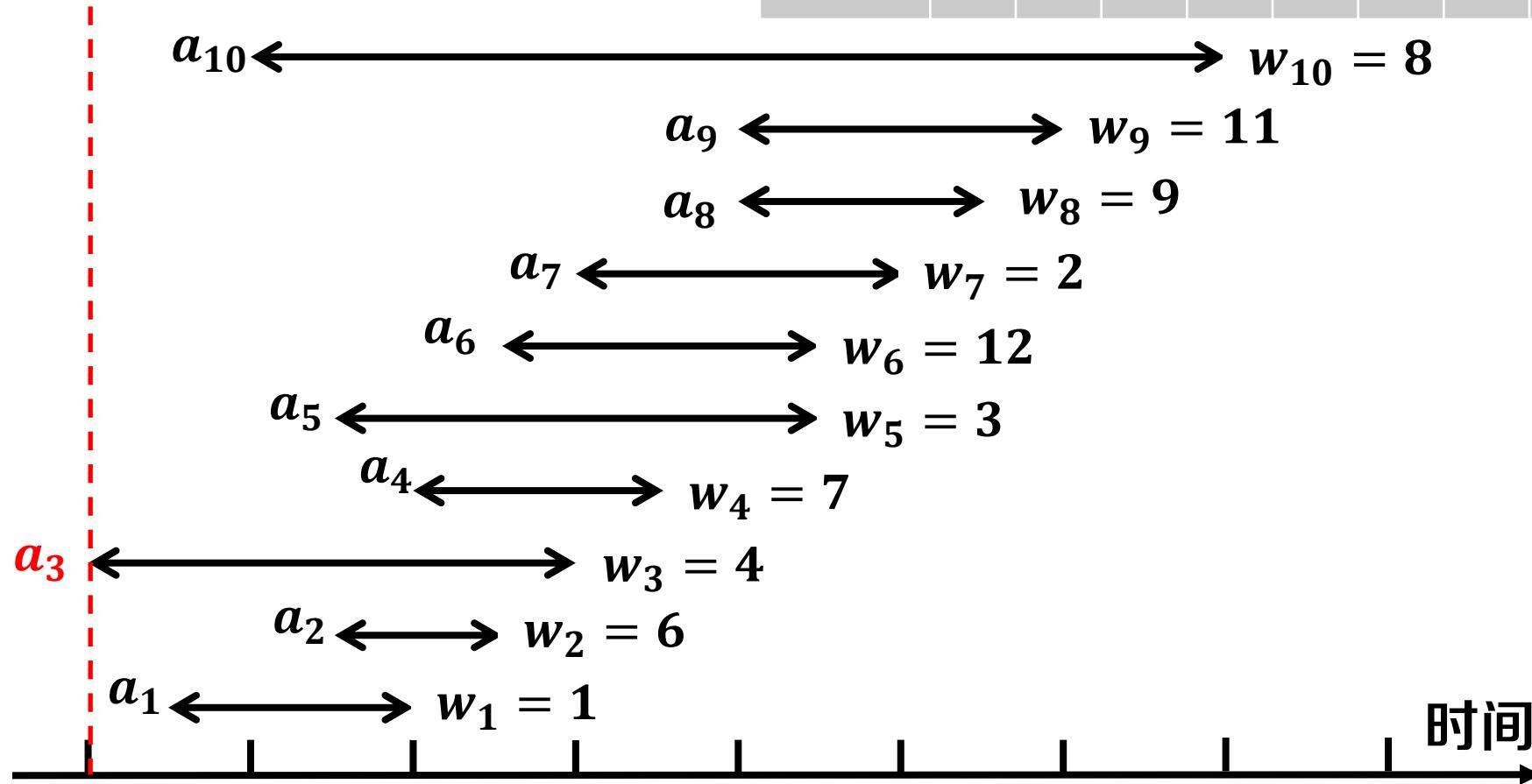
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 |   |   |   |   |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



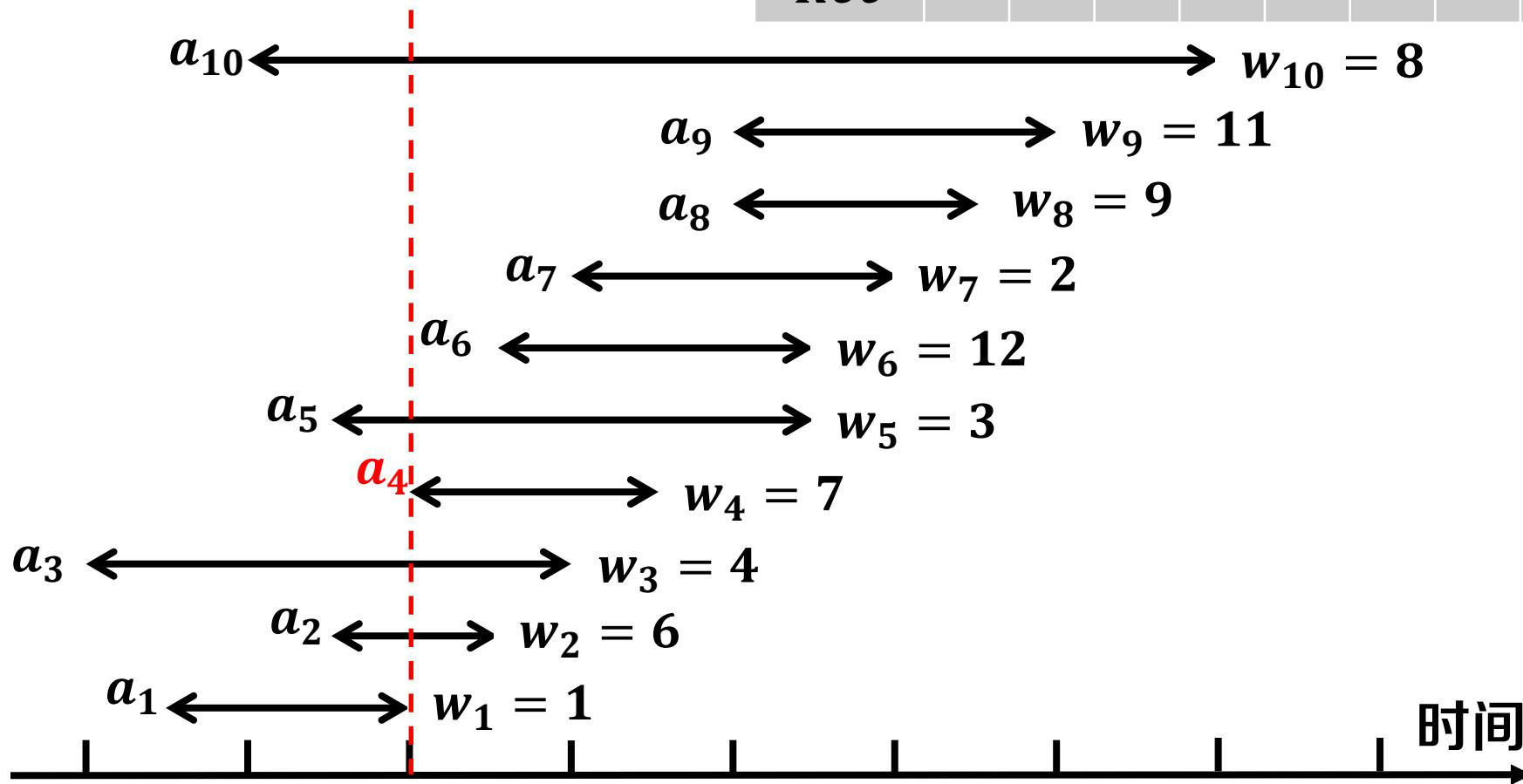
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 |   |   |   |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



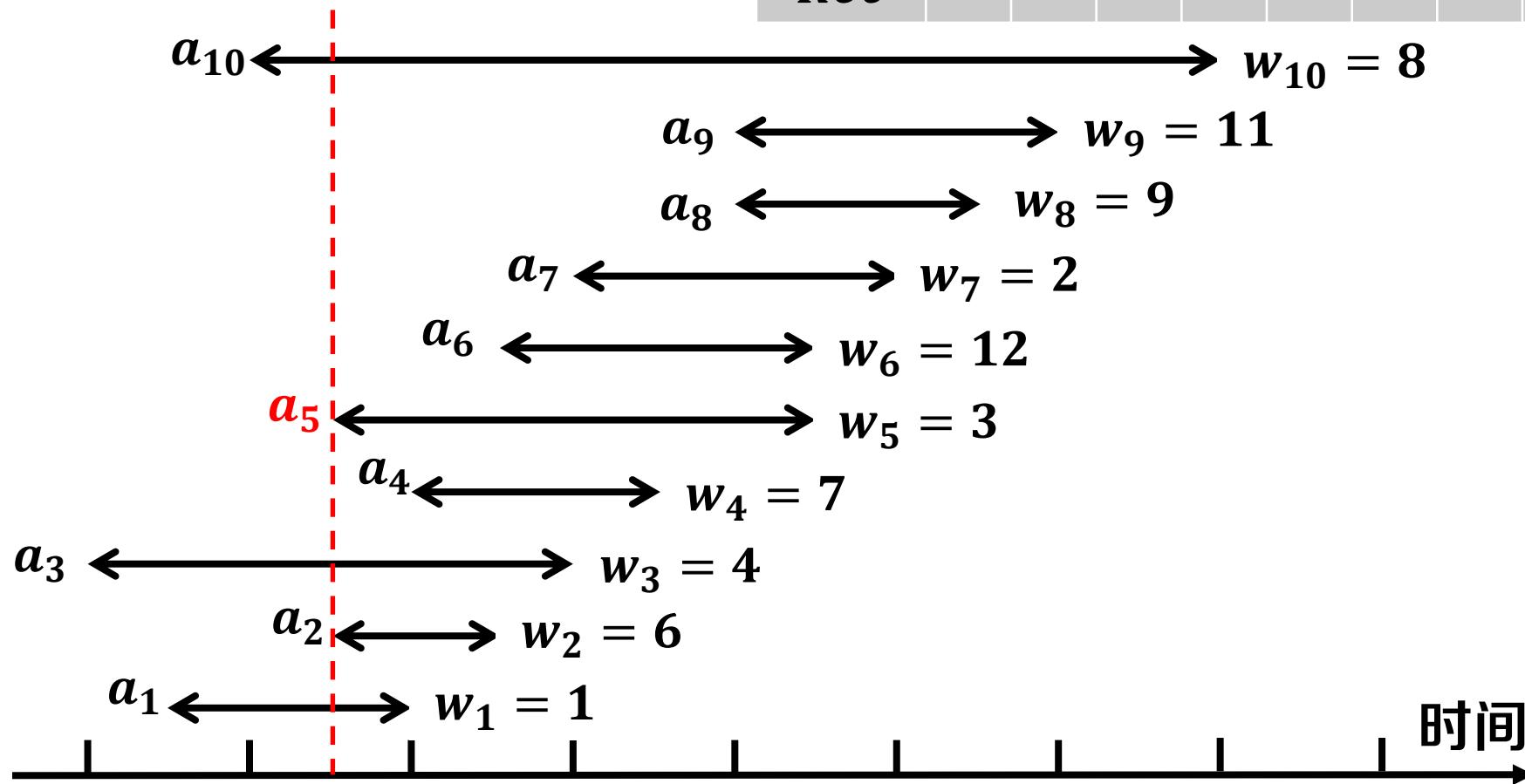
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 |   |   |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



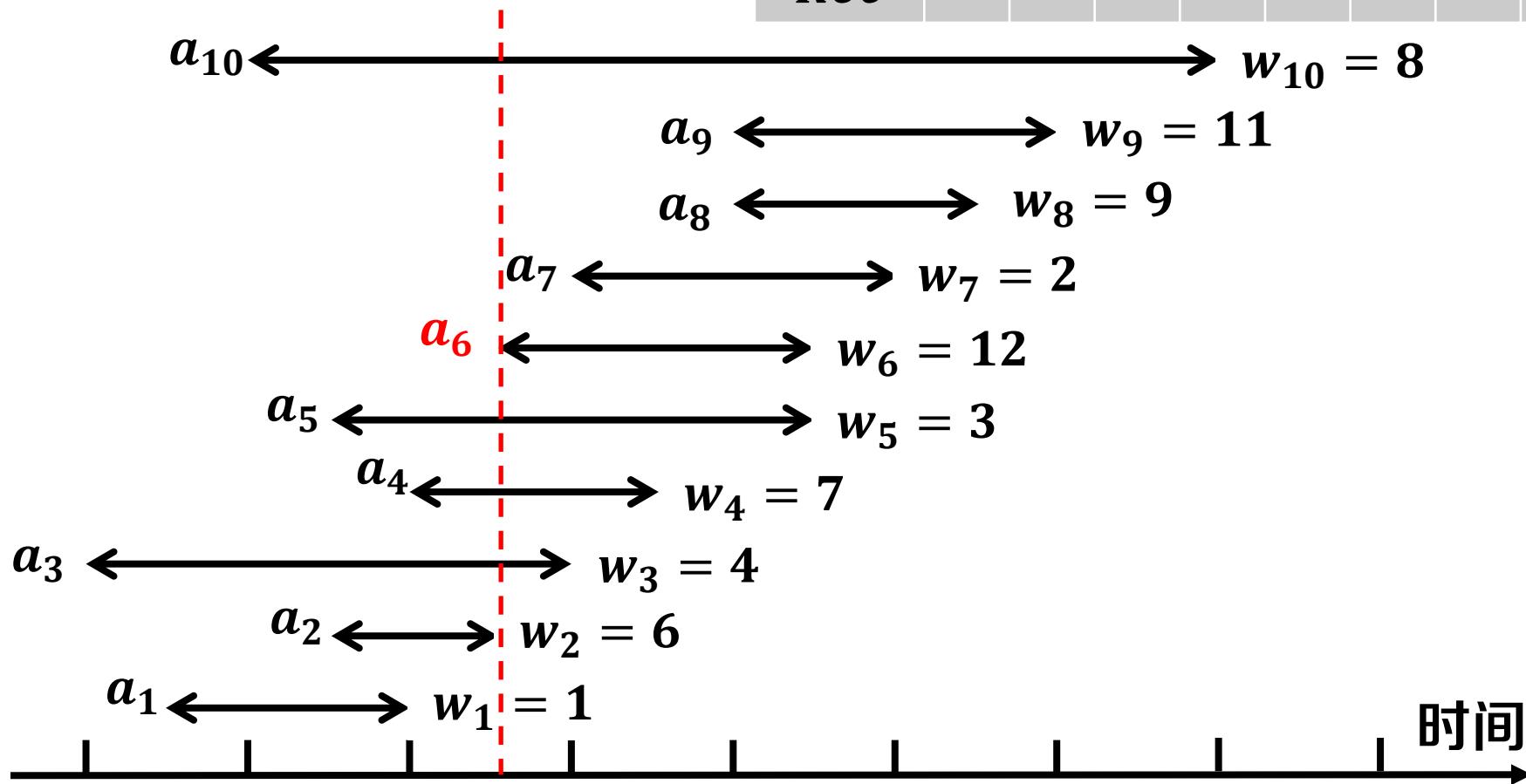
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 |   |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



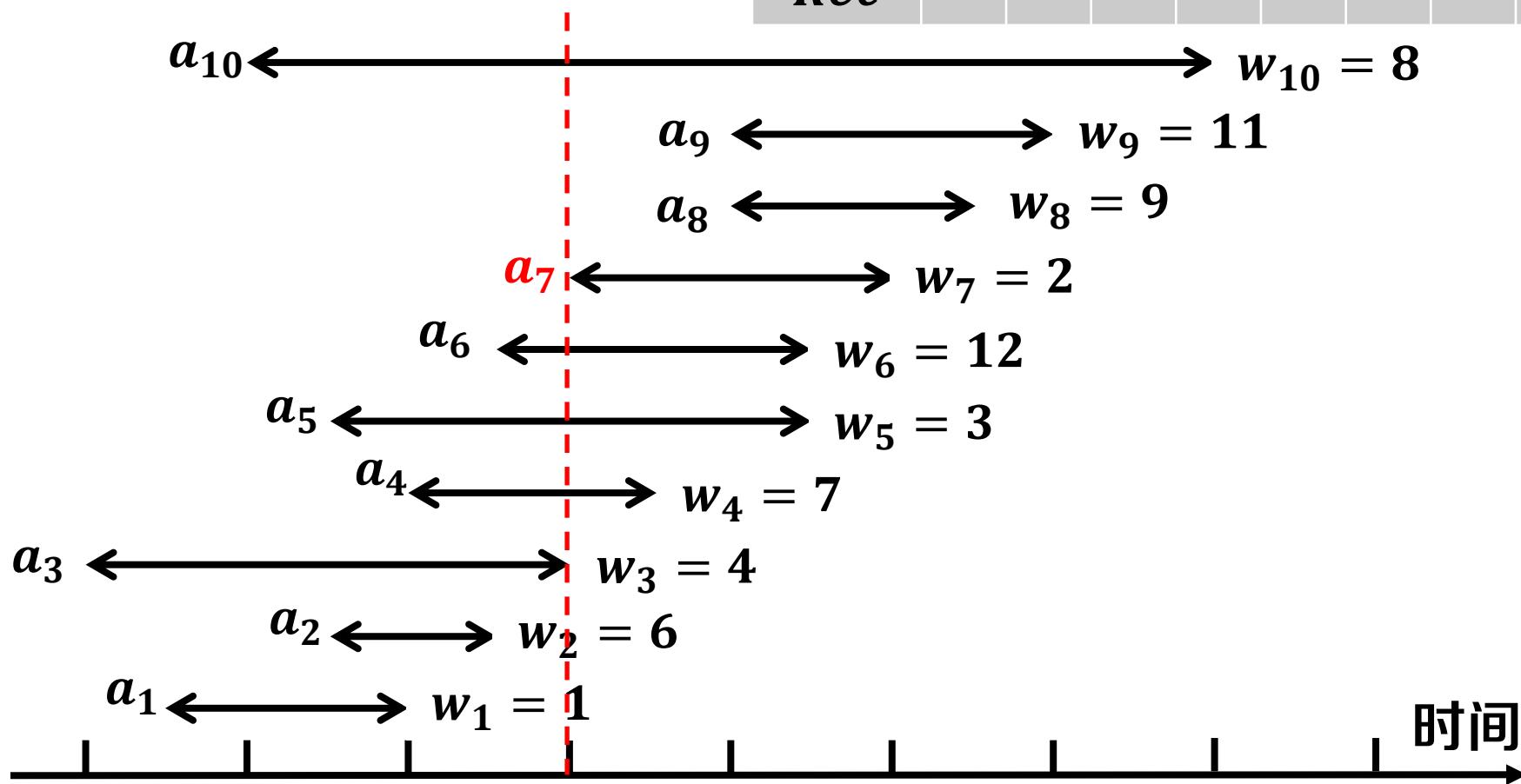
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 |   |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



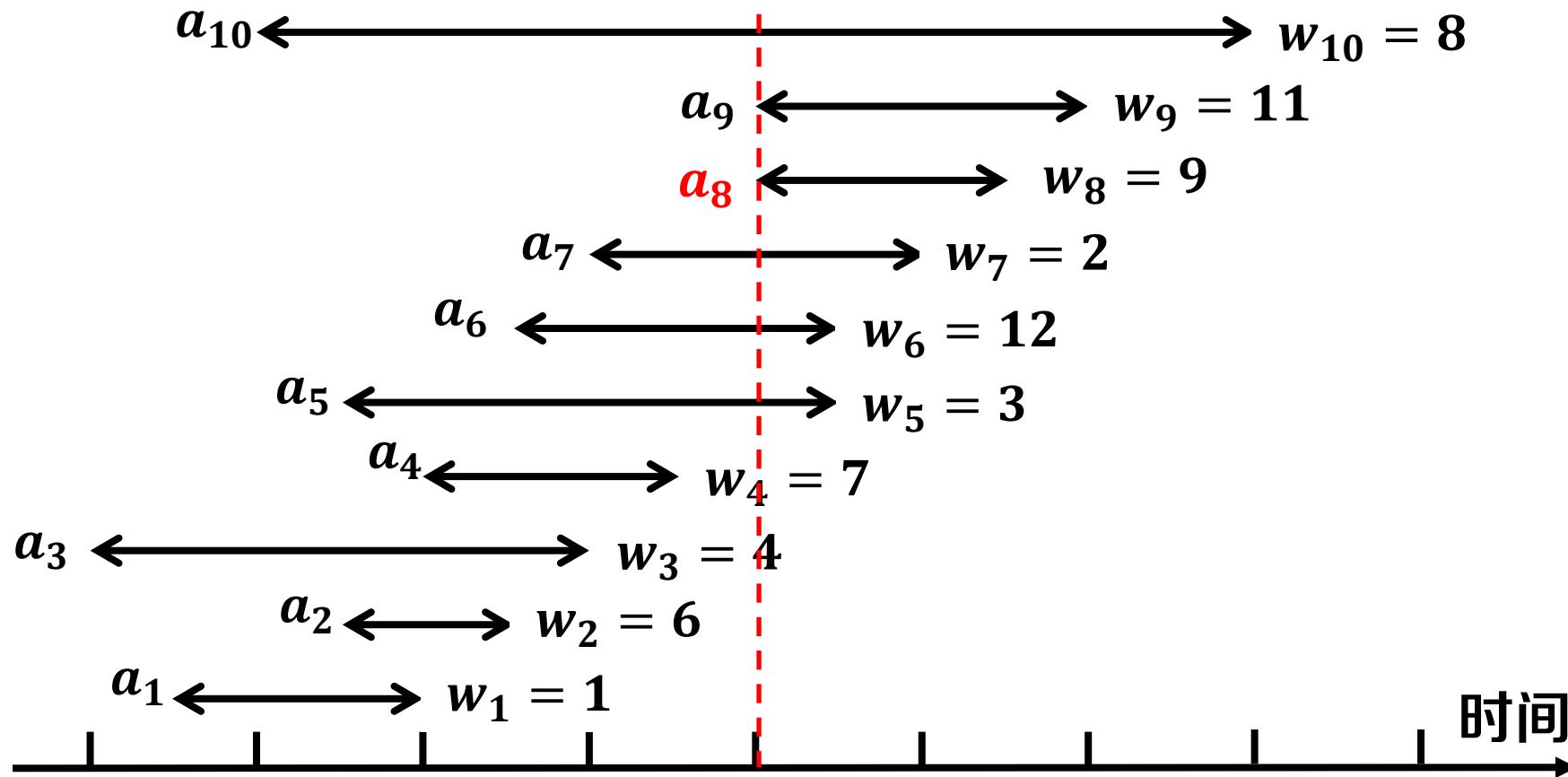
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 |   |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



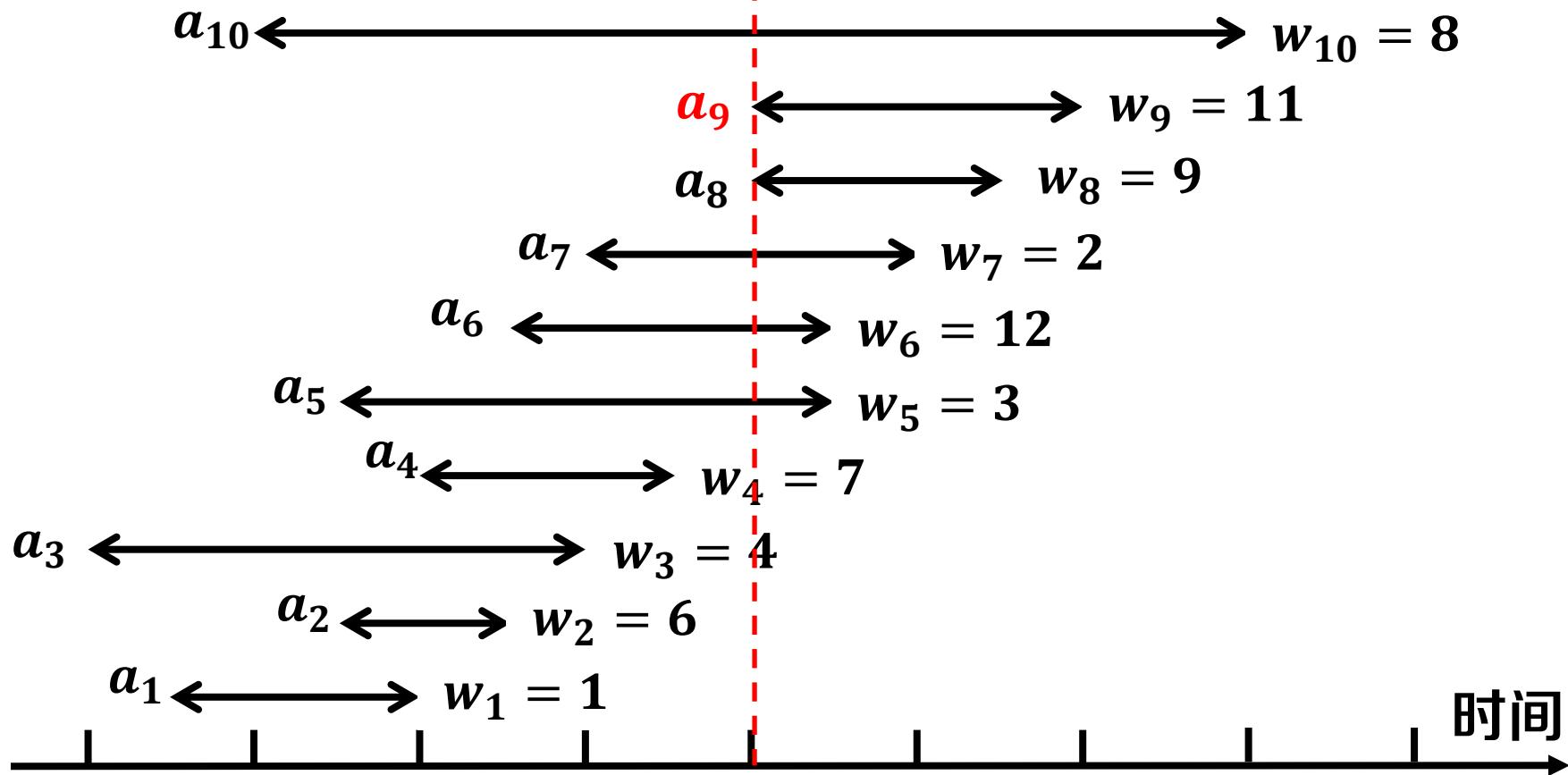
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 |    |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



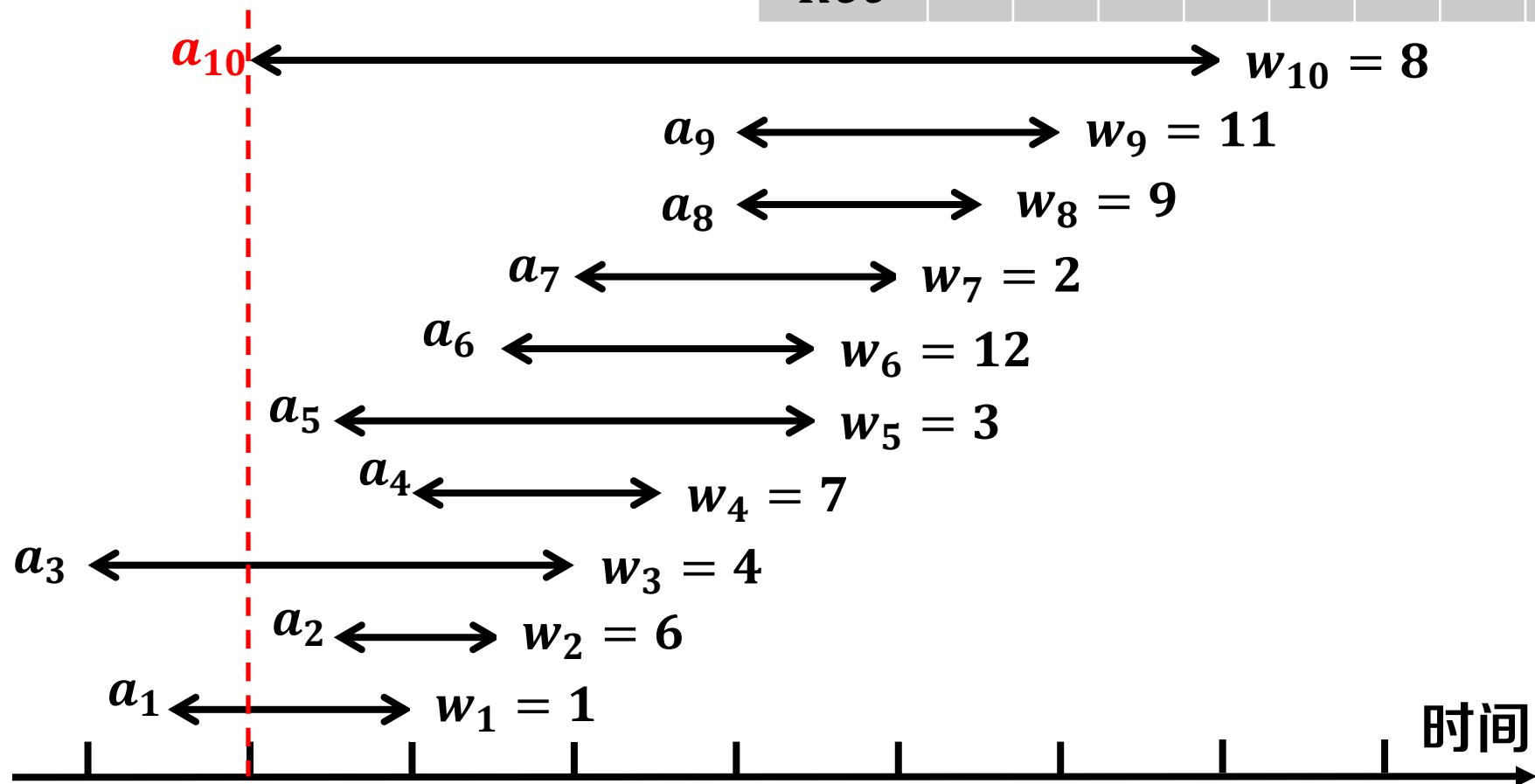
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ |   |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



# 算法实例

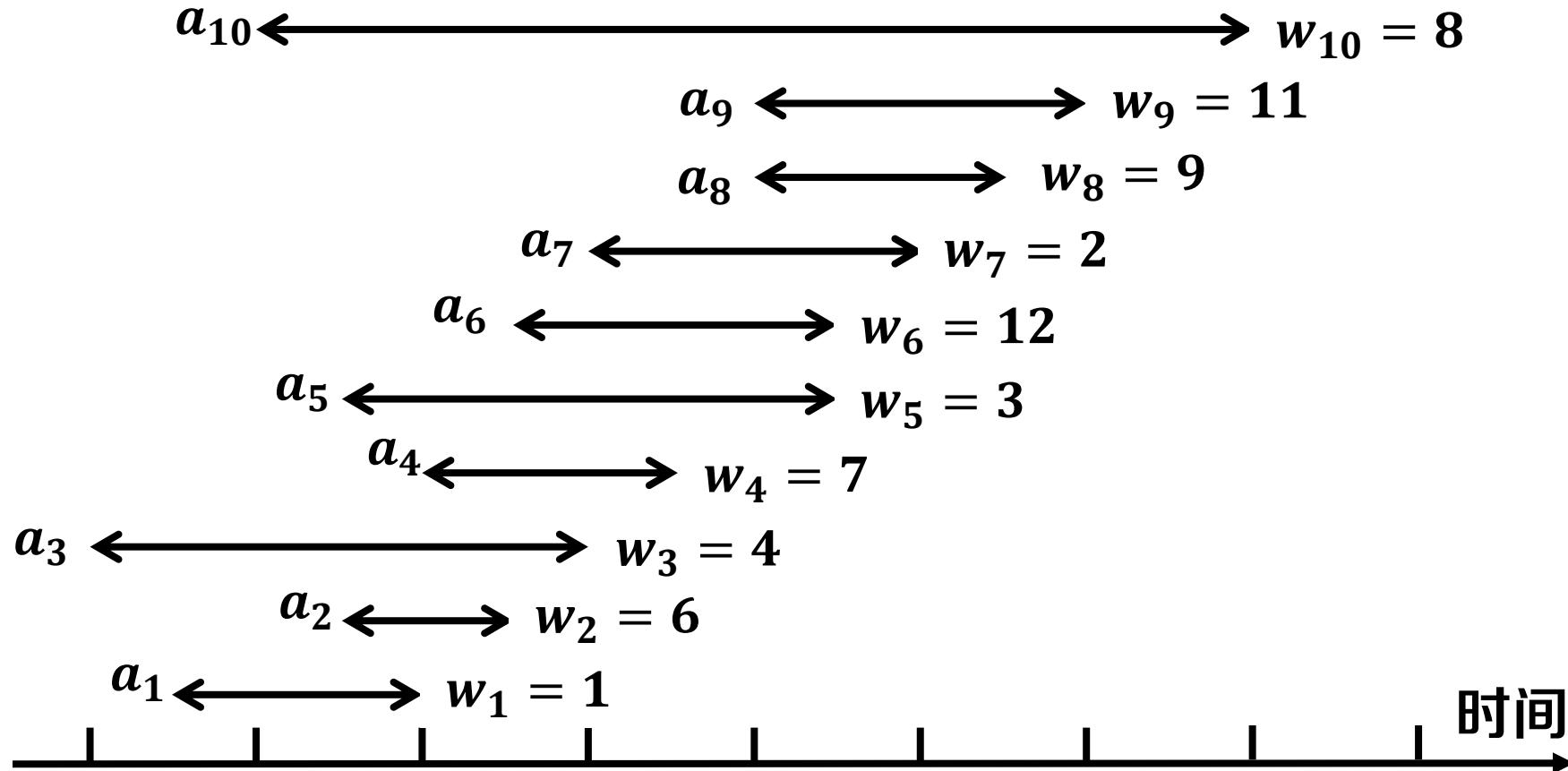
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

初始化

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



# 算法实例

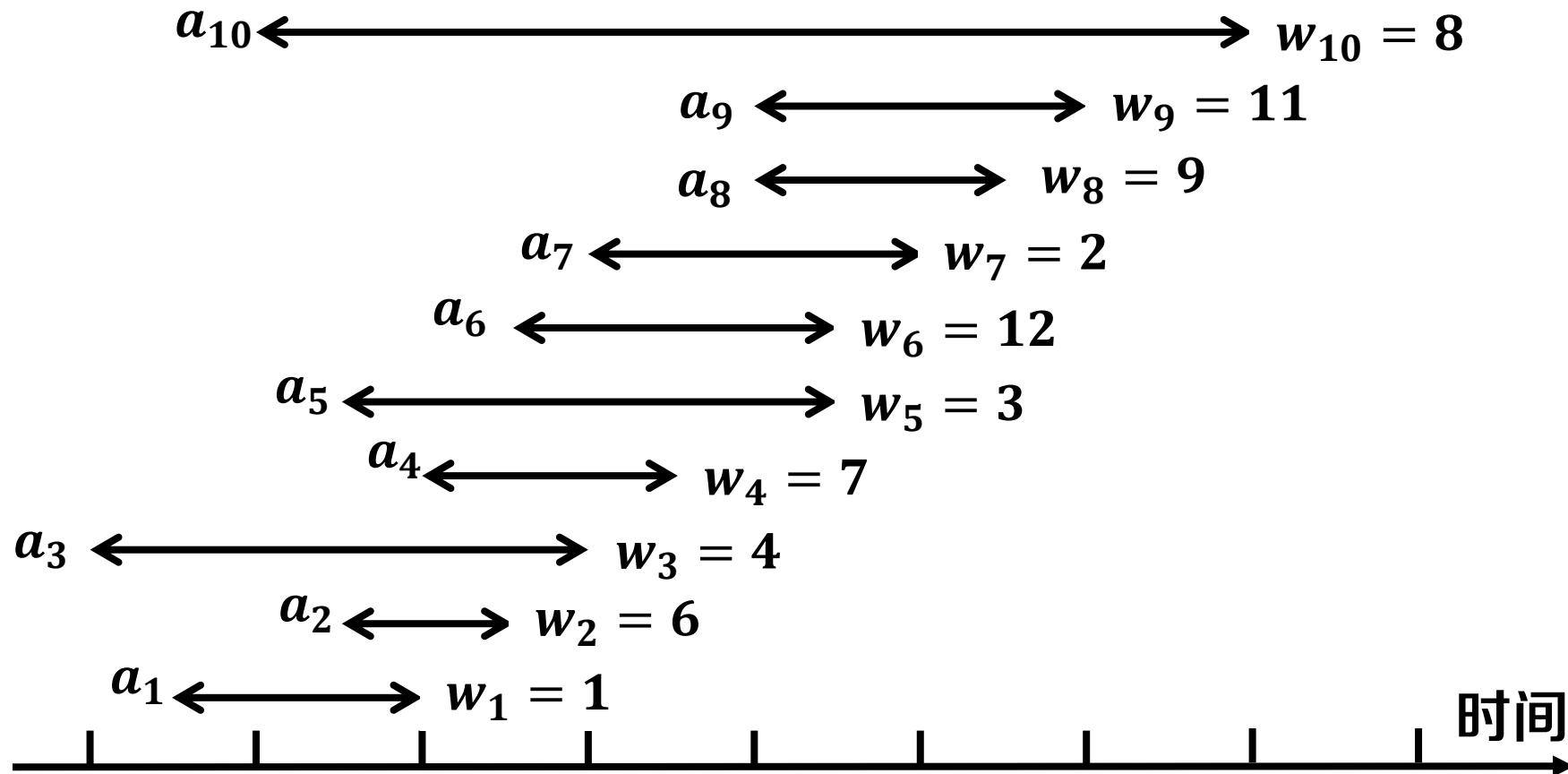
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  |



# 算法实例

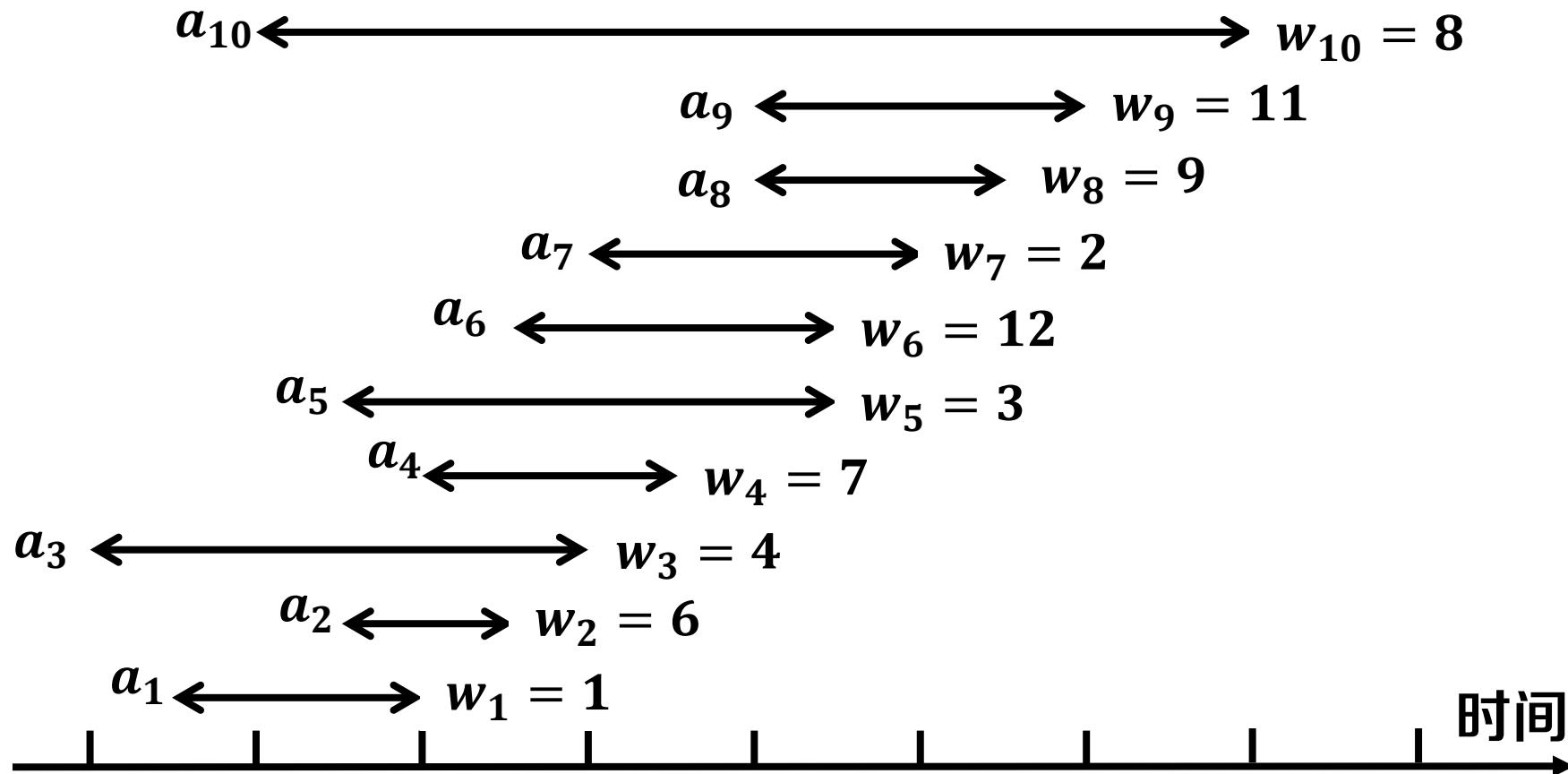
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  |



# 算法实例

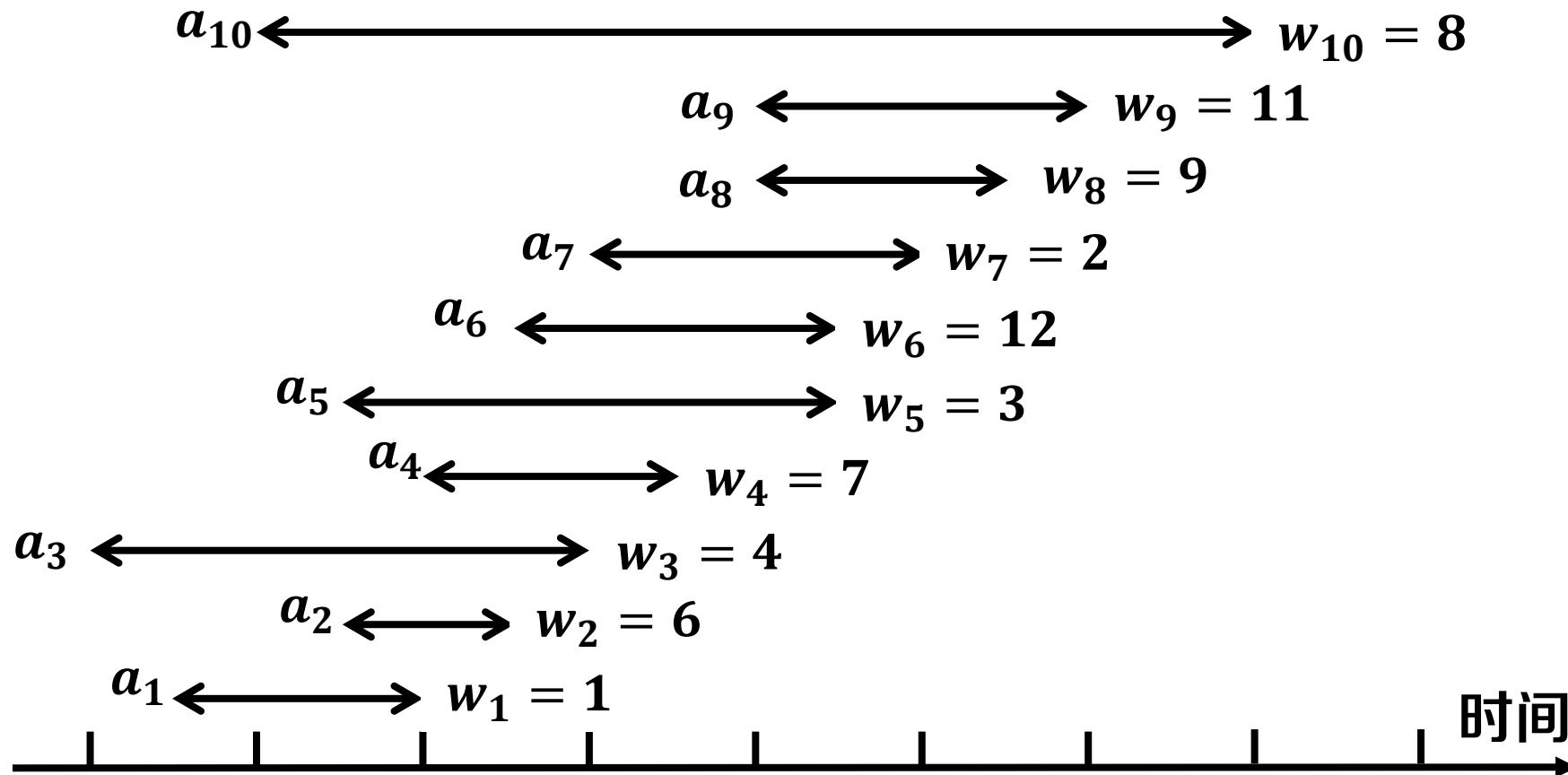
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



# 算法实例

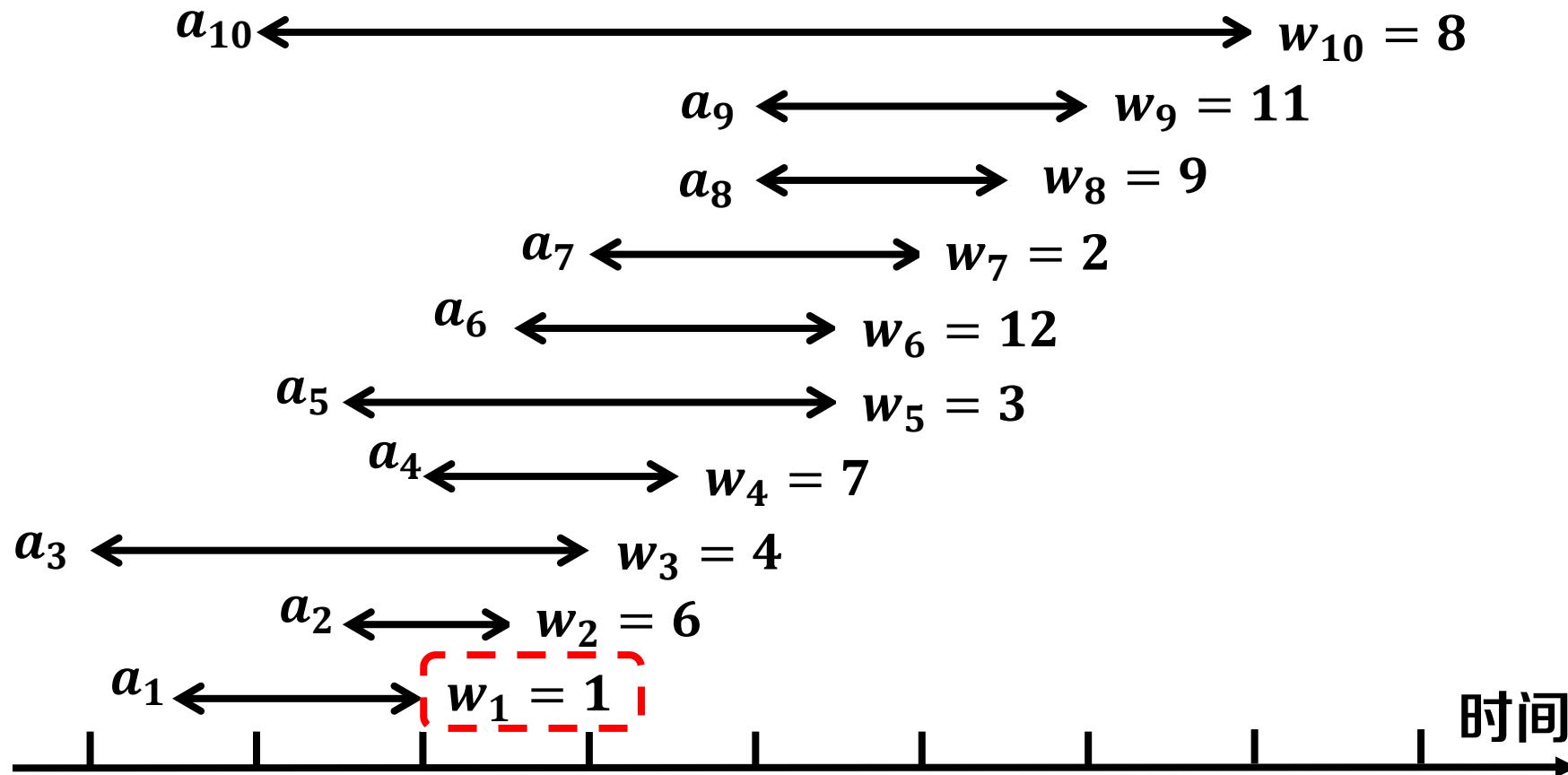
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



# 算法实例

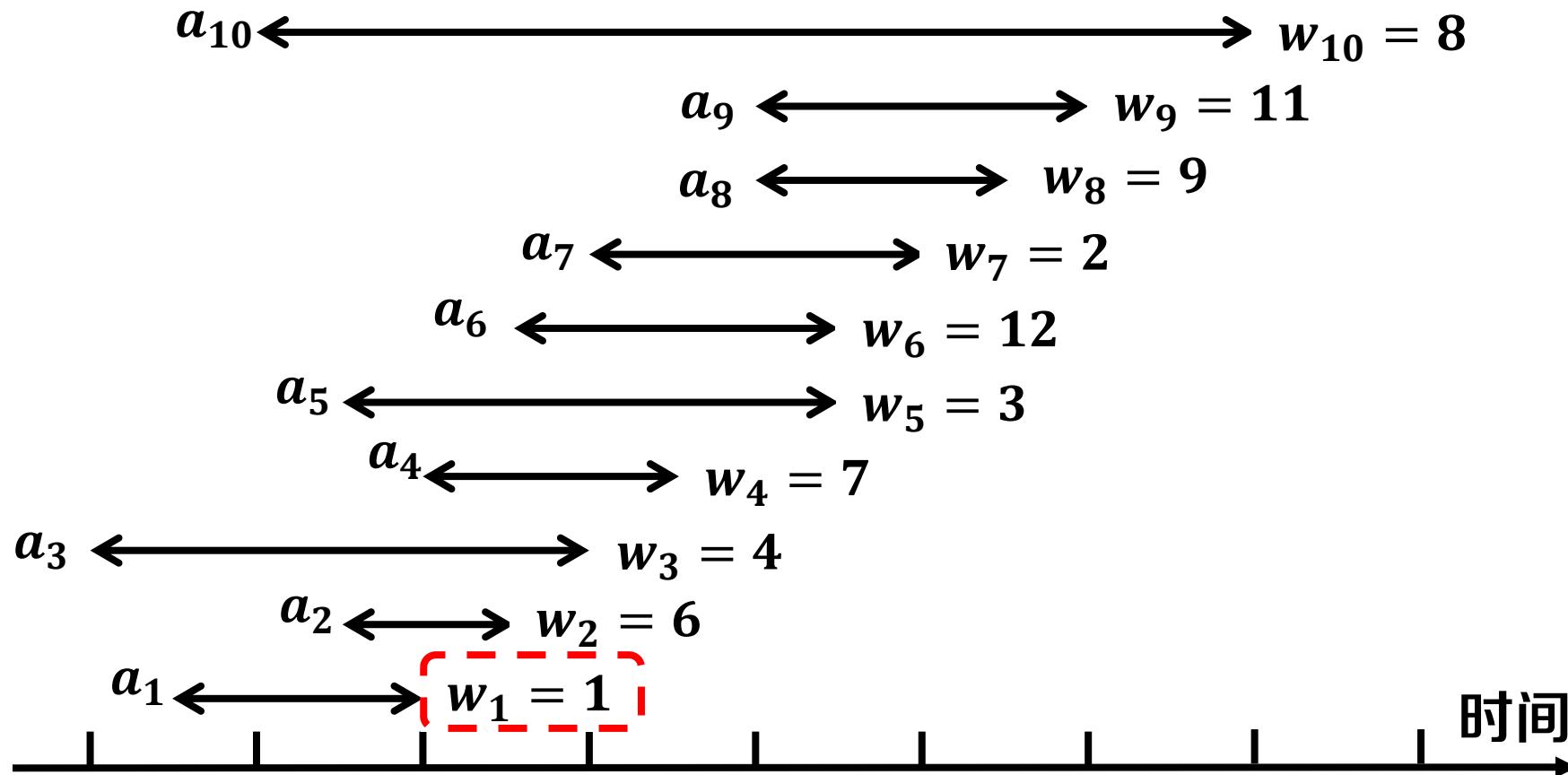
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 |   |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ |   |   |   |   |   |   |   |   |   |    |



# 算法实例

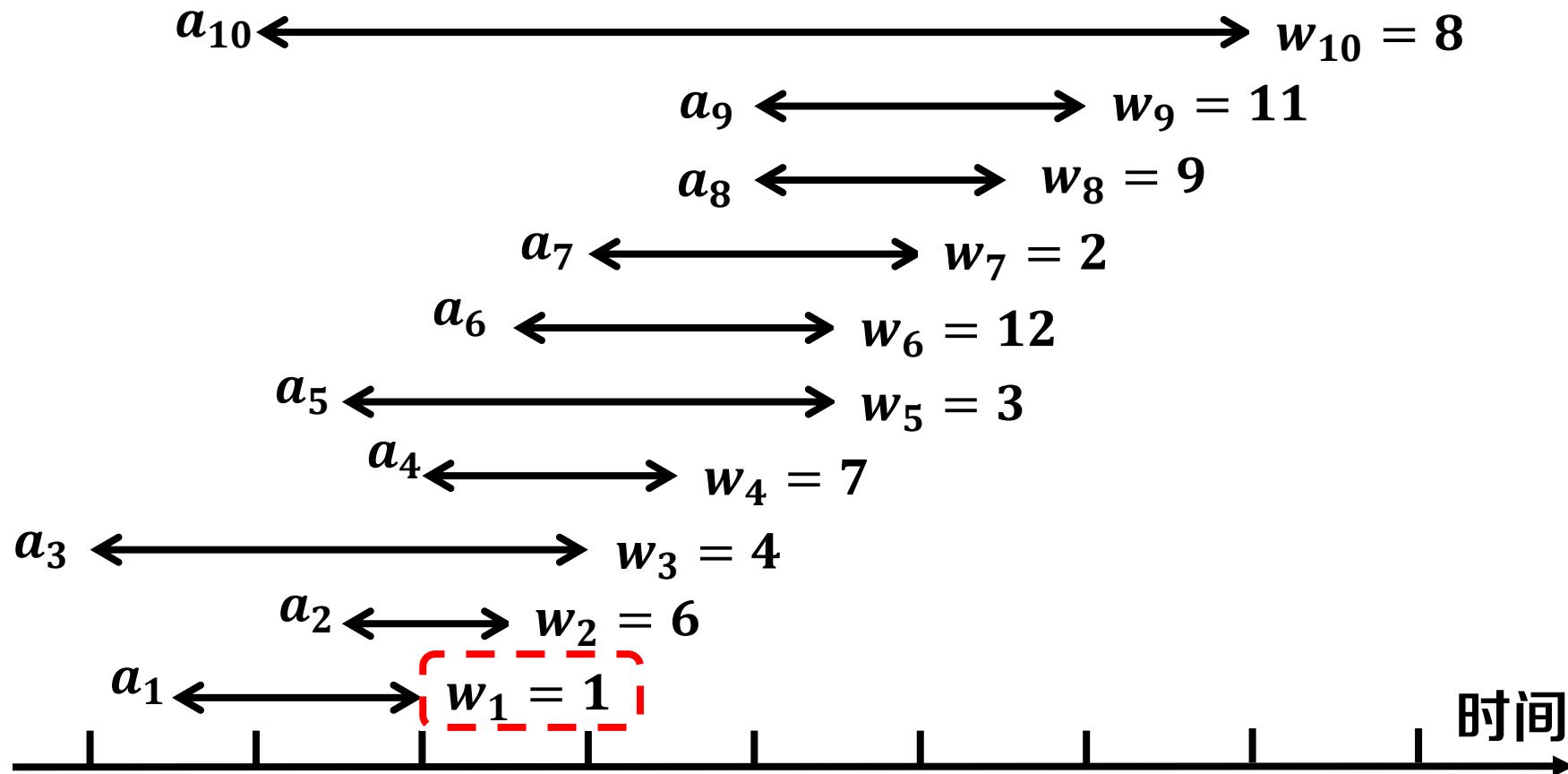
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 1 |   |   |   |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 |   |   |   |   |   |   |   |   |    |



# 算法实例

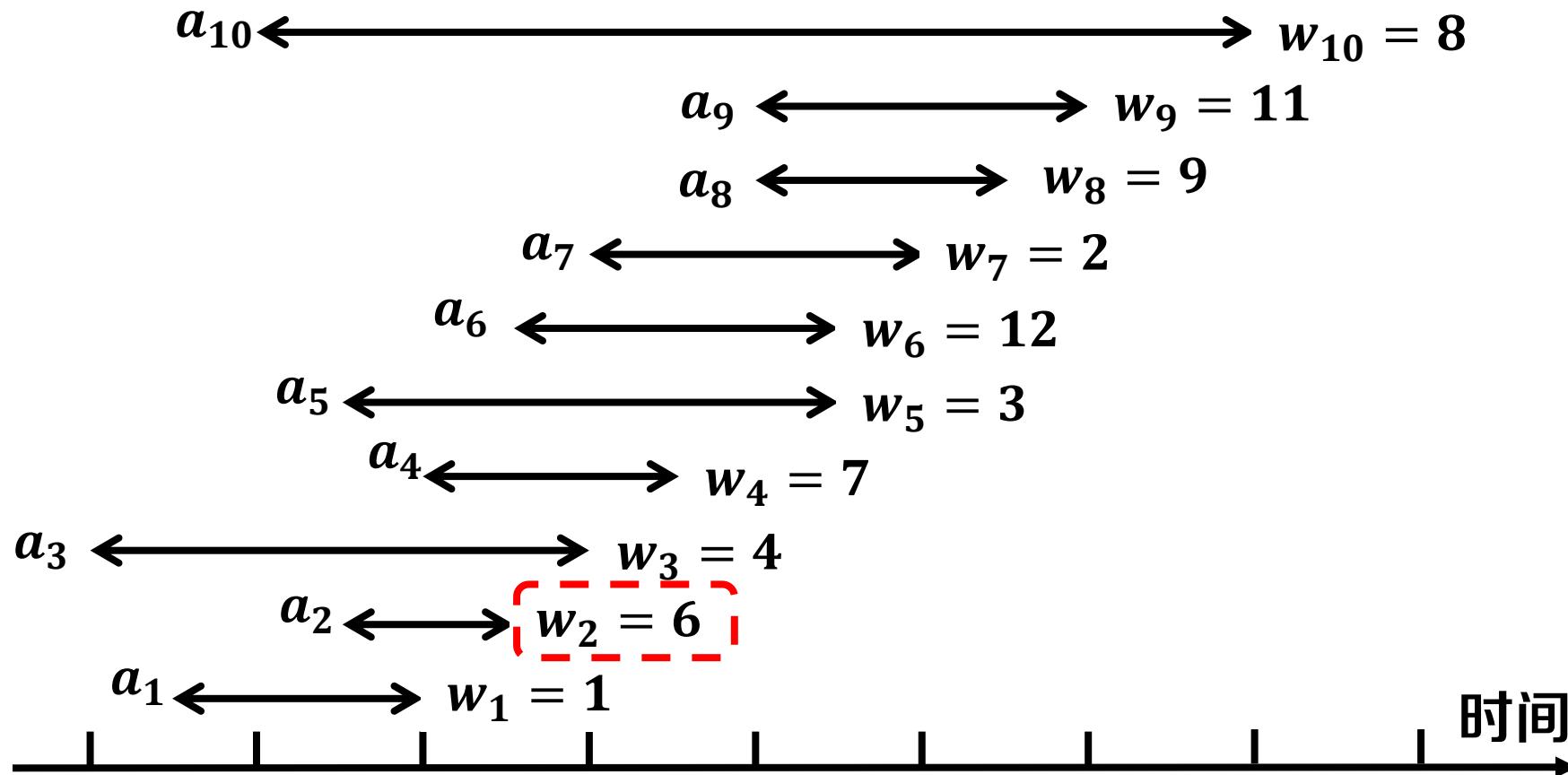
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 |   |   |   |   |   |   |   |    |



# 算法实例

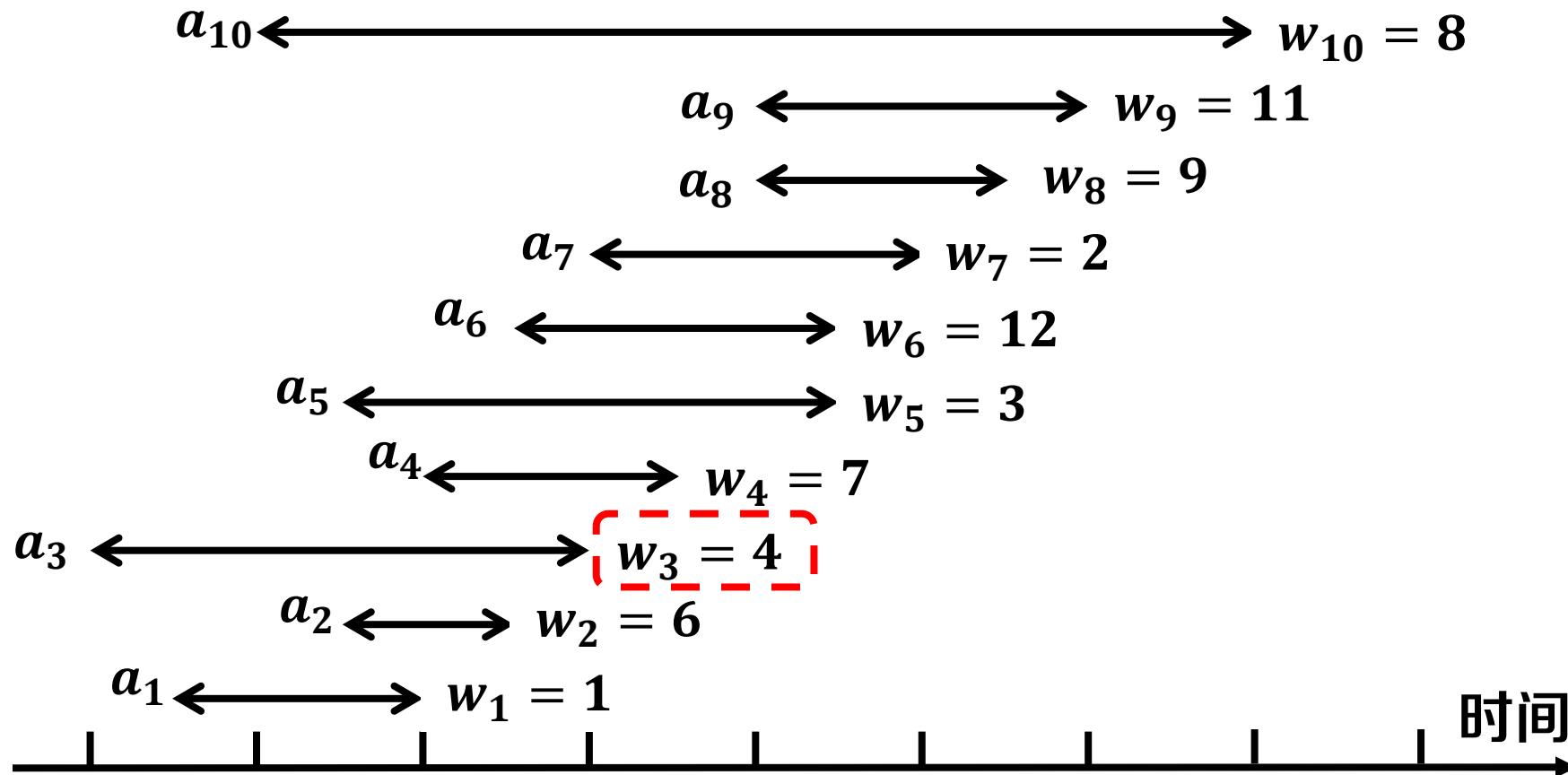
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 |   |   |   |   |   |   |    |



# 算法实例

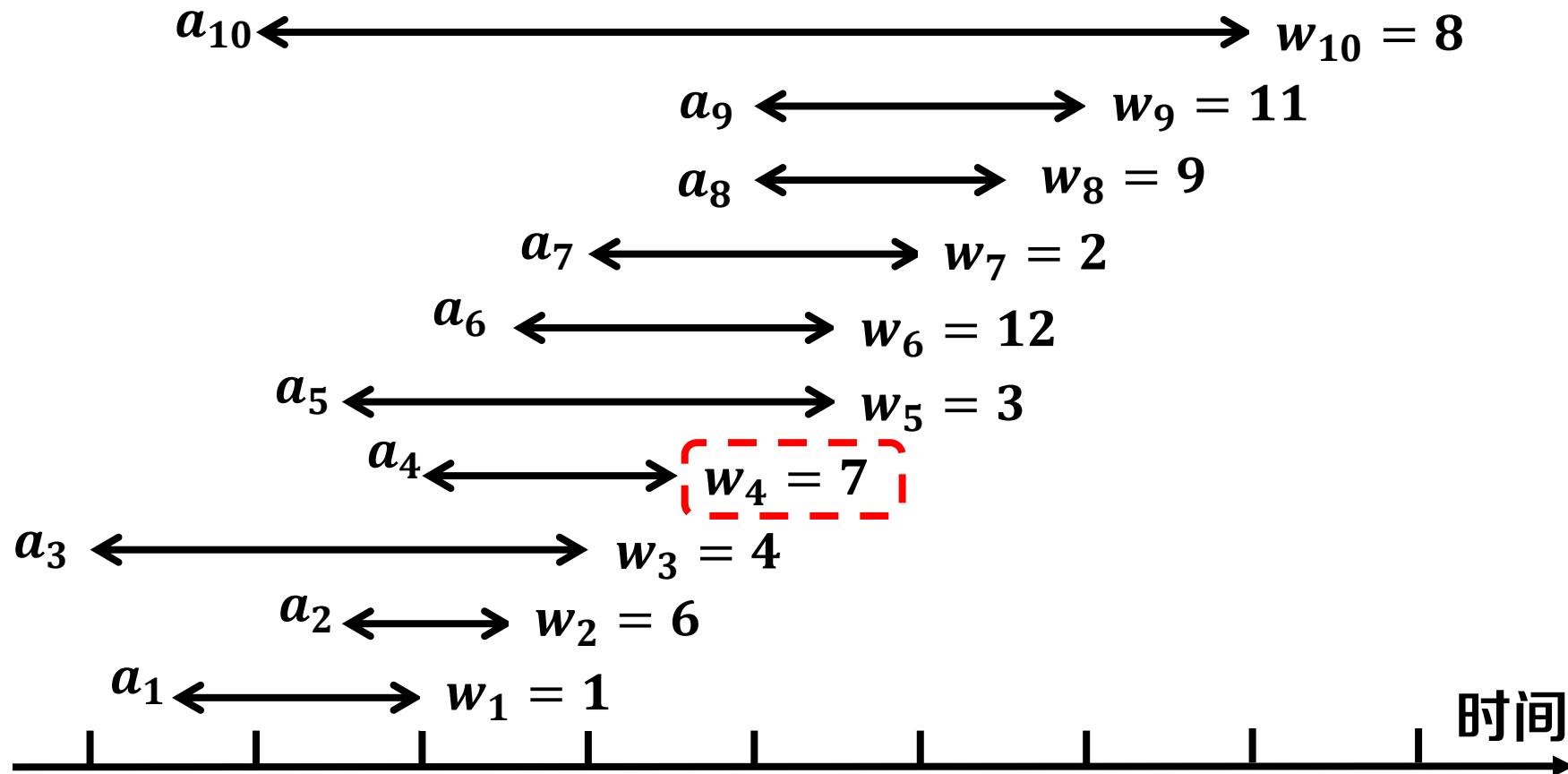
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 |   |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 |   |   |   |   |   |    |



# 算法实例

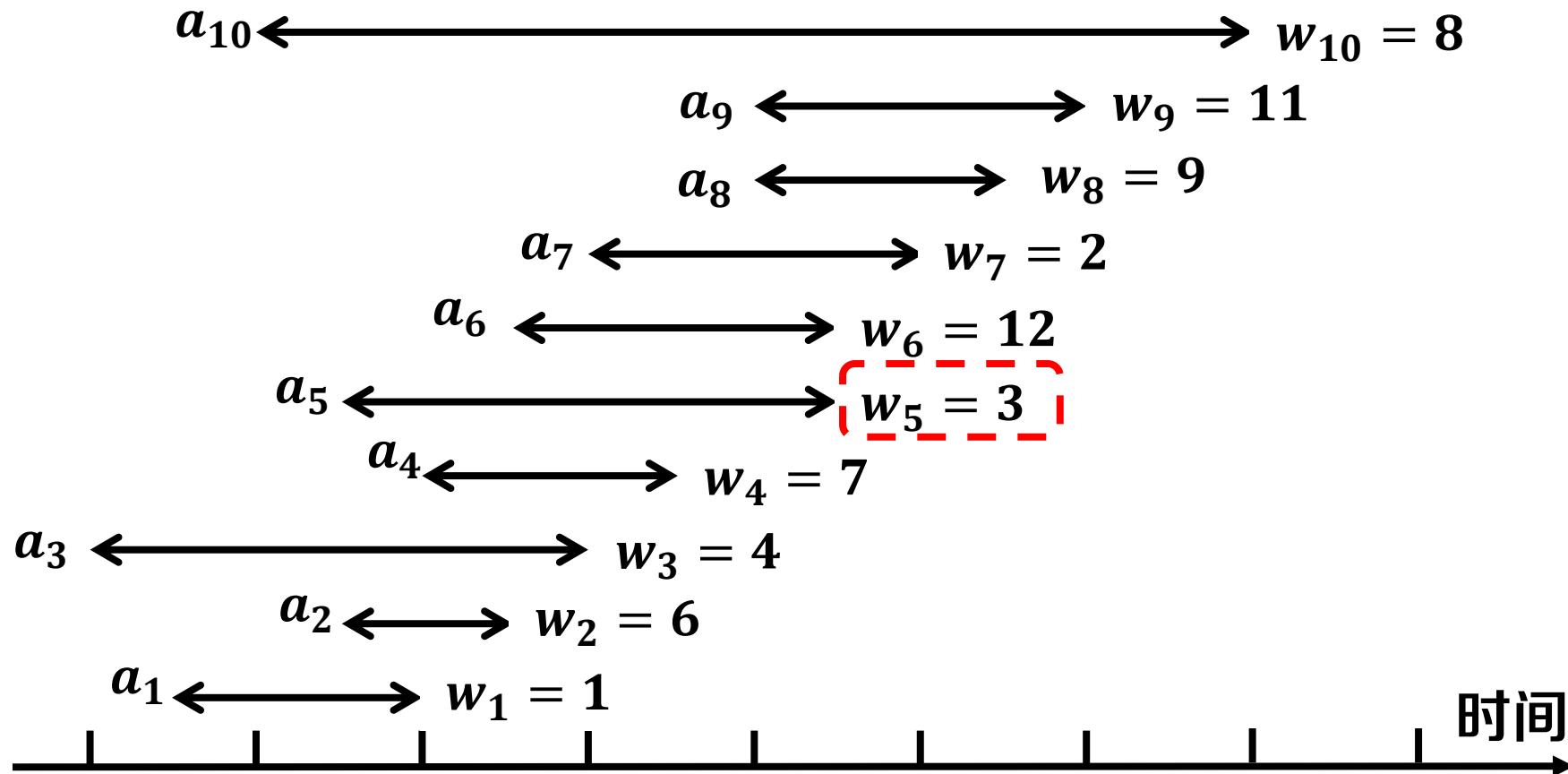
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 |   |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 |   |   |   |   |    |



# 算法实例

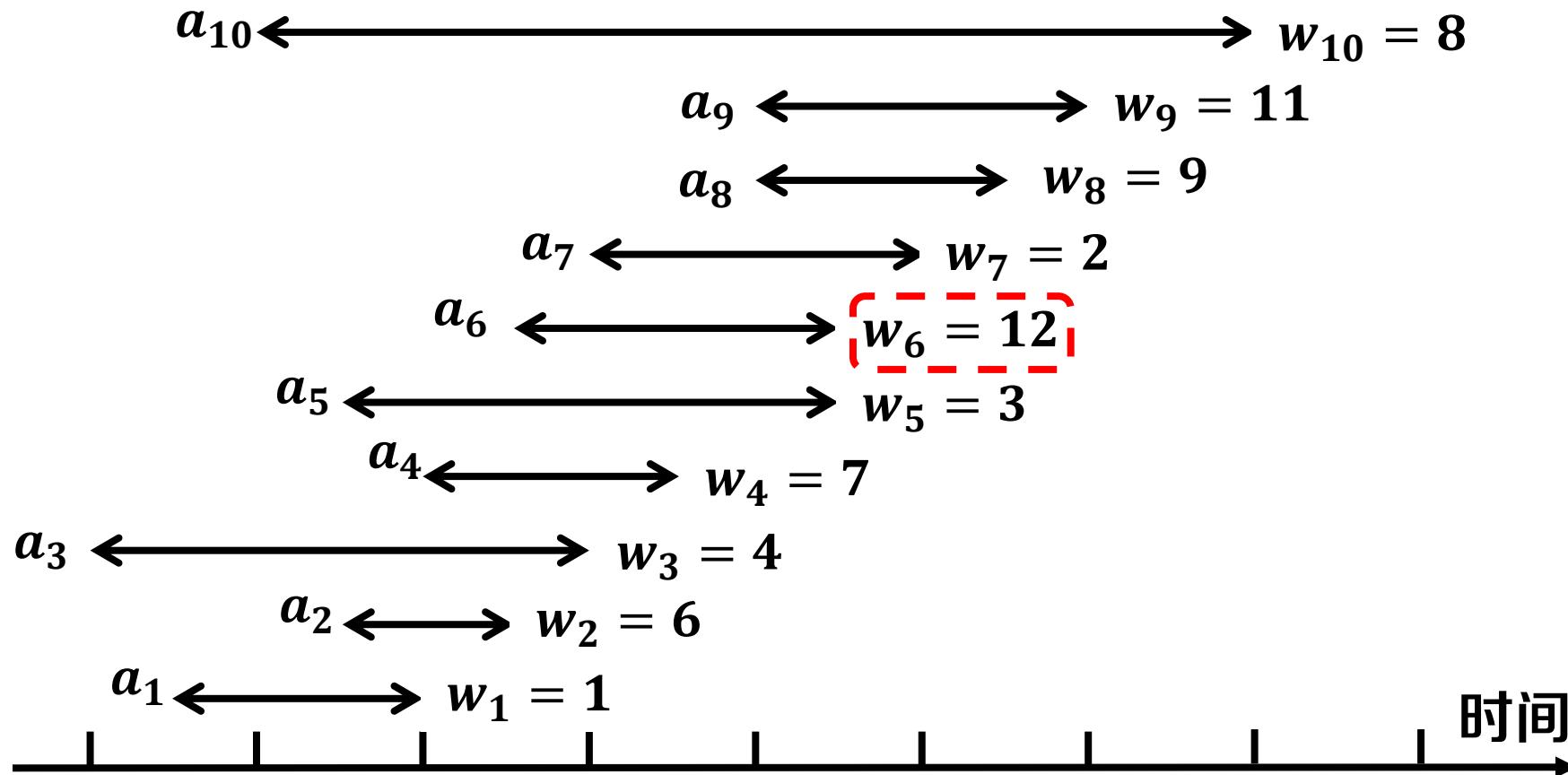
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |   |   |   |    |
|-----|---|---|---|---|---|---|----|---|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 |   |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 |   |   |   |    |



# 算法实例

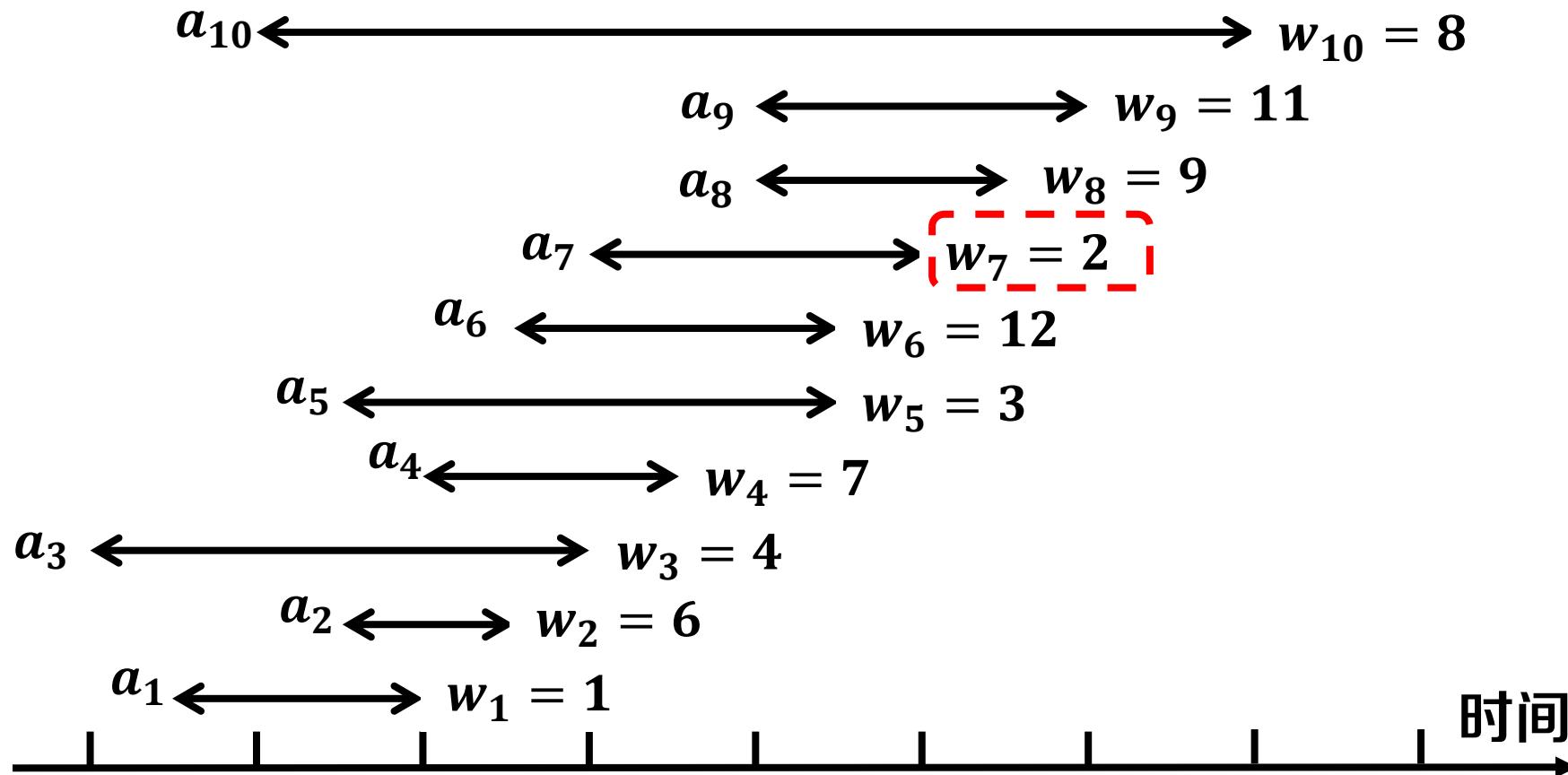
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |   |   |    |
|-----|---|---|---|---|---|---|----|----|---|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8 | 9 | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 |   |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 |   |   |    |



# 算法实例

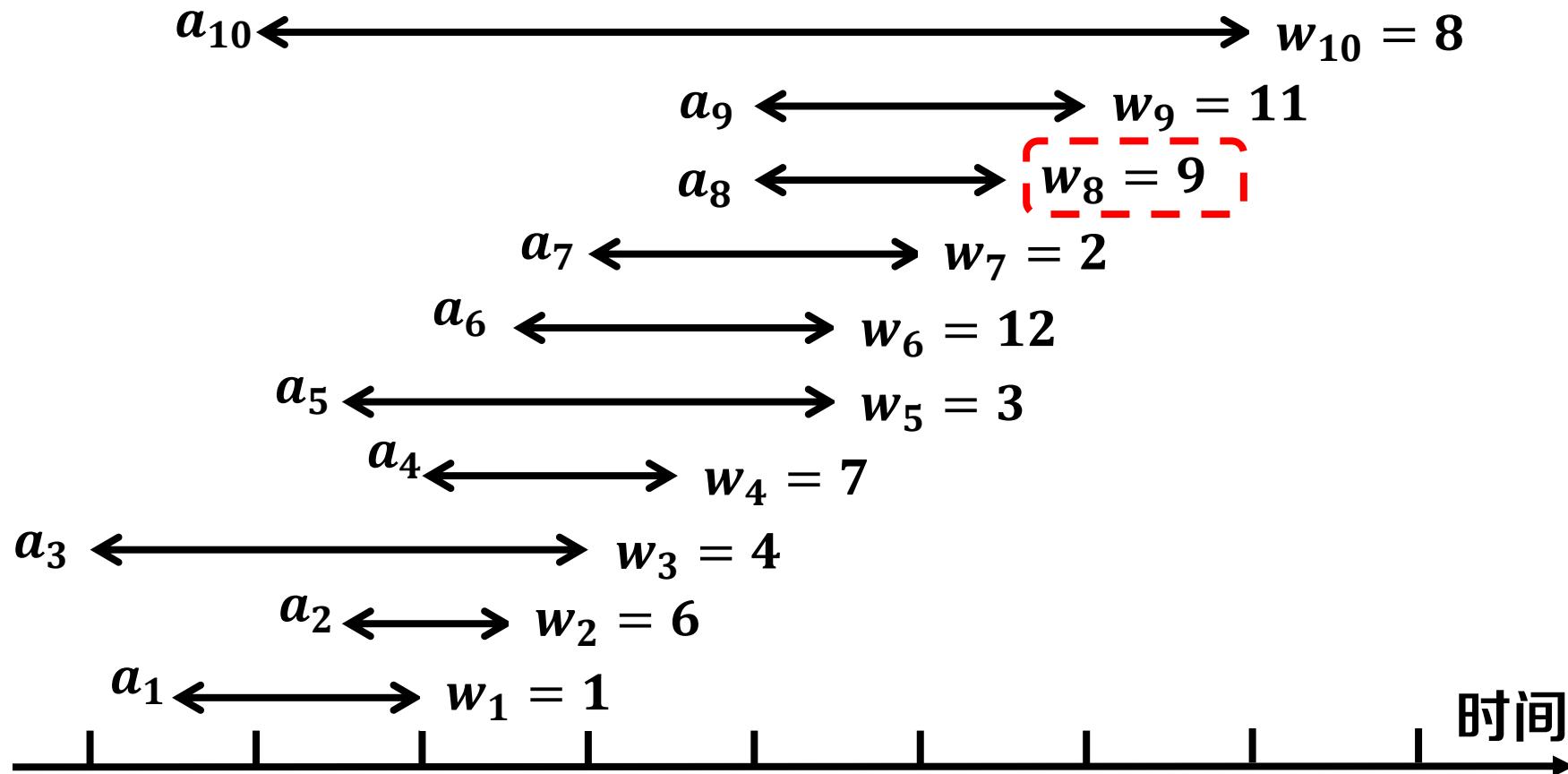
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |    |    |    |    |   |    |
|-----|---|---|---|---|---|----|----|----|----|---|----|
|     | 0 | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8  | 9 | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 18 | 18 | 18 | 18 |   |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |   |    |



# 算法实例

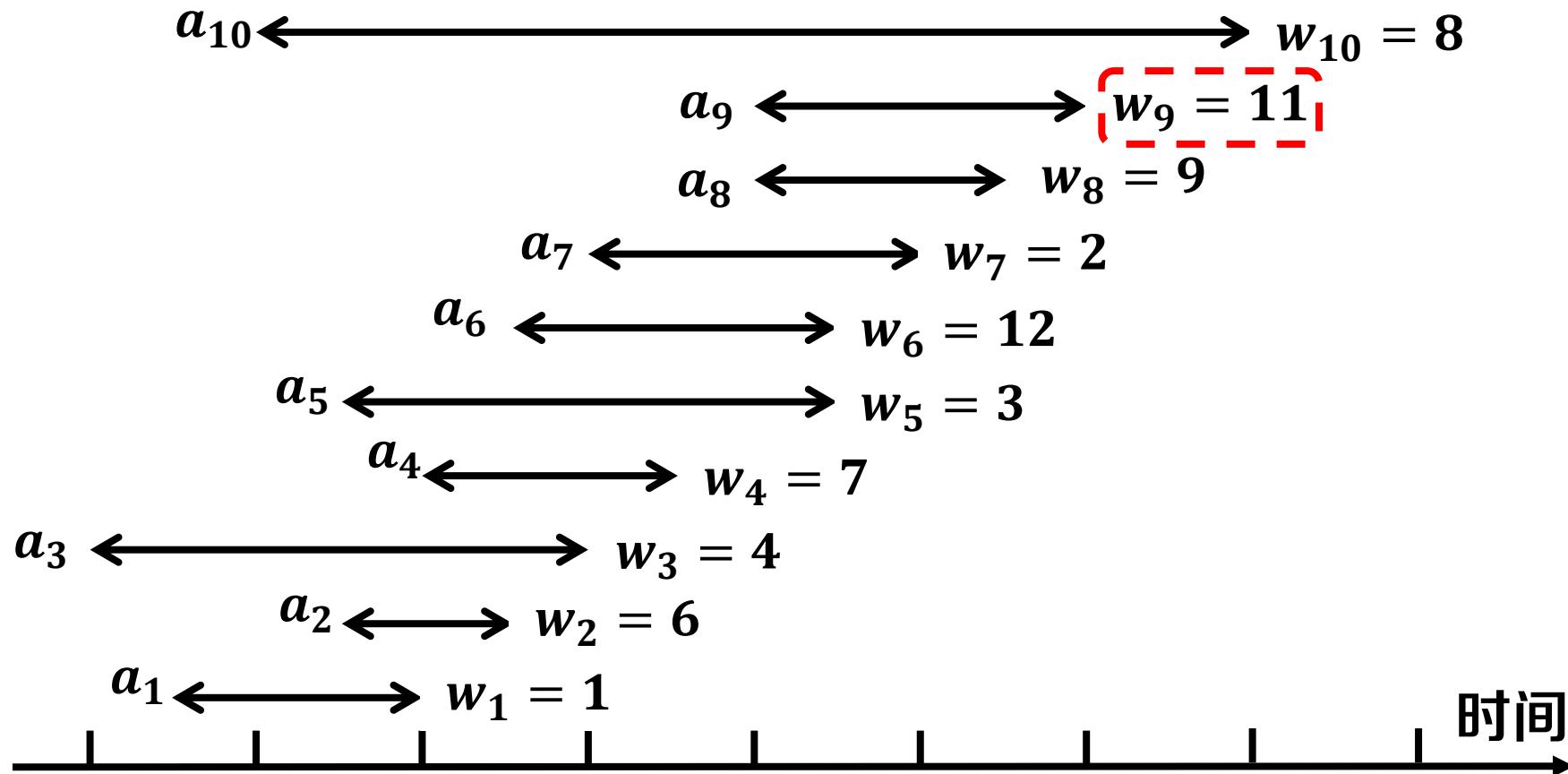
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |    |    |    |    |    |    |
|-----|---|---|---|---|---|----|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 18 | 18 | 18 | 19 | 19 |    |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |    |



# 算法实例

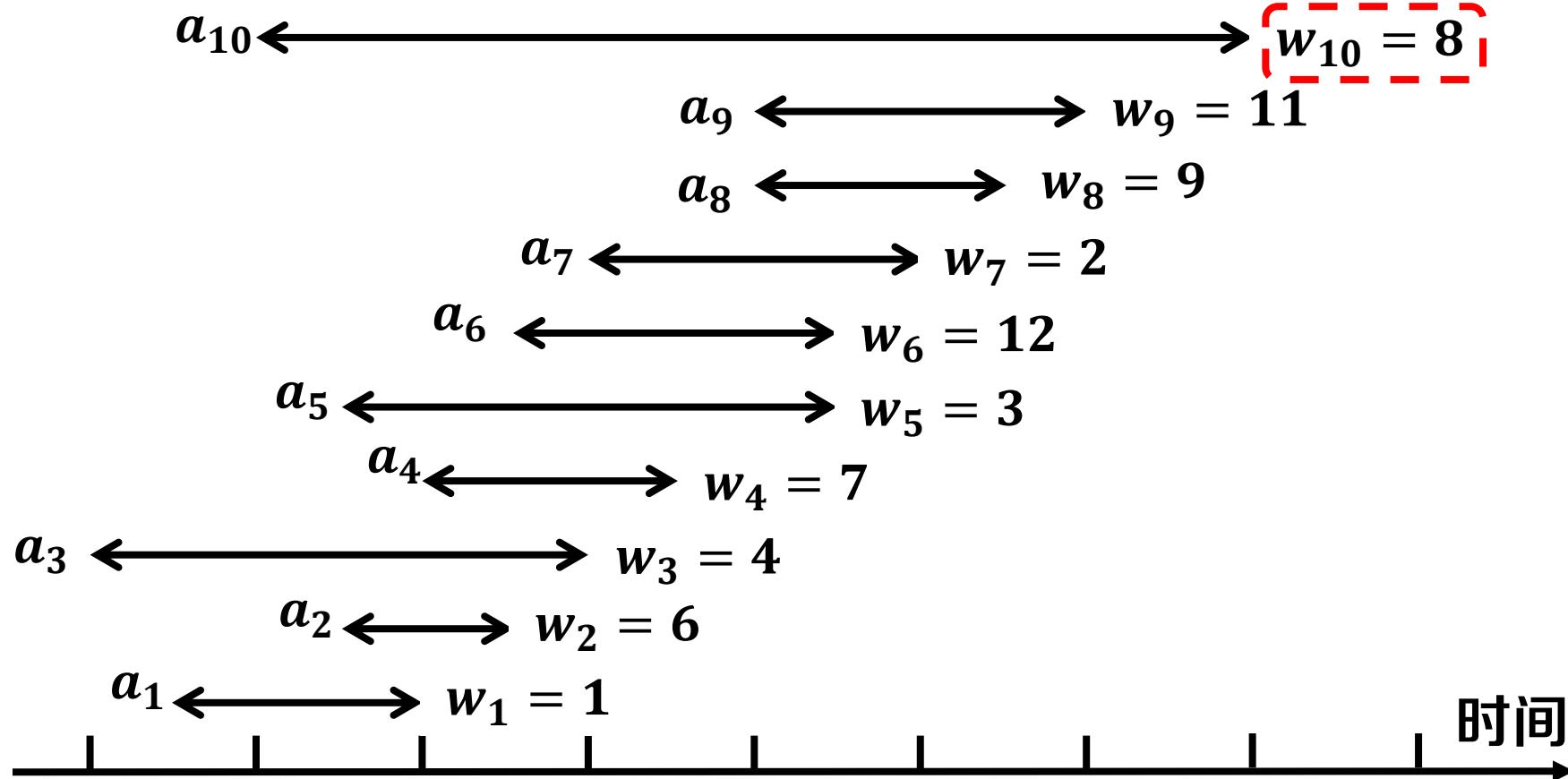
|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

$p[i]$ : 在  $a_i$  开始前最后结束的活动

$$D[i] = \max\{D[p[i]] + w_i, D[i - 1]\}$$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



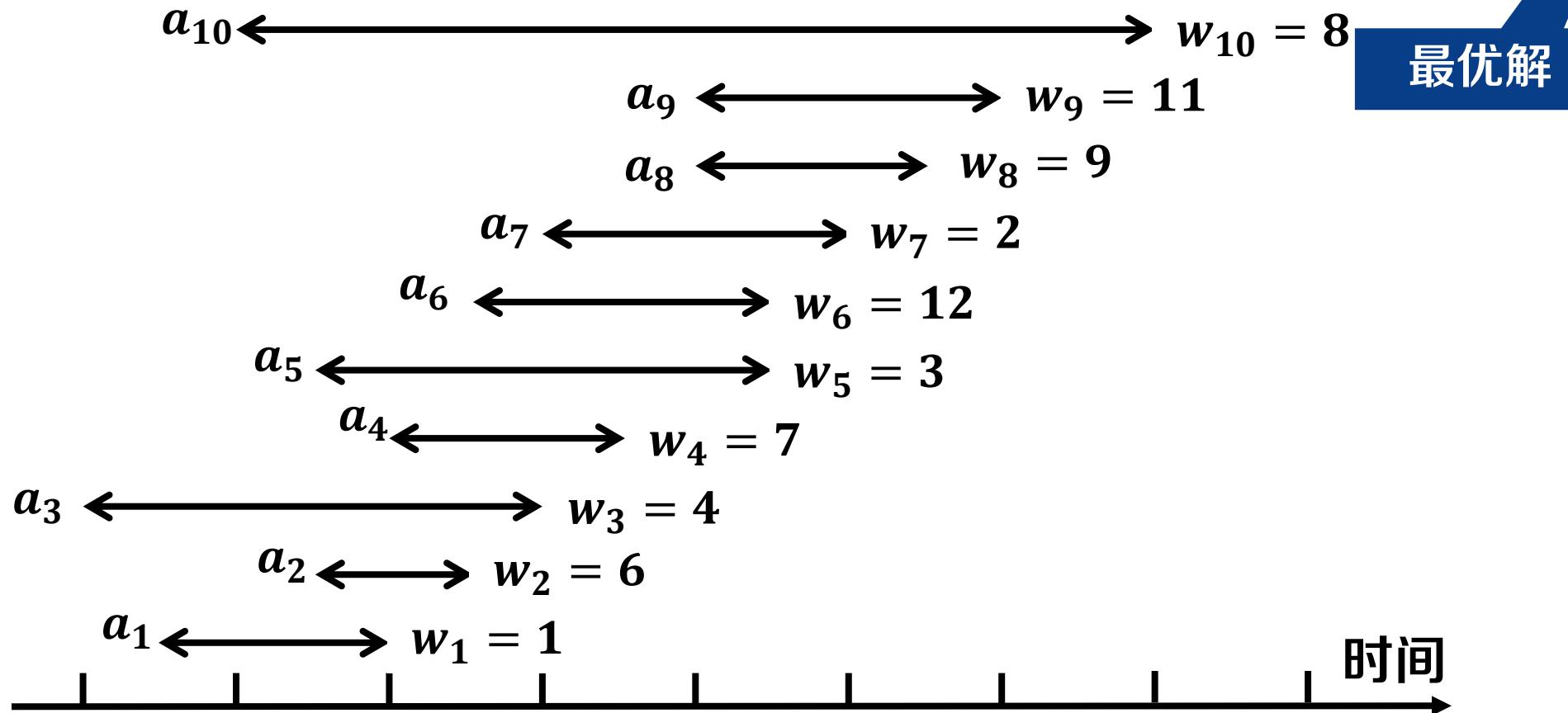


# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



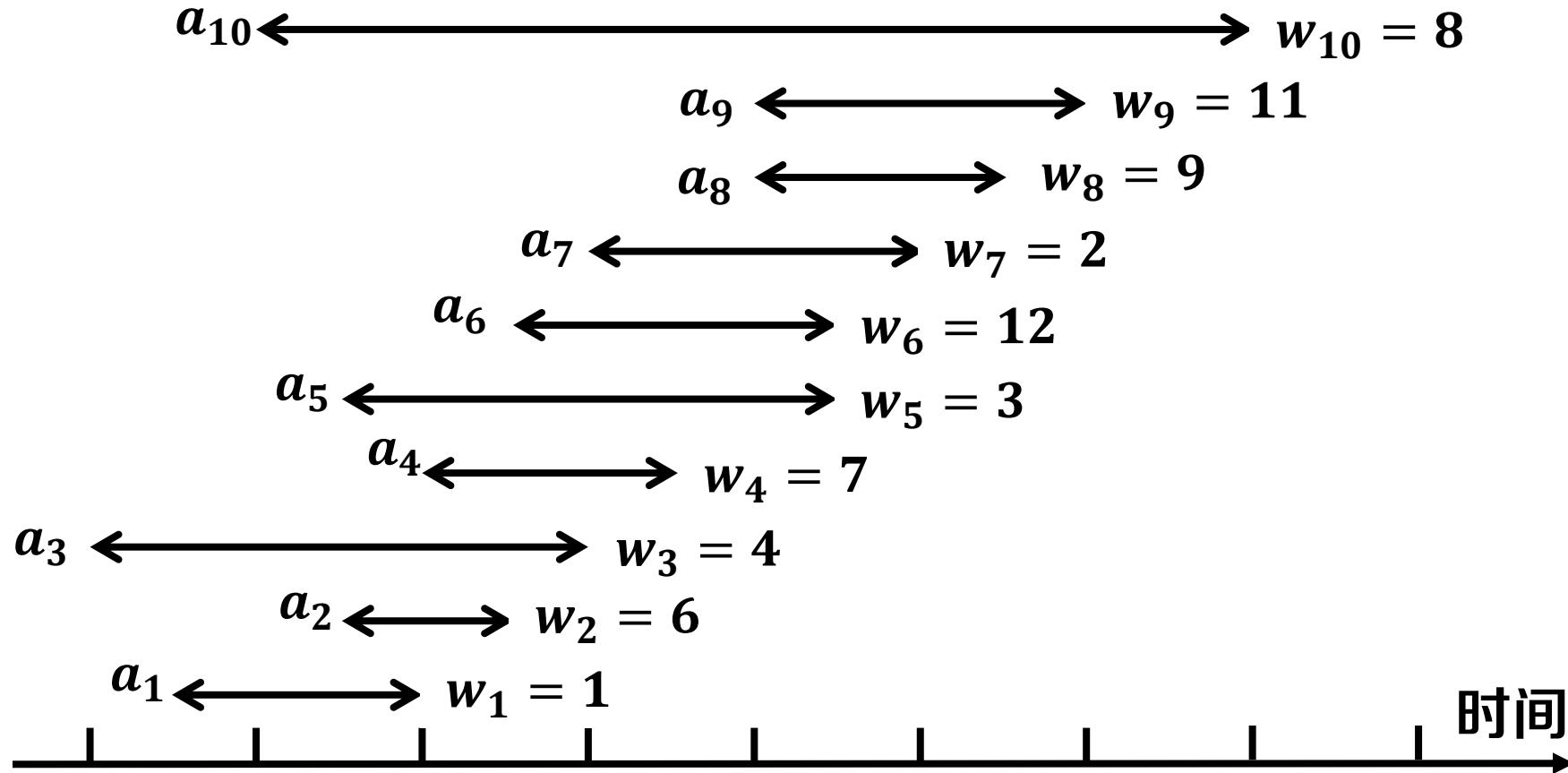
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



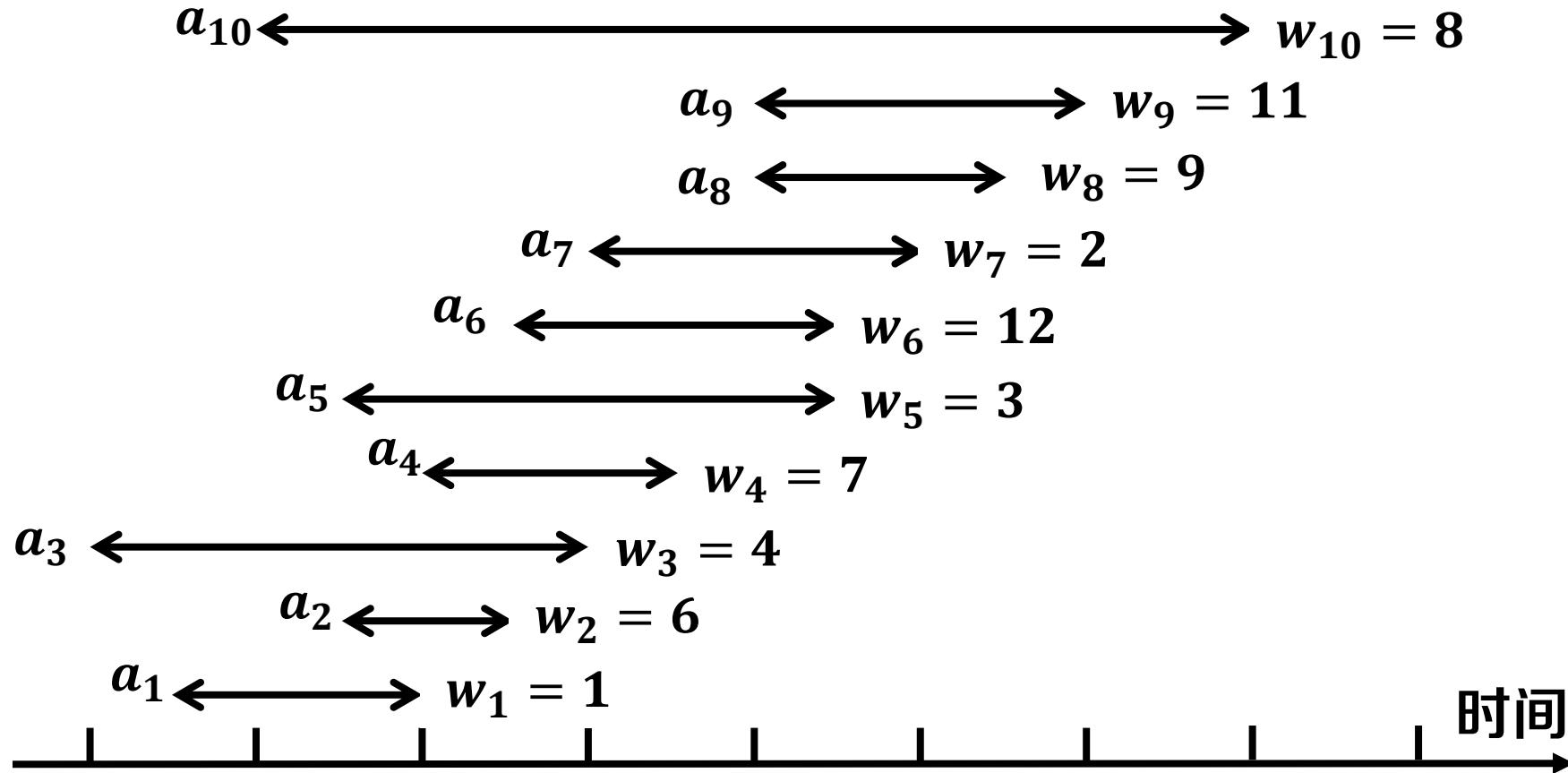
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{a_9\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



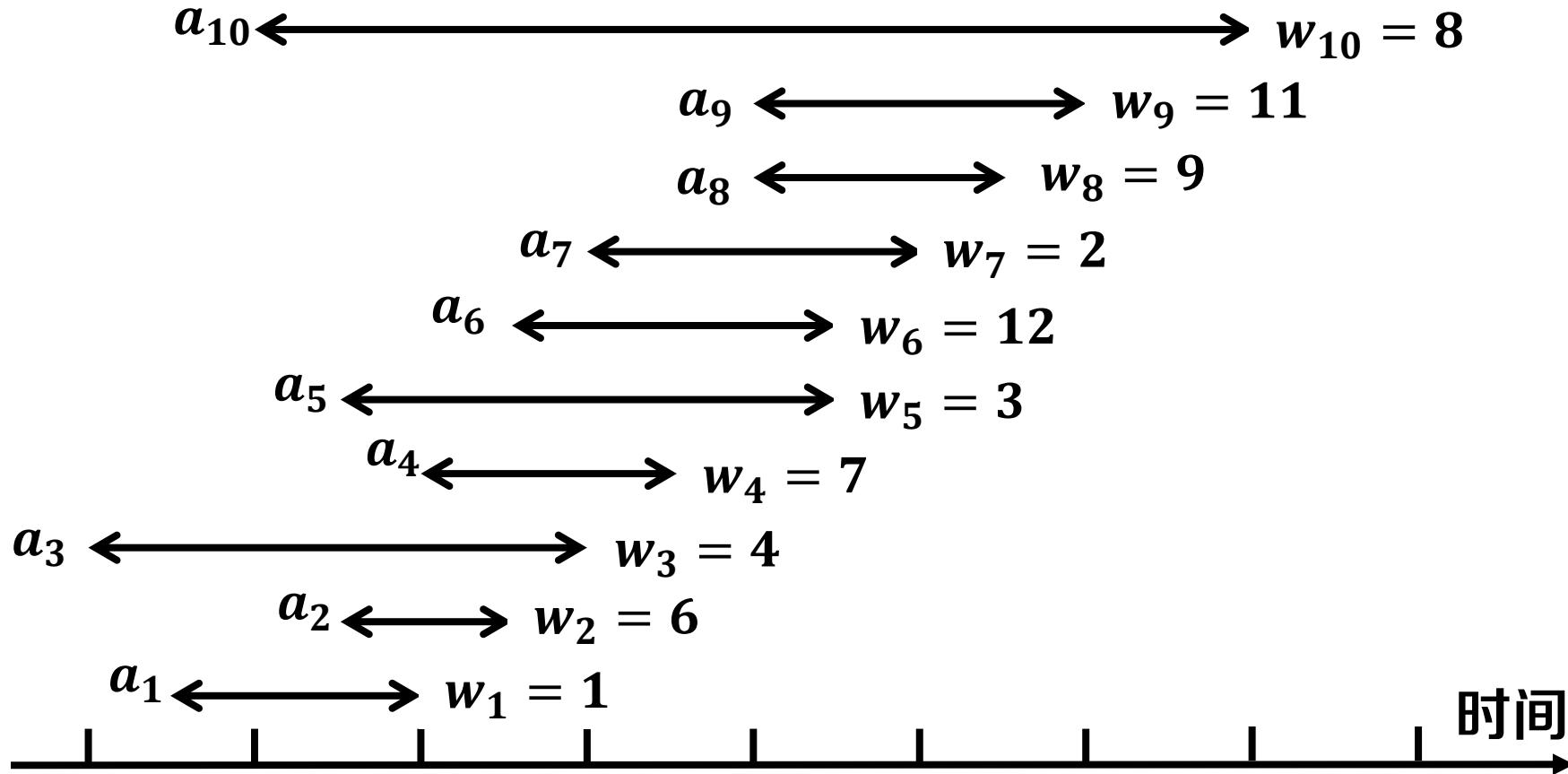
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{a_9\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



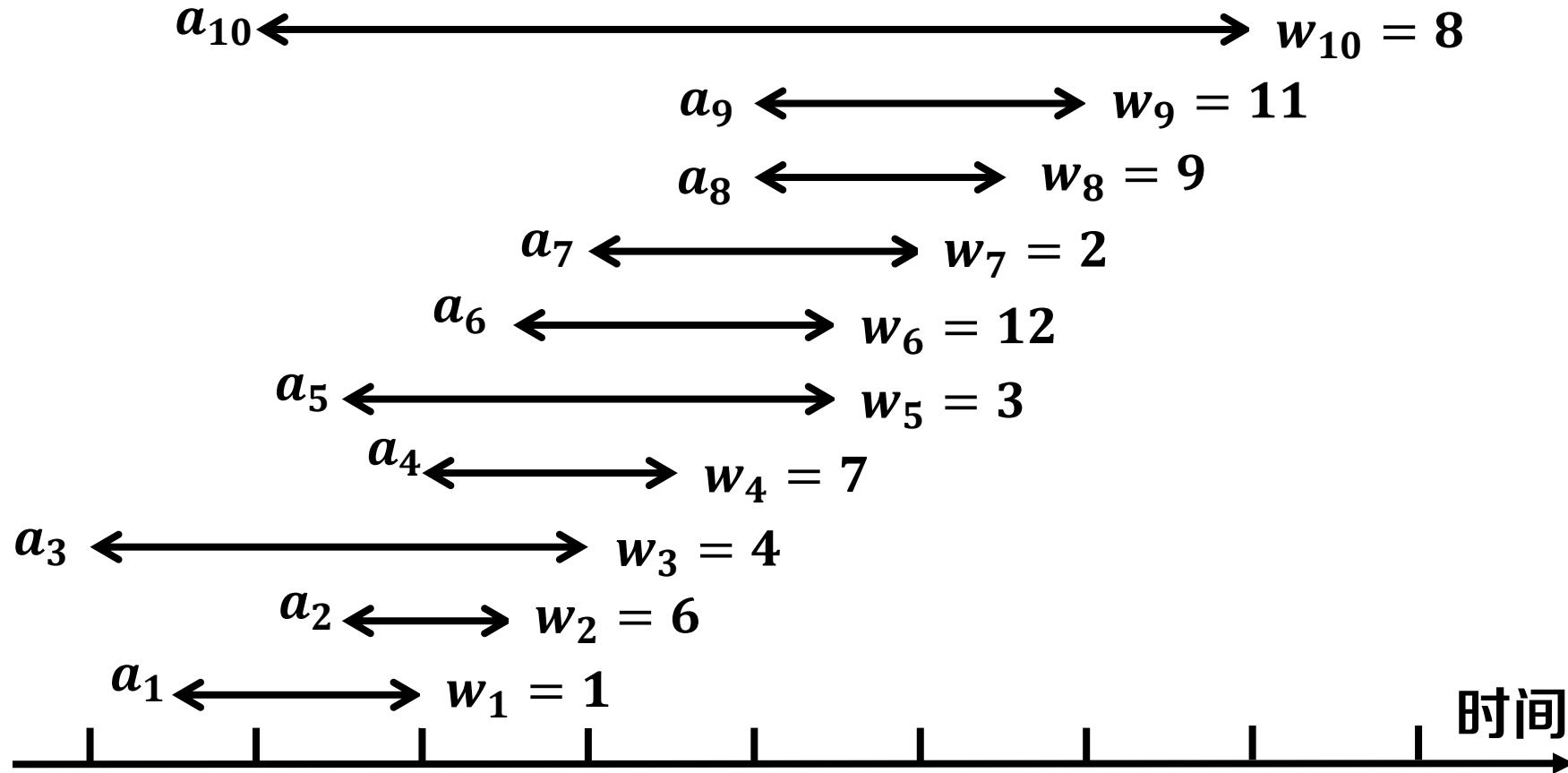
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{a_4, a_9\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



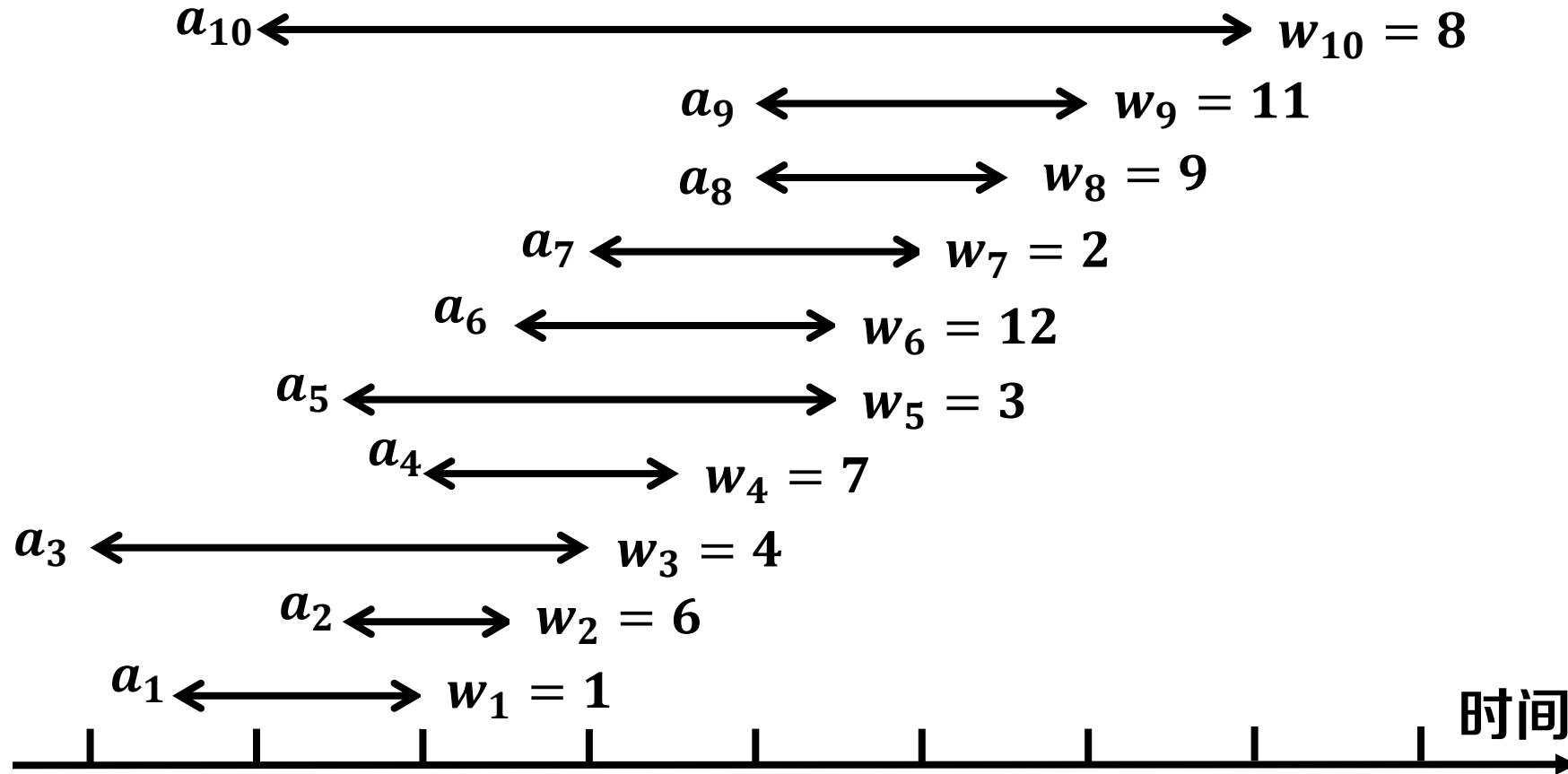
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{a_4, a_9\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



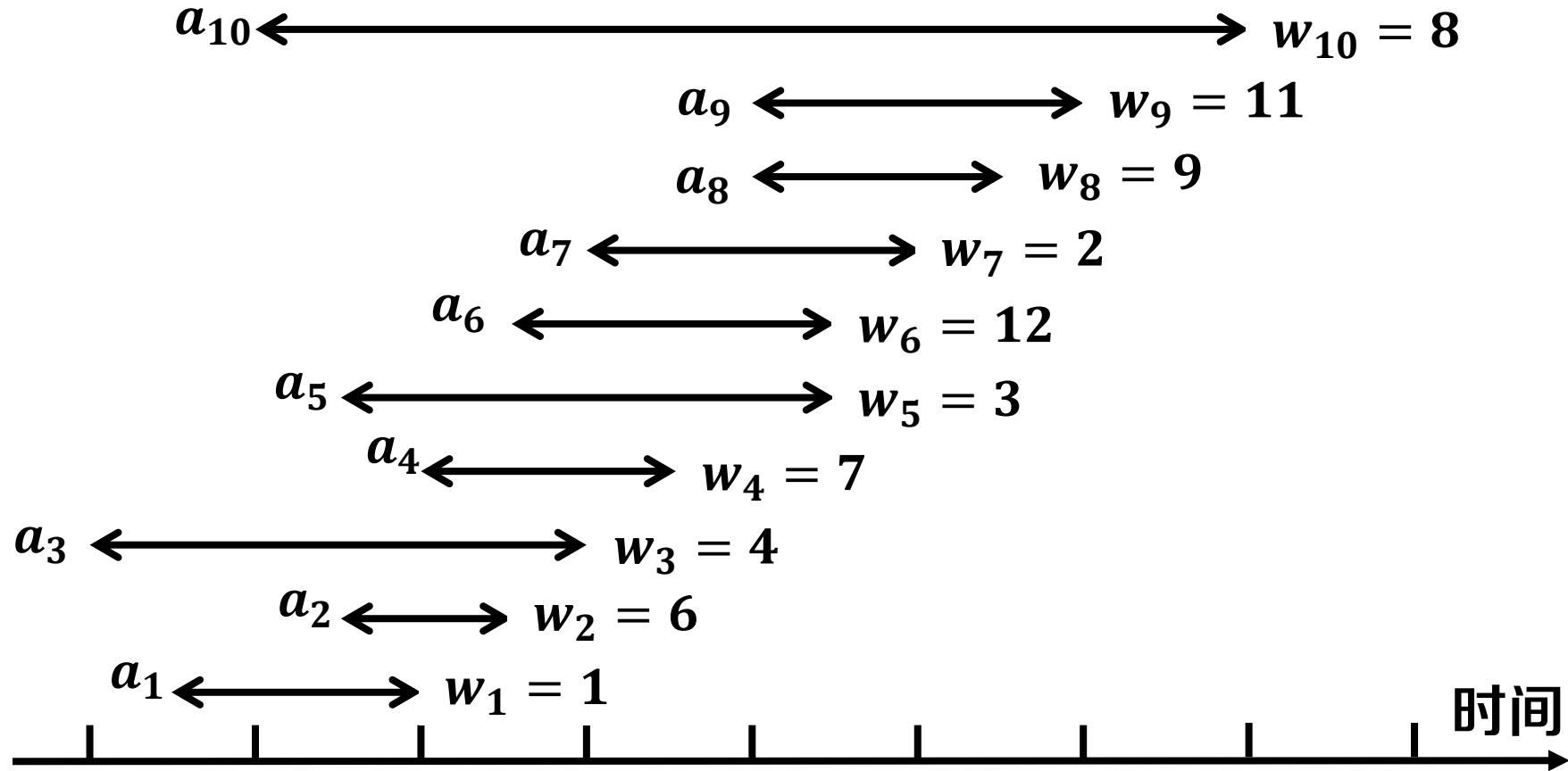
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{a_1, a_4, a_9\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |



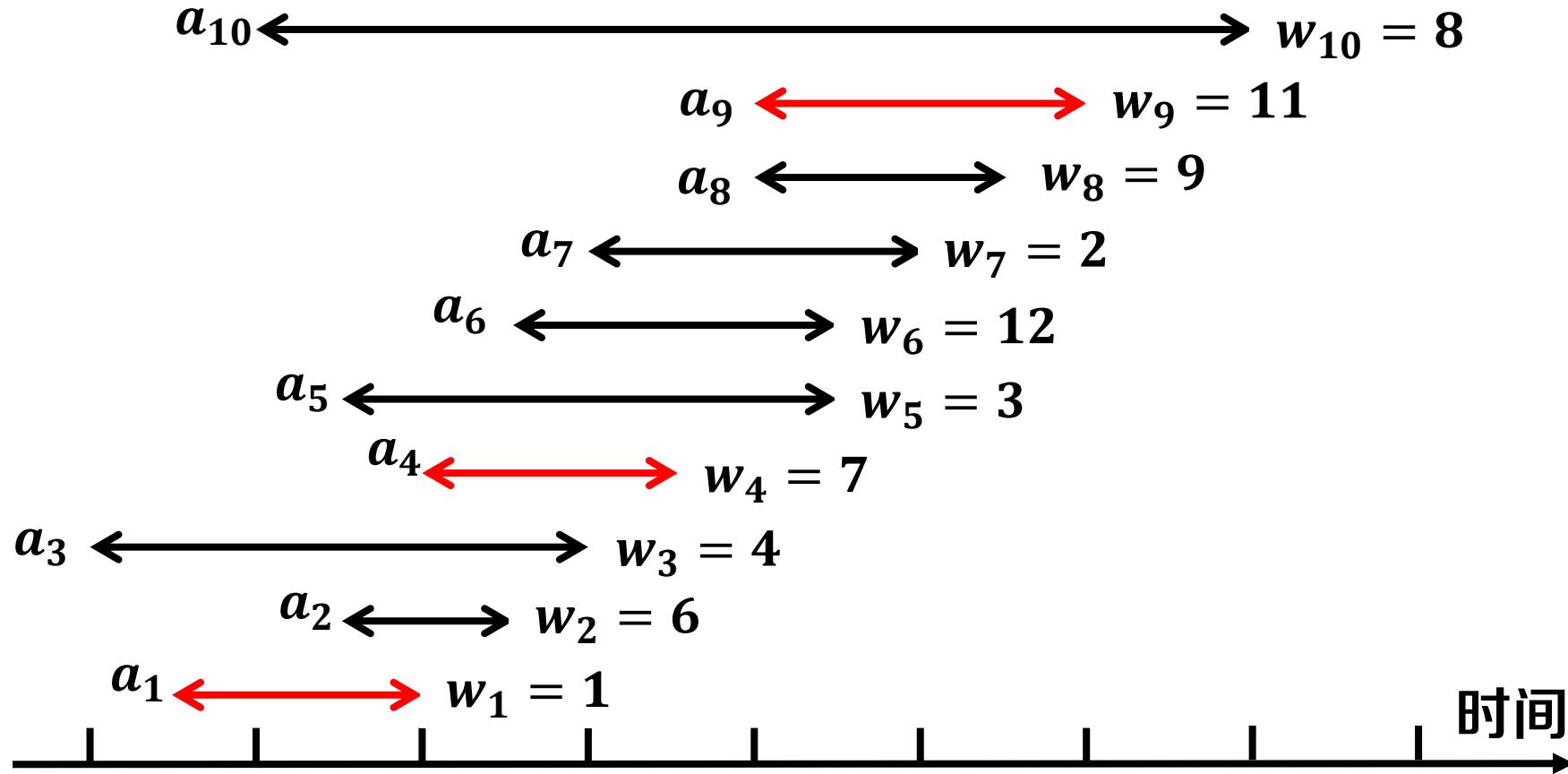
# 算法实例

|     |   |   |   |   |   |   |   |   |   |    |
|-----|---|---|---|---|---|---|---|---|---|----|
|     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $p$ | 0 | 0 | 0 | 1 | 0 | 2 | 3 | 4 | 4 | 0  |

|     |   |   |   |   |   |   |    |    |    |    |    |
|-----|---|---|---|---|---|---|----|----|----|----|----|
|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8  | 9  | 10 |
| $D$ | 0 | 1 | 6 | 6 | 8 | 8 | 18 | 18 | 18 | 19 | 19 |

活动集合  $S' = \{a_1, a_4, a_9\}$

|       |   |   |   |   |   |   |   |   |   |    |
|-------|---|---|---|---|---|---|---|---|---|----|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $Rec$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0  |





# 动态规划：伪代码

输入: 活动集合  $S = \{a_1, a_2, \dots, a_n\}$ ,  
每个活动  $a_i$  的起止时间  $s_i, f_i$  和权重  $w_i$

输出: 不冲突活动的最大子集  $S'$

//预处理

把活动按照结束时间升序排序

for  $i \leftarrow 1$  to  $n$  do

| 二分查找求解  $p[i]$

end

//初始化

新建数组  $D[0..n], Rec[1..n]$

$D[0] \leftarrow 0$

预处理



# 动态规划：伪代码

输入: 活动集合  $S = \{a_1, a_2, \dots, a_n\}$ ,  
每个活动  $a_i$  的起止时间  $s_i, f_i$  和权重  $w_i$

输出: 不冲突活动的最大子集  $S'$

//预处理

把活动按照结束时间升序排序

for  $i \leftarrow 1$  to  $n$  do

| 二分查找求解  $p[i]$

end

//初始化

新建数组  $D[0..n], Rec[1..n]$

$D[0] \leftarrow 0$

初始化



# 动态规划：伪代码

//动态规划

```
for  $j \leftarrow 1$  to  $n$  do
    if  $D[p[j]] + w_j > D[j - 1]$  then
         $D[j] \leftarrow D[p[j]] + w_j$ 
         $Rec[j] \leftarrow 1$ 
    end
    else
         $D[j] \leftarrow D[j - 1]$ 
         $Rec[j] \leftarrow 0$ 
    end
end
```

对每个子问题

# 动态规划：伪代码



//动态规划

for  $j \leftarrow 1$  to  $n$  do

  if  $D[p[j]] + w_j > D[j - 1]$  then

$D[j] \leftarrow D[p[j]] + w_j$

$Rec[j] \leftarrow 1$

  end

  else

$D[j] \leftarrow D[j - 1]$

$Rec[j] \leftarrow 0$

  end

end

选择活动 $a_j$

# 动态规划：伪代码



//动态规划

```
for  $j \leftarrow 1$  to  $n$  do
    if  $D[p[j]] + w_j > D[j - 1]$  then
        |  $D[j] \leftarrow D[p[j]] + w_j$ 
        |  $Rec[j] \leftarrow 1$ 
    end
    else
        |  $D[j] \leftarrow D[j - 1]$ 
        |  $Rec[j] \leftarrow 0$ 
    end
end
```

不选活动 $a_j$



# 动态规划：伪代码

//输出方案

```
k ← n
while k > 0 do
    if Rec[k] = 1 then
        print 选择活动 $a_k$ 
        k ← p[k]
    end
    else
        k ← k - 1
    end
end
return D[n]
```

选择活动 $a_k$



# 动态规划：伪代码

//输出方案

```
k ← n
while k > 0 do
    if Rec[k] = 1 then
        print 选择a[k]
        k ← p[k]
    end
    else
        k ← k - 1
    end
end
return D[n]
```

回溯子问题



# 动态规划：伪代码

//输出方案

```
k ← n
while k > 0 do
    if Rec[k] = 1 then
        | print 选择 $a[k]$ 
        | k ← p[k]
    end
    else
        | k ← k - 1
    end
end
return D[n]
```

不选活动 $a_k$



# 动态规划：复杂度分析

输入: 活动集合  $S = \{a_1, a_2, \dots, a_n\}$ ,  
每个活动  $a_i$  的起止时间  $s_i, f_i$  和权重  $w_i$

输出: 不冲突活动的最大子集  $S'$

//预处理和初始化

把活动按照结束时间升序排序 ——  $O(n \log n)$   
for  $i \leftarrow 1$  to  $n$  do  
| 二分查找求解  $p[i]$   
end }  $O(n \log n)$

新建数组  $D[0..n]$ ,  $Rec[1..n]$

$D[0] \leftarrow 0$

//动态规划

for  $j \leftarrow 1$  to  $n$  do  
| if  $D[p[j]] + w_j > D[j - 1]$  then  
| |  $D[j] \leftarrow D[p[j]] + w_j$   
| |  $Rec[j] \leftarrow 1$   
| end  
| else  
| |  $D[j] \leftarrow D[j - 1]$   
| |  $Rec[j] \leftarrow 0$   
| end  
end }  $O(n)$

//输出方案

$k \leftarrow n$   
while  $k > 0$  do  
| if  $Rec[k] = 1$  then  
| | print 选择  $a[k]$   
| |  $k \leftarrow p[k]$   
| end  
| else  
| |  $k \leftarrow k - 1$   
| end  
end  
return  $D[n]$

}  $O(n)$

时间复杂度:  $O(n \log n)$

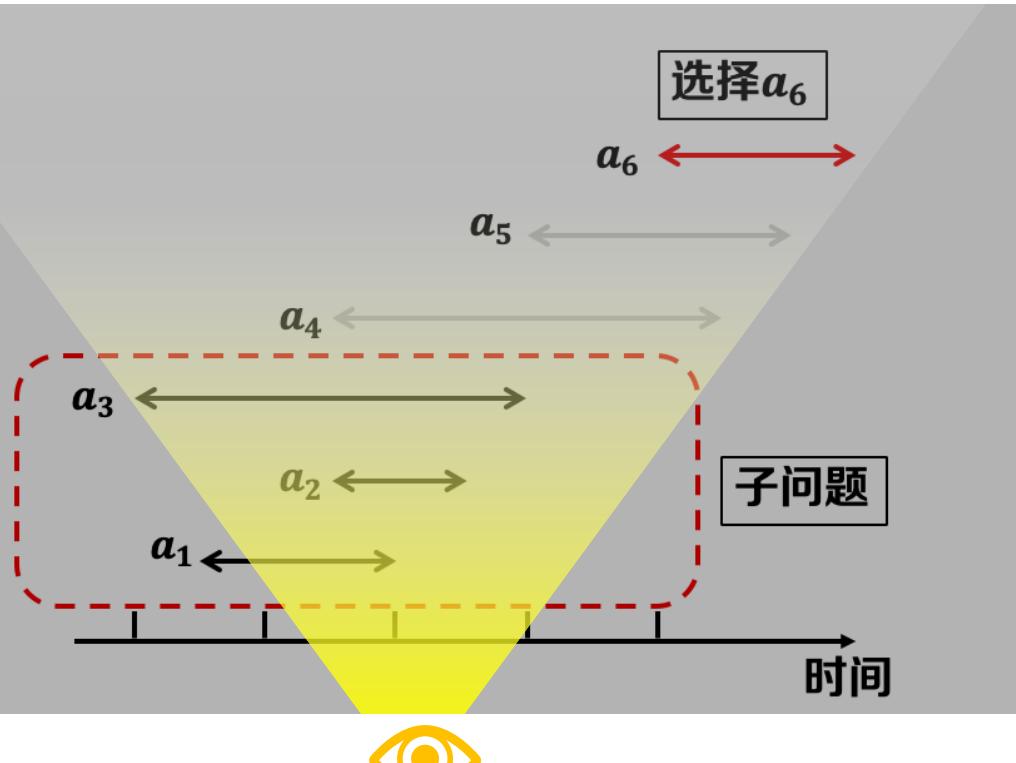
# 活动选择问题：动态规划 vs. 贪心策略

带权活动选择问题

权重均为1

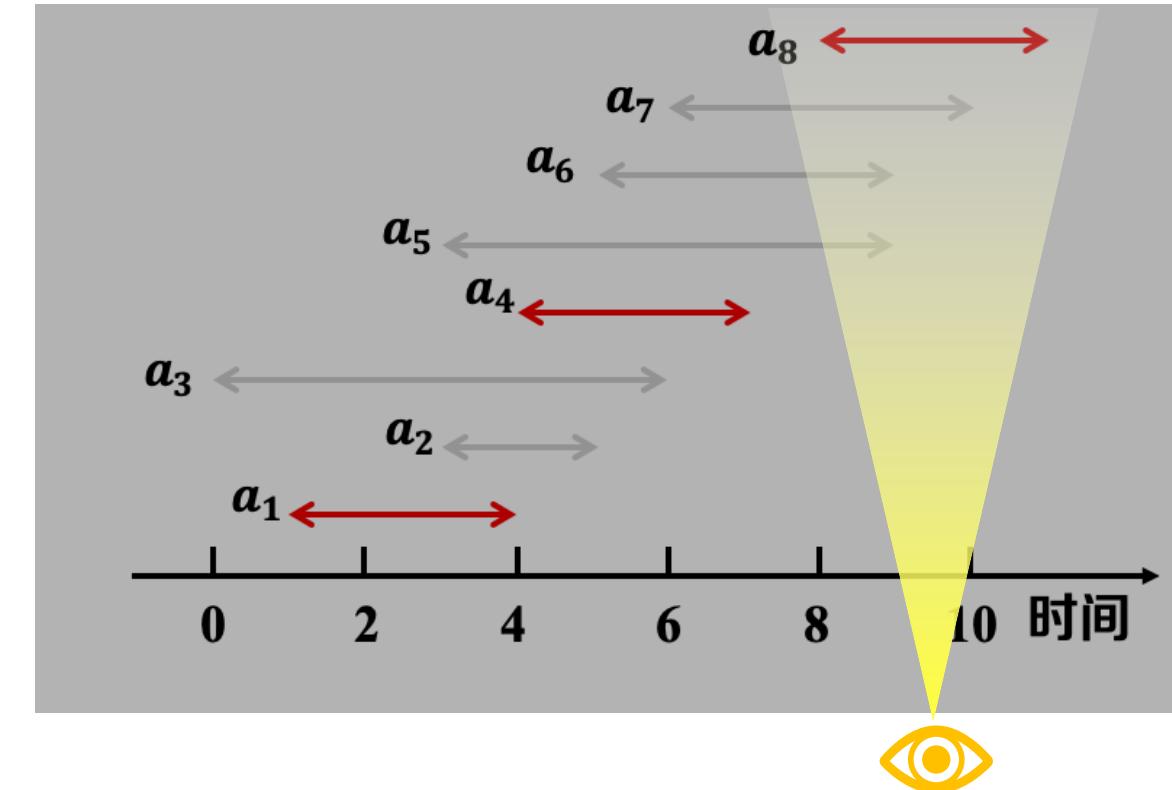
性质更好

活动选择问题



求解子问题，组合最优解

动态规划：考察全局



直接做决策，构造最优解

贪心策略：考察局部