分而治之篇：最大子数组问题I

童咏昕

北京航空航天大学
计算机学院

中国大学MOOC北航《算法设计与分析》

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组为数组X中连续的一段序列

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$
- 子数组$X[1..10]$

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$
- 子数组$X[1..10]$
  - 求和为：$-1-3+3+5-4+3+2-2+3+6=12$

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$
- 子数组$X[1..10]$
  - 求和为：$-1-3+3+5-4+3+2-2+3+6=12$
- 子数组$X[3..10]$

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$
- 子数组$X[1..10]$
  - 求和为：$-1-3+3+5-4+3+2-2+3+6=12$
- 子数组$X[3..10]$
  - 求和为：$3+5-4+3+2-2+3+6=16$

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$
- 子数组$X[1..10]$
  - 求和为：$-1-3+3+5-4+3+2-2+3+6=12$
- 子数组$X[3..10]$
  - 求和为：$3+5-4+3+2-2+3+6=16$

问题：寻找数组X最大的非空子数组？

问题背景：子数组

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 子数组$X[3..7]$
  - 求和为：$3+5-4+3+2=9$
- 子数组$X[1..10]$
  - 求和为：$-1-3+3+5-4+3+2-2+3+6=12$
- 子数组$X[3..10]$
  - 求和为：$3+5-4+3+2-2+3+6=16$

问题：寻找数组X最大的非空子数组？
答案：$X[3..10]=16$

问题定义

- 形式化定义

最大子数组问题

Max Continuous Subarray, MCS

输入

- 给定一个数组$X[1..n]$，对于任意一对数组下标为l, r $(l \leq r)$的非空子数组，其和记为

$$S(l,r) = \sum_{i=l}^{r} X[i]$$

输出

- 求出$S(l,r)$的最大值，记为S_{max}

蛮力枚举

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

蛮力枚举

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 数组$X[1..n]$，其所有的下标$l, r(l \leq r)$组合分为以下两种情况

蛮力枚举

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 数组$X[1..n]$，其所有的下标$l, r(l \leq r)$组合分为以下两种情况
  - 当$l = r$时，一共$C_n^1 = n$种组合

蛮力枚举

	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 数组$X[1..n]$，其所有的下标$l, r(l \leq r)$组合分为以下两种情况
  - 当$l = r$时，一共$C_n^1 = n$种组合
  - 当$l < r$时，一共C_n^2种组合

蛮力枚举

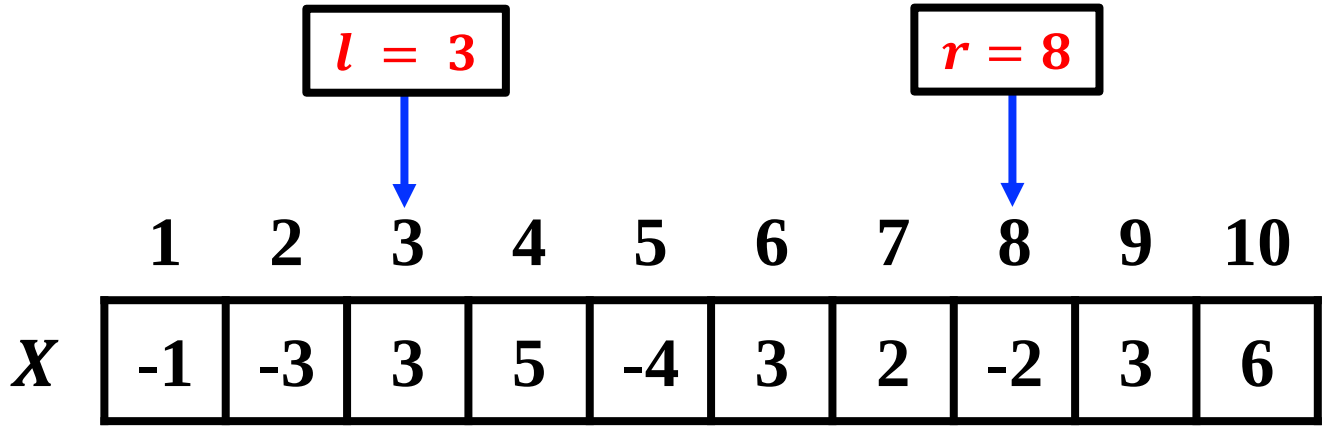
	1	2	3	4	5	6	7	8	9	10
X	-1	-3	3	5	-4	3	2	-2	3	6

- 数组$X[1..n]$，其所有的下标$l, r(l \le r)$组合分为以下两种情况
  - 当$l = r$时，一共$C_n^1 = n$种组合
  - 当$l < r$时，一共C_n^2种组合

- 枚举$n + C_n^2$种下标l, r组合，求出最大子数组之和

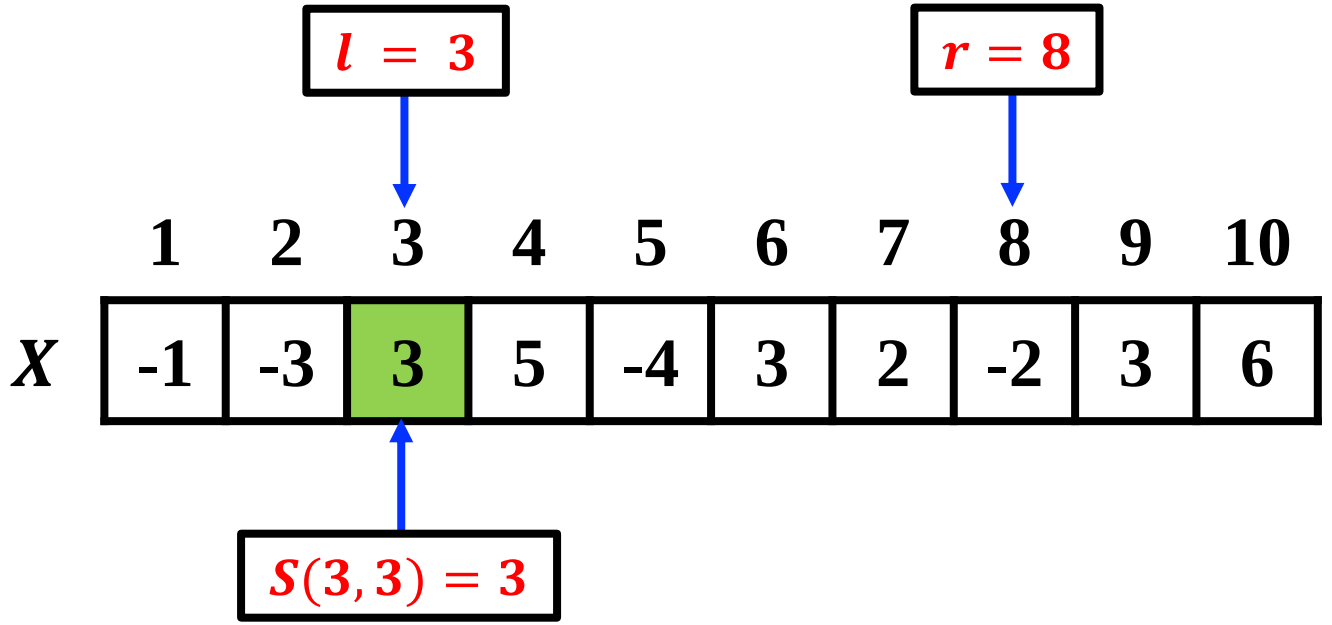
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



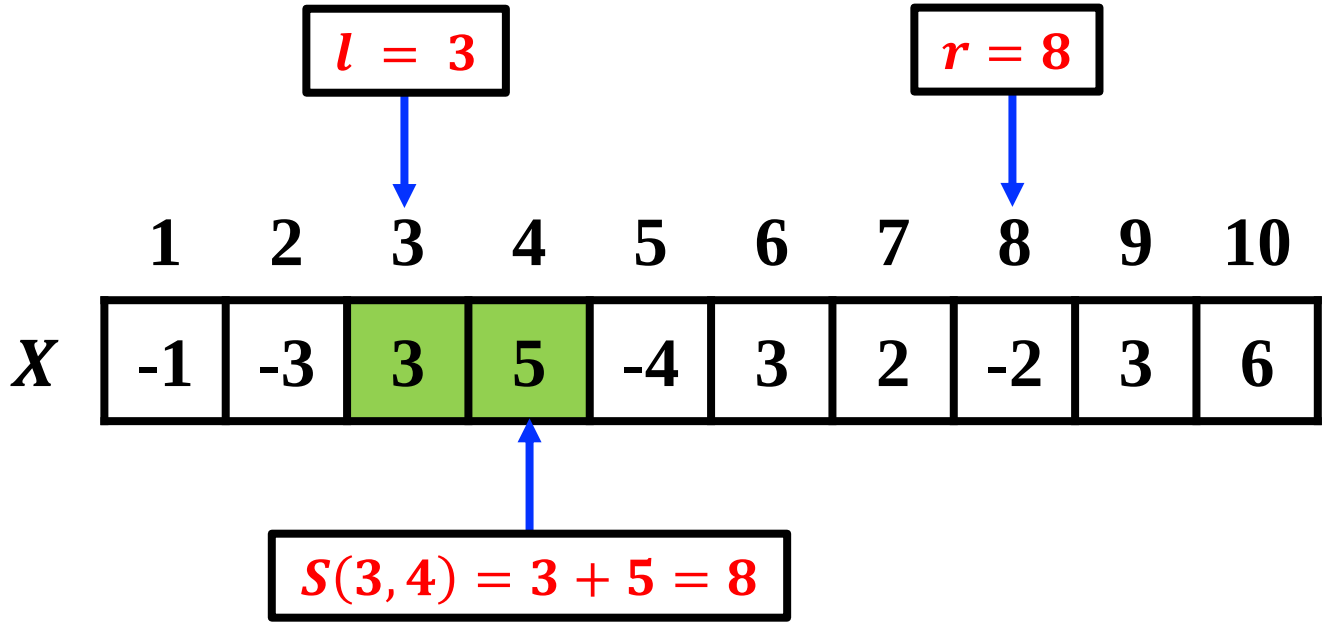
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



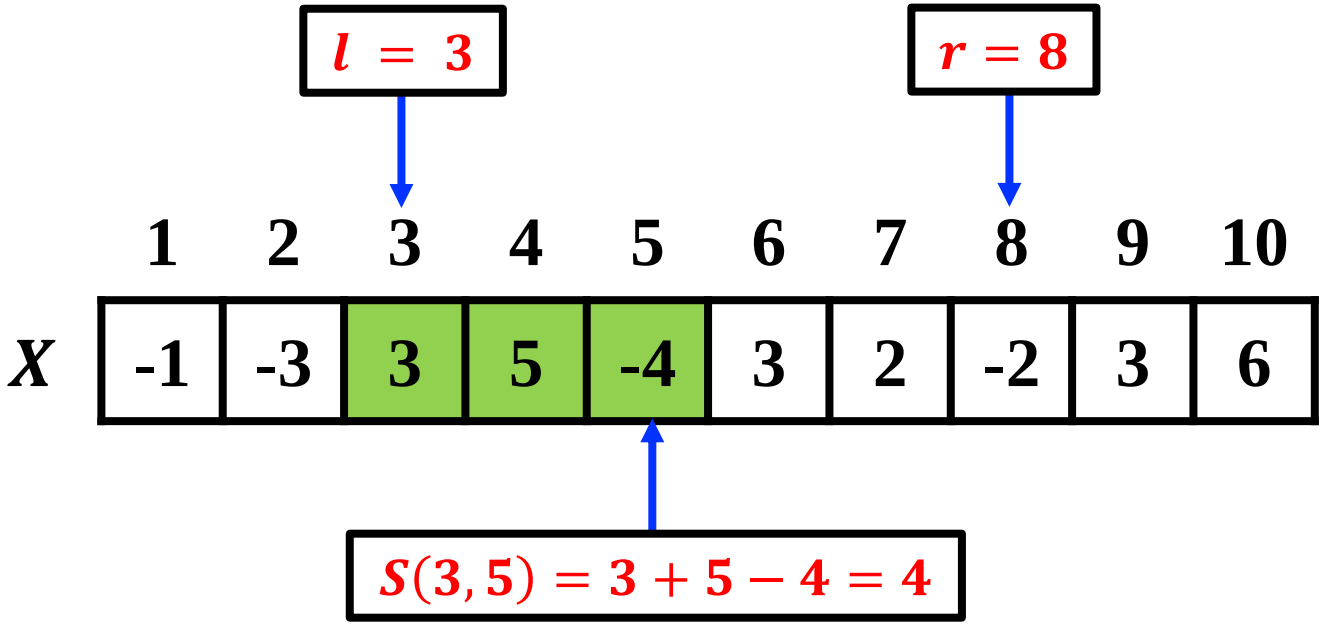
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



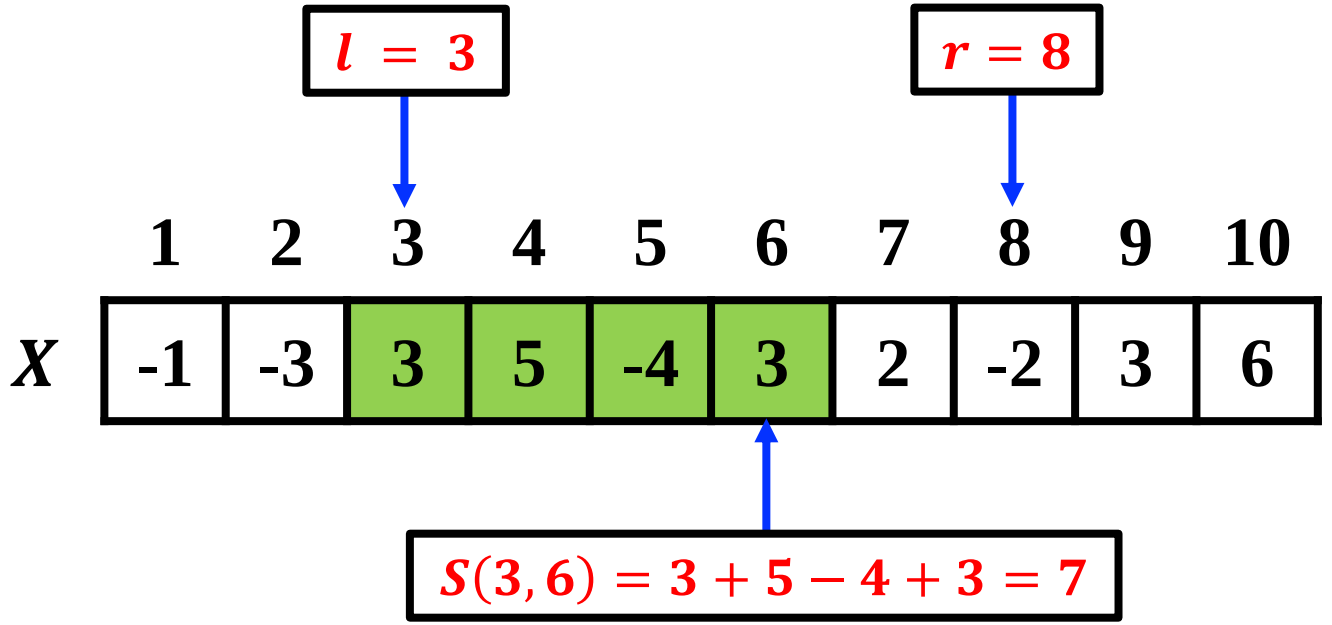
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



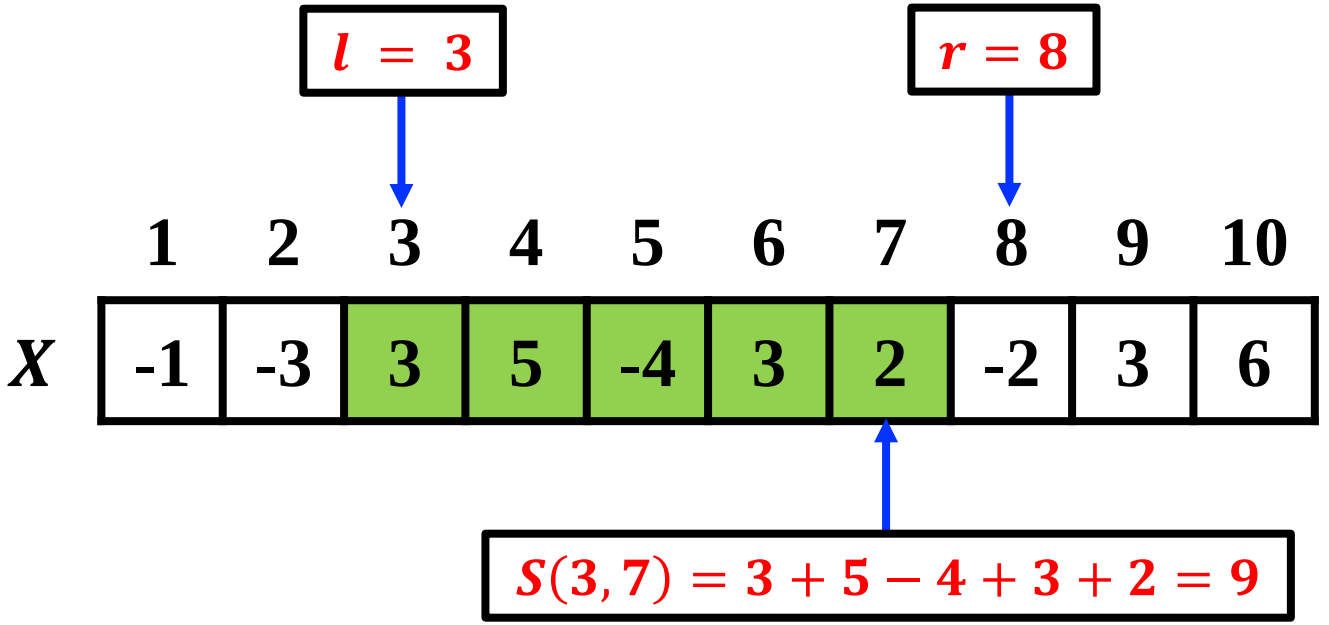
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



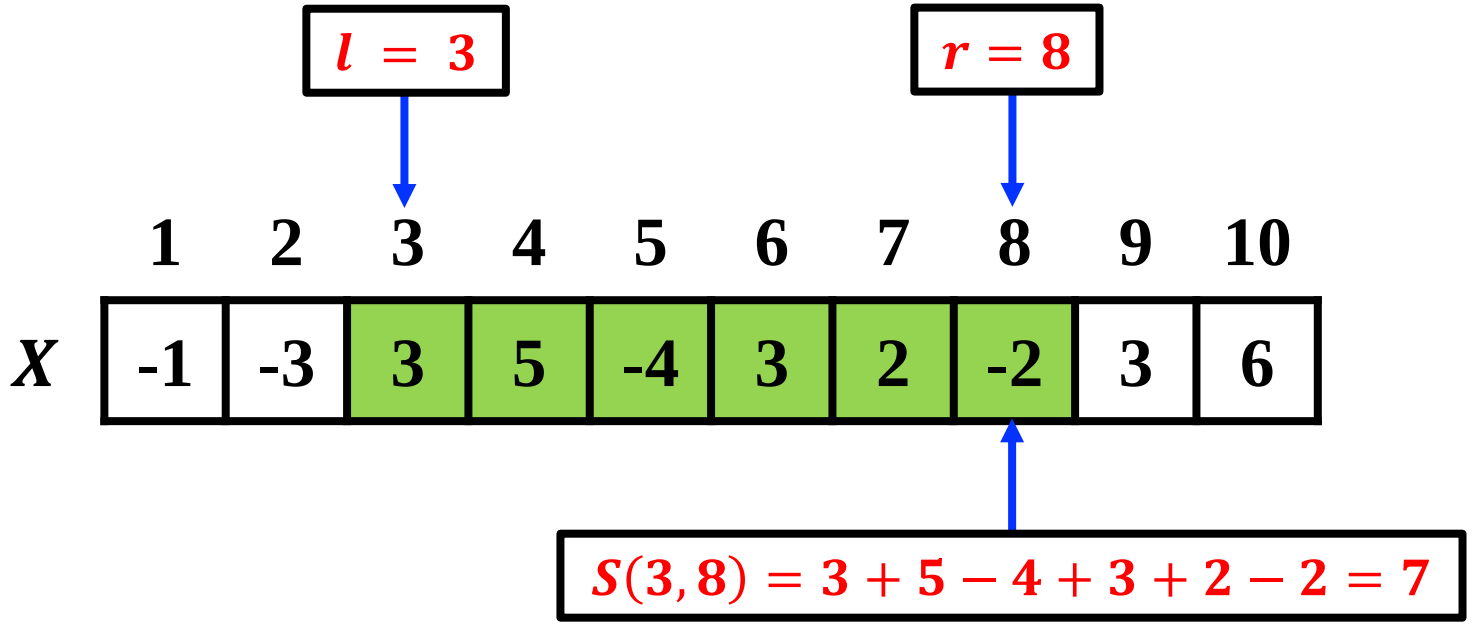
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



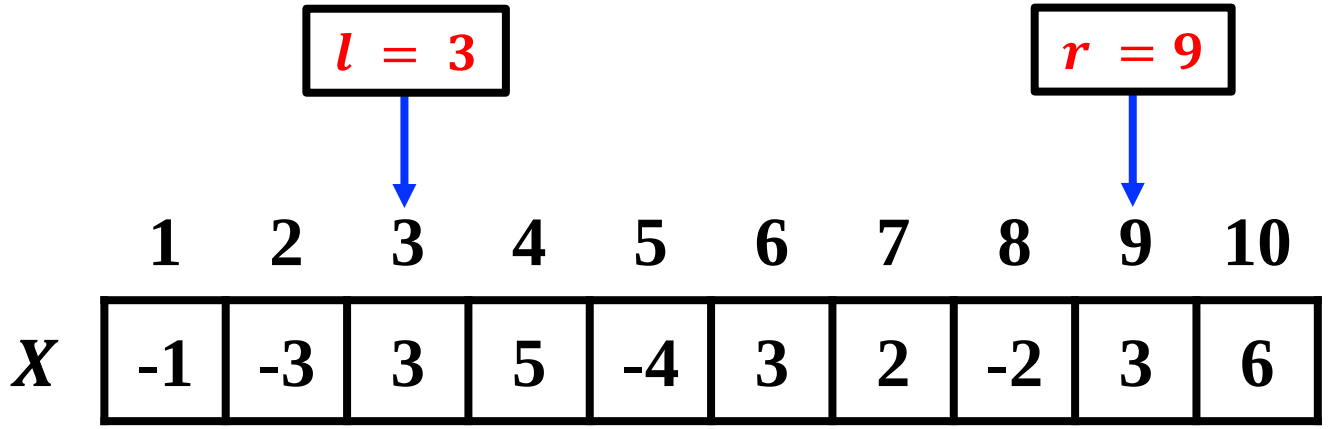
蛮力枚举：算法实例

- $l = 3,\ r = 8$
- 计算$S(3,8)$：



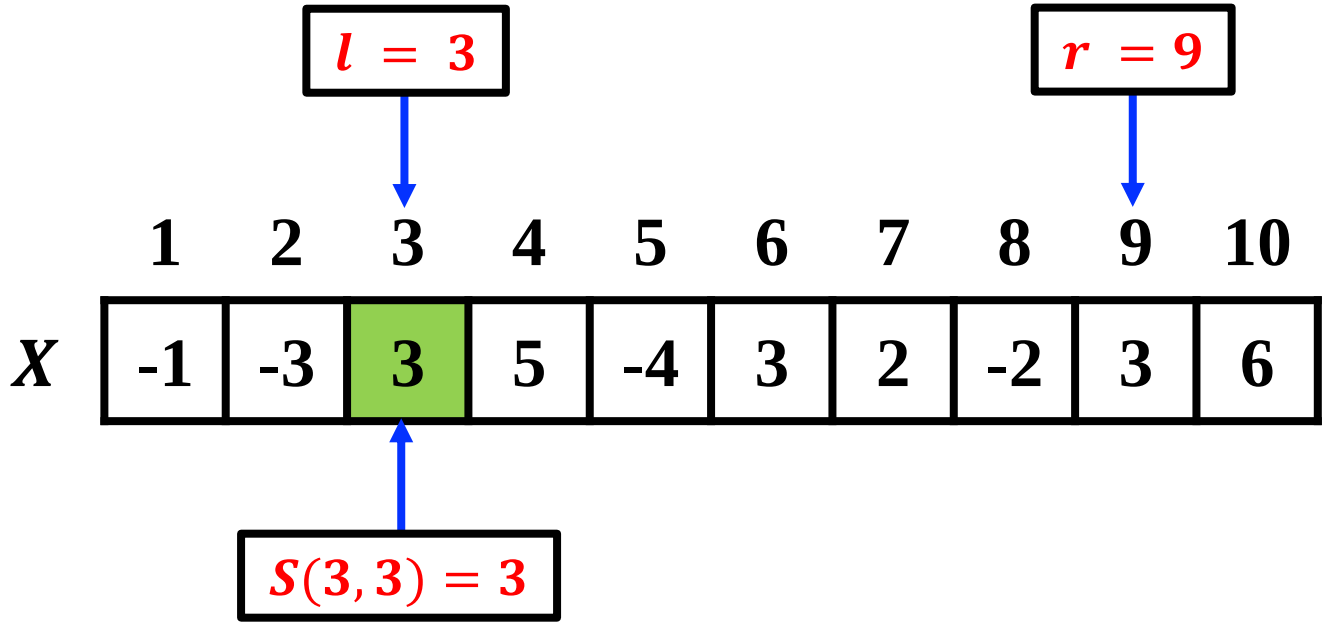
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3, 9)$：



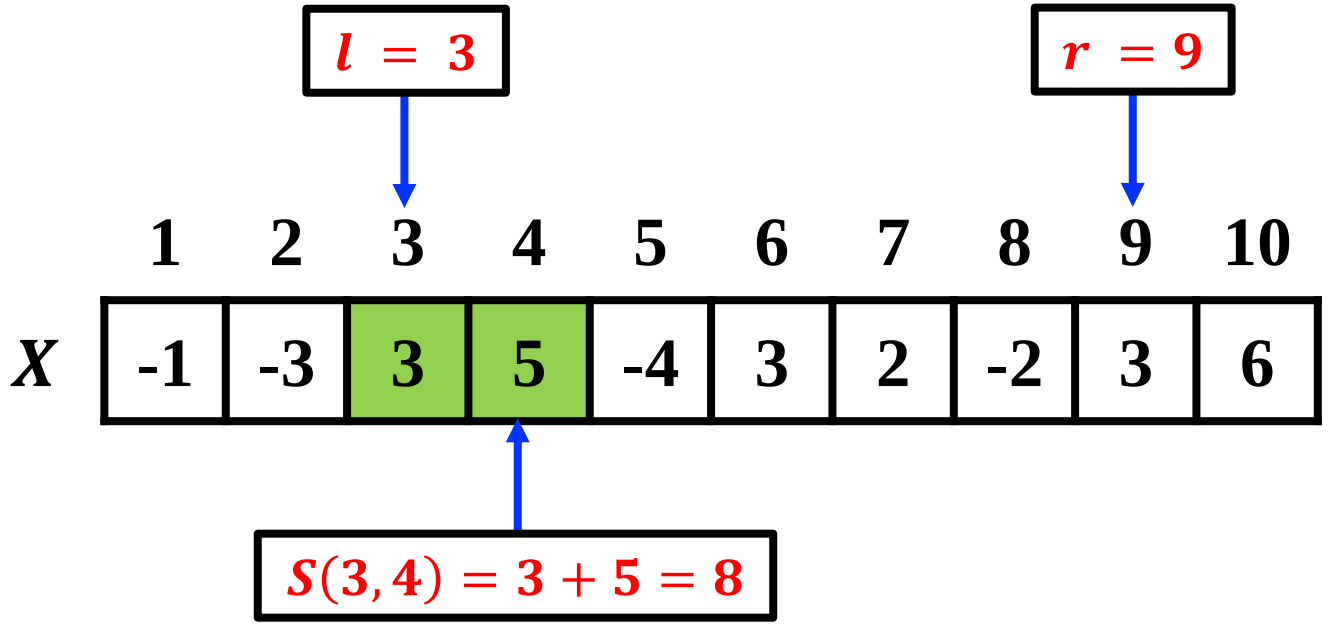
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



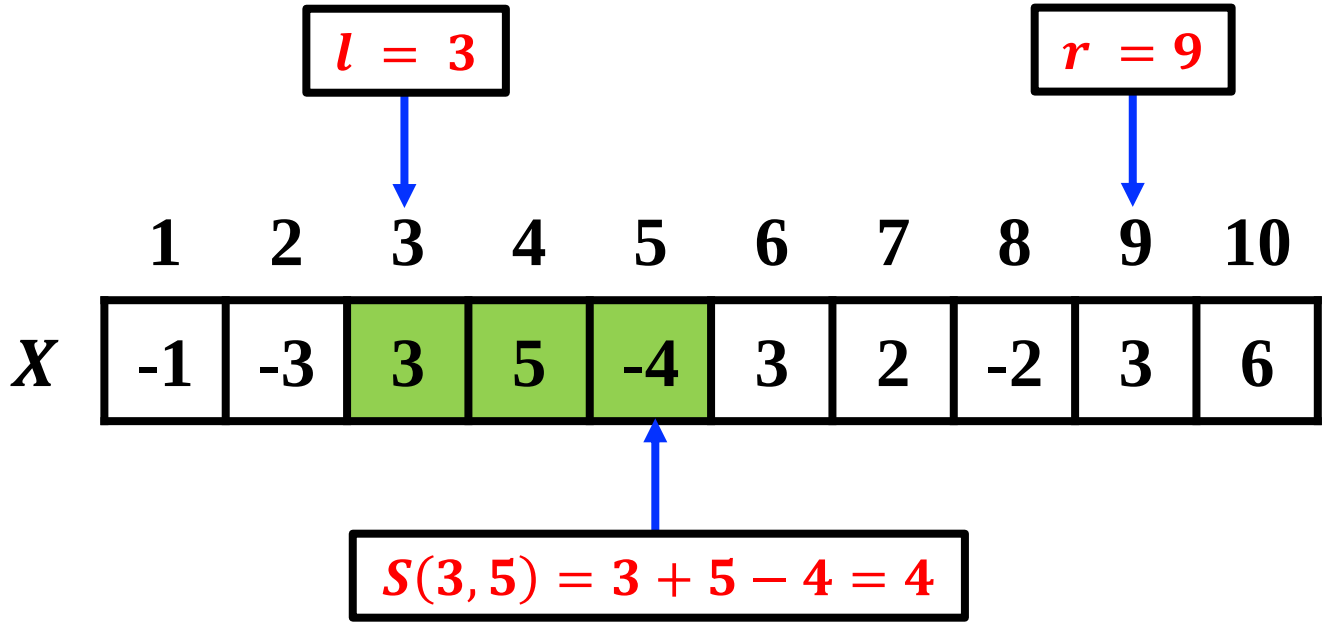
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



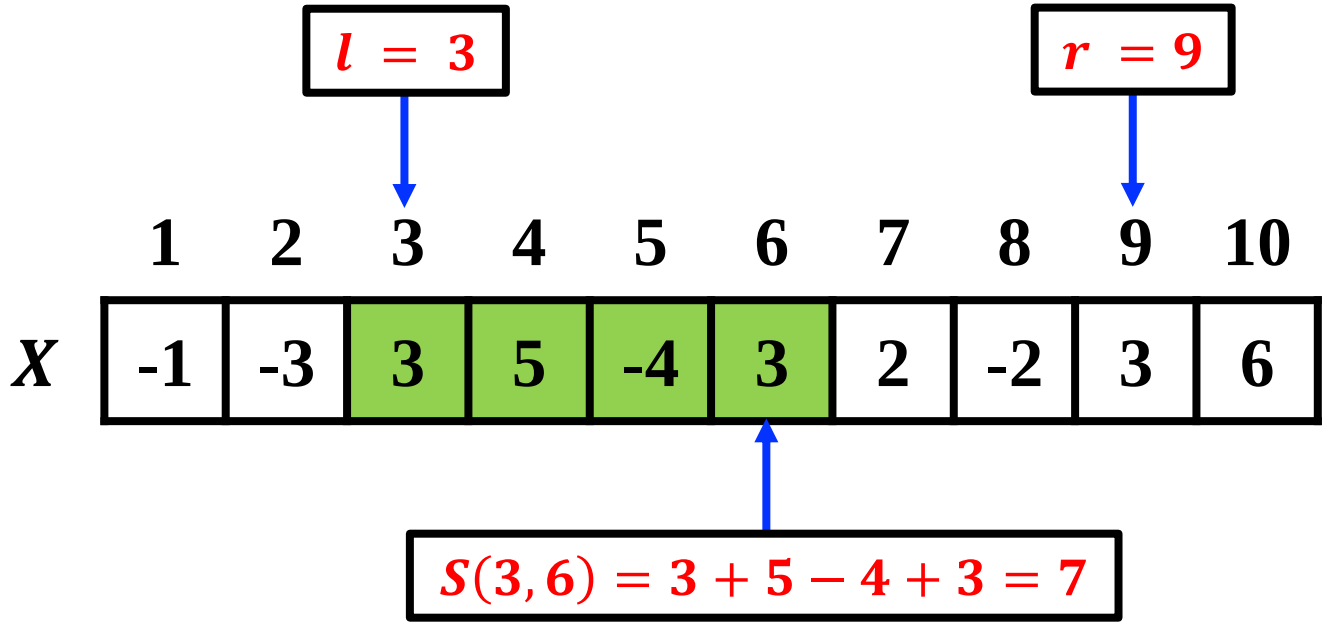
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



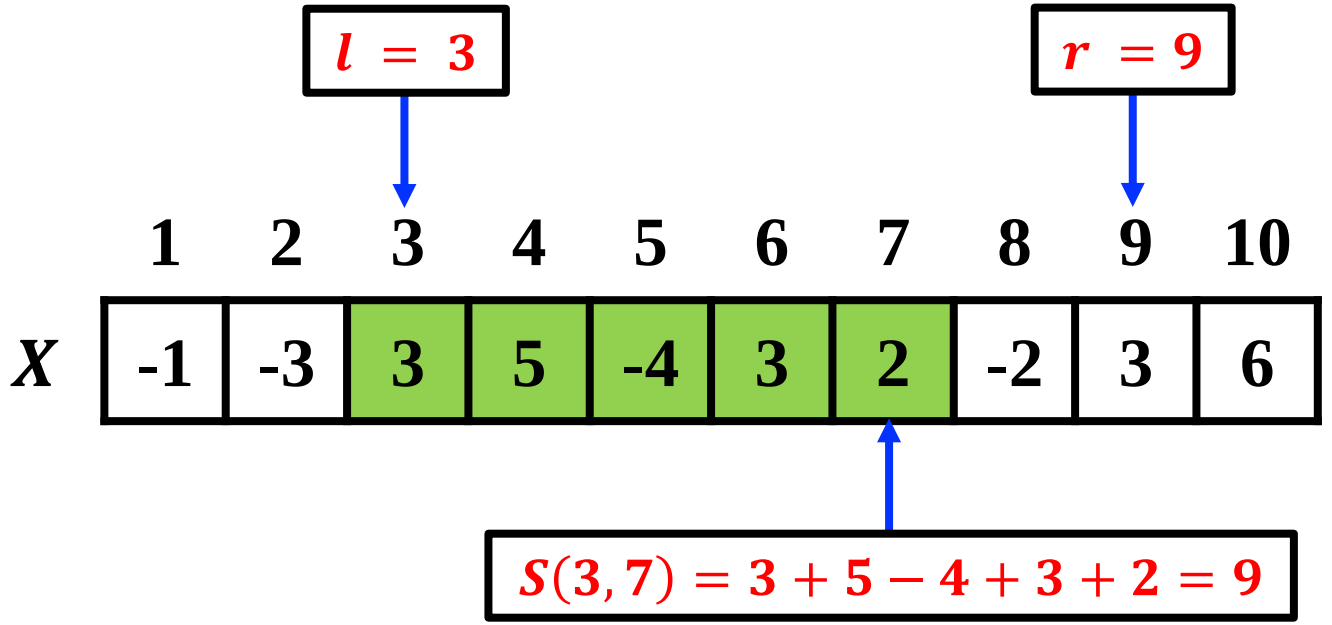
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



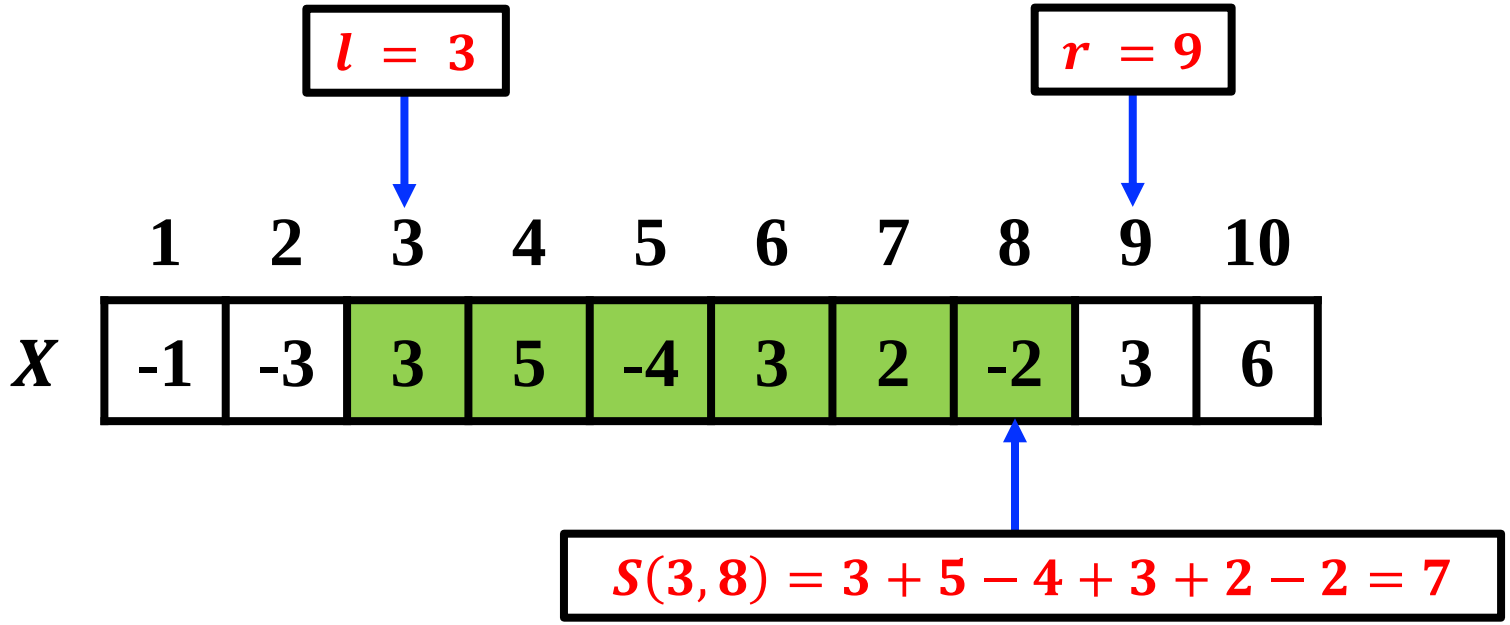
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



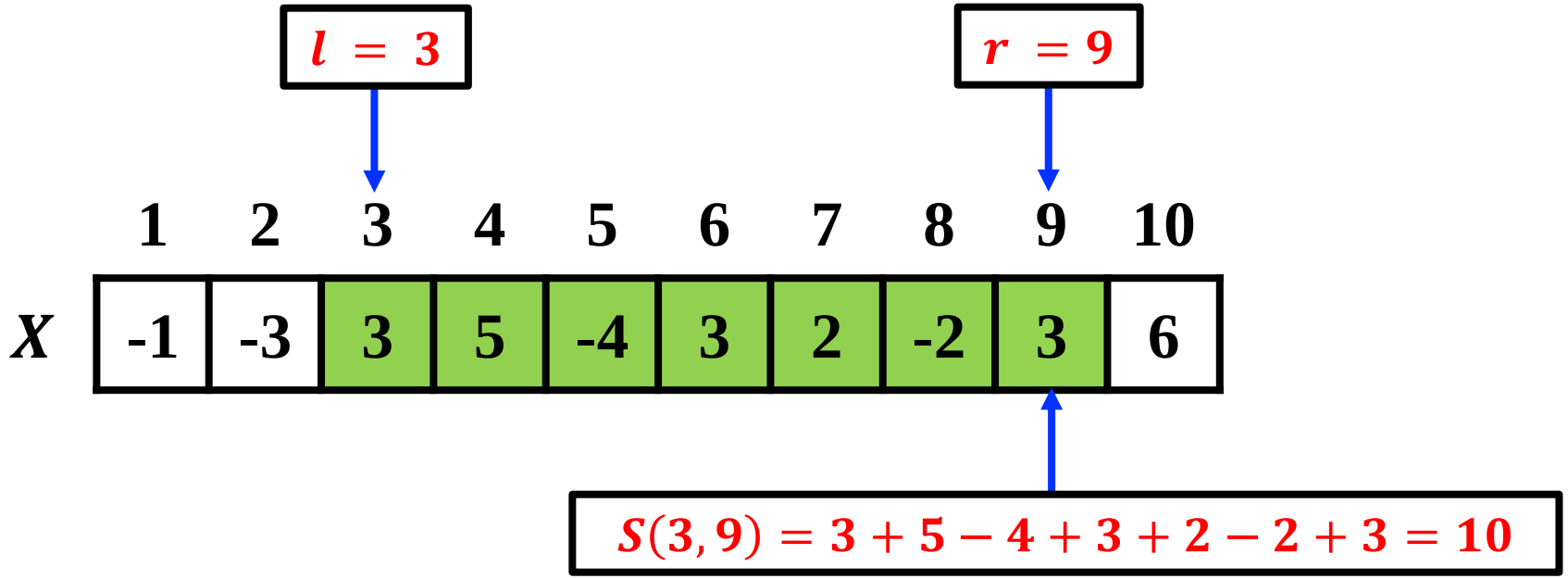
蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



蛮力枚举：算法实例

- $l = 3,\ r = 9$
- 计算$S(3,9)$：



蛮力枚举：伪代码

- 枚举$n + C_n^2$种可能的区间$[l, r](l \leq r)$，求解最大子数组之和S_{max}

输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    for r ← l to n do
        S(l, r) ← 0
        for i ← l to r do
            S(l, r) ← S(l, r) + X[i]
        end
        S_max ← max{S_max, S(l, r)}
    end
end
return S_max
```

枚举所有的起始下标l

蛮力枚举：伪代码

- 枚举$n + C_n^2$种可能的区间$[l, r](l \leq r)$，求解最大子数组之和S_{max}

输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    for r ← l to n do
        S(l,r) ← 0
        for i ← l to r do
            S(l,r) ← S(l,r) +X[i]
        end
        S_max ← max{S_max, S(l,r)}
    end
end
return S_max
```

枚举所有的终止下标r

蛮力枚举：伪代码

- 枚举$n + C_n^2$种可能的区间$[l, r](l \leq r)$，求解最大子数组之和S_{max}

输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    for r ← l to n do
        S(l, r) ← 0
        for i ← l to r do
            S(l, r) ← S(l, r) + X[i]
        end
        S_max ← max{S_max, S(l, r)}
    end
end
return S_max
```

迭代计算$S(l, r)$并更新S_{max}

蛮力枚举：伪代码

- 枚举$n + C_n^2$种可能的区间$[l, r](l \leq r)$，求解最大子数组之和S_{max}

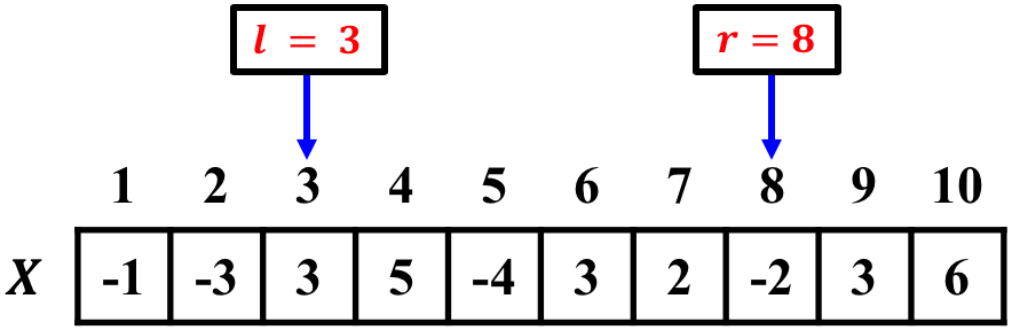
输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    for r ← l to n do
        S(l, r) ← 0
        for i ← l to r do
            S(l, r) ← S(l, r) + X[i]
        end
        S_max ← max{S_max, S(l, r)}
    end
end
return S_max
```

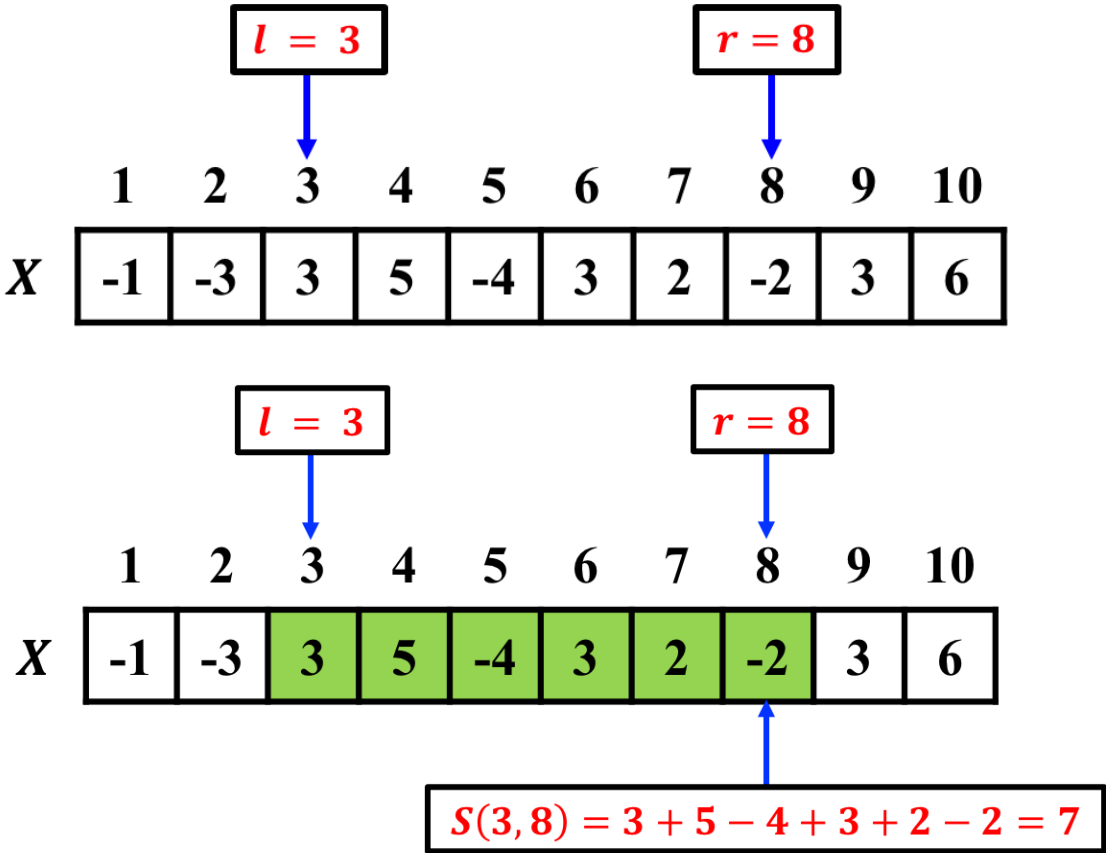
$O(n)$ $O(n^2)$ $O(n^3)$

时间复杂度：$O(n^3)$

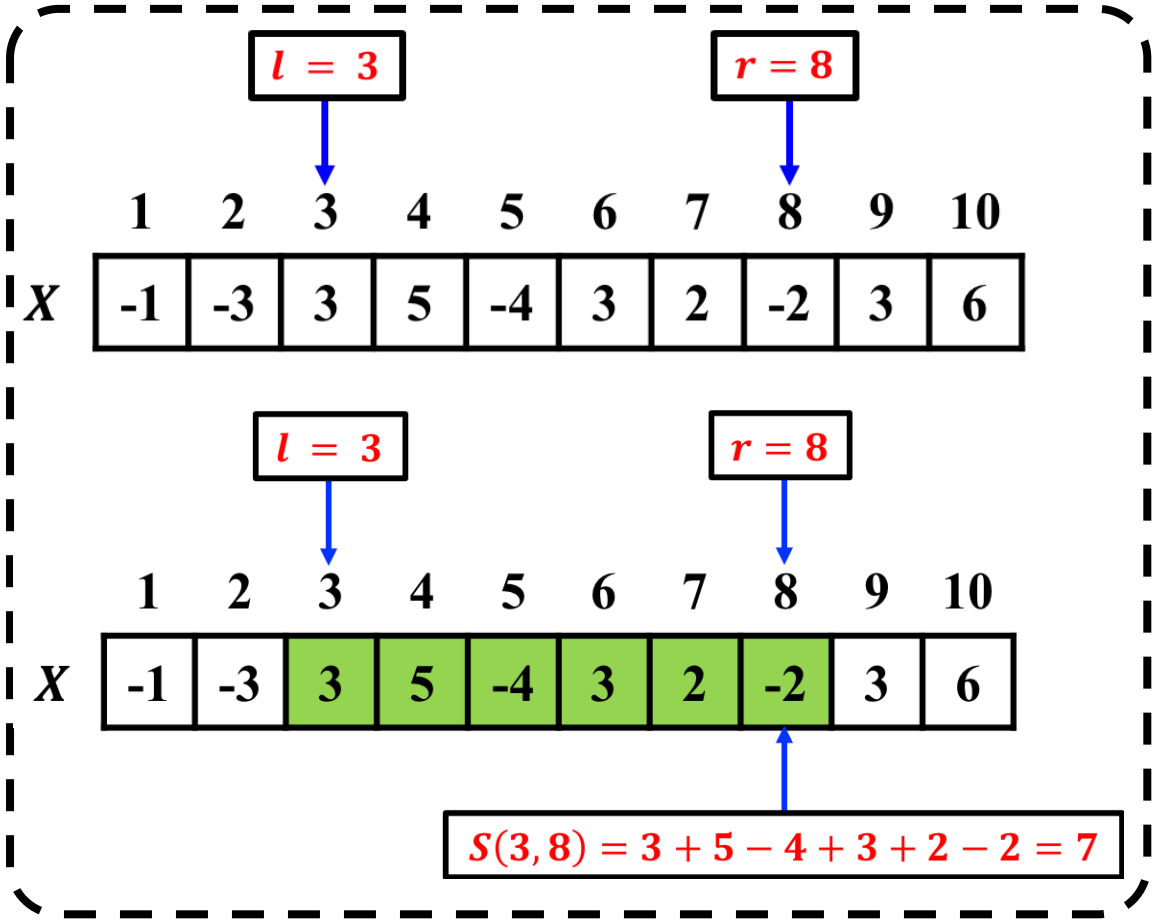
蛮力枚举：实例分析



蛮力枚举：实例分析

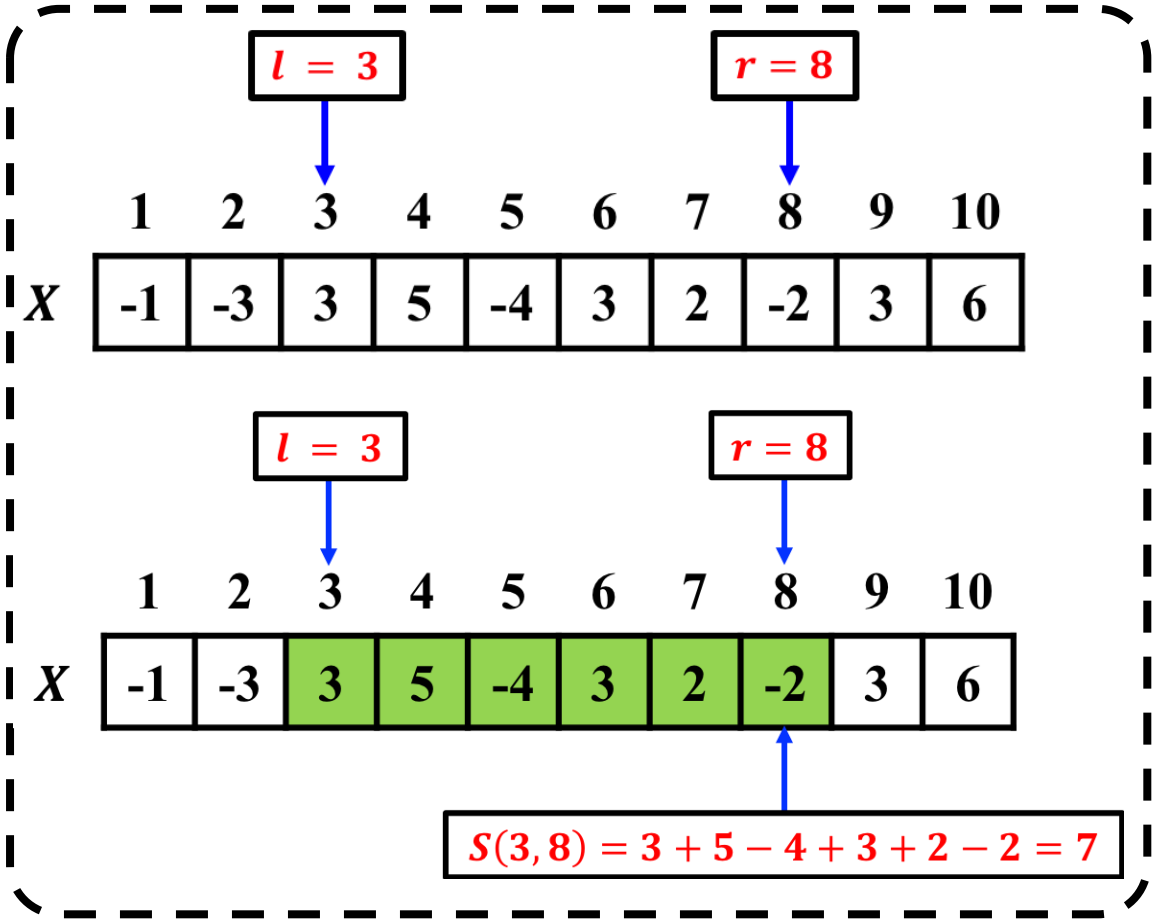

l = 3
r = 8
1 2 3 4 5 6 7 8 9 10
X
-1 -3 3 5 -4 3 2 -2 3 6
l = 3
r = 8
1 2 3 4 5 6 7 8 9 10
X
-1 -3 3 5 -4 3 2 -2 3 6
S(3,8) = 3 + 5 − 4 + 3 + 2 − 2 = 7

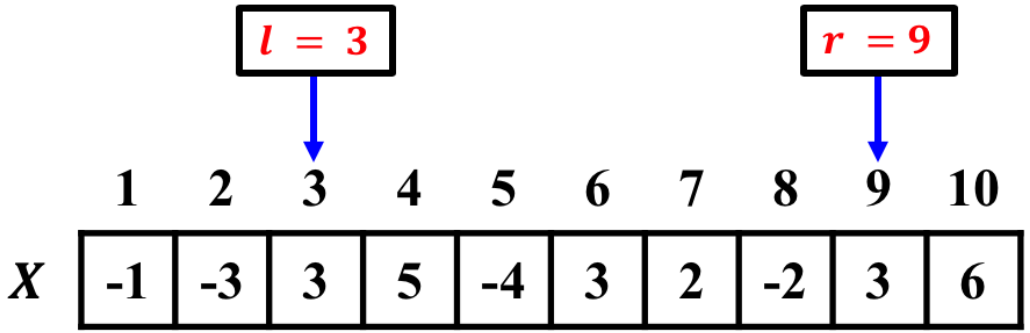
蛮力枚举：实例分析



计算$S(3,8)$循环6次

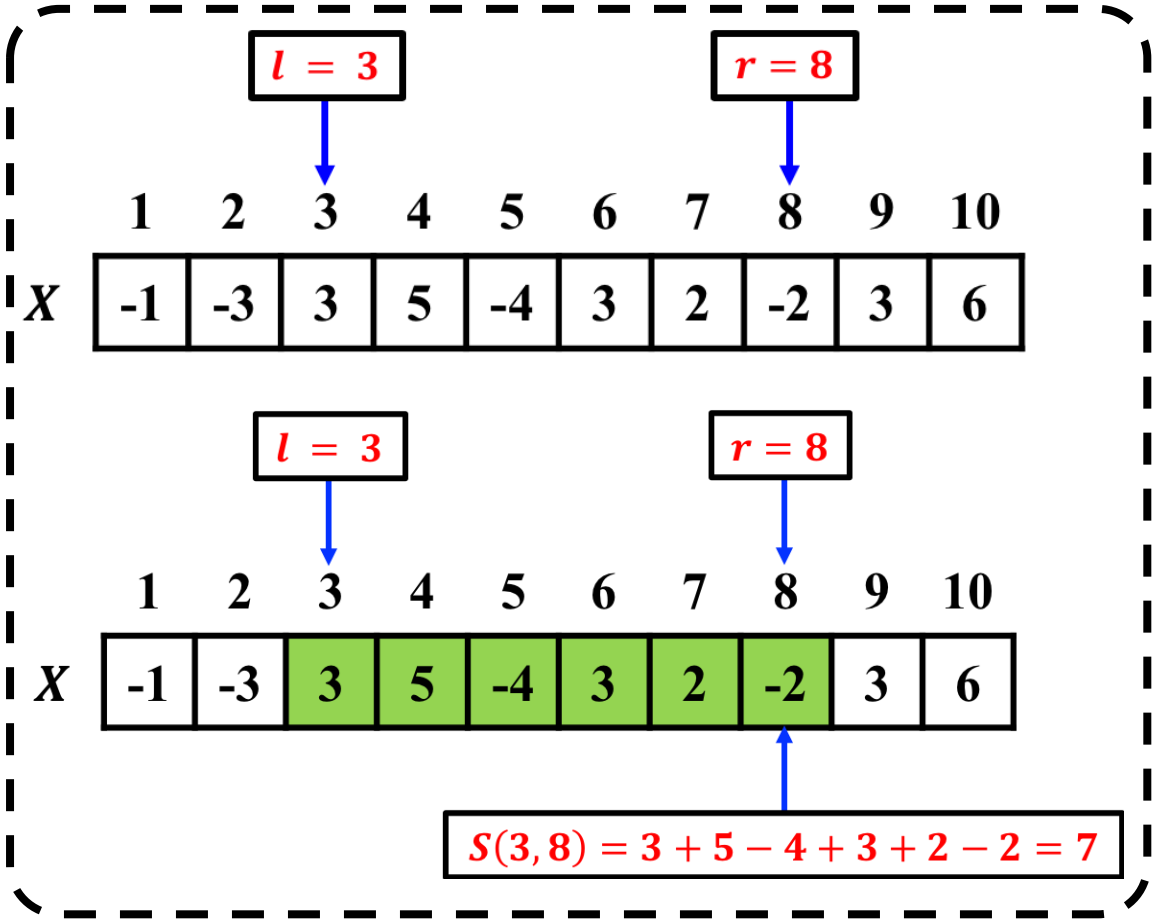
蛮力枚举：实例分析

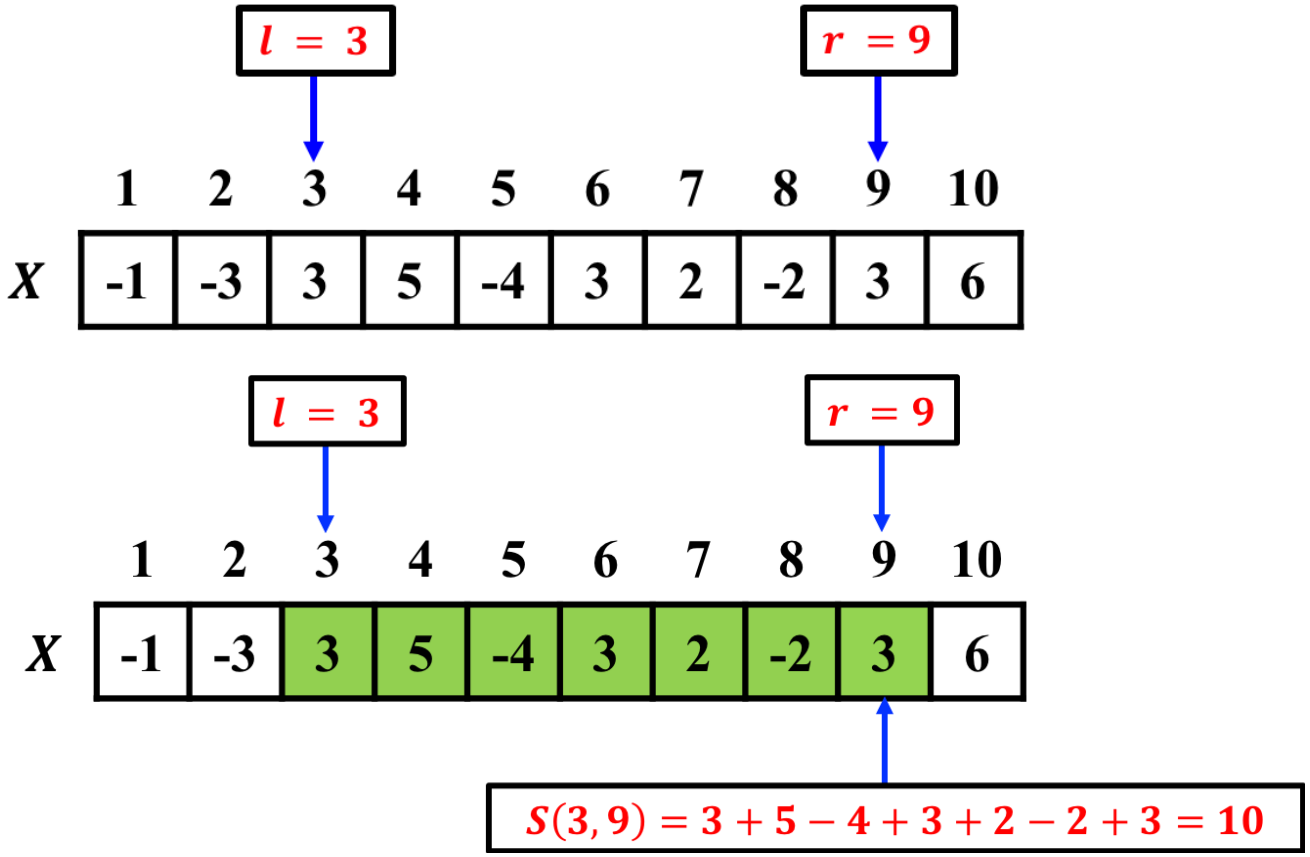




计算$S(3,8)$循环6次

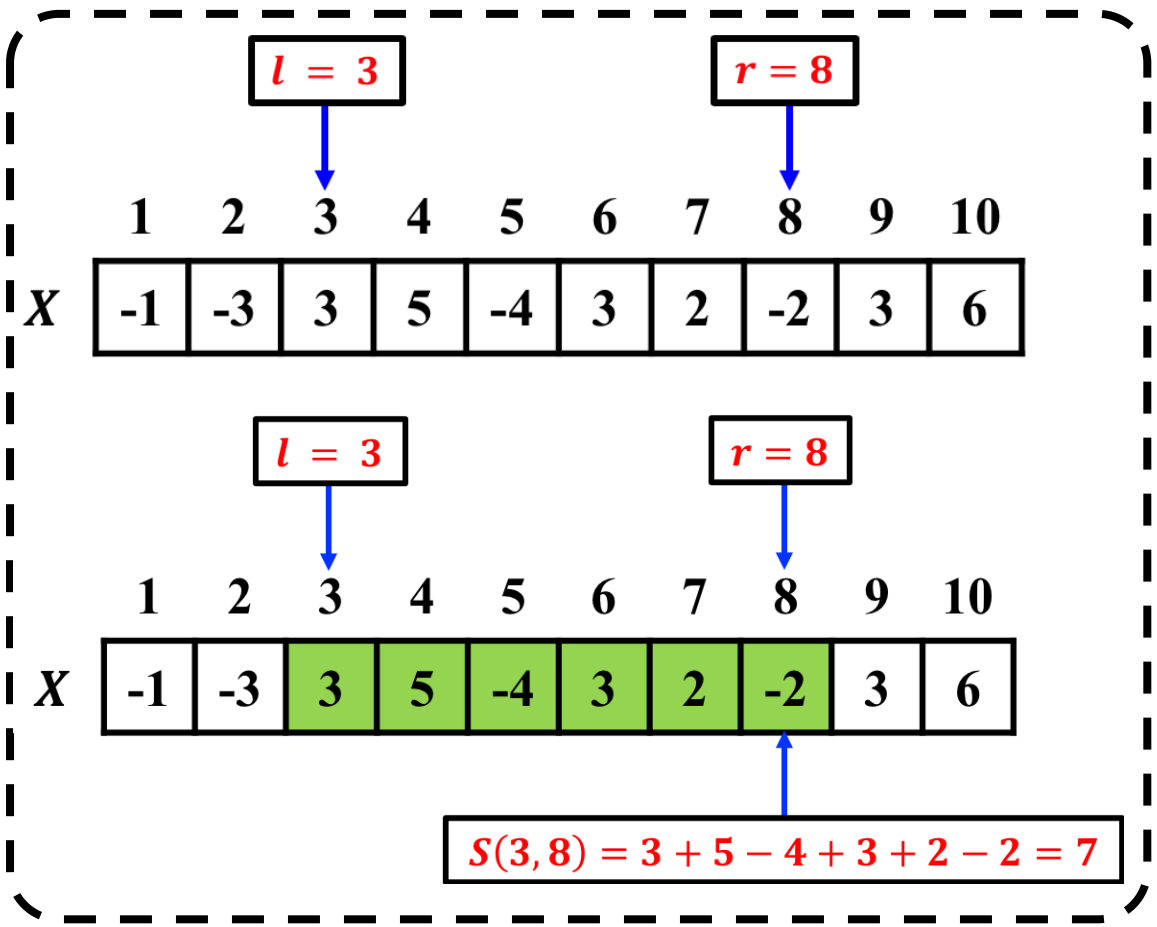
蛮力枚举：实例分析



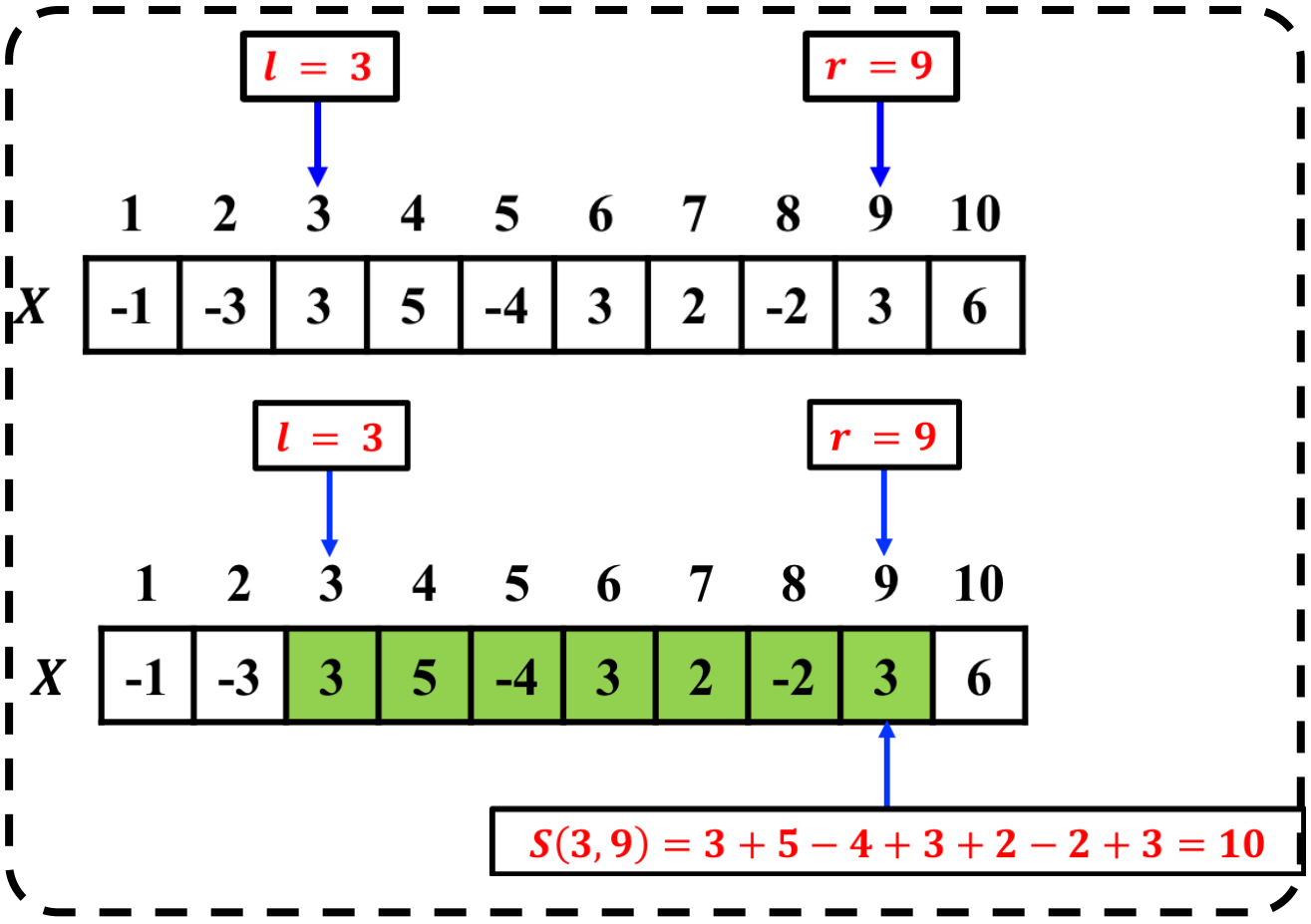


计算$S(3,8)$循环6次

蛮力枚举：实例分析

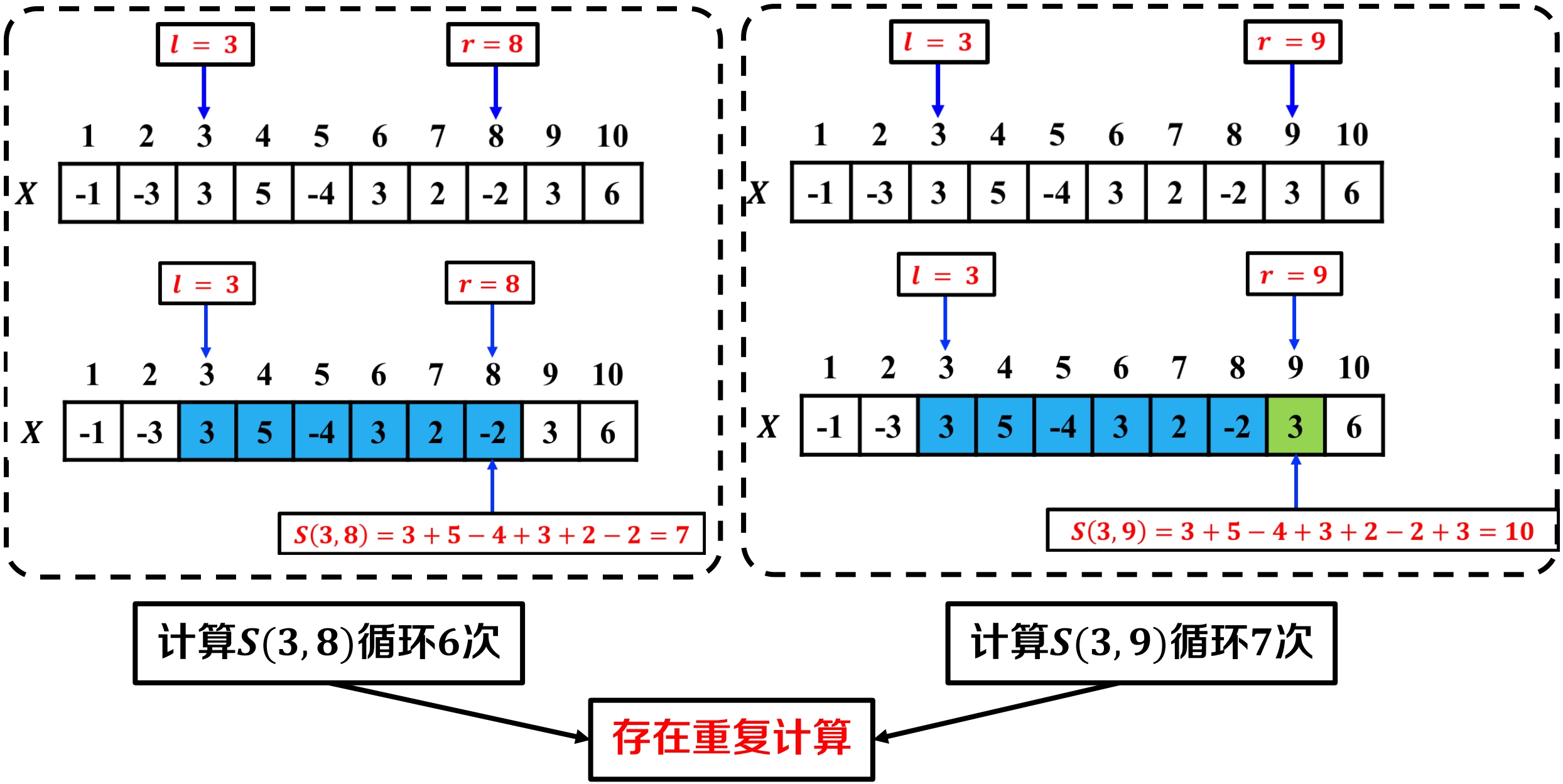


计算$S(3,8)$循环6次



计算$S(3,9)$循环7次

蛮力枚举：实例分析

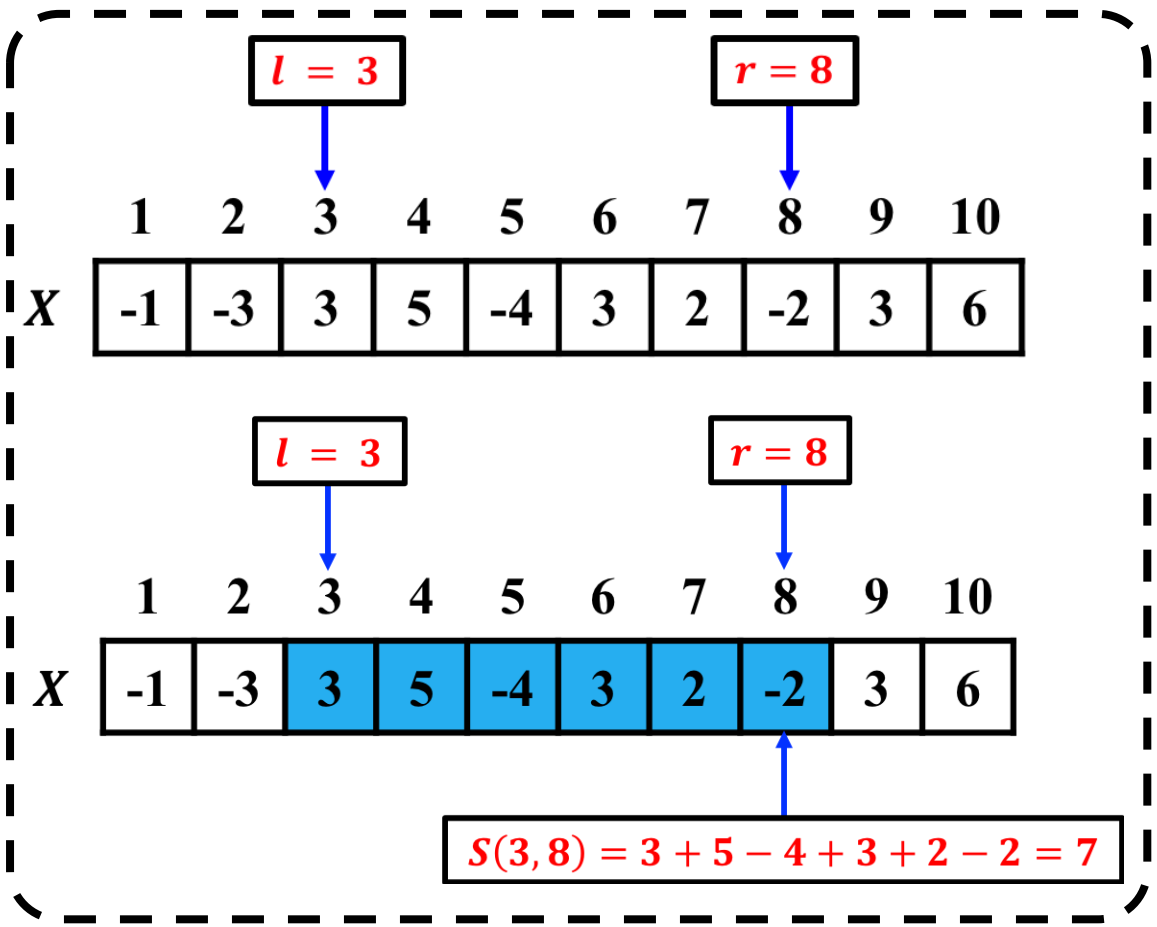


计算$S(3,8)$循环6次

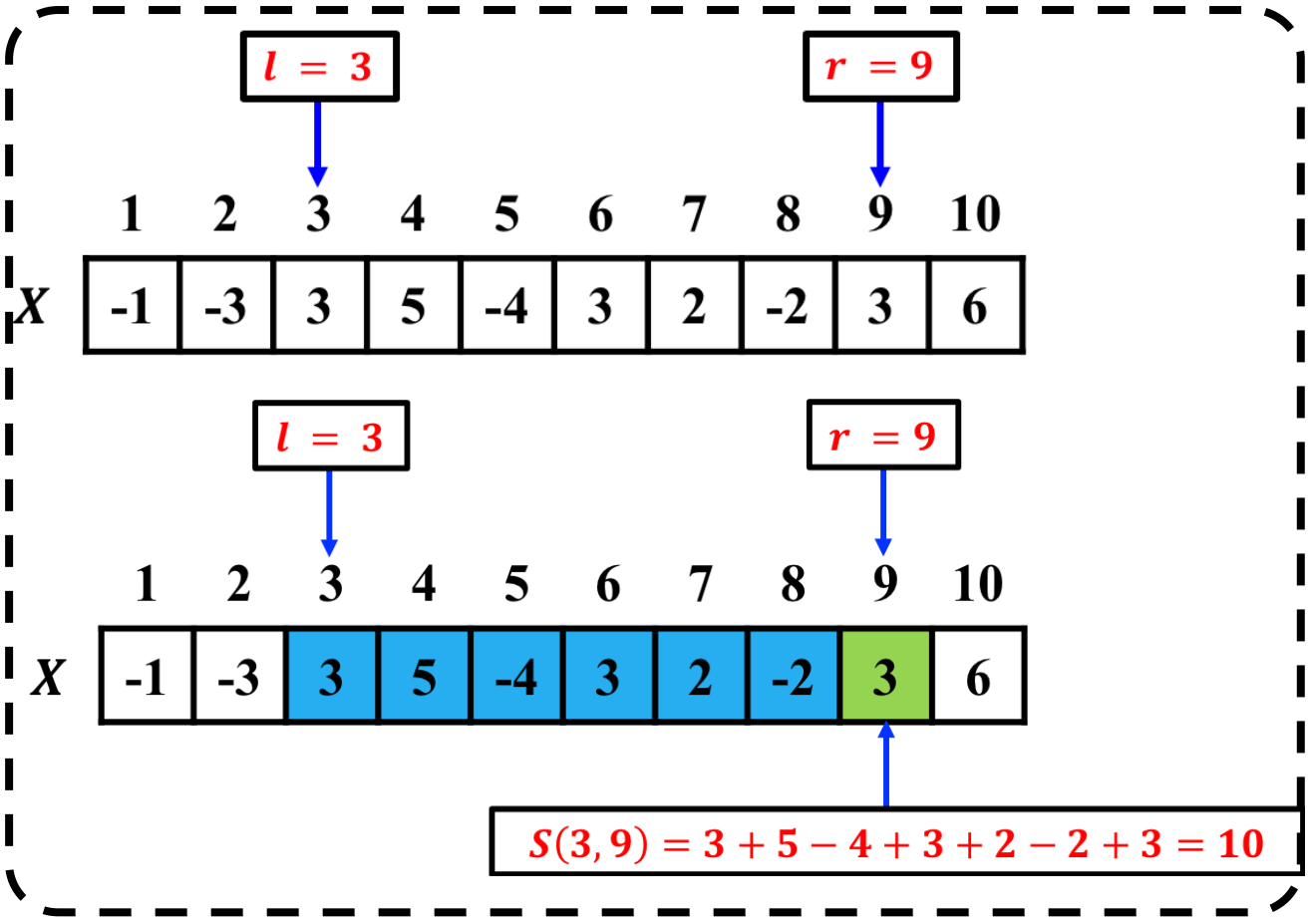
计算$S(3,9)$循环7次

存在重复计算

蛮力枚举：实例分析



计算$S(3,8)$循环6次

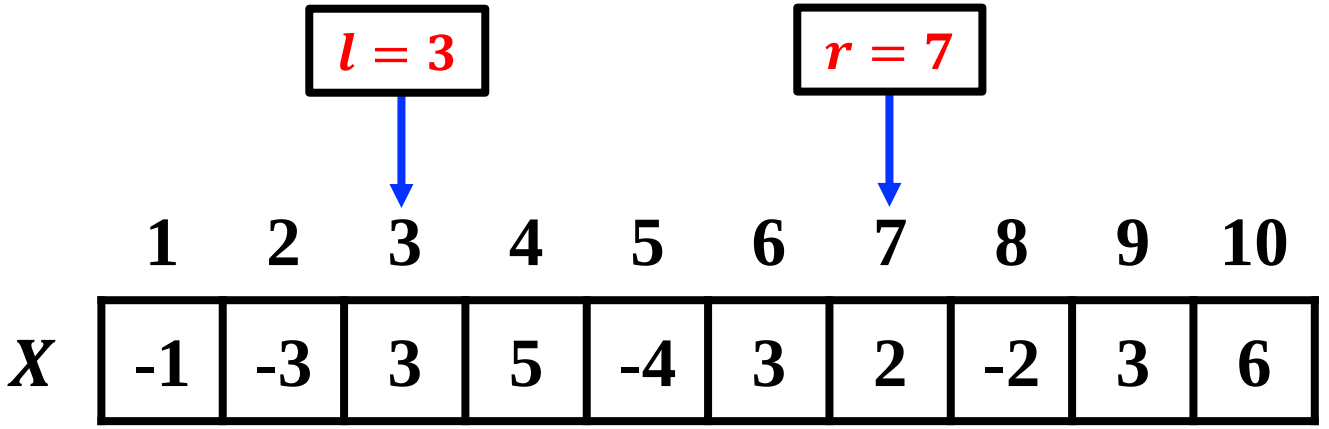


计算$S(3,9)$循环7次

问题：如何减少重复计算？

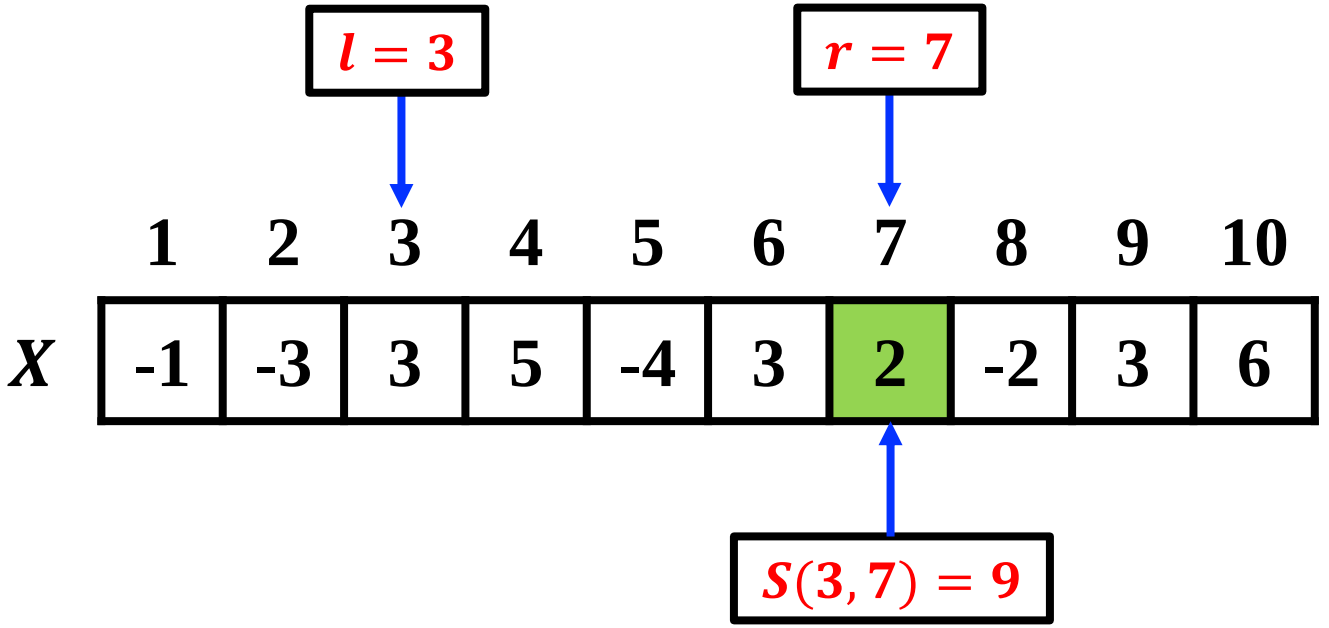
优化枚举：算法实例

- $l = 3,\ r = 7$



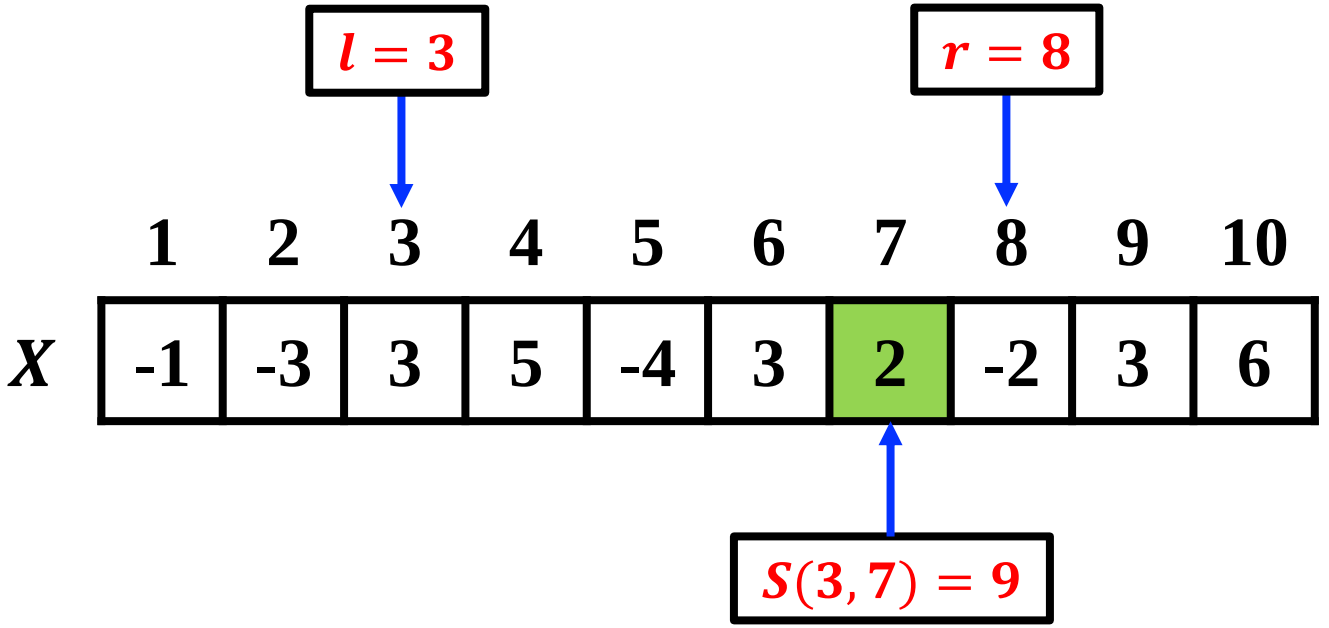
优化枚举：算法实例

- $l = 3,\ r = 7,\ S(3,7) = 9$



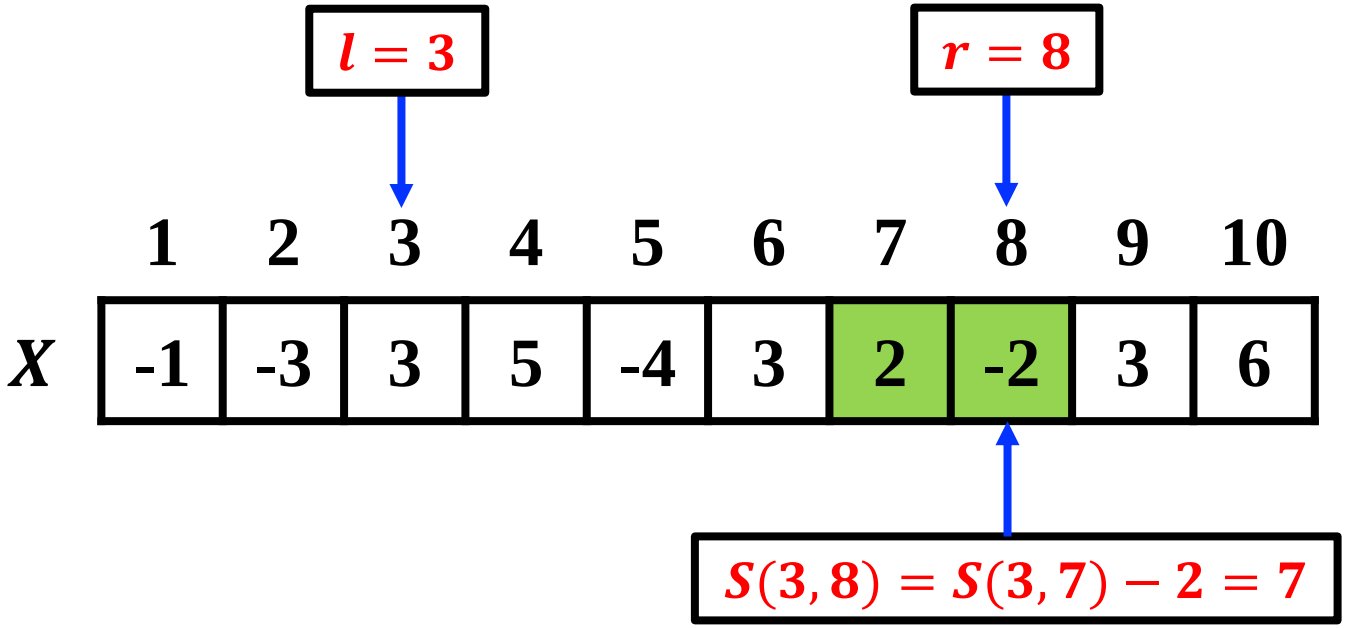
优化枚举：算法实例

- $l = 3,\ r = 8,\ S(3,8) = ?$



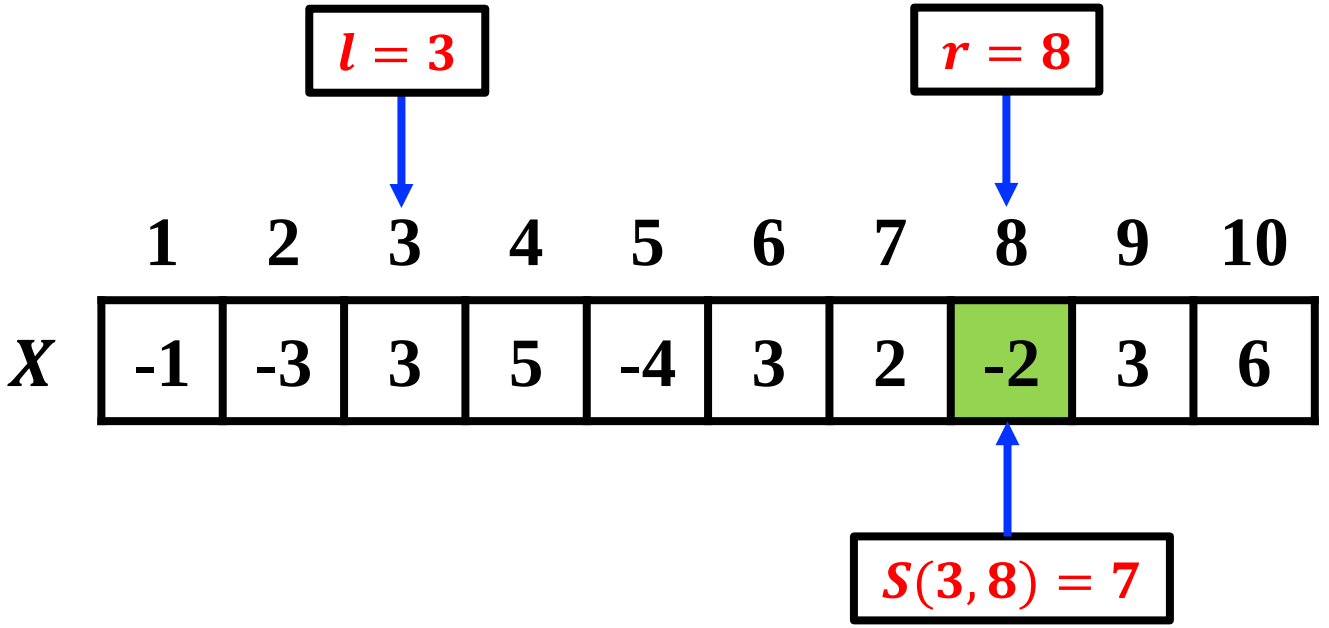
优化枚举：算法实例

- $l = 3,\ r = 8,\ S(3,8) = 7$



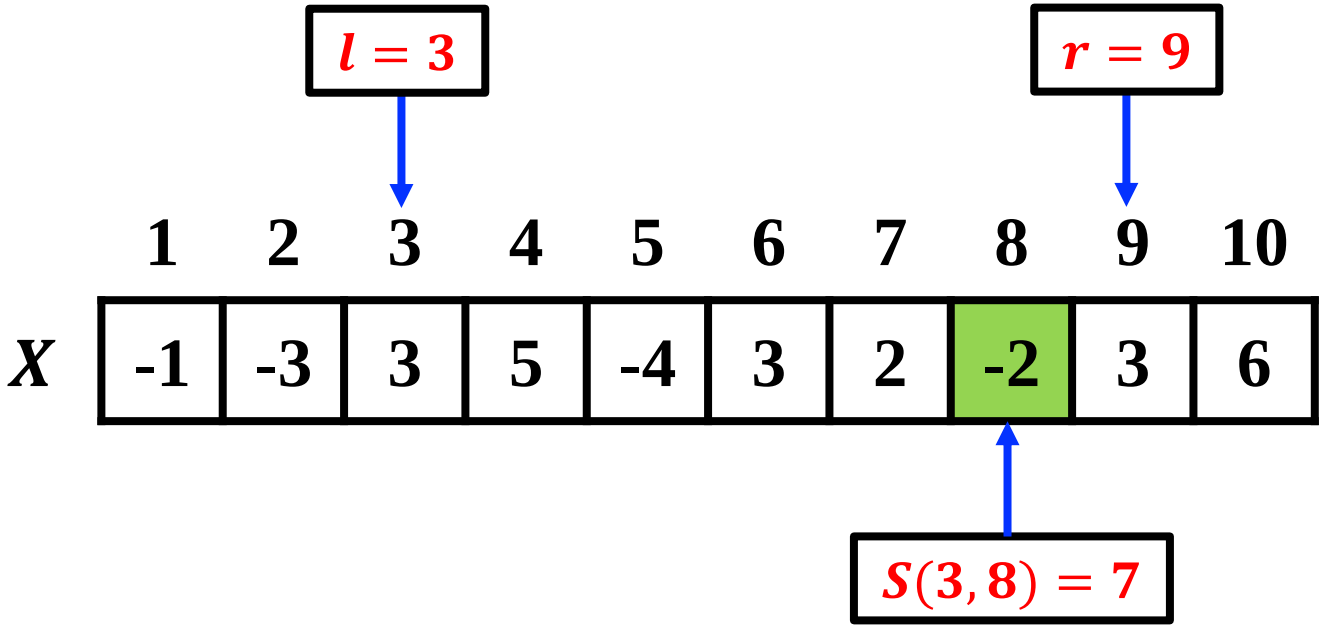
优化枚举：算法实例

- $l = 3,\ r = 8,\ S(3,8) = 7$



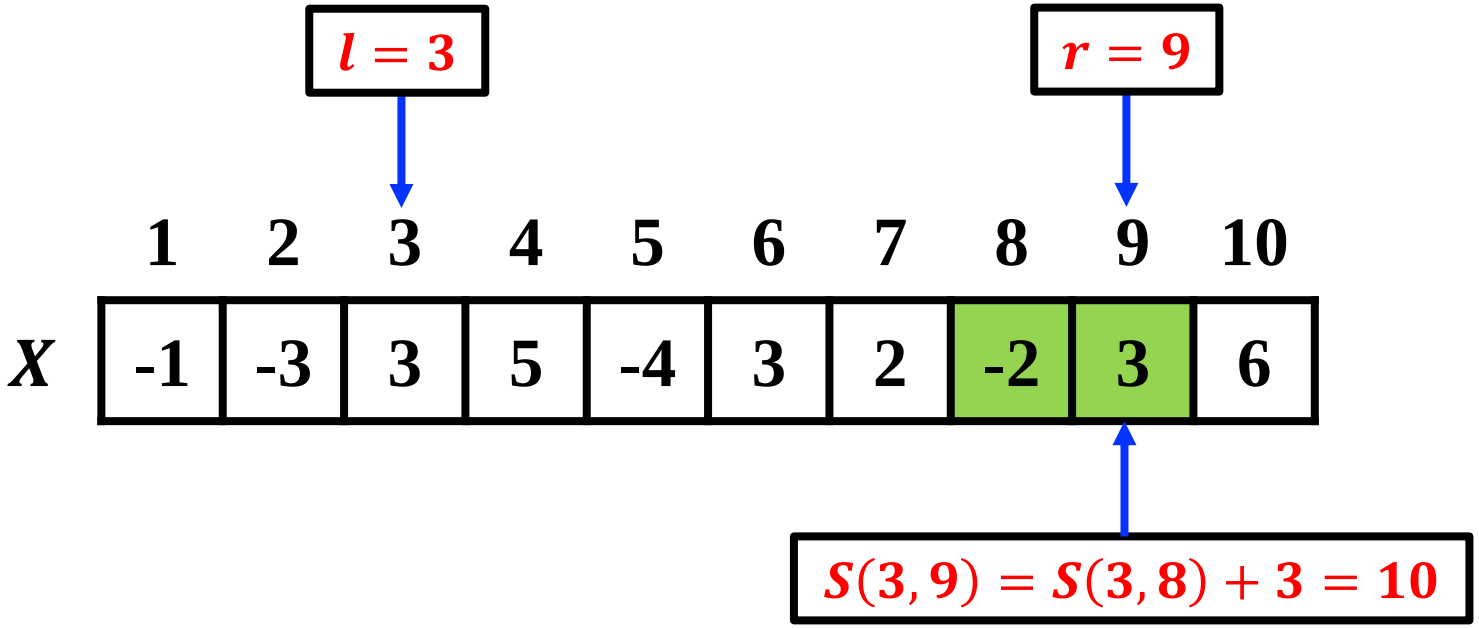
优化枚举：算法实例

- $l = 3,\ r = 9,\ S(3,9) = ?$

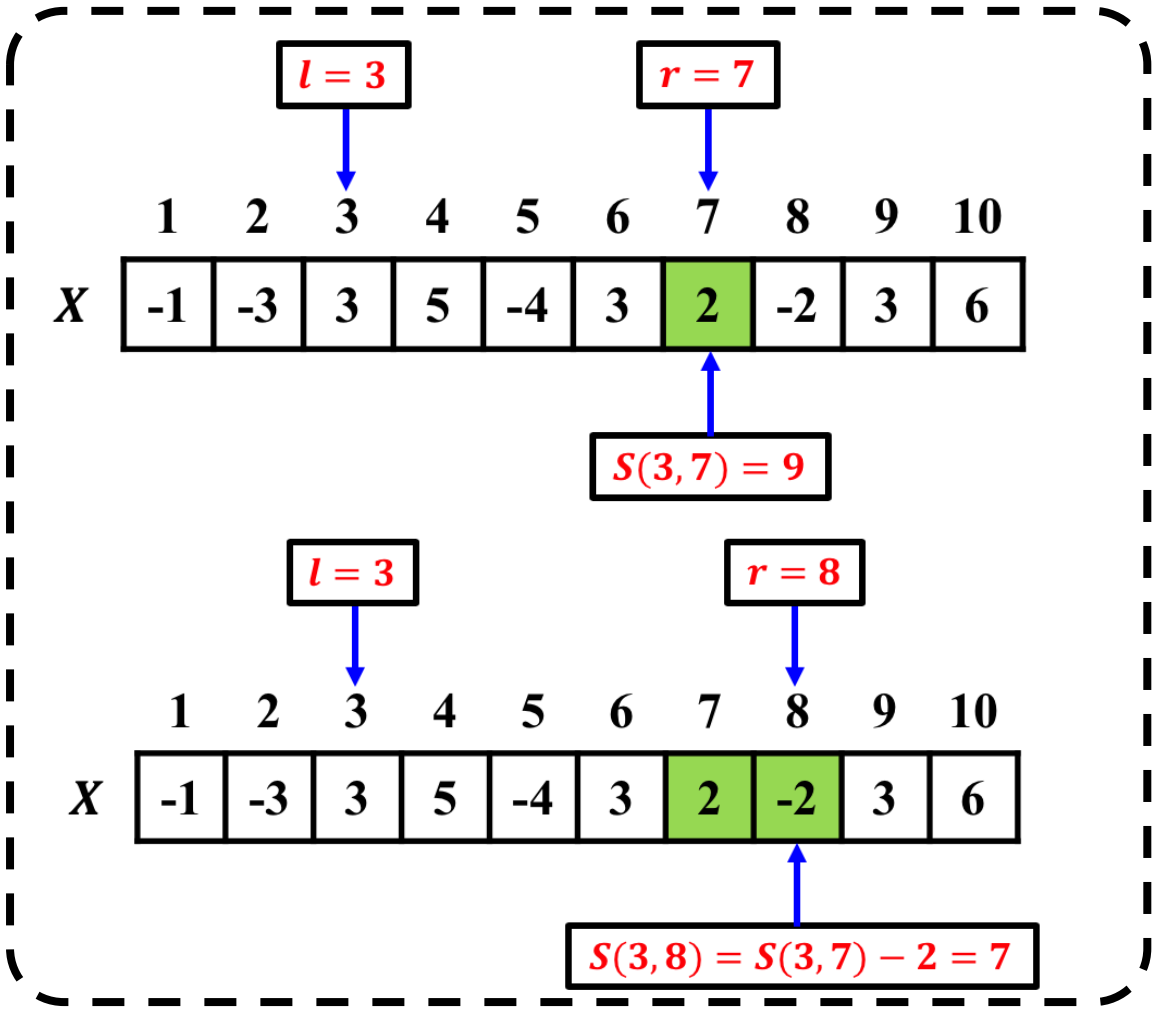


优化枚举：算法实例

- $l = 3,\ r = 9,\ S(3,9) = ?$

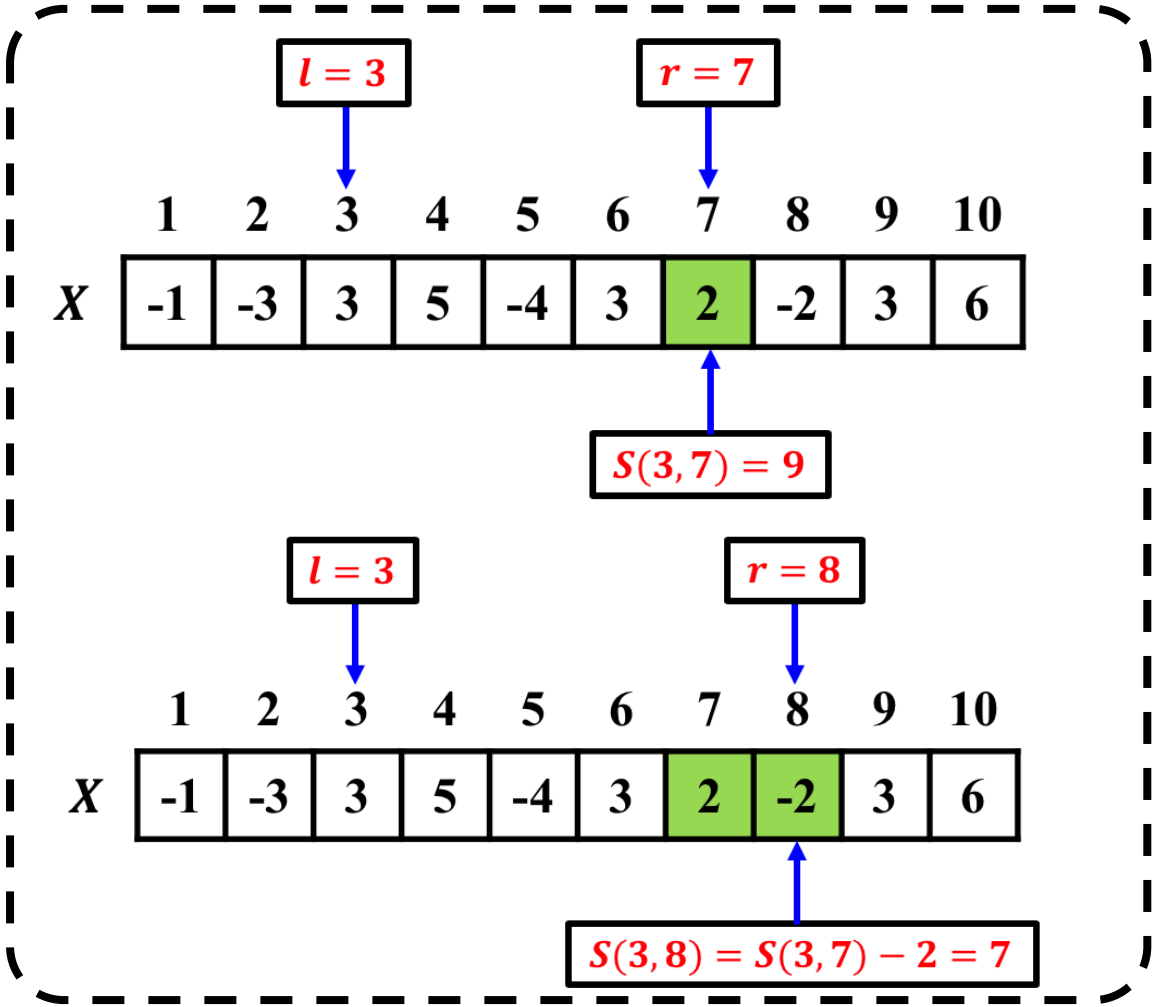


优化枚举：实例分析

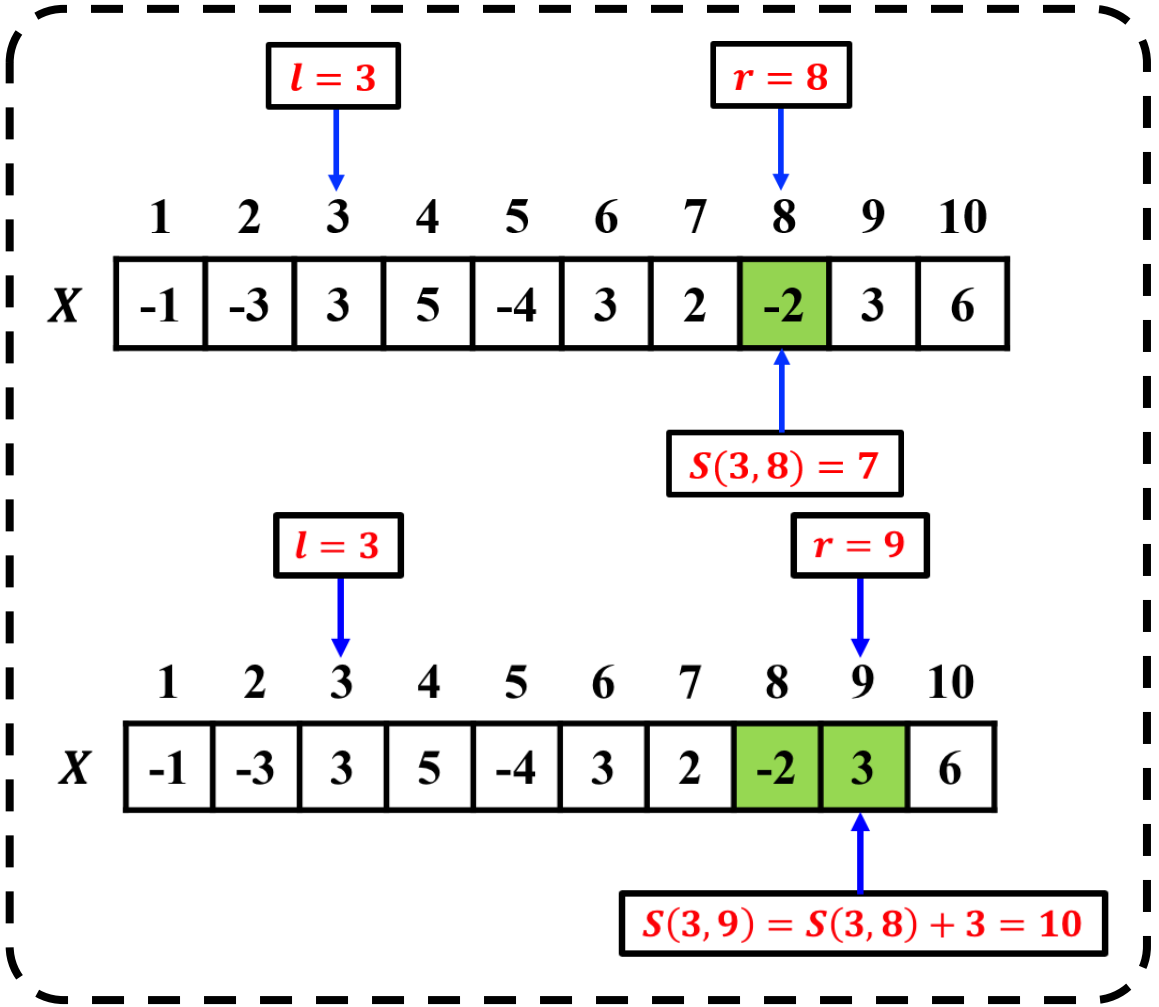


计算$S(3,8)$操作1次

优化枚举：实例分析

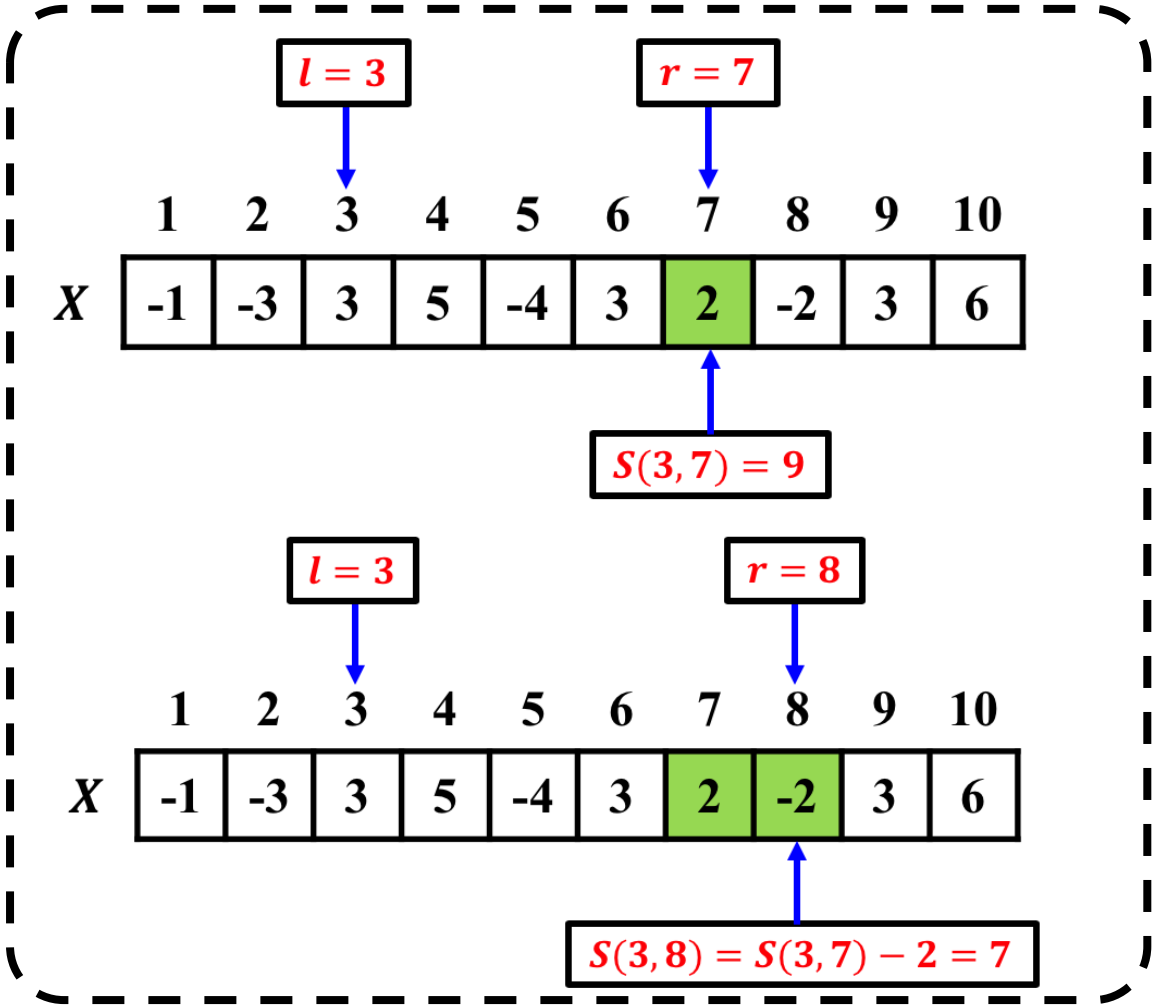


计算$S(3,8)$操作1次

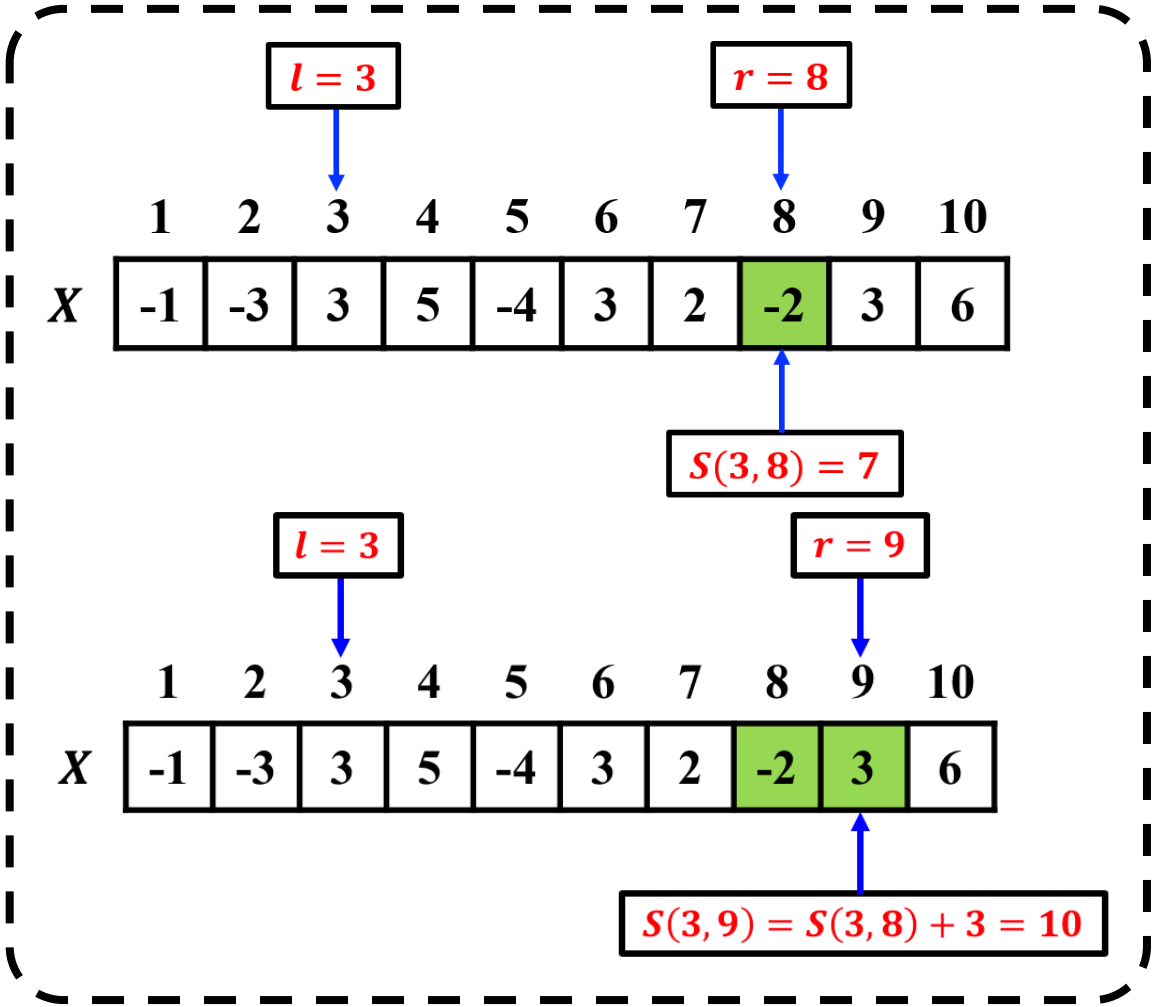


计算$S(3,9)$操作1次

优化枚举：实例分析



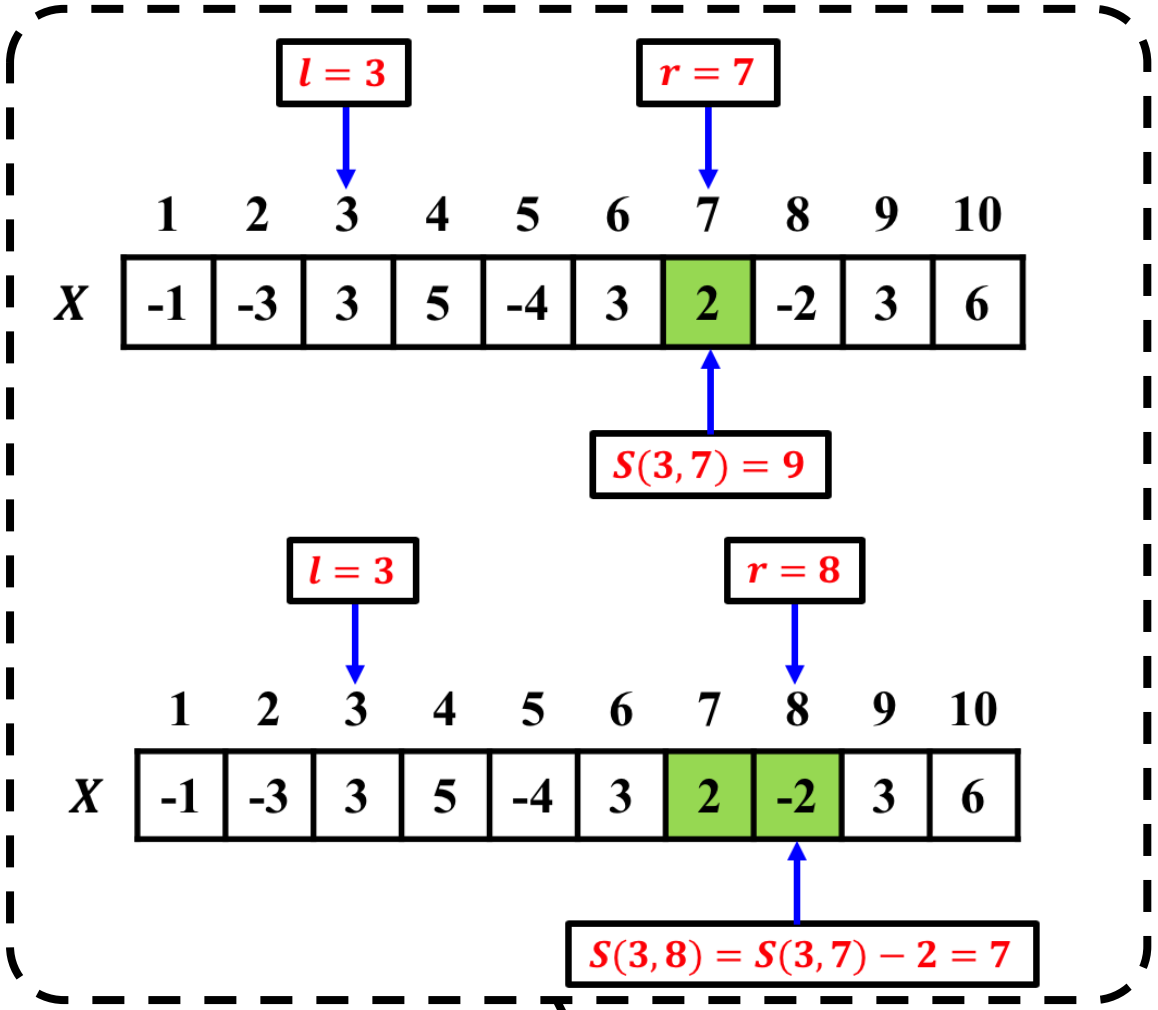
计算$S(3,8)$操作1次

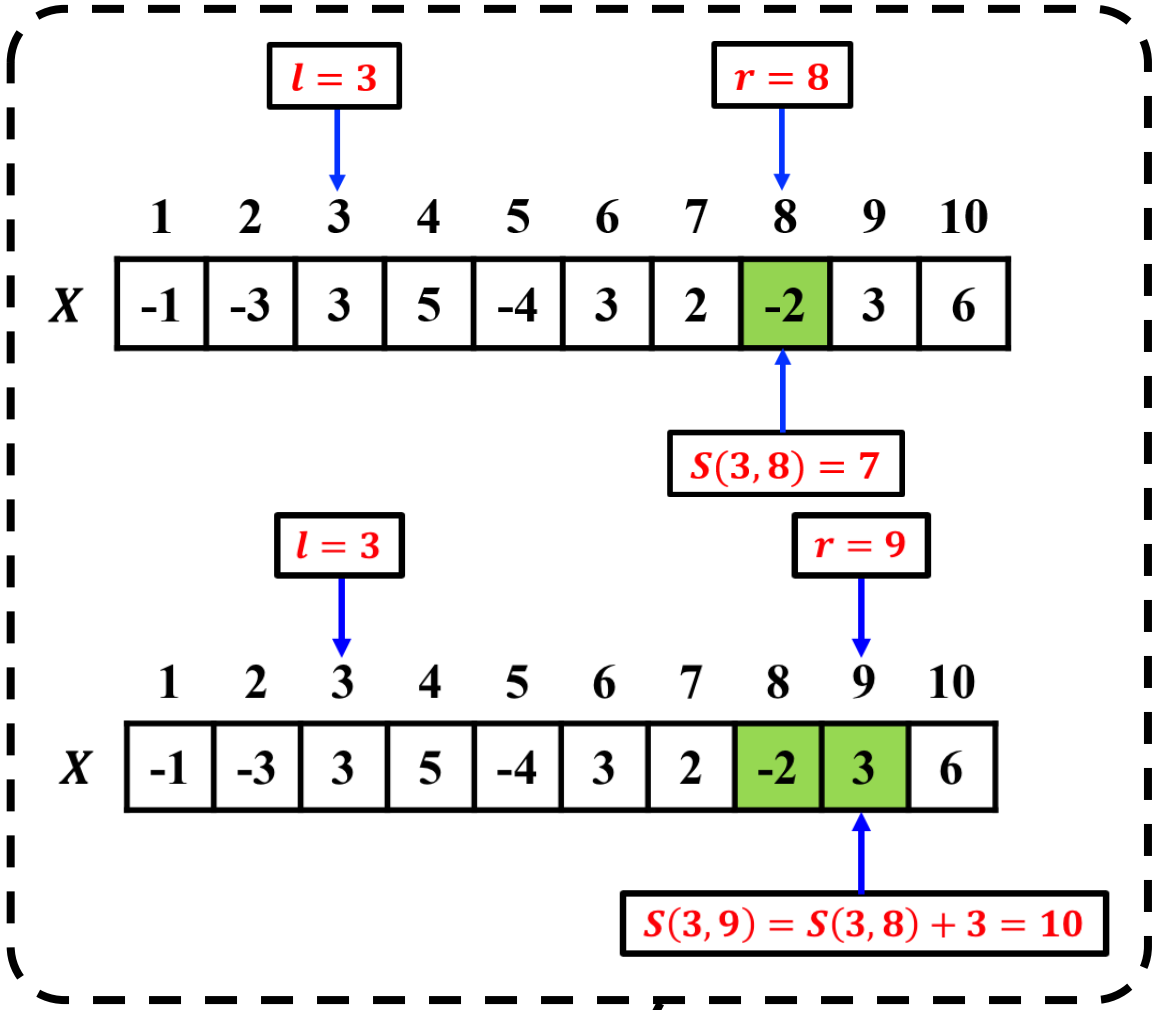


计算$S(3,9)$操作1次

重复利用之前已经计算的数据

优化枚举：实例分析





$$S(l,r) = \sum_{i=l}^{r} X[i] = S(l,r-1) + X[r]$$

优化枚举：伪代码

- 核心思想: $S(l,r)=\sum_{i=l}^{r}X[i]=\ S(l,r-1)+X[r]$

输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    S ← 0
    for r ← l to n do
        S ← S + X[r]
        S_max ← max{S_max, S}
    end
end
return S_max
```

迭代求解S

优化枚举：伪代码

- 核心思想: $S(l,r)=\sum_{i=l}^{r}X[i]=\ S(l,r-1)+X[r]$

输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    S ← 0
    for r ← l to n do
        S ← S + X[r]
        S_max ← max{S_max, S}
    end
end
return S_max
```

更新S_{max}

优化枚举：伪代码

- 核心思想: $S(l,r)=\sum_{i=l}^{r}X[i]=\ S(l,r-1)+X[r]$

输入: 数组$X[1..n]$
输出: 最大子数组之和S_{max}

```
S_max ← −∞
for l ← 1 to n do
    S ← 0
    for r ← l to n do
        S ← S + X[r]
        S_max ← max{S_max, S}
    end
end
return S_max
```

$O(n)$ $O(n^2)$

时间复杂度：$O(n^2)$

分而治之：一般步骤

分而治之：一般步骤

$X[1..n]$

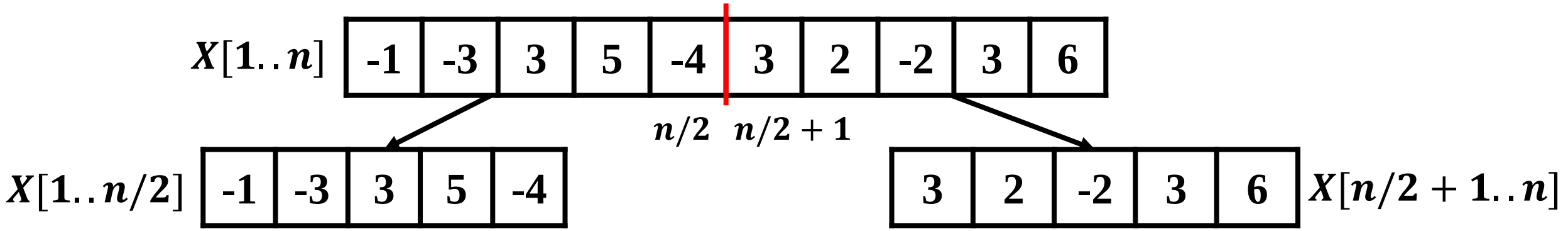
-1	-3	3	5	-4	3	2	-2	3	6

分解原问题

解决子问题

合并问题解

分而治之：一般步骤



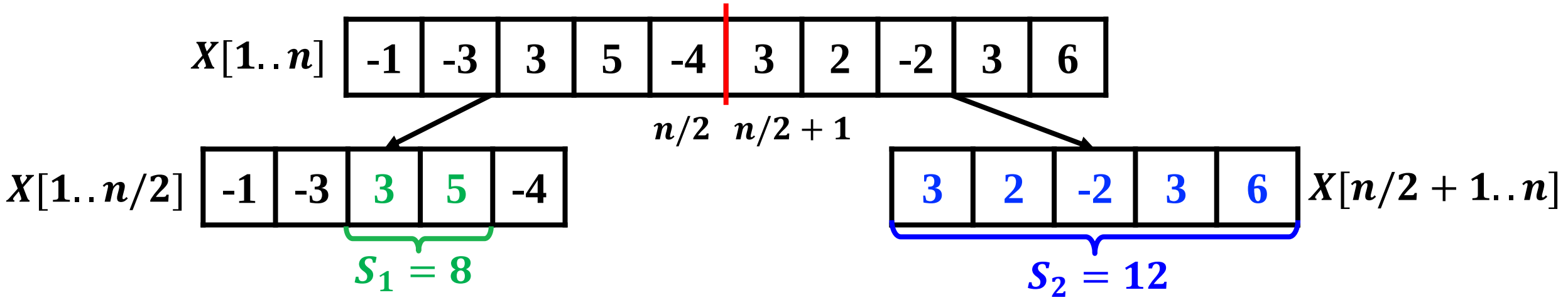
- 将数组$X[1..n]$分为$X[1..n/2]$和$X[n/2+1..n]$

分解原问题

解决子问题

合并问题解

分而治之：一般步骤



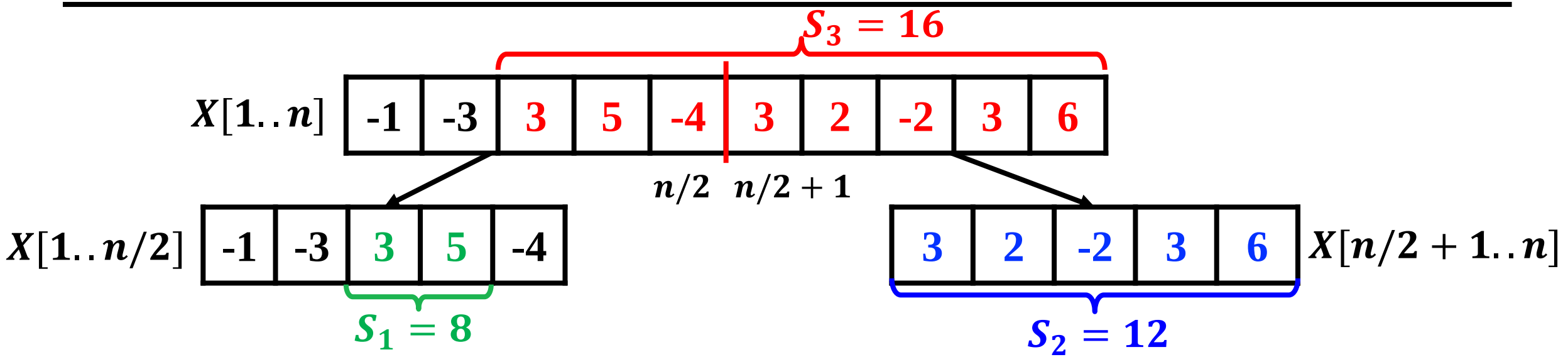
- 将数组$X[1..n]$分为$X[1..n/2]$和$X[n/2+1..n]$
- 递归求解子问题
  - S_1：数组$X[1..n/2]$ 的最大子数组
  - S_2：数组$X[n/2+1..n]$ 的最大子数组

分解原问题

解决子问题

合并问题解

分而治之：一般步骤



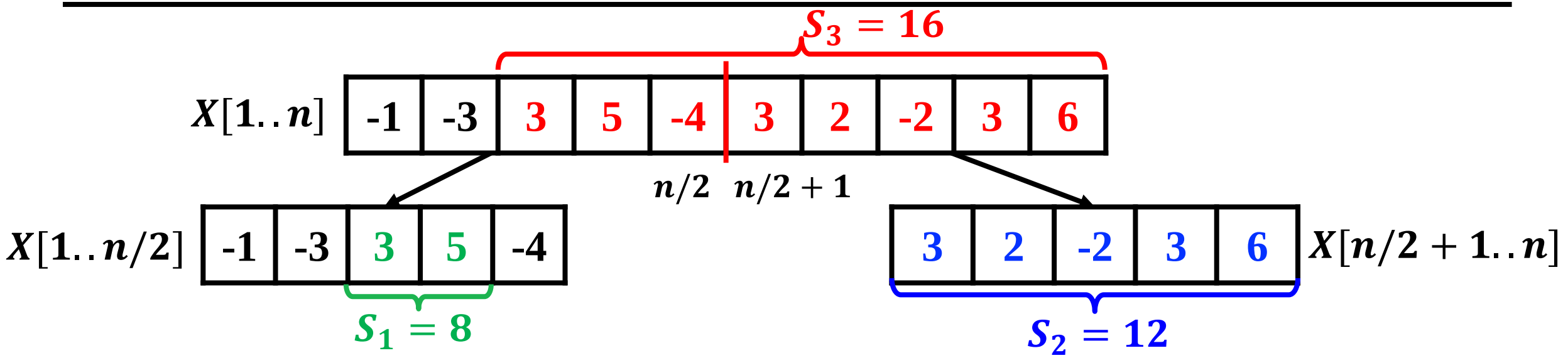
- 将数组$X[1..n]$分为$X[1..n/2]$和$X[n/2+1..n]$
- 递归求解子问题
  - S_1：数组$X[1..n/2]$ 的最大子数组
  - S_2：数组$X[n/2+1..n]$ 的最大子数组
- 合并子问题，得到S_{max}
  - S_3：跨中点的最大子数组

分解原问题

解决子问题

合并问题解

分而治之：一般步骤



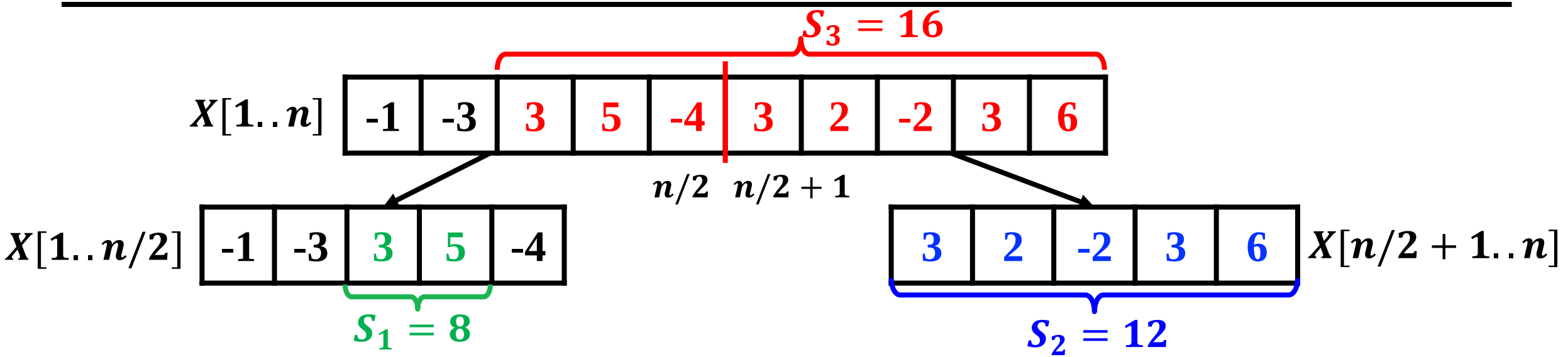
- 将数组$X[1..n]$分为$X[1..n/2]$和$X[n/2+1..n]$
- 递归求解子问题
  - S_1：数组$X[1..n/2]$ 的最大子数组
  - S_2：数组$X[n/2+1..n]$ 的最大子数组
- 合并子问题，得到S_{max}
  - S_3：跨中点的最大子数组
  - 数组X的最大子数组之和$S_{max} = max\{S_1, S_2, S_3\}$

分解原问题

解决子问题

合并问题解

分而治之：一般步骤



- 将数组$X[1..n]$分为$X[1..n/2]$和$X[n/2+1..n]$
- 递归求解子问题
  - S_1：数组$X[1..n/2]$ 的最大子数组
  - S_2：数组$X[n/2+1..n]$ 的最大子数组
- 合并子问题，得到S_{max}
  - S_3：跨中点的最大子数组
  - 数组X的最大子数组之和$S_{max} = max\{S_1, S_2, S_3\}$

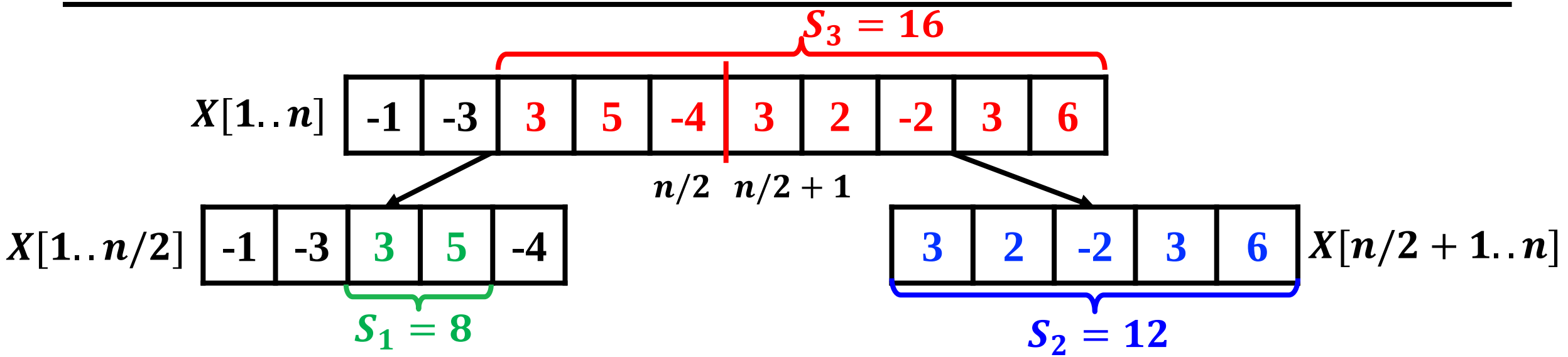
分解原问题

可递归求解

解决子问题

合并问题解

分而治之：一般步骤



- 将数组$X[1..n]$分为$X[1..n/2]$和$X[n/2+1..n]$
- 递归求解子问题
  - S_1：数组$X[1..n/2]$ 的最大子数组
  - S_2：数组$X[n/2+1..n]$ 的最大子数组
- 合并子问题，得到S_{max}
  - S_3：跨中点的最大子数组
  - 数组X的最大子数组之和$S_{max} = max\{S_1, S_2, S_3\}$

分解原问题

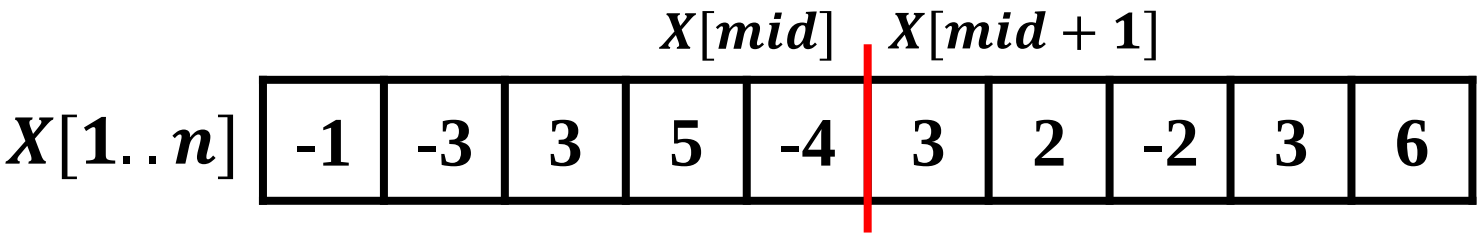
可递归求解

解决子问题

问题：如何求解S_3？

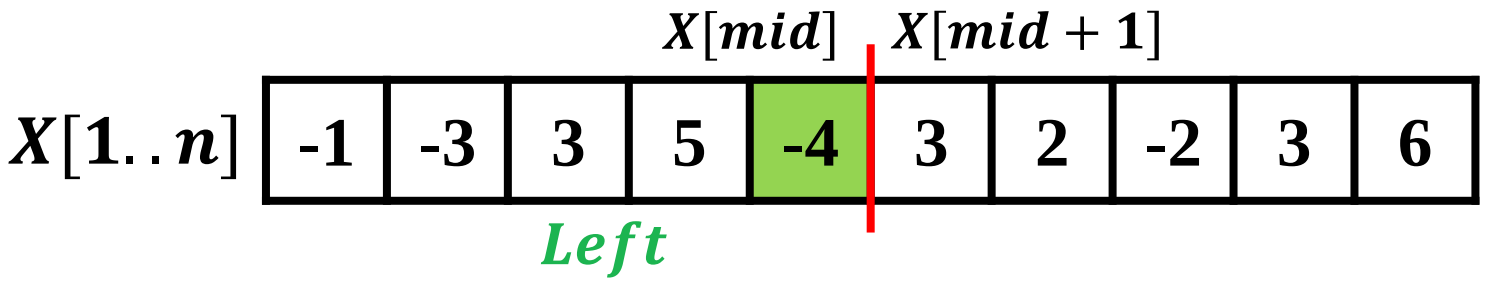
合并问题解

合并问题解：求解S_3



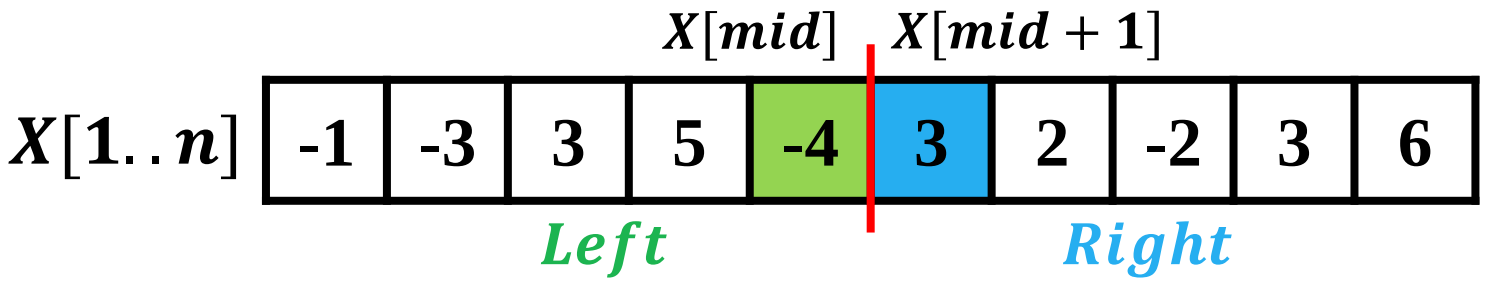
- 记$mid = n/2$

合并问题解：求解S_3



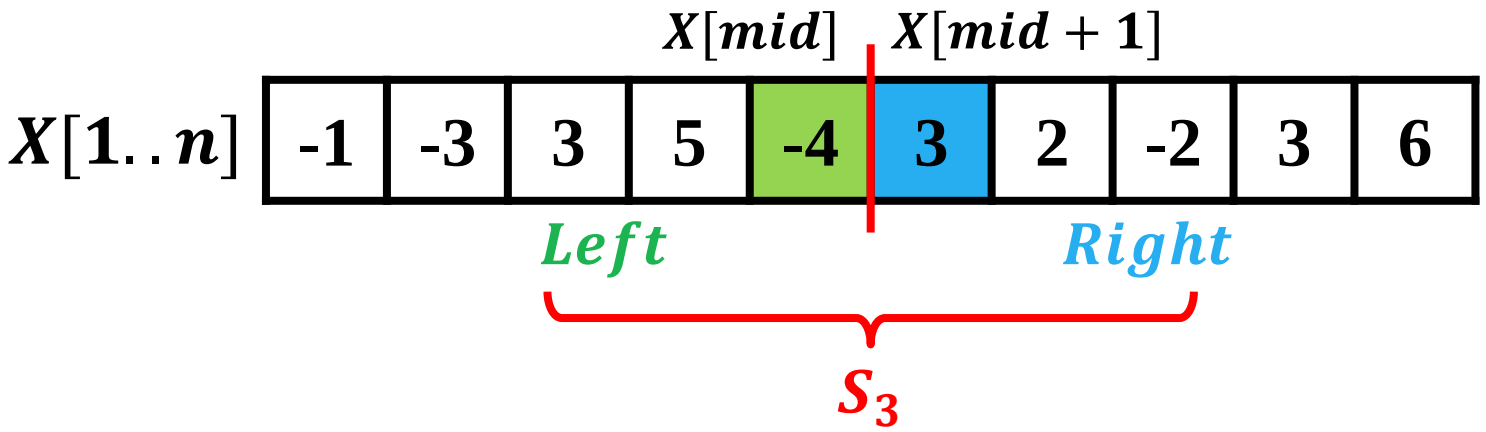
- 记$mid = n/2$
- S_3可分为左右两部分
  - $Left$：以$X[mid]$为结尾的最大子数组之和

合并问题解：求解S_3



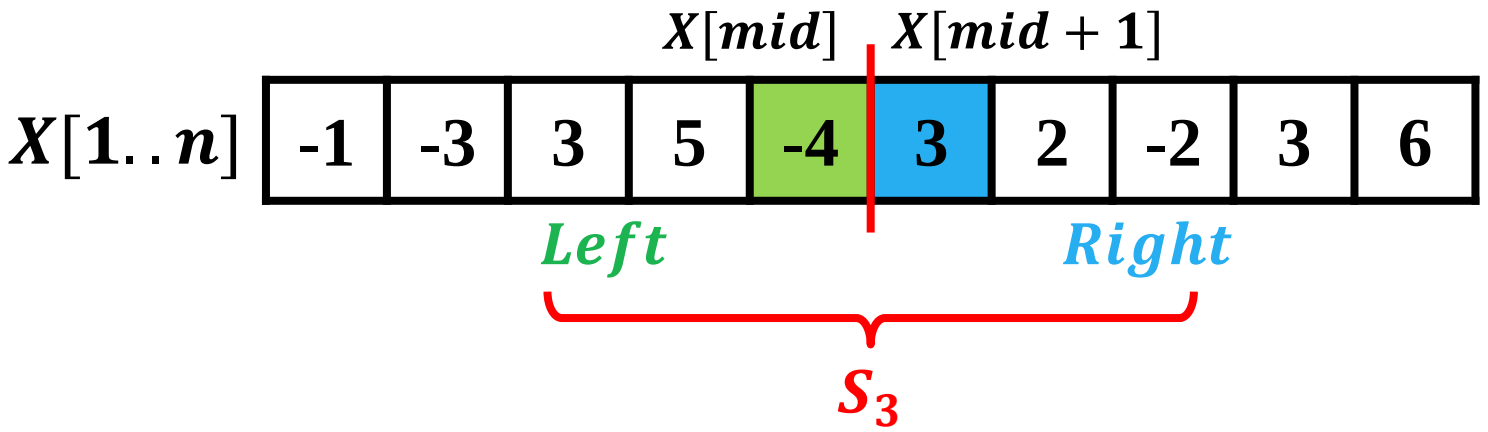
- 记$mid = n/2$
- S_3可分为左右两部分
  - $Left$：以$X[mid]$为结尾的最大子数组之和
  - $Right$：以$X[mid+1]$为开头的最大子数组之和

合并问题解：求解S_3



- 记$mid = n/2$
- S_3可分为左右两部分，$S_3 = Left + Right$
  - $Left$：以$X[mid]$为结尾的最大子数组之和
  - $Right$：以$X[mid+1]$为开头的最大子数组之和

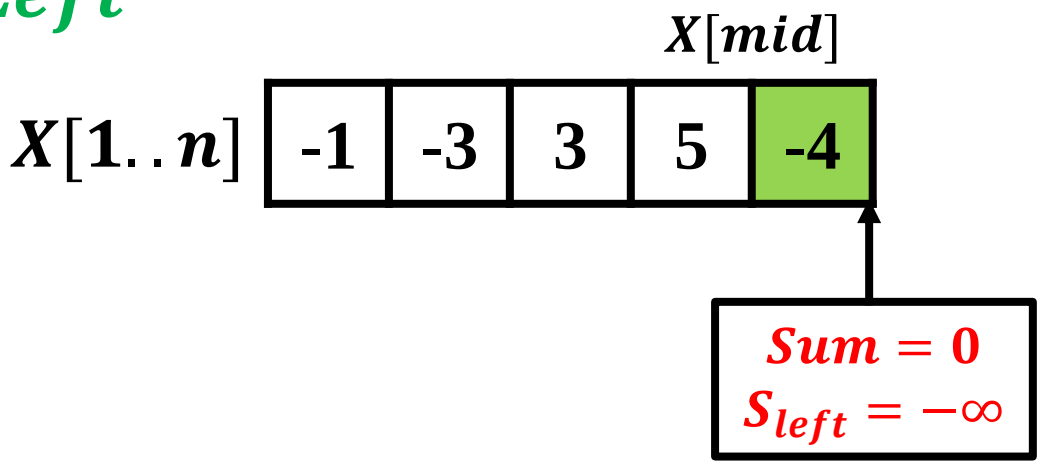
合并问题解：求解S_3



- 记$mid = n/2$
- S_3可分为左右两部分，$S_3 = Left + Right$
  - $Left$：以$X[mid]$为结尾的最大子数组之和
  - $Right$：以$X[mid + 1]$为开头的最大子数组之和
  - 分别求出$Left$和$Right$，便可求出S_3

合并问题解：求解S_3

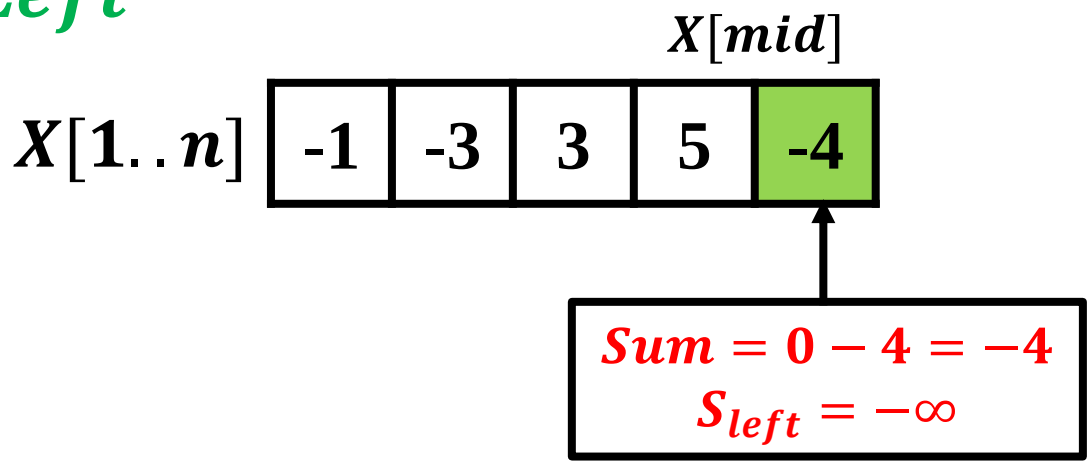
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

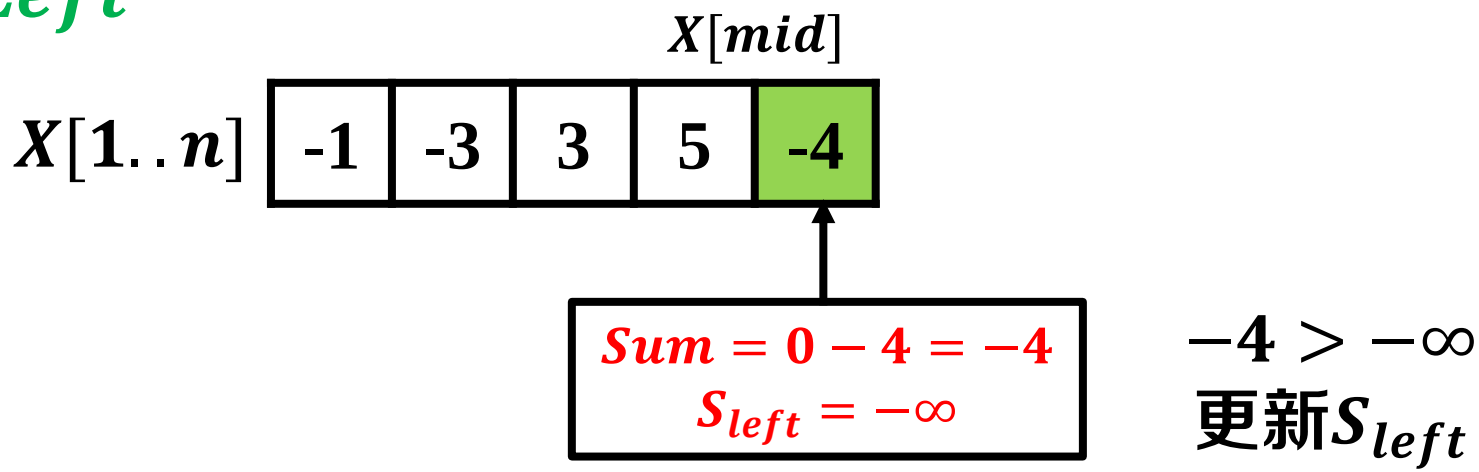
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

- 求解$Left$



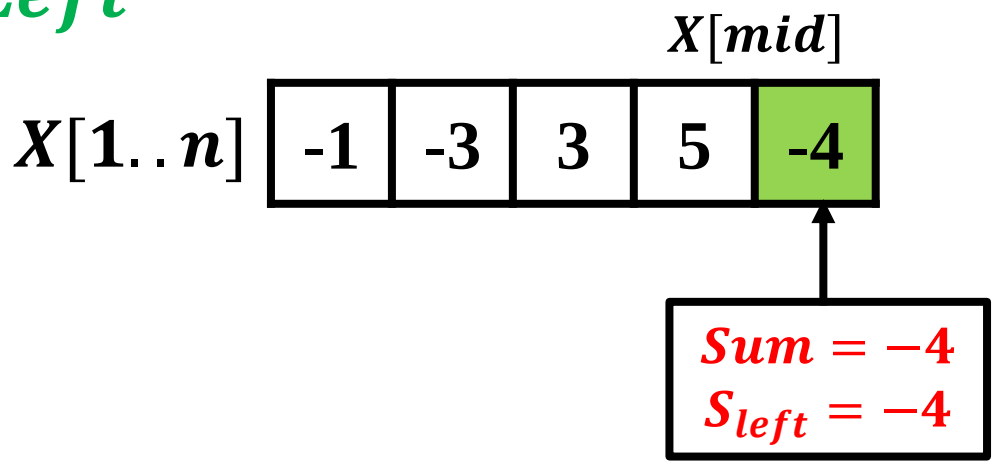
$-4 > -\infty$

更新S_{left}

- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

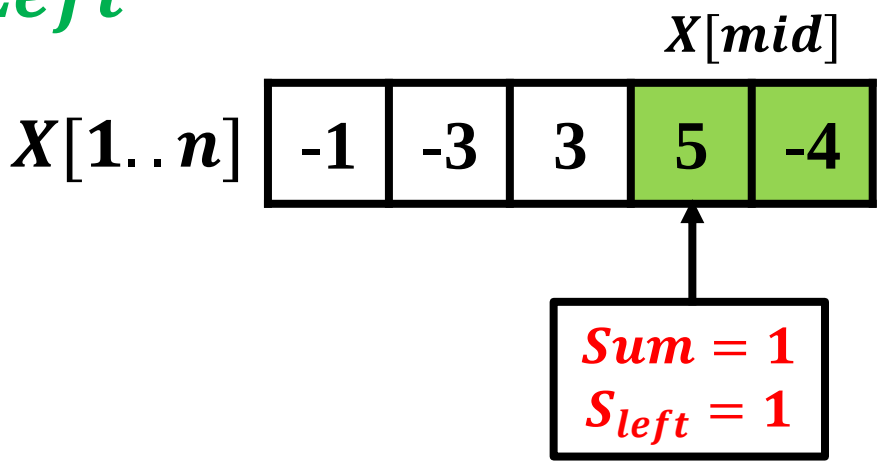
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

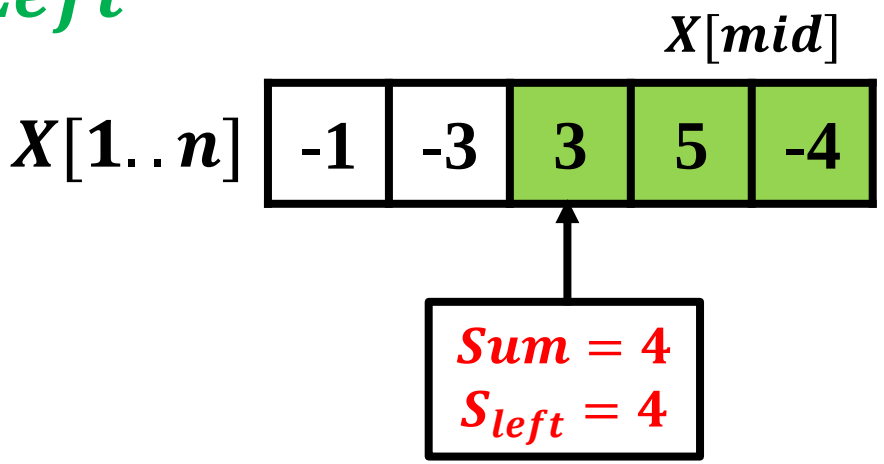
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

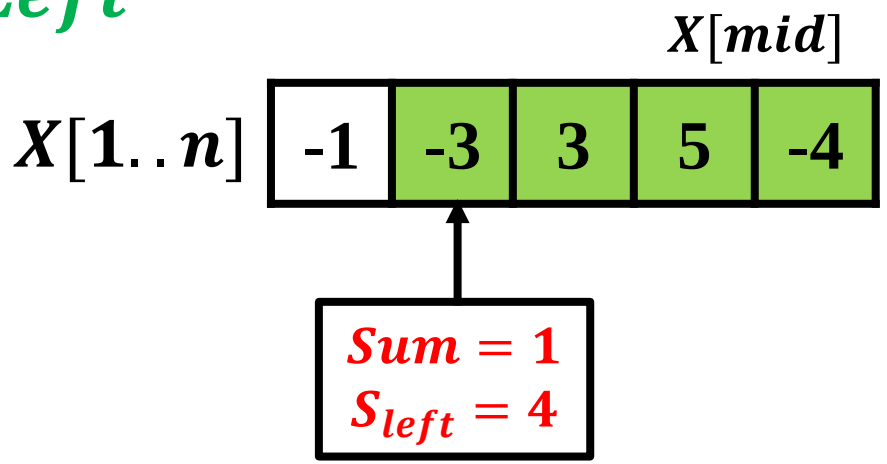
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

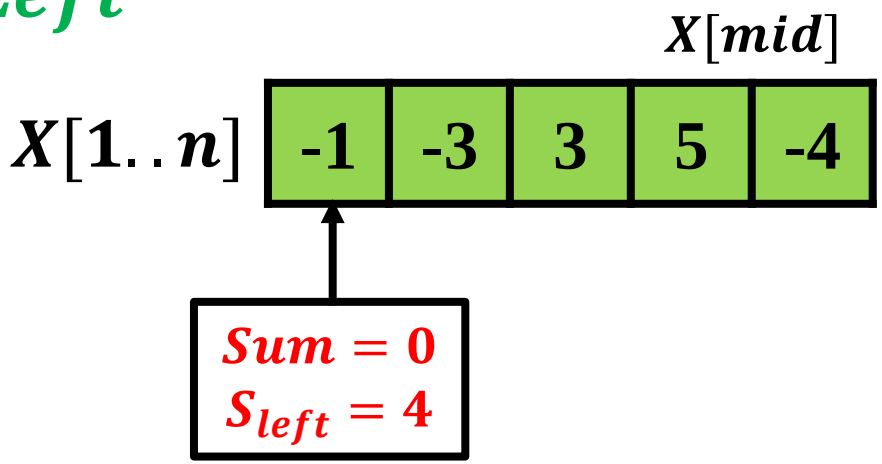
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

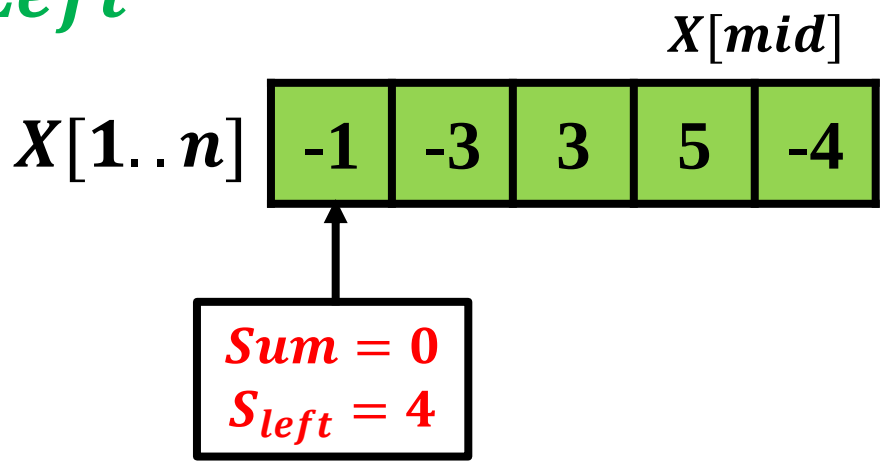
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值

合并问题解：求解S_3

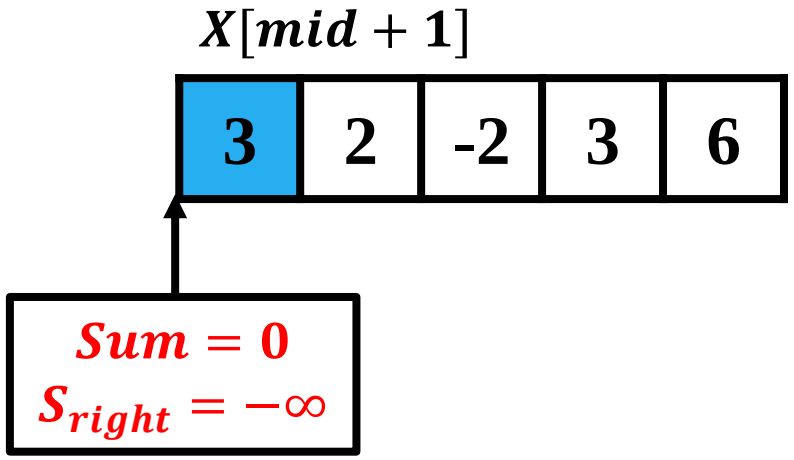
- 求解$Left$



- 记$mid = n/2$
- 从$X[mid]$向前遍历求和，并记录最大值
- 以$X[mid]$为结尾的最大子数组之和$S_{left} = 4$

合并问题解：求解S_3

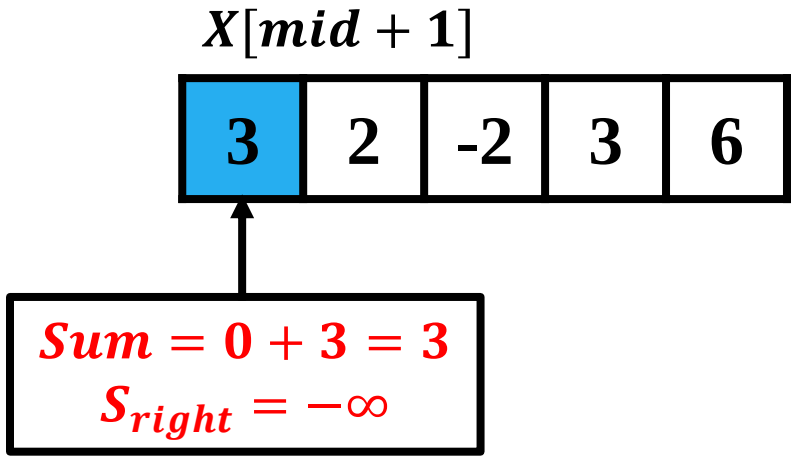
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

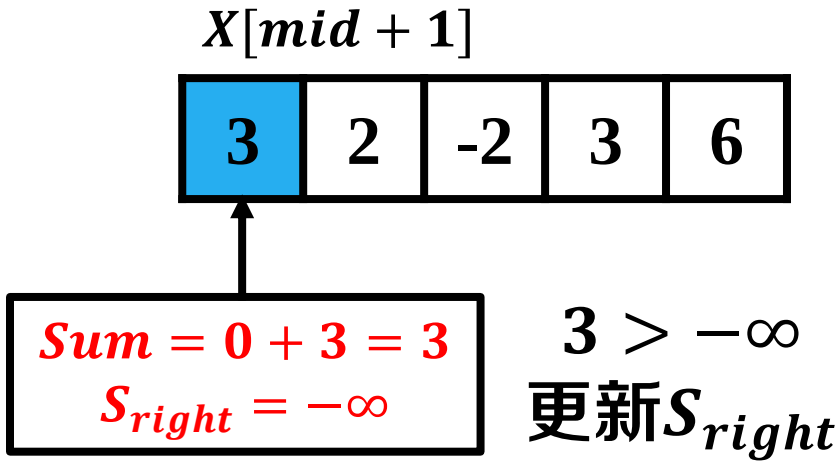
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

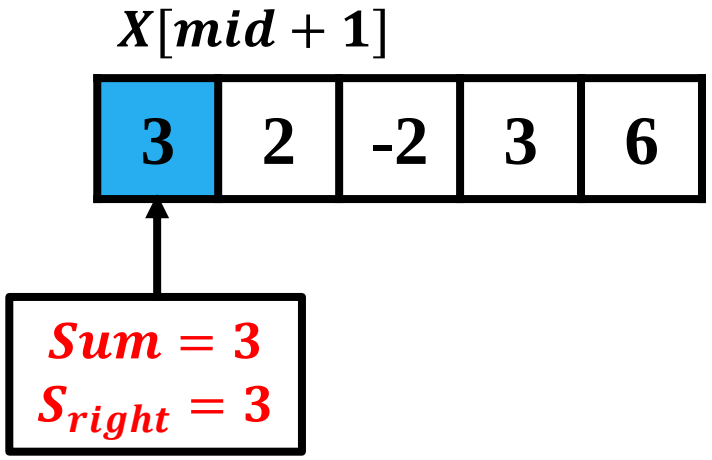
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

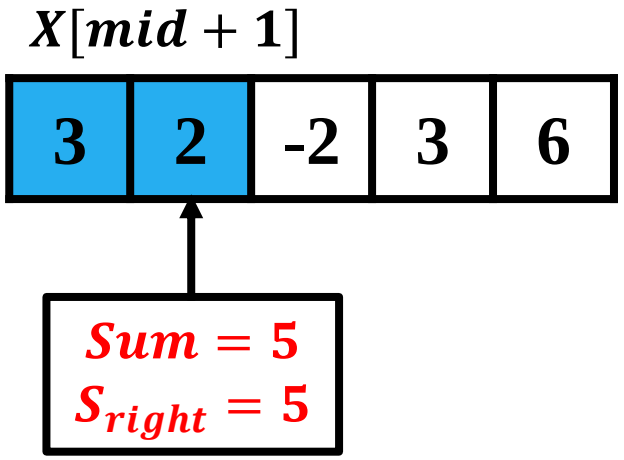
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

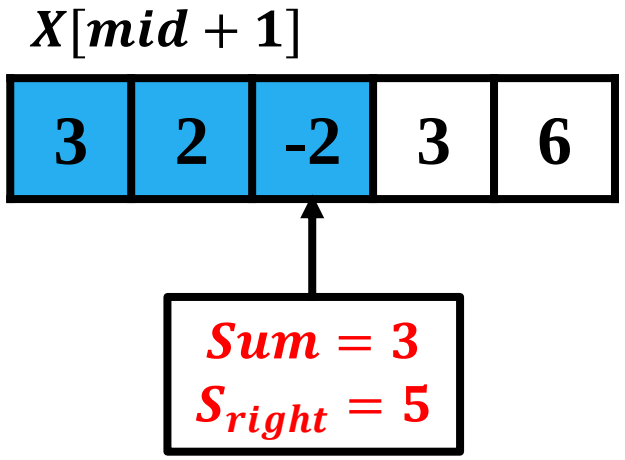
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid + 1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

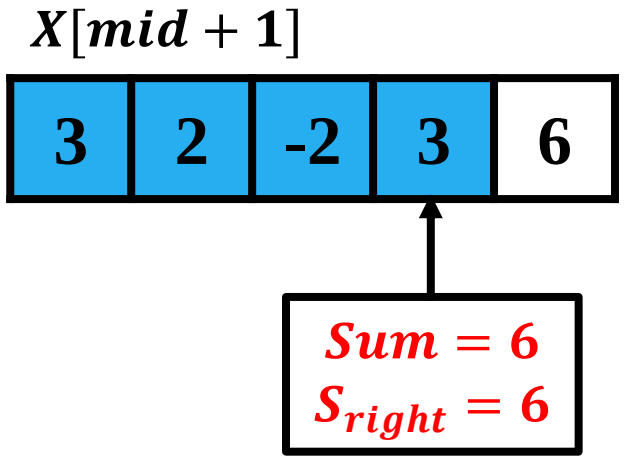
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid + 1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

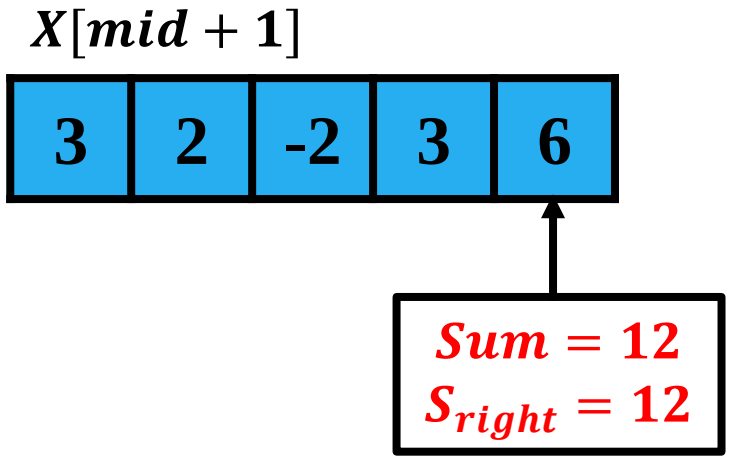
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值

合并问题解：求解S_3

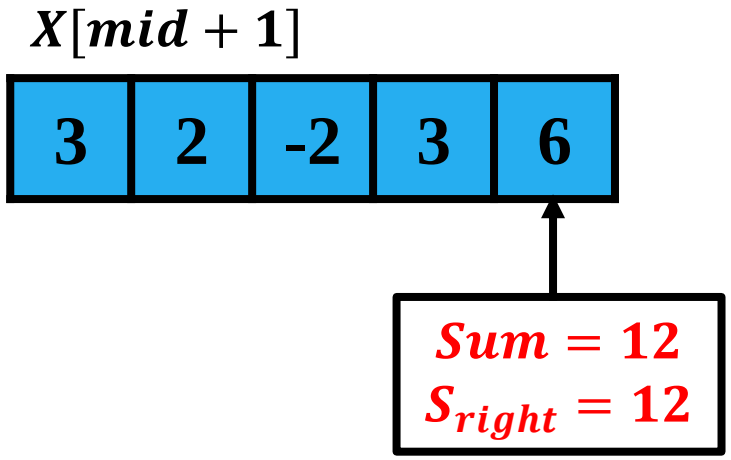
- 求解$Right$



- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值

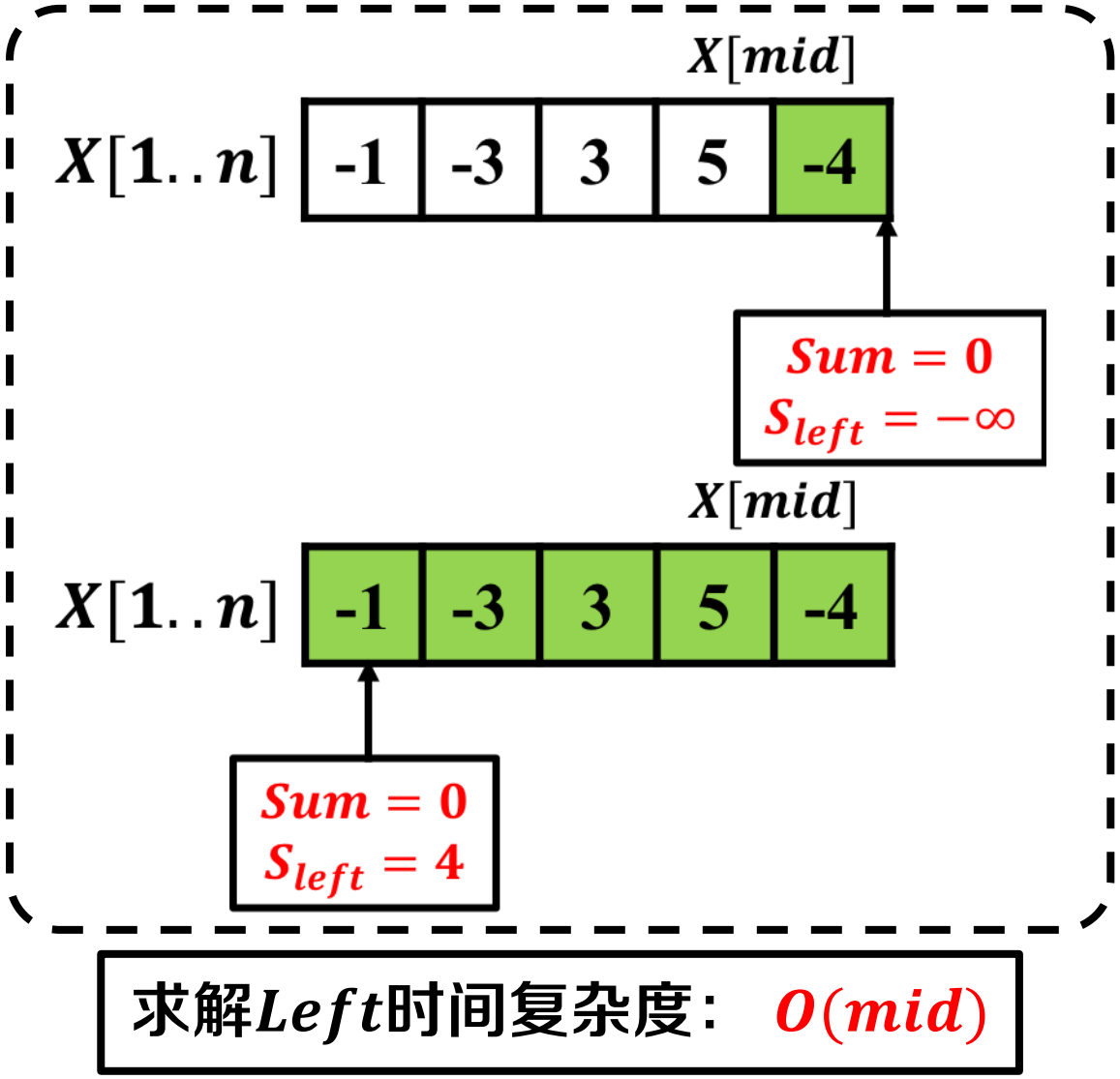
合并问题解：求解S_3

- 求解$Right$



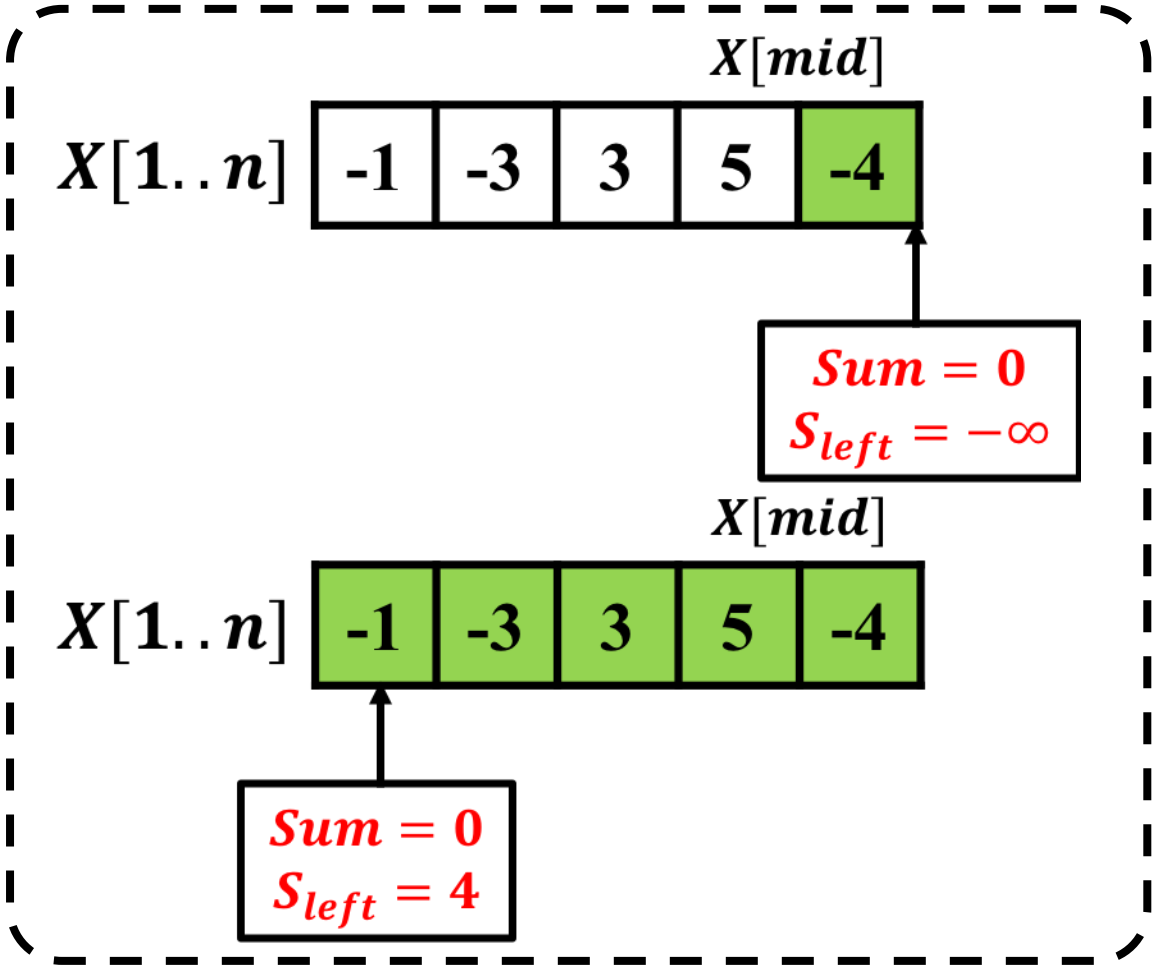
- 记$mid = n/2$
- 从$X[mid+1]$向后遍历求和，并记录最大值
- 以$X[mid+1]$为开头的最大子数组之和$S_{right} = 12$

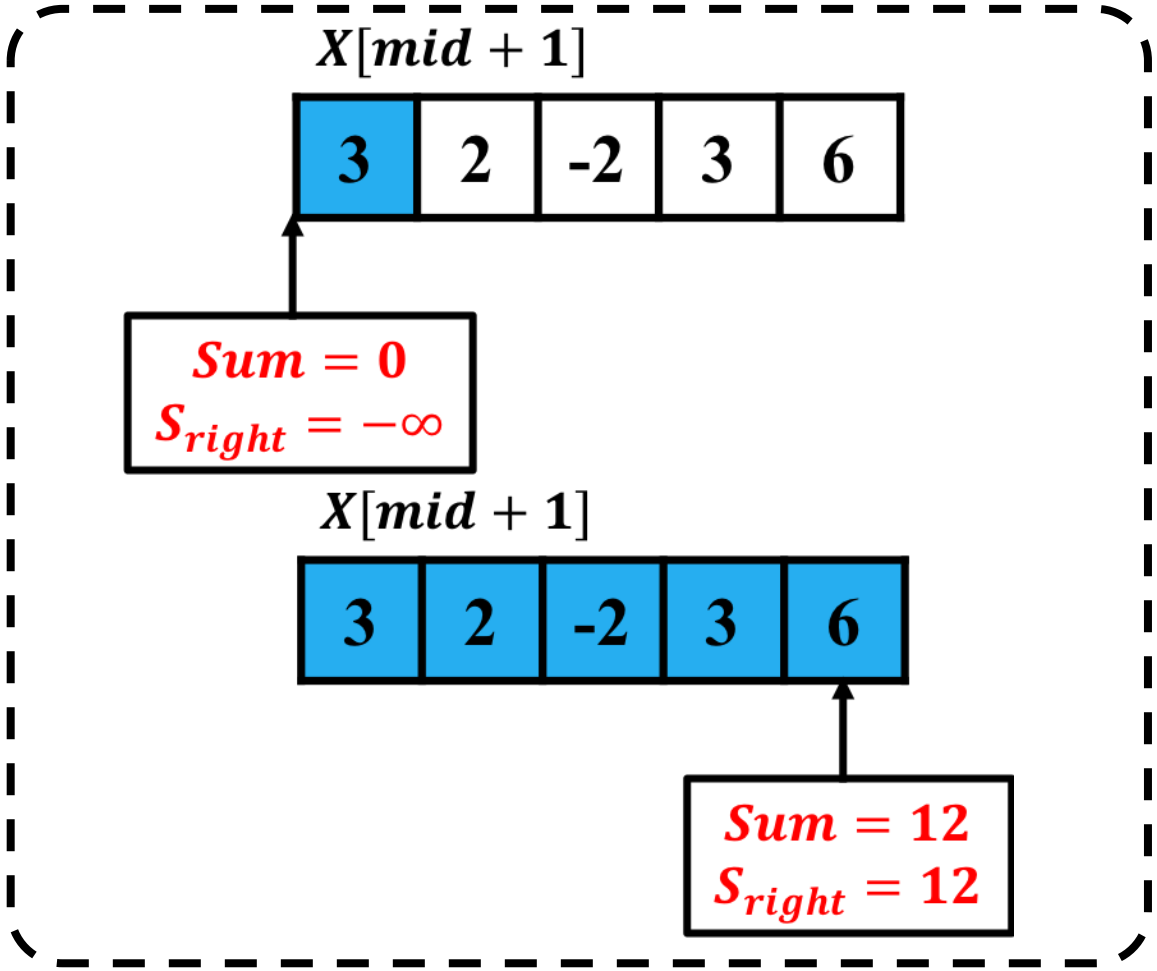
求解S_3：时间复杂度



求解$Left$时间复杂度：$O(mid)$

求解S_3：时间复杂度

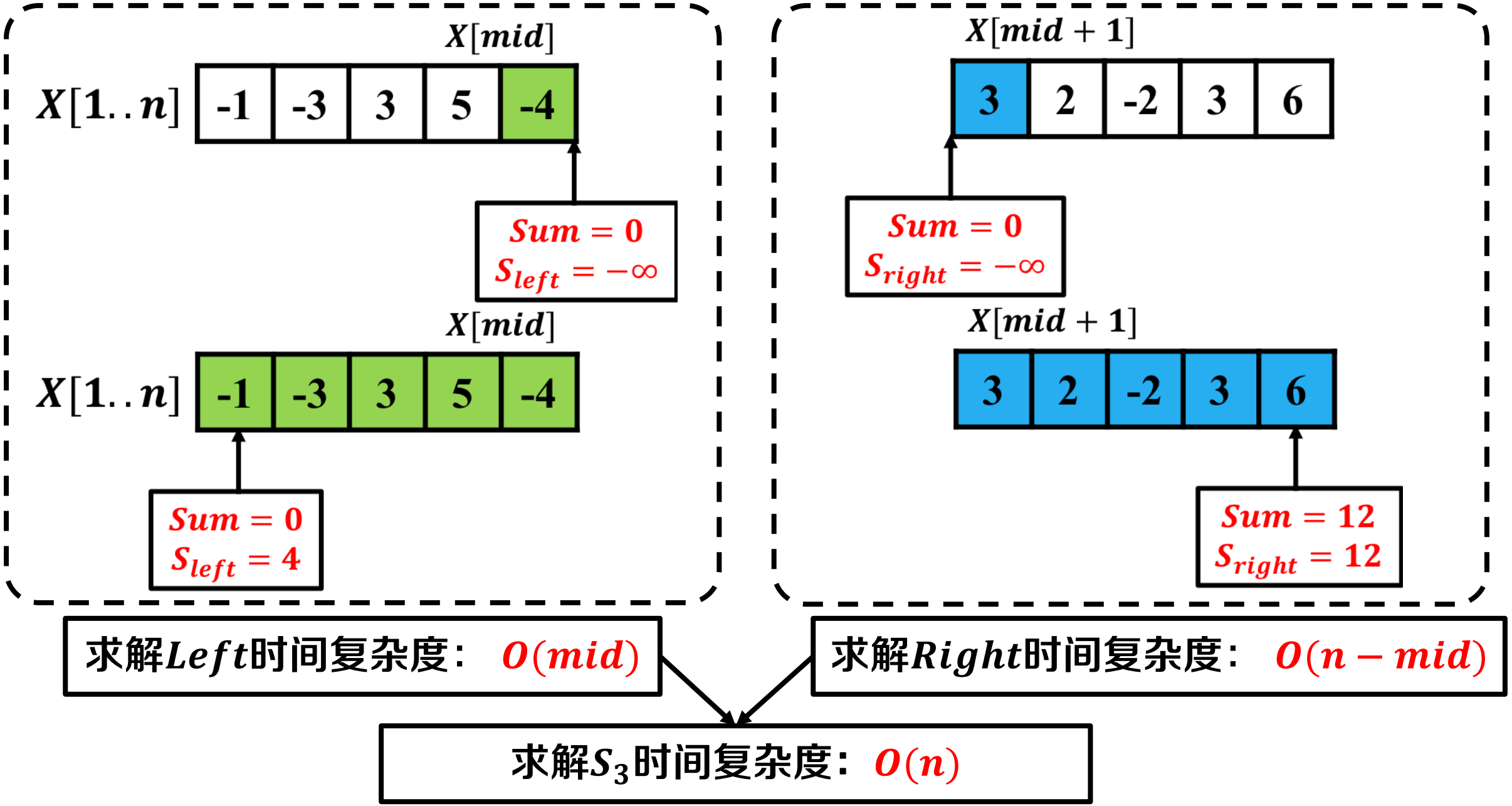




求解$Left$时间复杂度：$O(mid)$

求解$Right$时间复杂度：$O(n-mid)$

求解S_3：时间复杂度


X[mid]
X[1..n]
-1
-3
3
5
-4
Sum = 0
S_left = −∞
X[mid]
X[1..n]
-1
-3
3
5
-4
Sum = 0
S_left = 4
X[mid + 1]
3
2
-2
3
6
Sum = 0
S_right = −∞
X[mid + 1]
3
2
-2
3
6
Sum = 12
S_right = 12
求解Left时间复杂度： O(mid)
求解Right时间复杂度： O(n − mid)
求解S_3时间复杂度： O(n)

求解S_3：伪代码

- **CrossingSubArray**$(X, low, mid, high)$

求解S_3：伪代码

- **CrossingSubArray($X, low, mid, high$)**

输入: 数组X, 数组下标$low, mid, high$
输出: 跨越中点的最大子数组之和S_3

$S_{left} \leftarrow -\infty$
$Sum \leftarrow 0$
for $l \leftarrow mid$ *downto* low **do**
  $Sum \leftarrow Sum + X[l]$
  $S_{left} \leftarrow \max\{S_{left}, Sum\}$
end
$S_{right} \leftarrow -\infty$
$Sum \leftarrow 0$
for $r \leftarrow mid + 1$ *to* $high$ **do**
  $Sum \leftarrow Sum + X[r]$
  $S_{right} \leftarrow \max\{S_{right}, Sum\}$
end
$S_3 \leftarrow S_{left} + S_{right}$
return S_3

求解$Left$

求解S_3：伪代码

- **CrossingSubArray($X, low, mid, high$)**

输入: 数组X, 数组下标$low, mid, high$
输出: 跨越中点的最大子数组之和S_3

$S_{left} \leftarrow -\infty$
$Sum \leftarrow 0$
for $l \leftarrow mid$ *downto* low **do**
  $Sum \leftarrow Sum + X[l]$
  $S_{left} \leftarrow \max\{S_{left}, Sum\}$
end
$S_{right} \leftarrow -\infty$
$Sum \leftarrow 0$
for $r \leftarrow mid + 1$ *to* $high$ **do**
  $Sum \leftarrow Sum + X[r]$
  $S_{right} \leftarrow \max\{S_{right}, Sum\}$
end
$S_3 \leftarrow S_{left} + S_{right}$
return S_3

求解***Right***

求解S_3：伪代码

- **CrossingSubArray($X, low, mid, high$)**

输入: 数组X, 数组下标$low, mid, high$
输出: 跨越中点的最大子数组之和S_3

$S_{left} \leftarrow -\infty$
$Sum \leftarrow 0$
for $l \leftarrow mid$ *downto* low **do**
  $Sum \leftarrow Sum + X[l]$
  $S_{left} \leftarrow \max\{S_{left}, Sum\}$
end
$S_{right} \leftarrow -\infty$
$Sum \leftarrow 0$
for $r \leftarrow mid + 1$ *to* $high$ **do**
  $Sum \leftarrow Sum + X[r]$
  $S_{right} \leftarrow \max\{S_{right}, Sum\}$
end
$S_3 \leftarrow S_{left} + S_{right}$
return S_3

求解S_3

求解S_3：伪代码

- **CrossingSubArray($X, low, mid, high$)**

输入: 数组X, 数组下标$low, mid, high$
输出: 跨越中点的最大子数组之和S_3

$S_{left} \leftarrow -\infty$
$Sum \leftarrow 0$
for $l \leftarrow mid$ *downto* low **do**
  $Sum \leftarrow Sum + X[l]$
  $S_{left} \leftarrow \max\{S_{left}, Sum\}$
end
$S_{right} \leftarrow -\infty$
$Sum \leftarrow 0$
for $r \leftarrow mid + 1$ *to* $high$ **do**
  $Sum \leftarrow Sum + X[r]$
  $S_{right} \leftarrow \max\{S_{right}, Sum\}$
end
$S_3 \leftarrow S_{left} + S_{right}$
return S_3

$O(mid)$

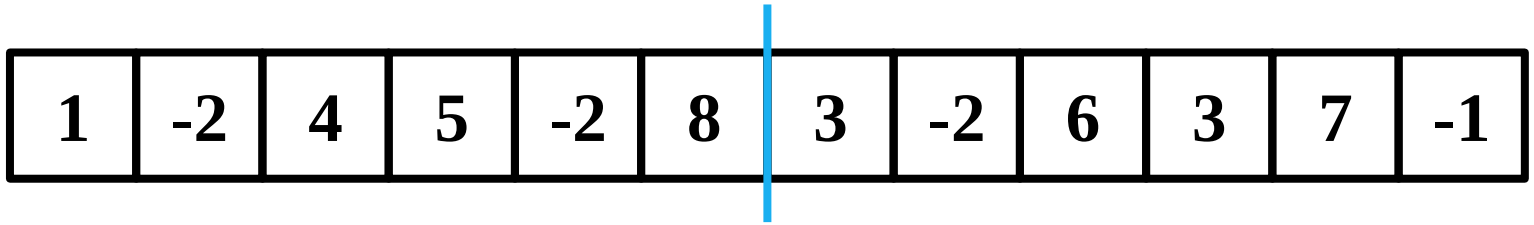
$O(n - mid)$

时间复杂度：$O(n)$

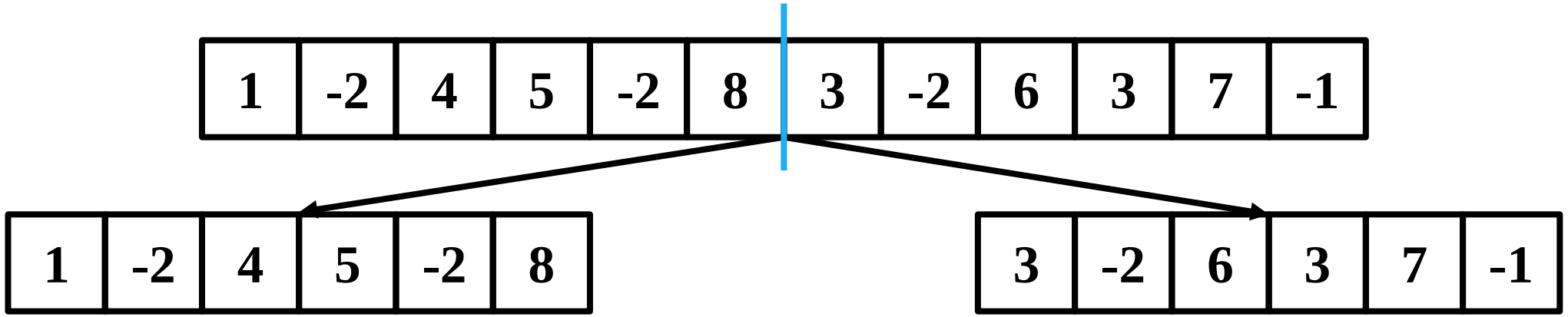
算法实例

1	-2	4	5	-2	8	3	-2	6	3	7	-1

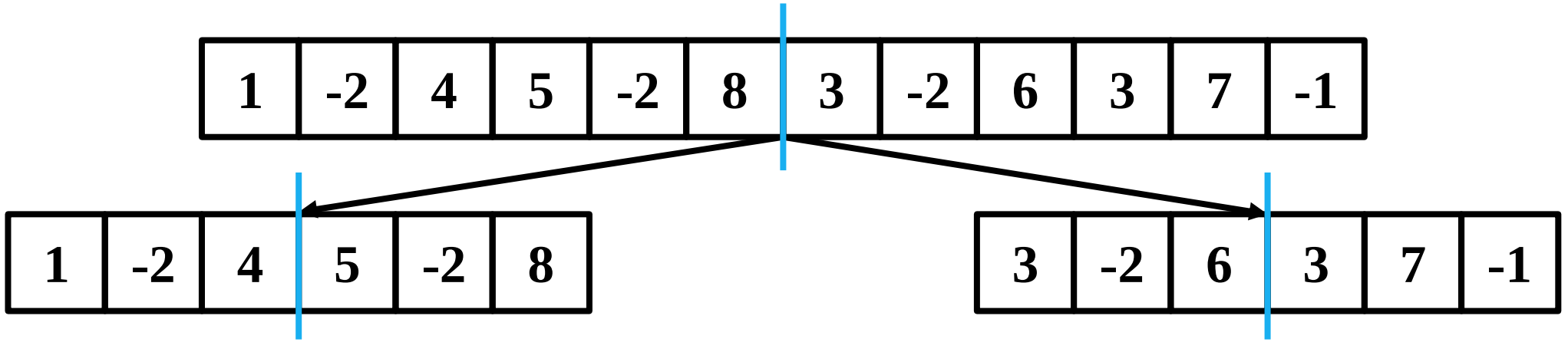
算法实例



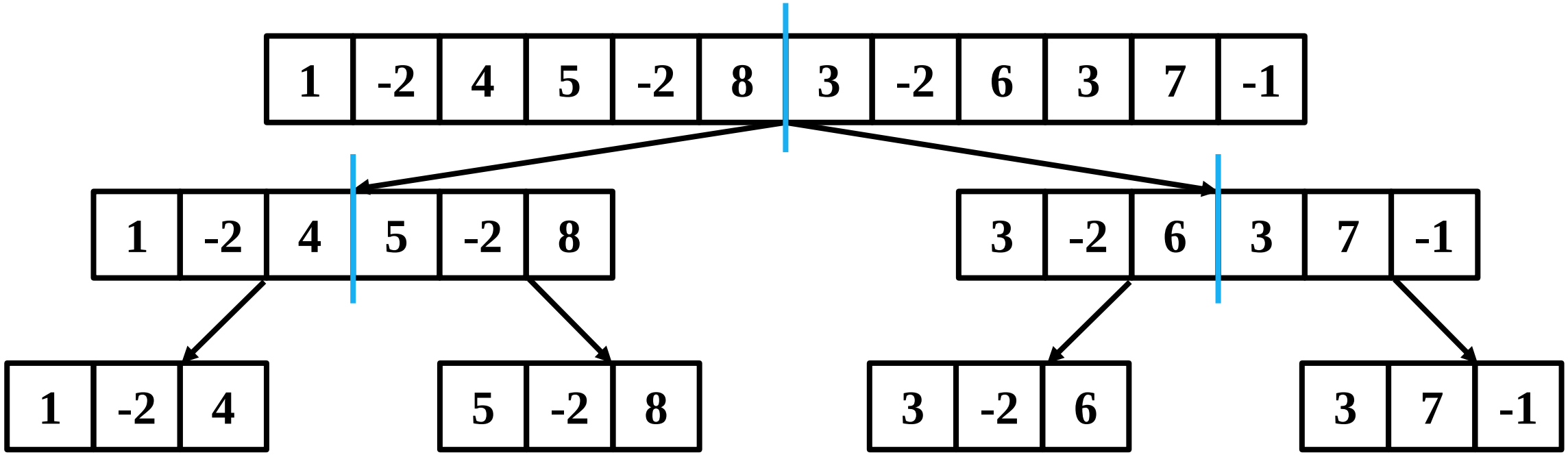
算法实例



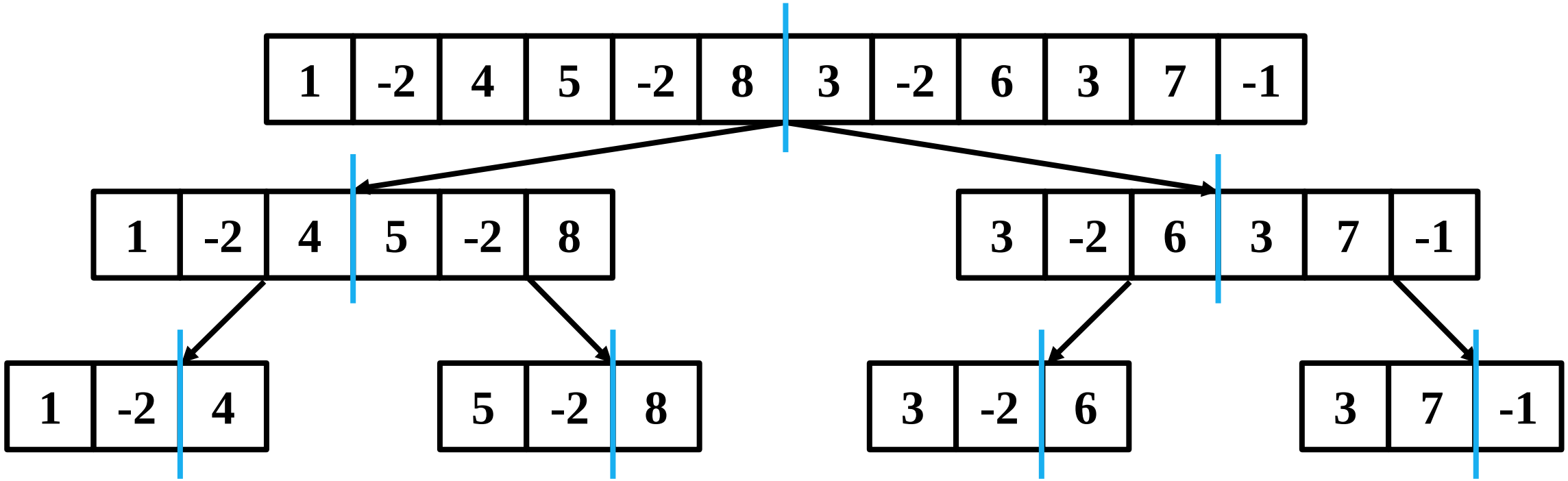
算法实例



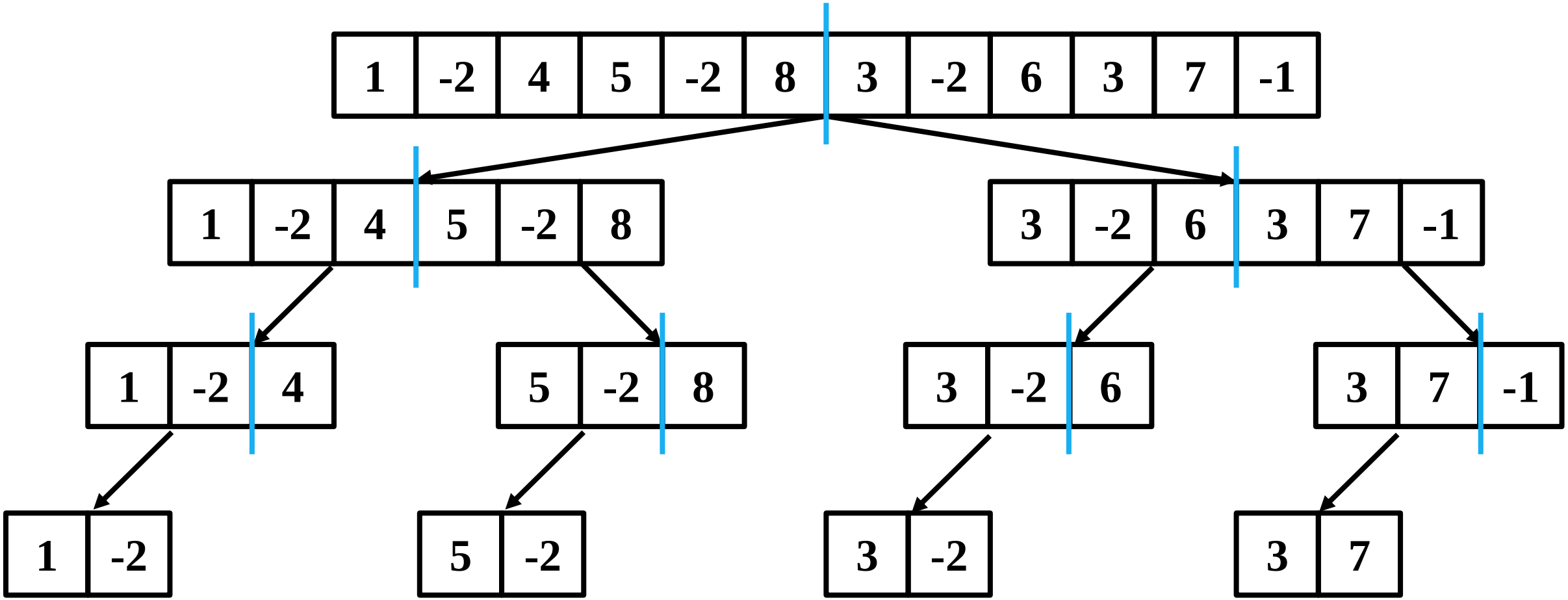
算法实例



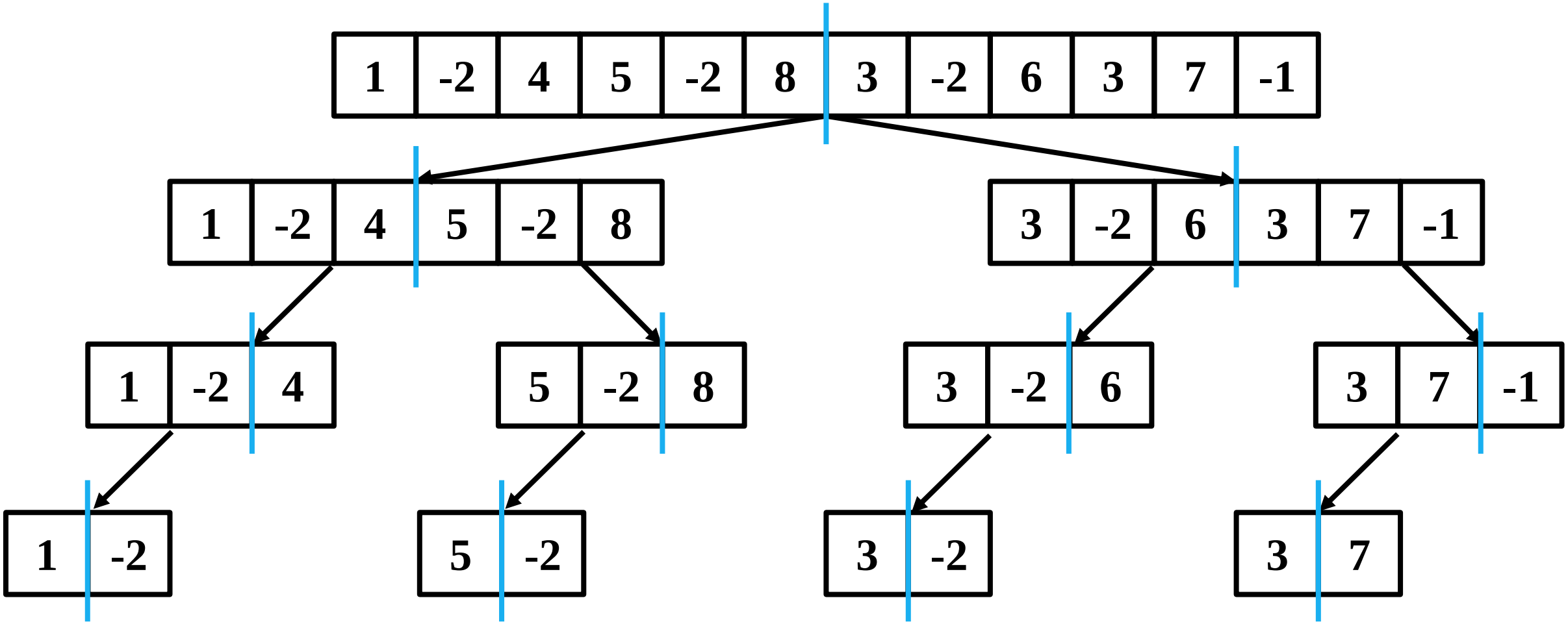
算法实例



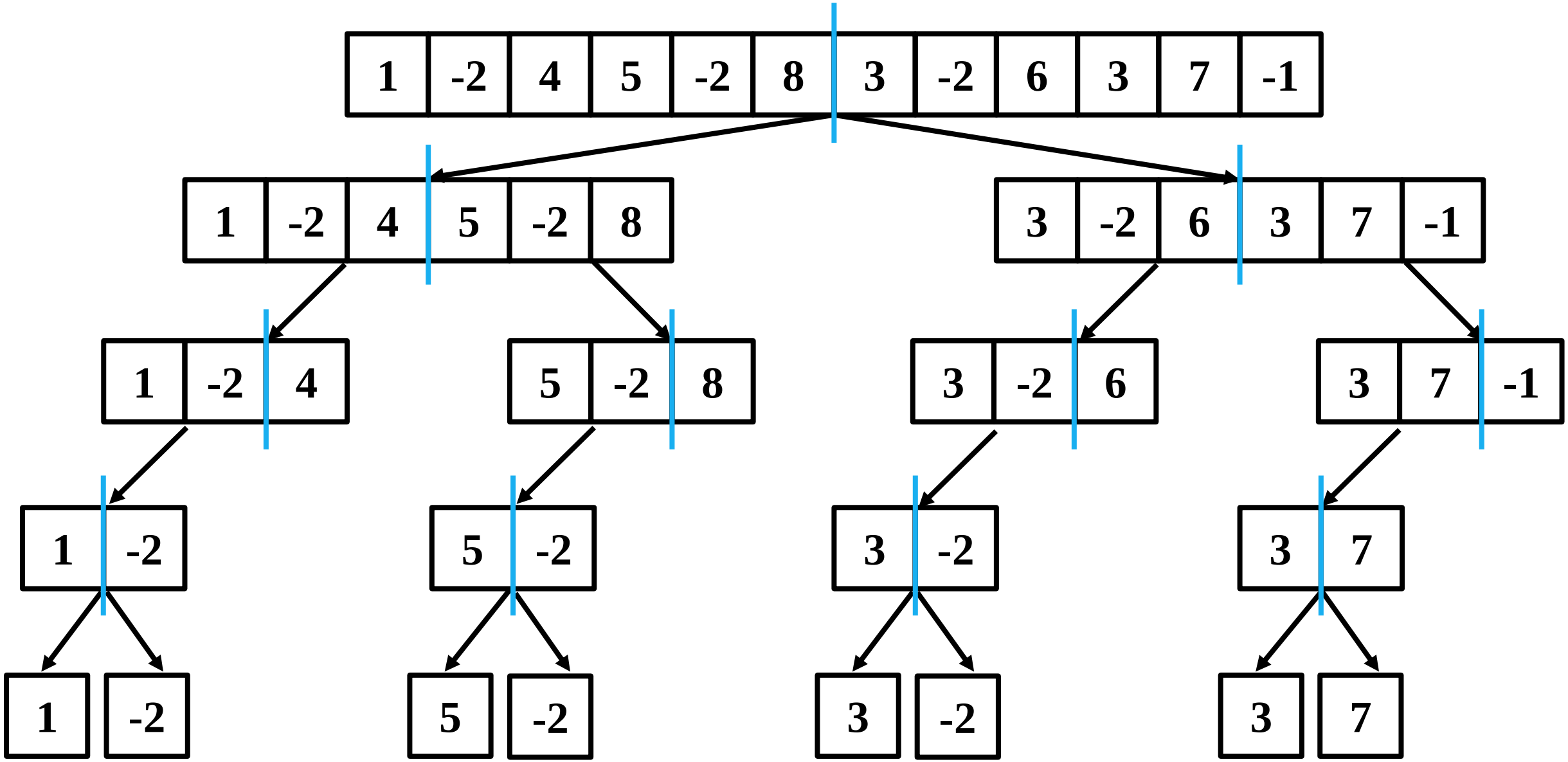
算法实例



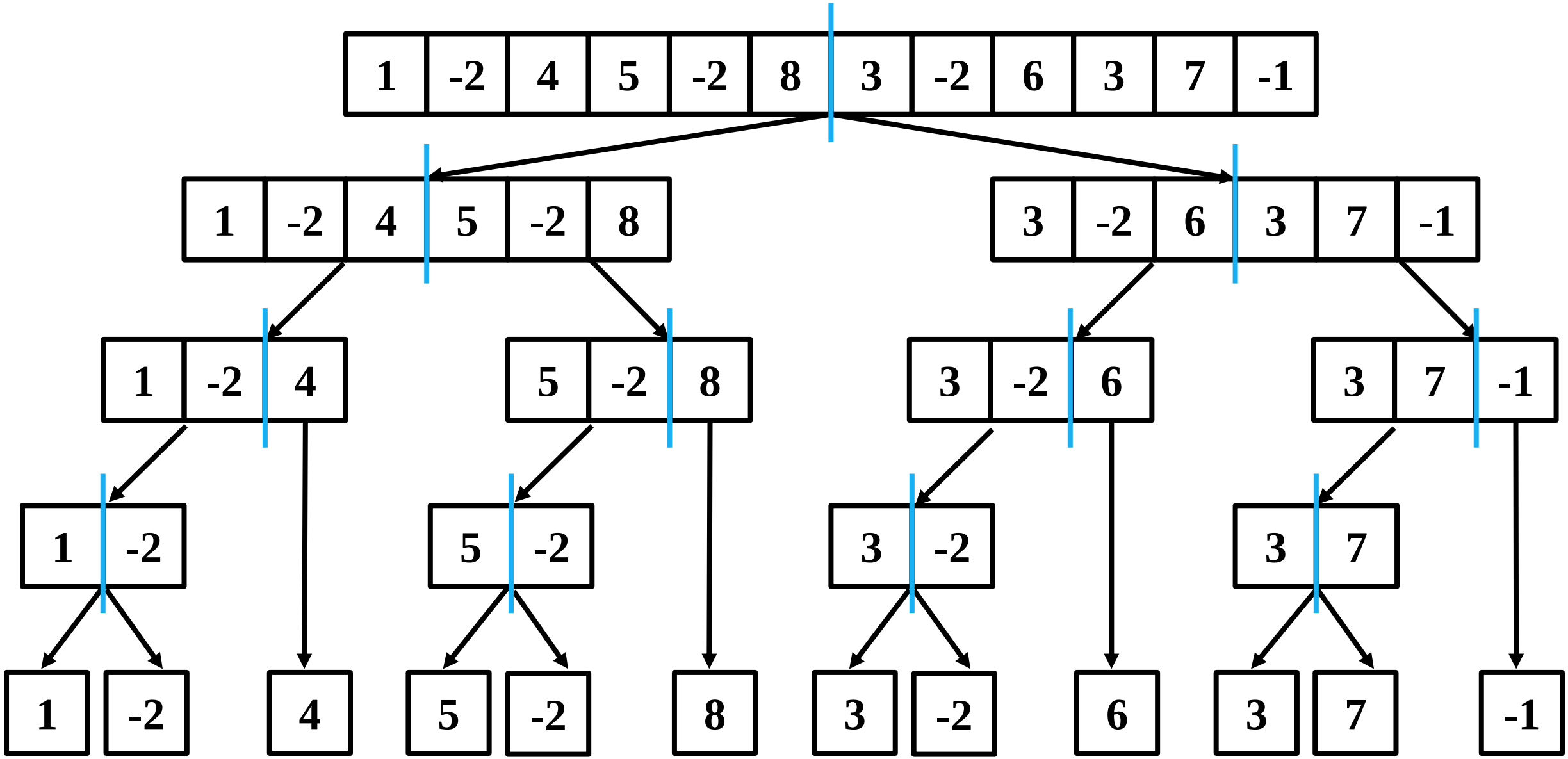
算法实例



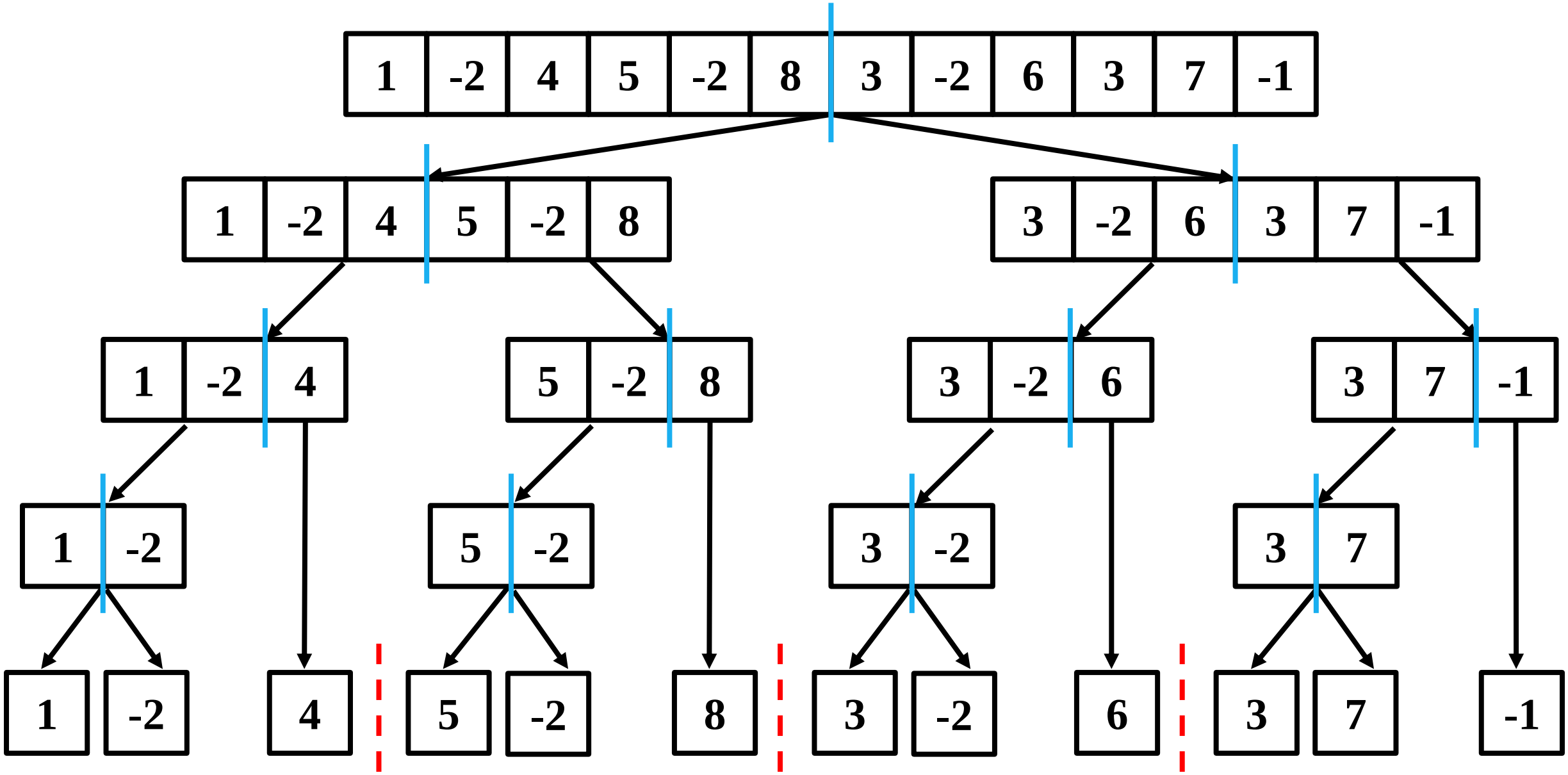
算法实例


1 -2 4 5 -2 8 3 -2 6 3 7 -1
1 -2 4 5 -2 8
3 -2 6 3 7 -1
1 -2 4
5 -2 8
3 -2 6
3 7 -1
1 -2
5 -2
3 -2
3 7
1
-2
5
-2
3
-2
3
7

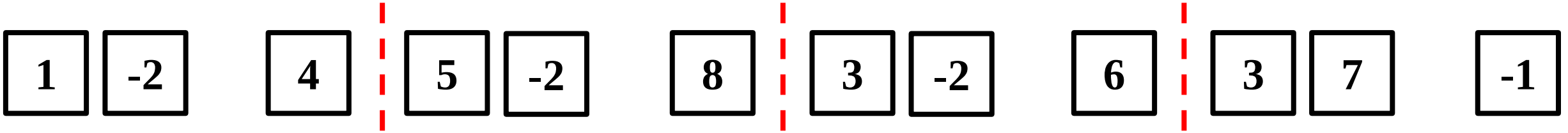
算法实例


1 -2 4 5 -2 8 3 -2 6 3 7 -1
1 -2 4 5 -2 8
3 -2 6 3 7 -1
1 -2 4
5 -2 8
3 -2 6
3 7 -1
1 -2
5 -2
3 -2
3 7
1
-2
4
5
-2
8
3
-2
6
3
7
-1

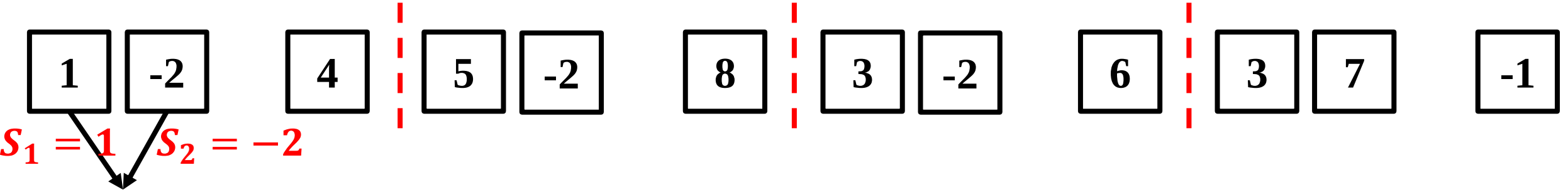
算法实例


1 -2 4 5 -2 8 3 -2 6 3 7 -1
1 -2 4 5 -2 8
3 -2 6 3 7 -1
1 -2 4
5 -2 8
3 -2 6
3 7 -1
1 -2
5 -2
3 -2
3 7
1
-2
4
5
-2
8
3
-2
6
3
7
-1

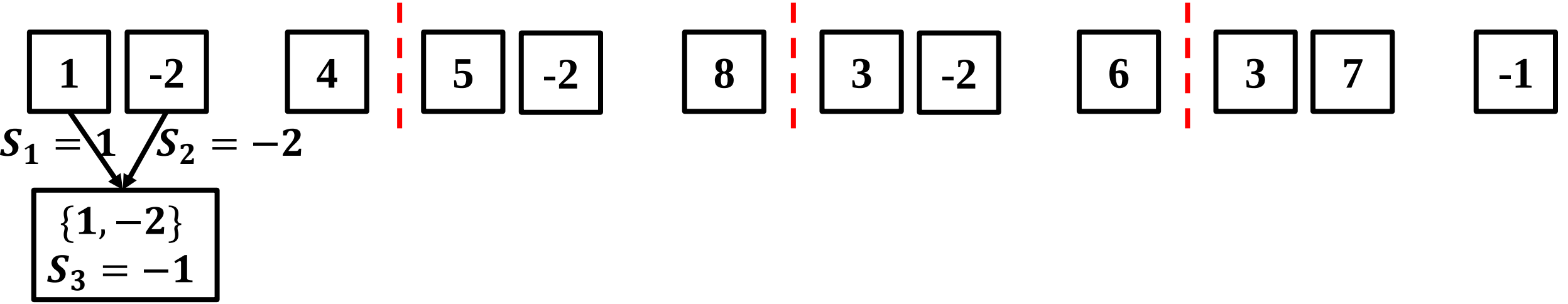
算法实例



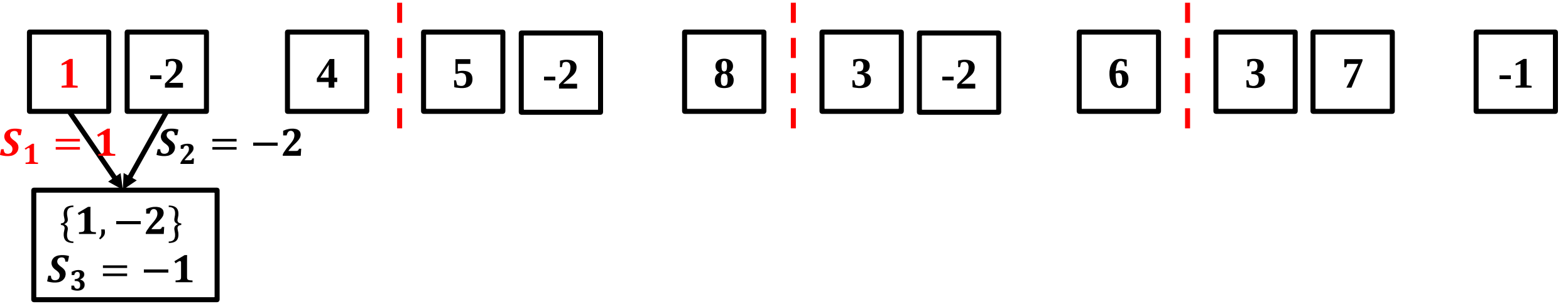
算法实例



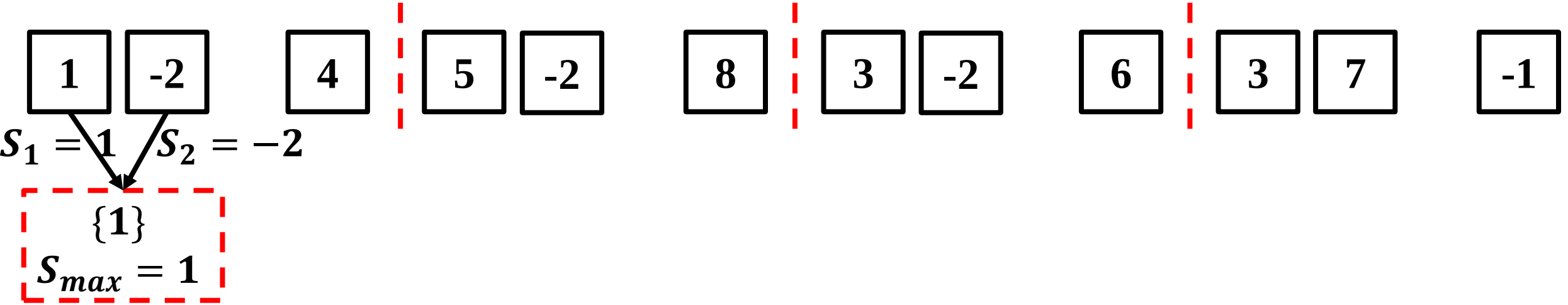
算法实例



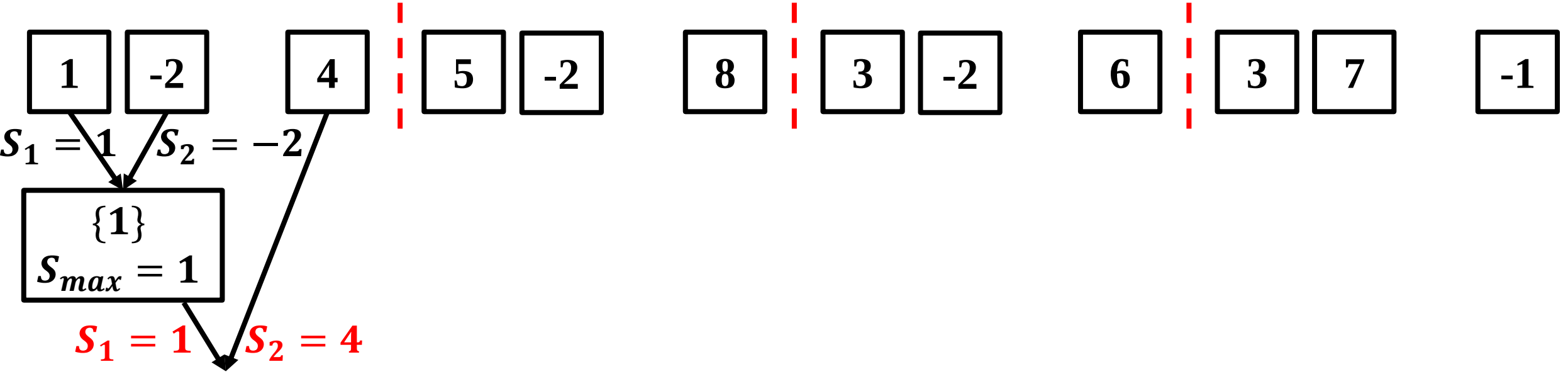
算法实例



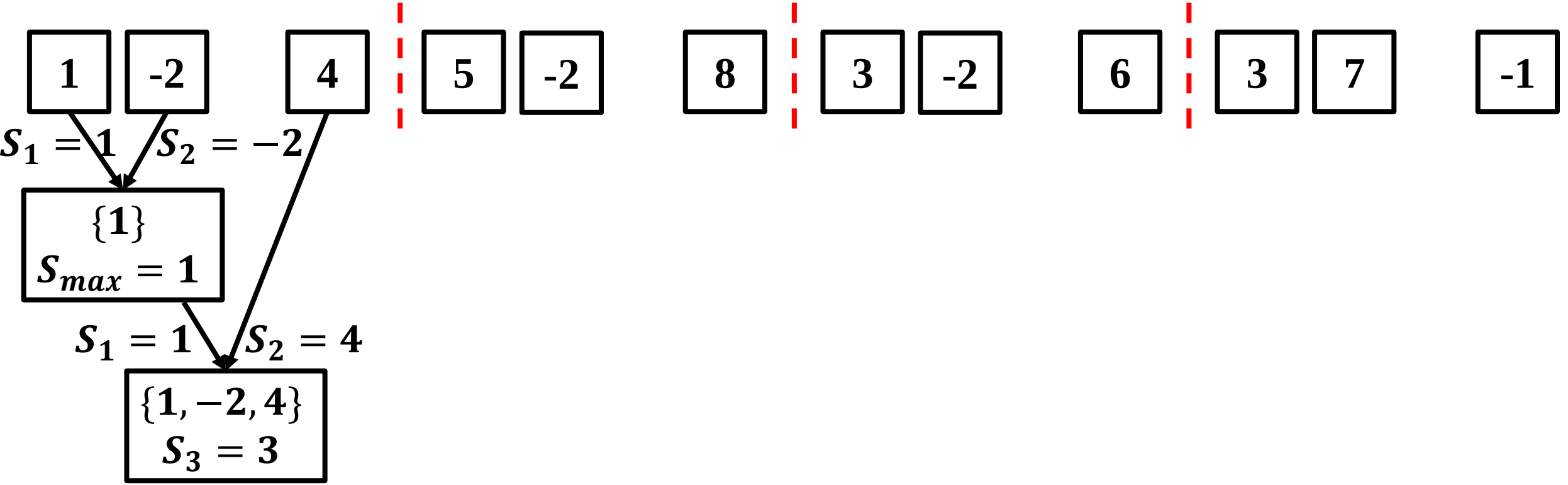
算法实例



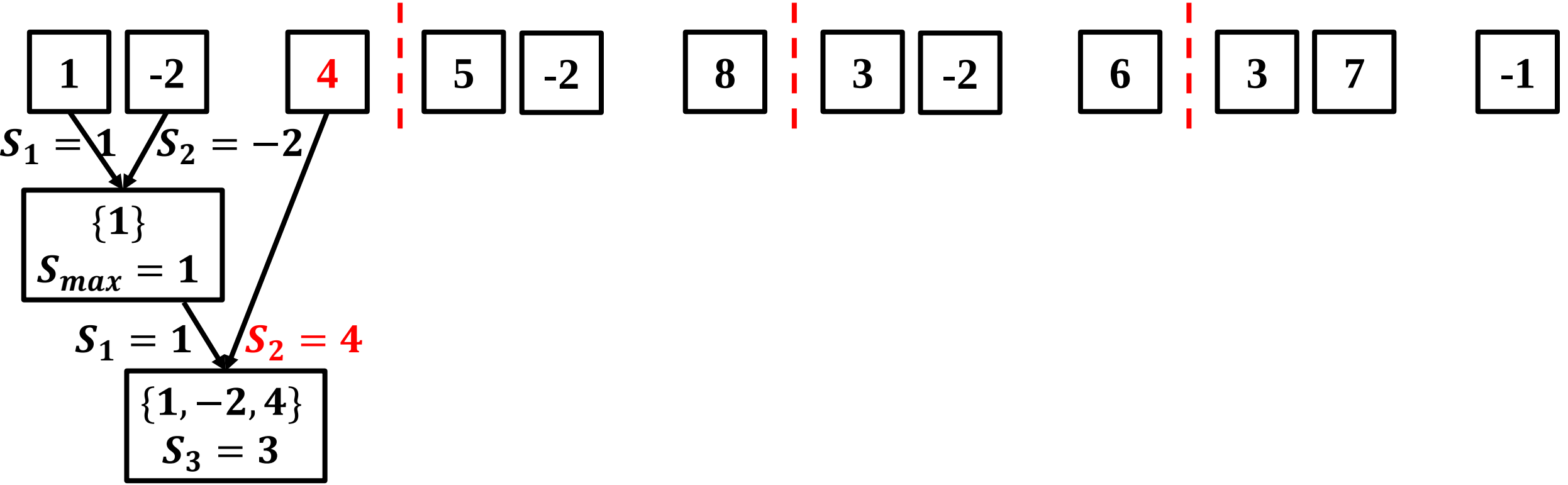
算法实例



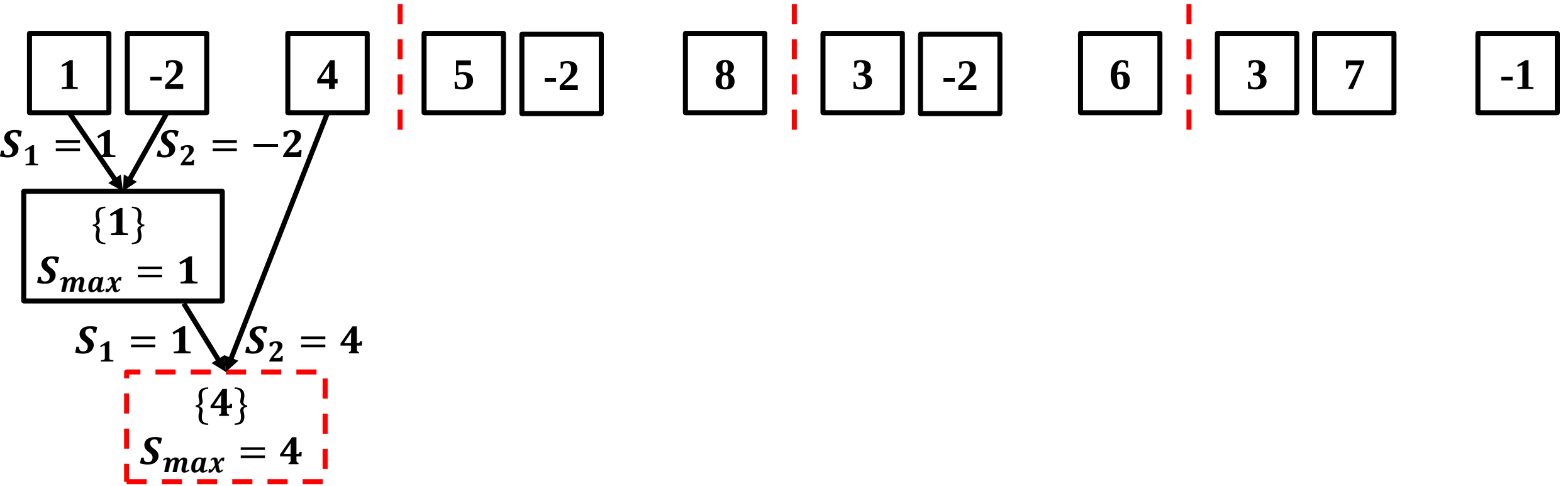
算法实例



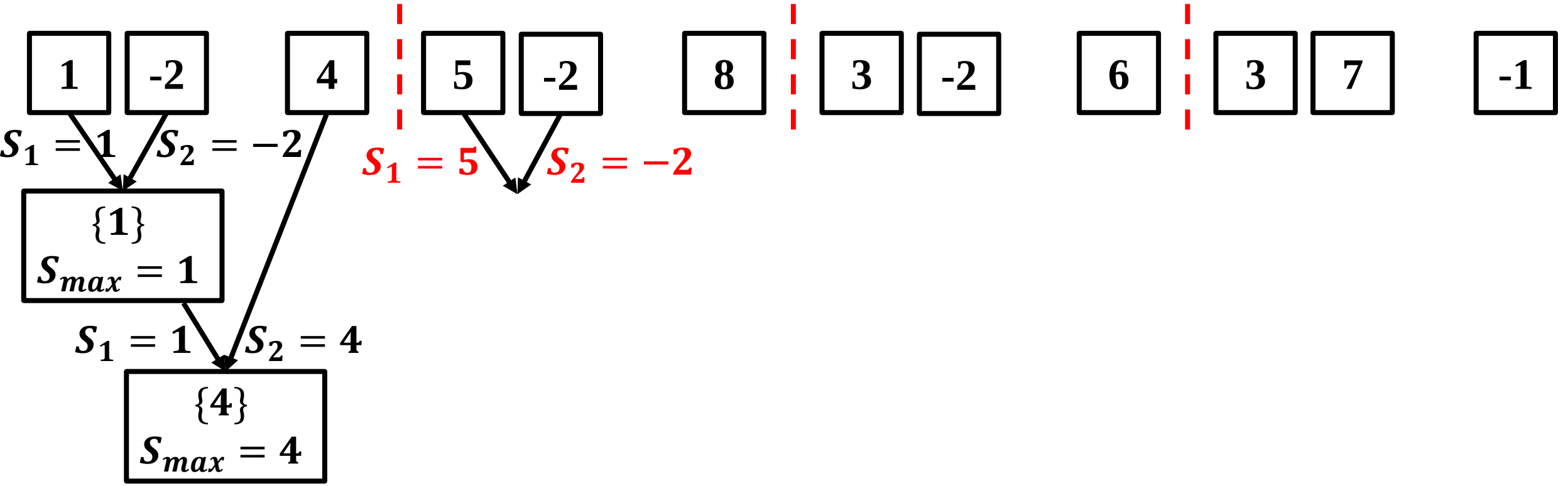
算法实例



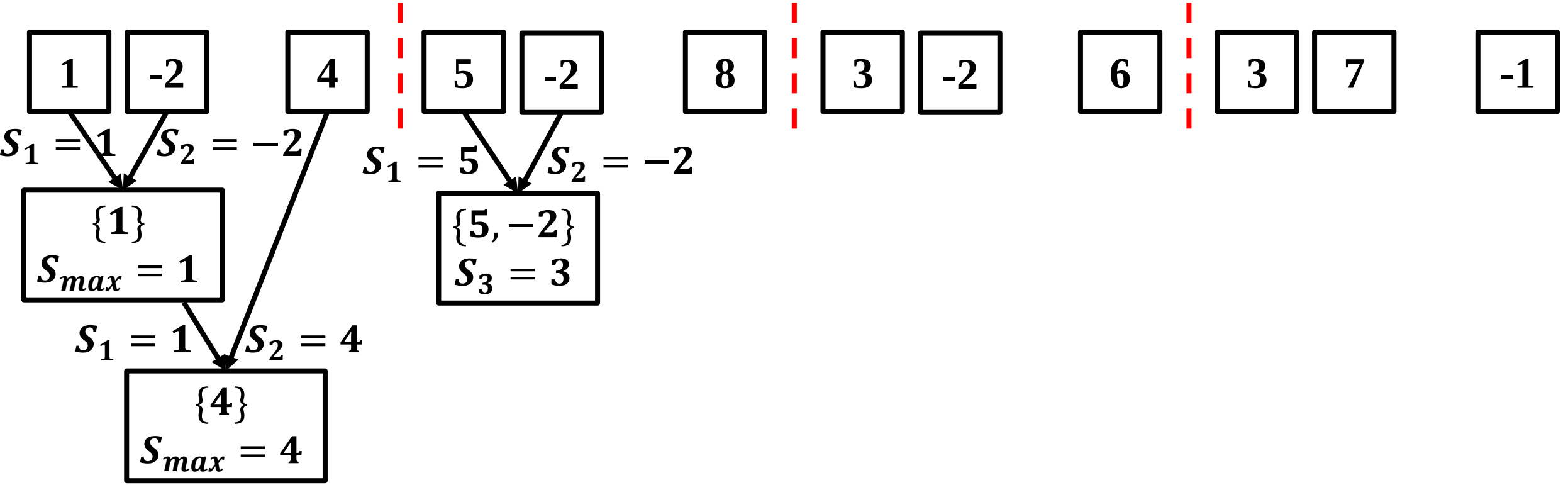
算法实例



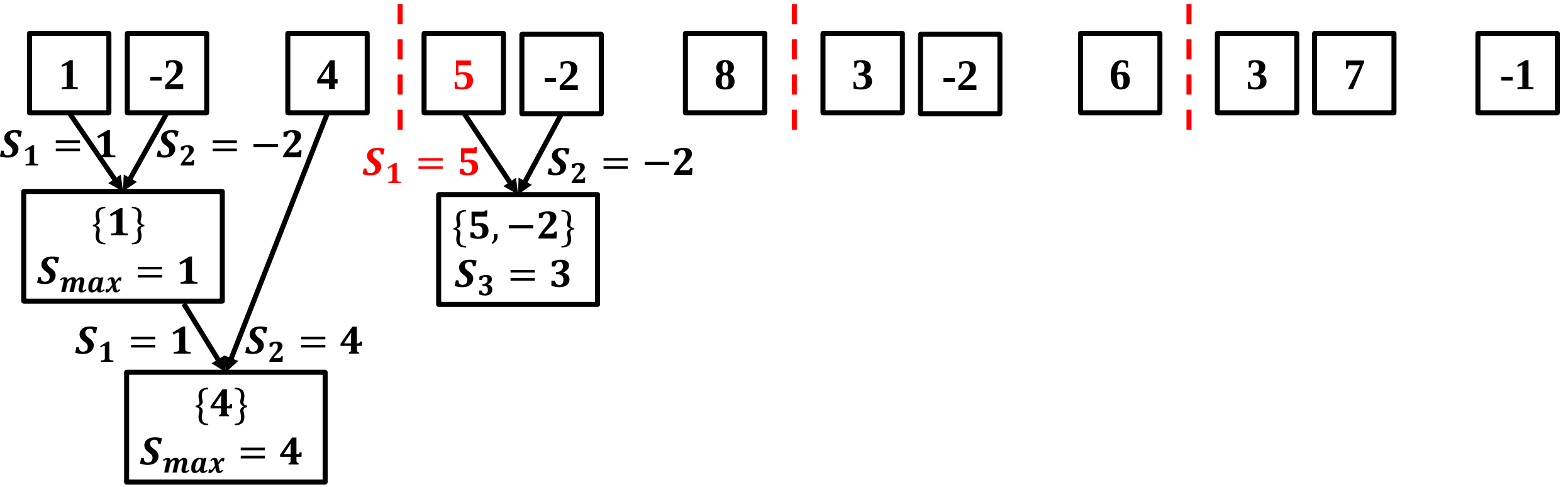
算法实例



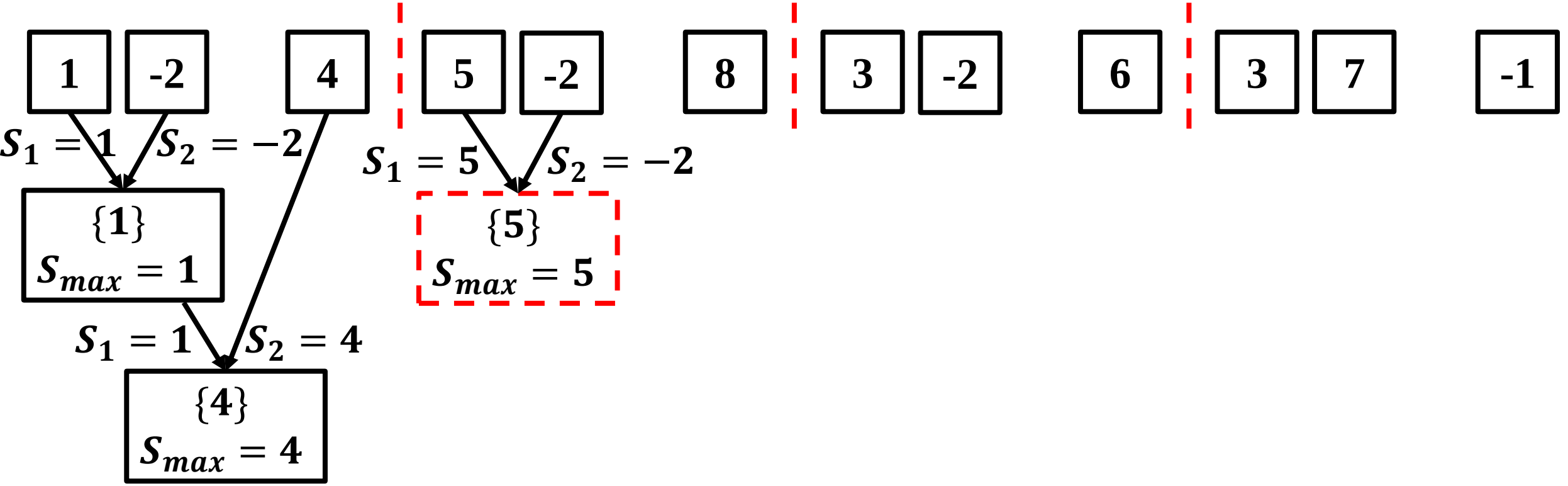
算法实例



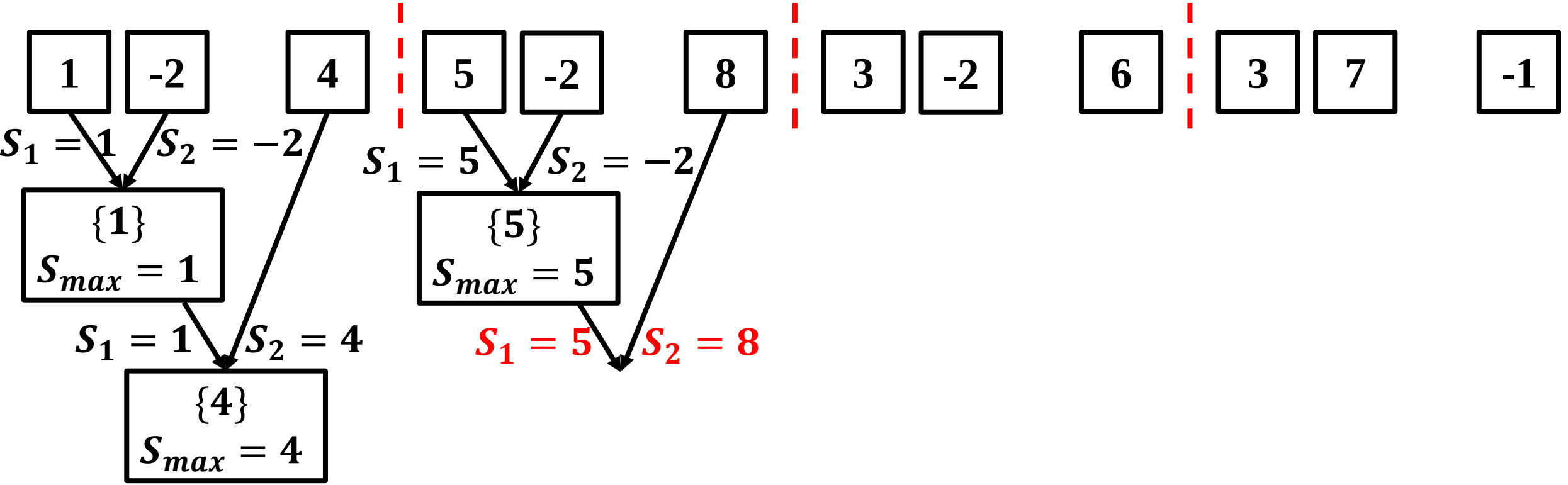
算法实例



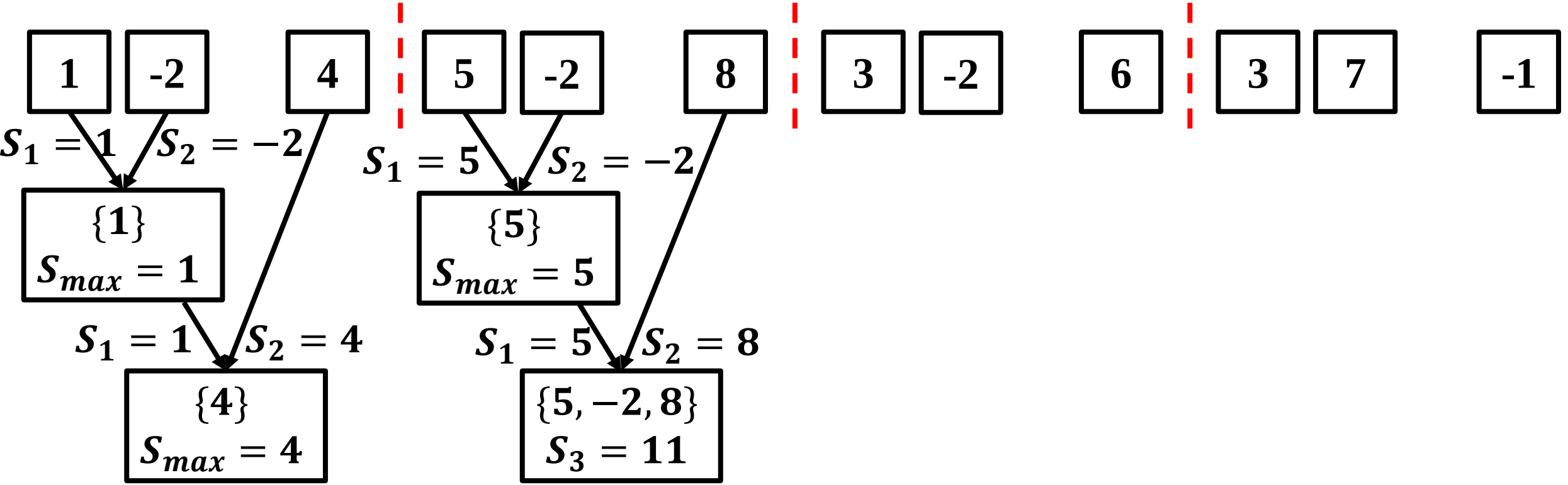
算法实例



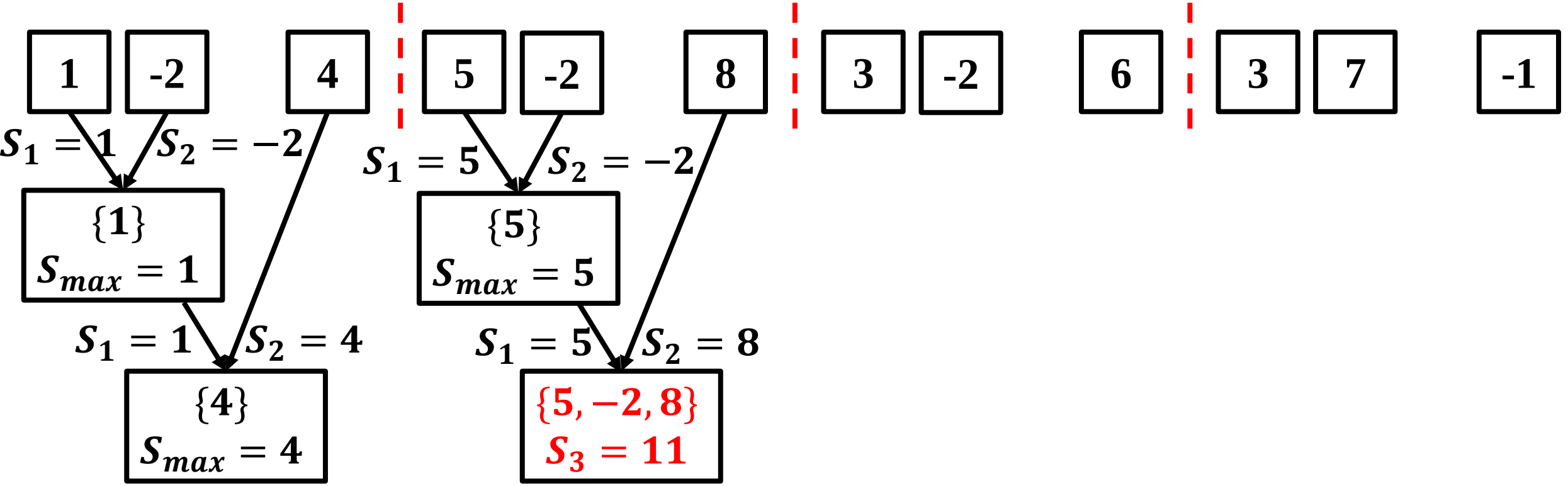
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$

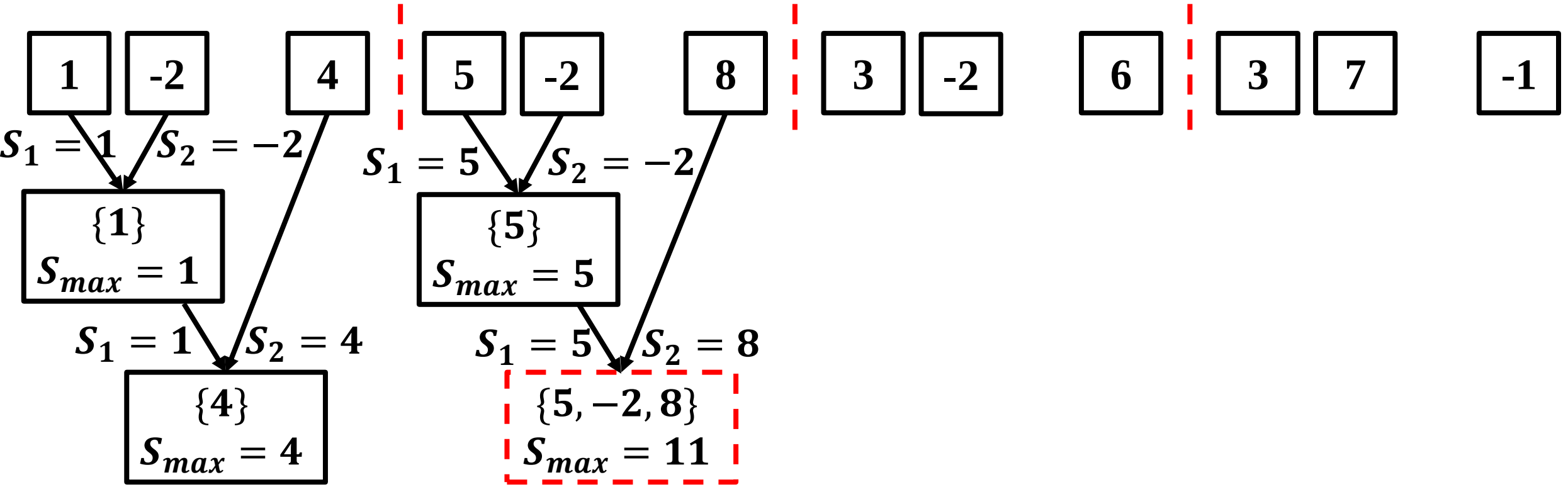
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_3 = 11$

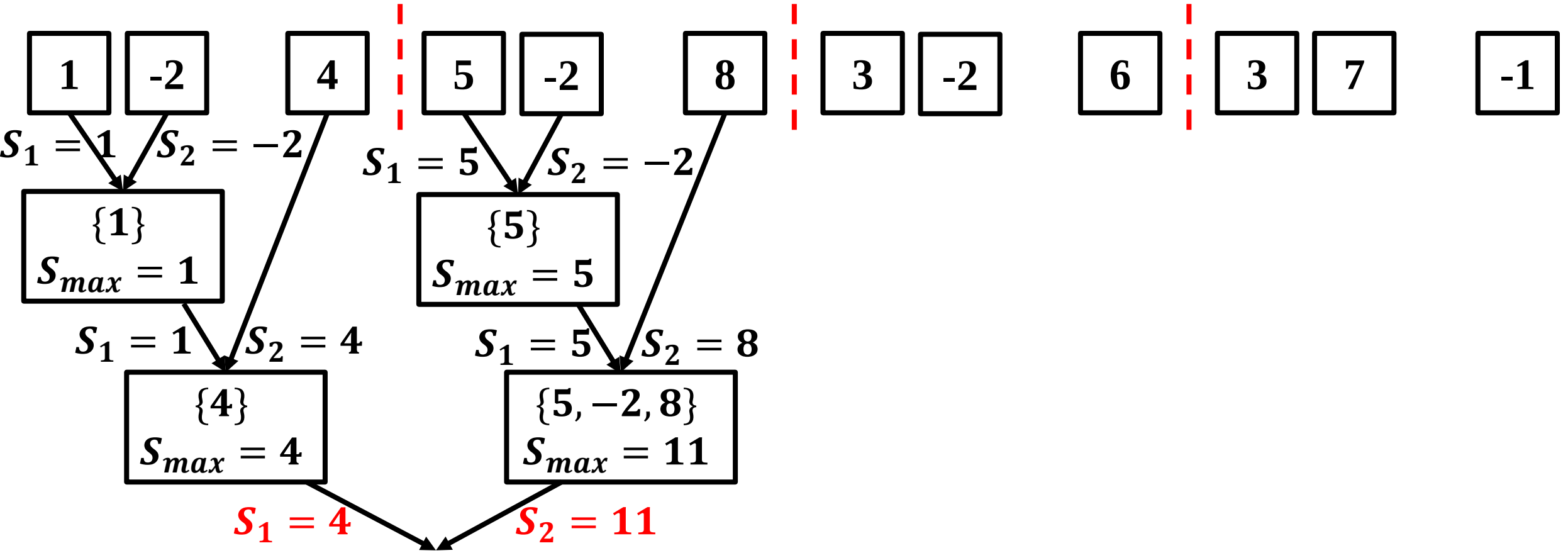
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_3 = 11$

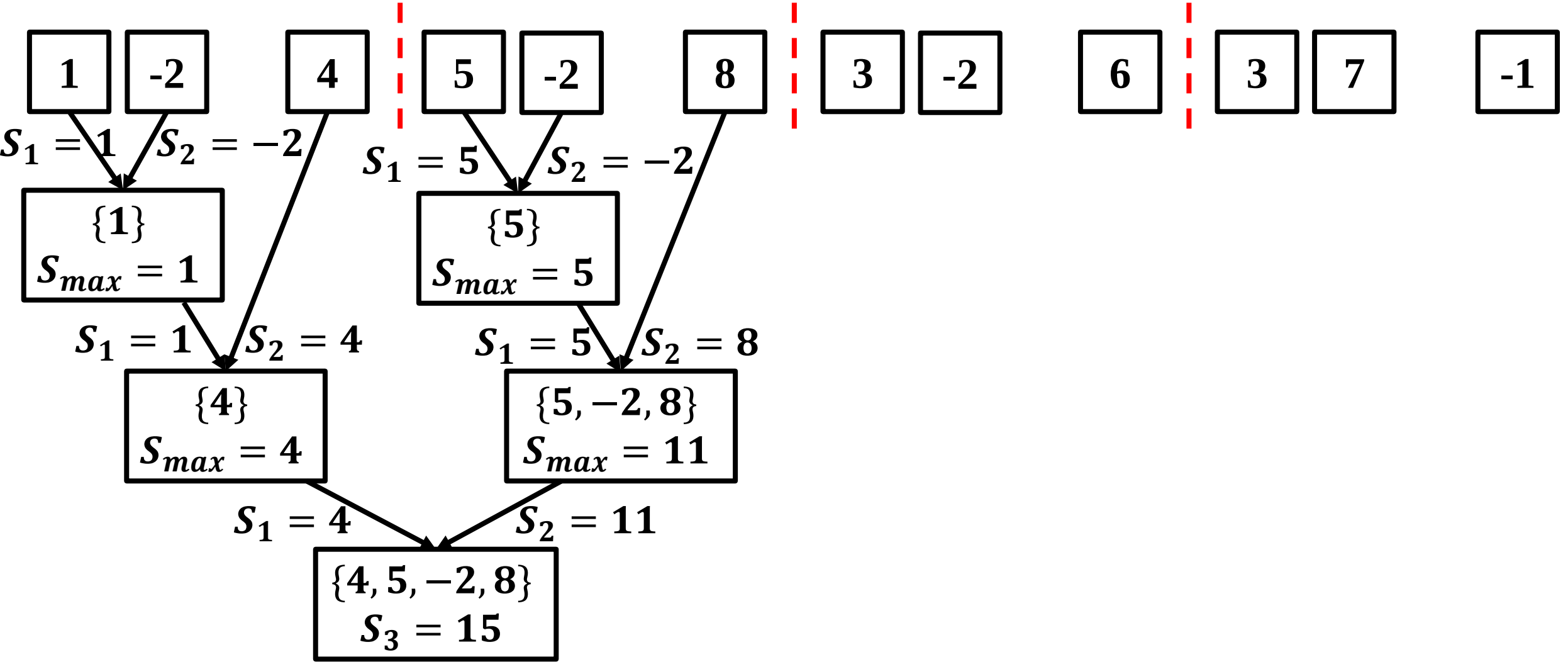
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$

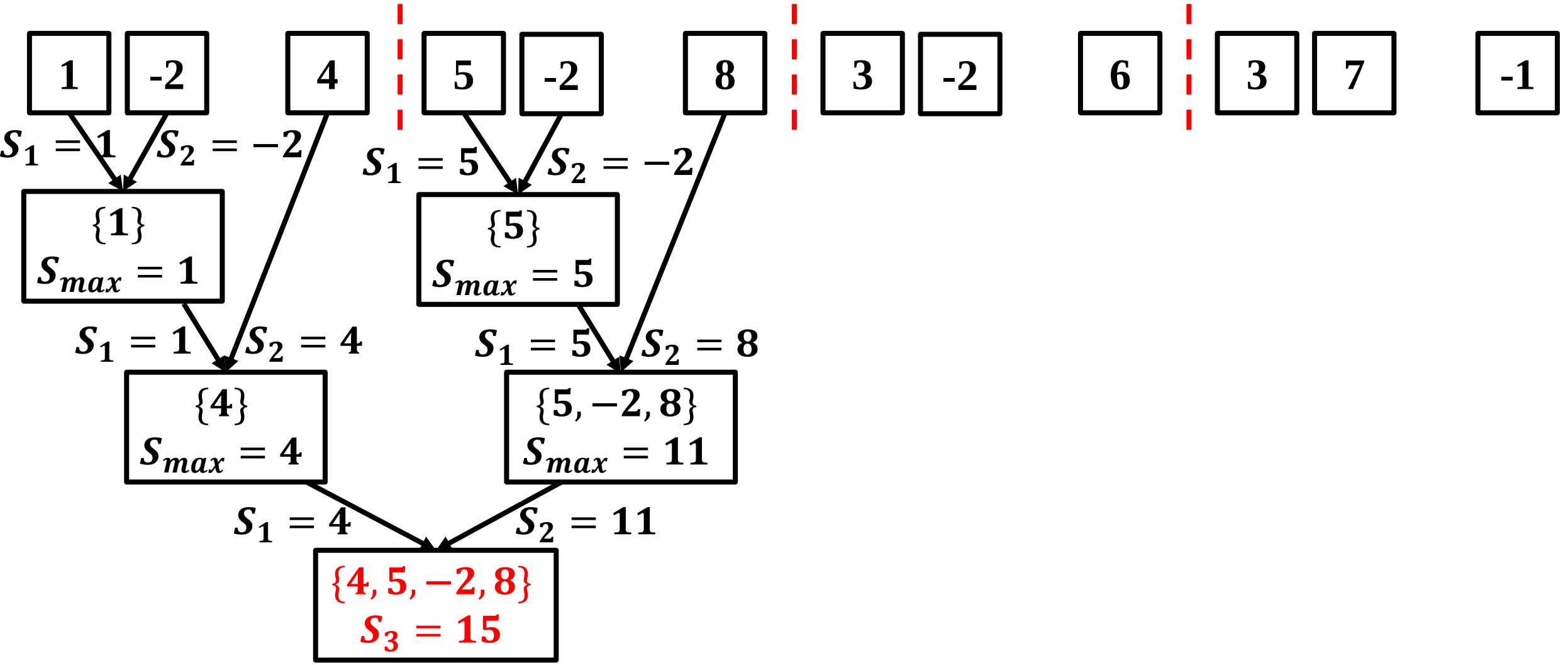
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$

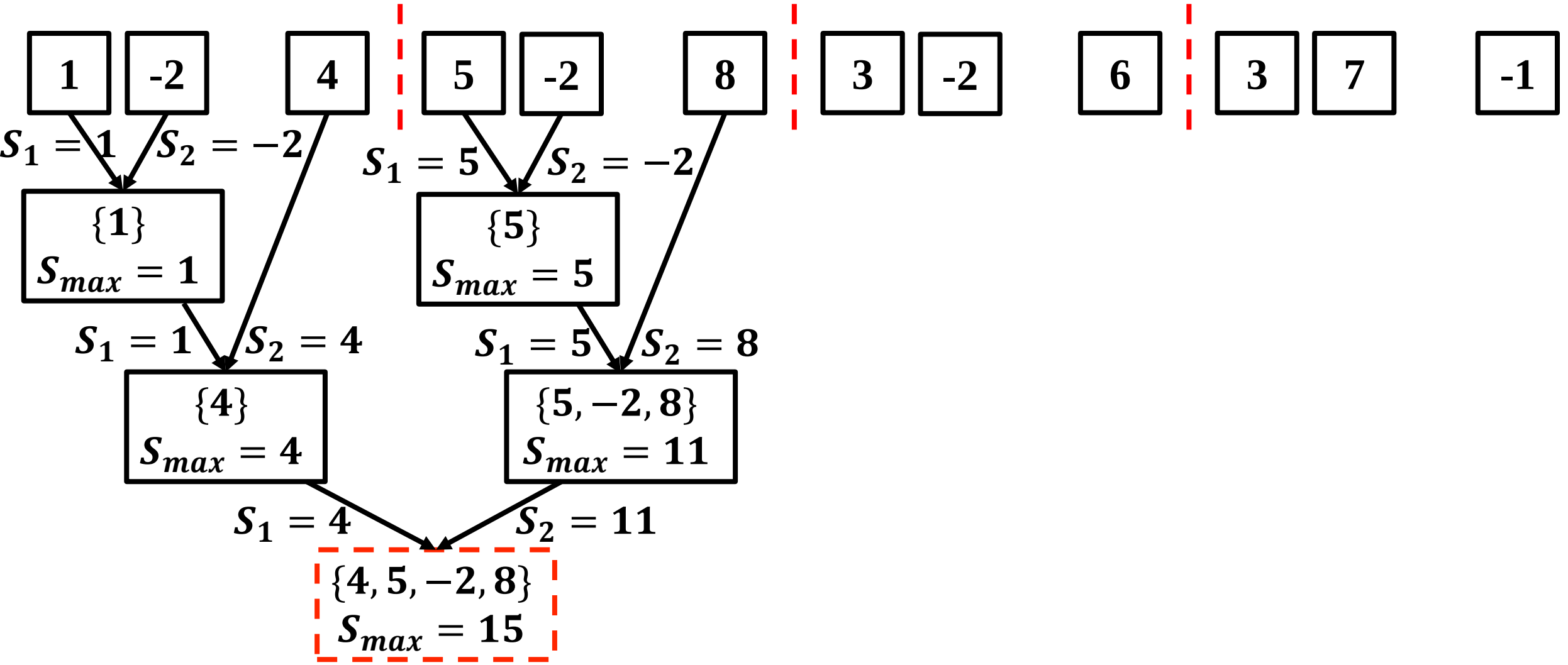
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_3 = 15$

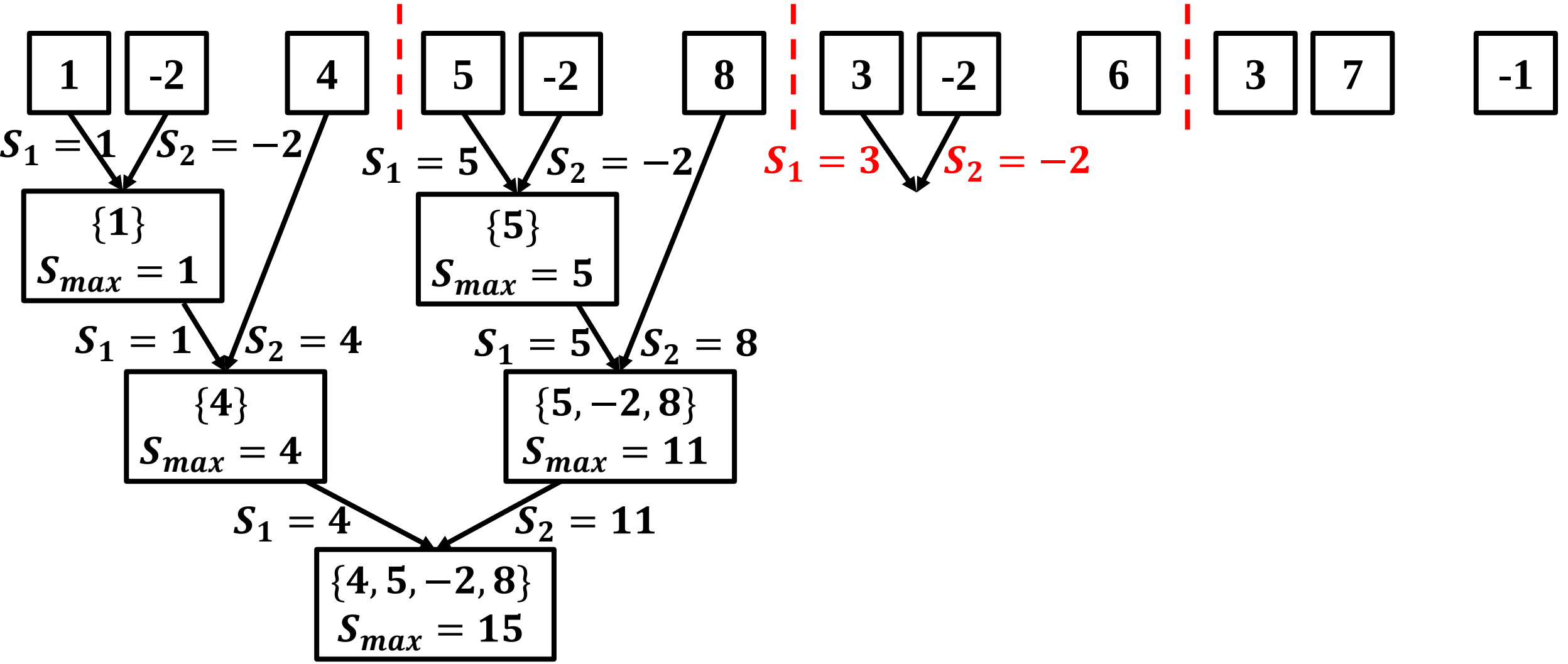
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_3 = 15$

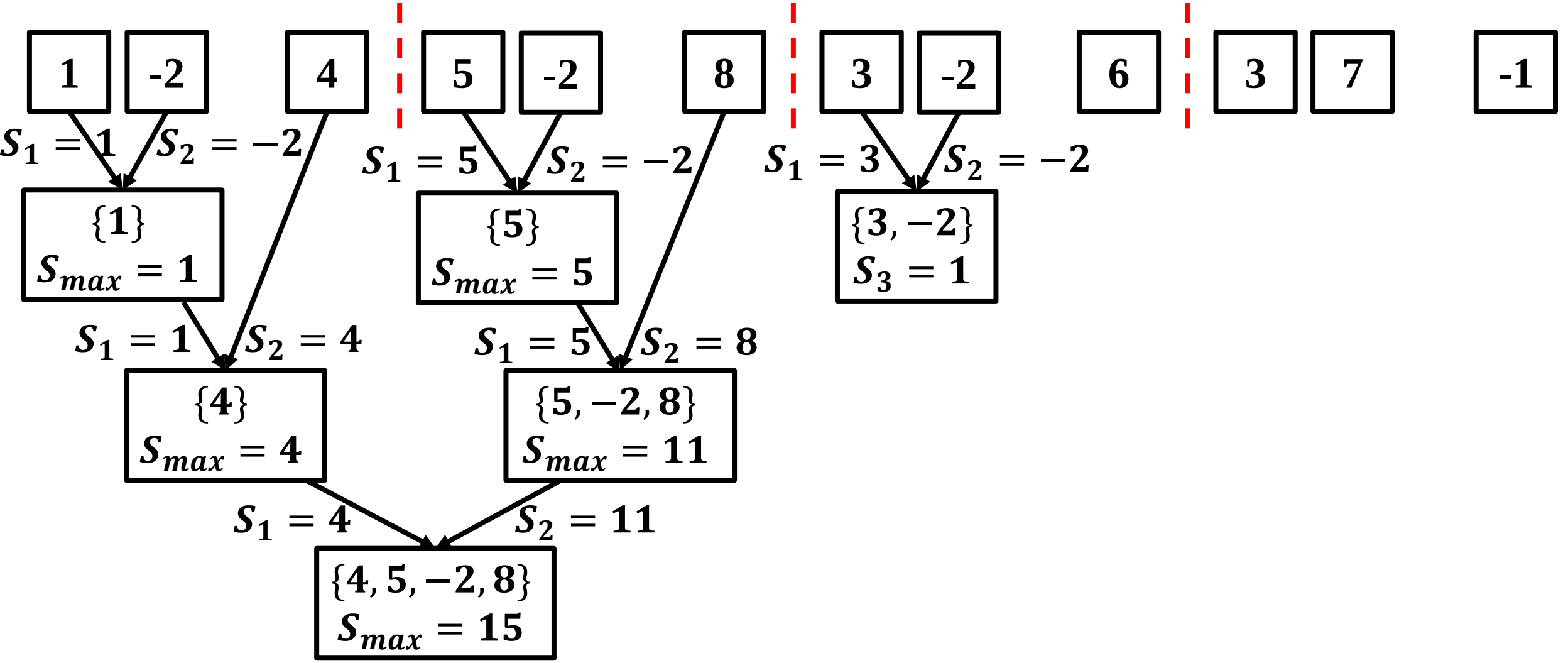
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

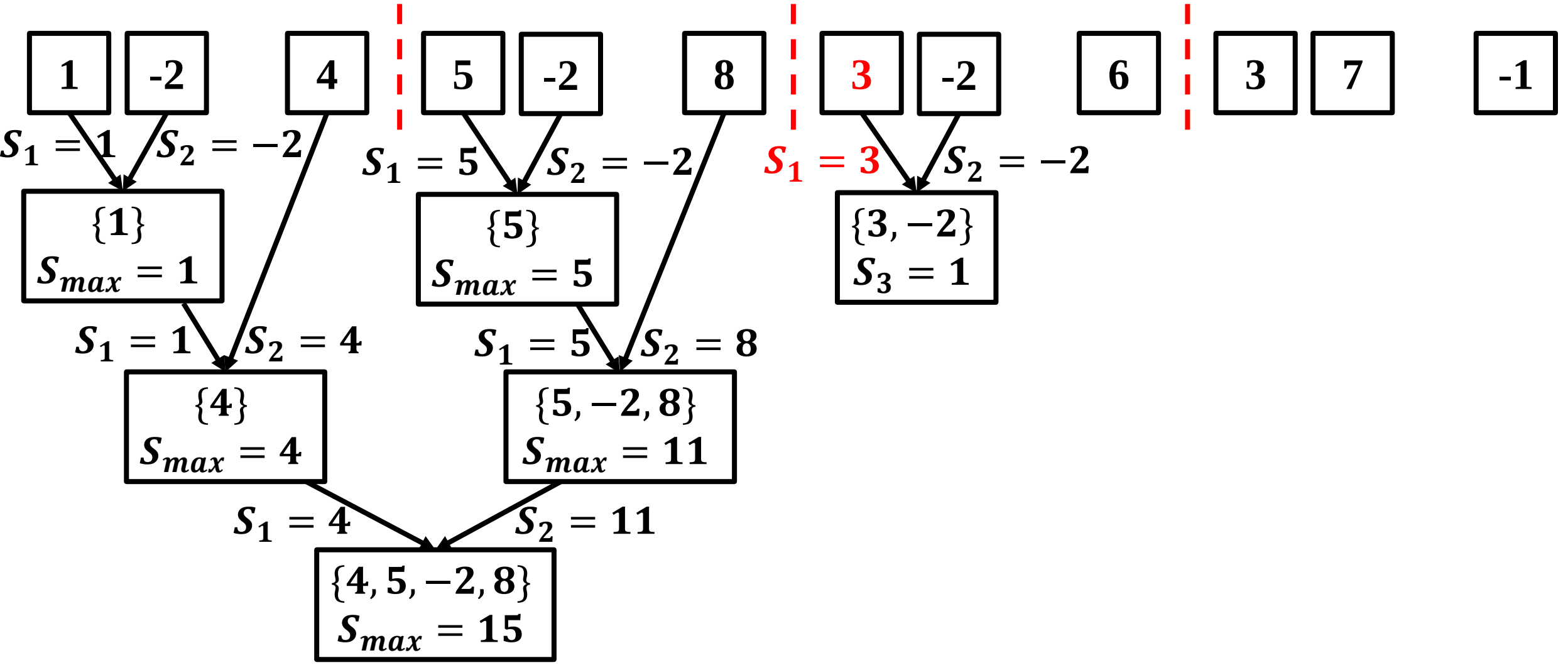
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

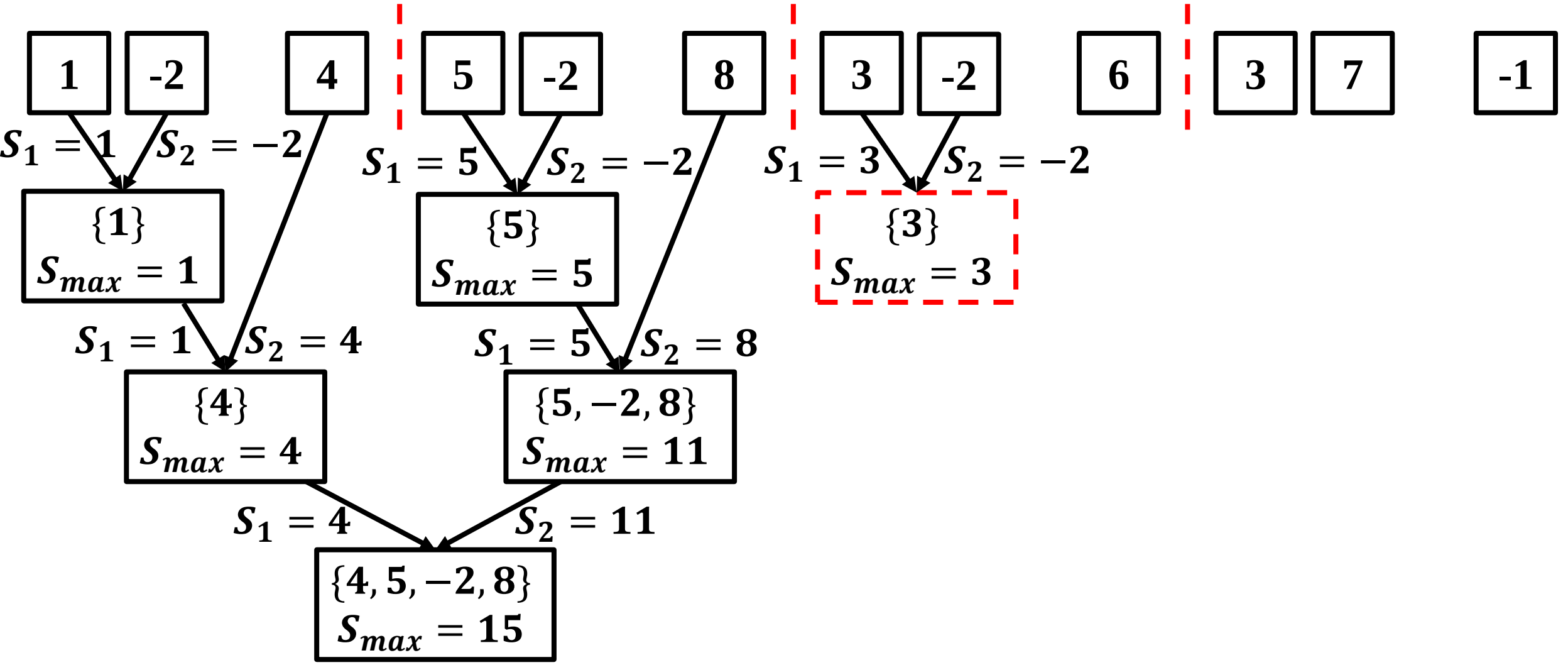
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3, -2\}$
$S_3 = 1$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

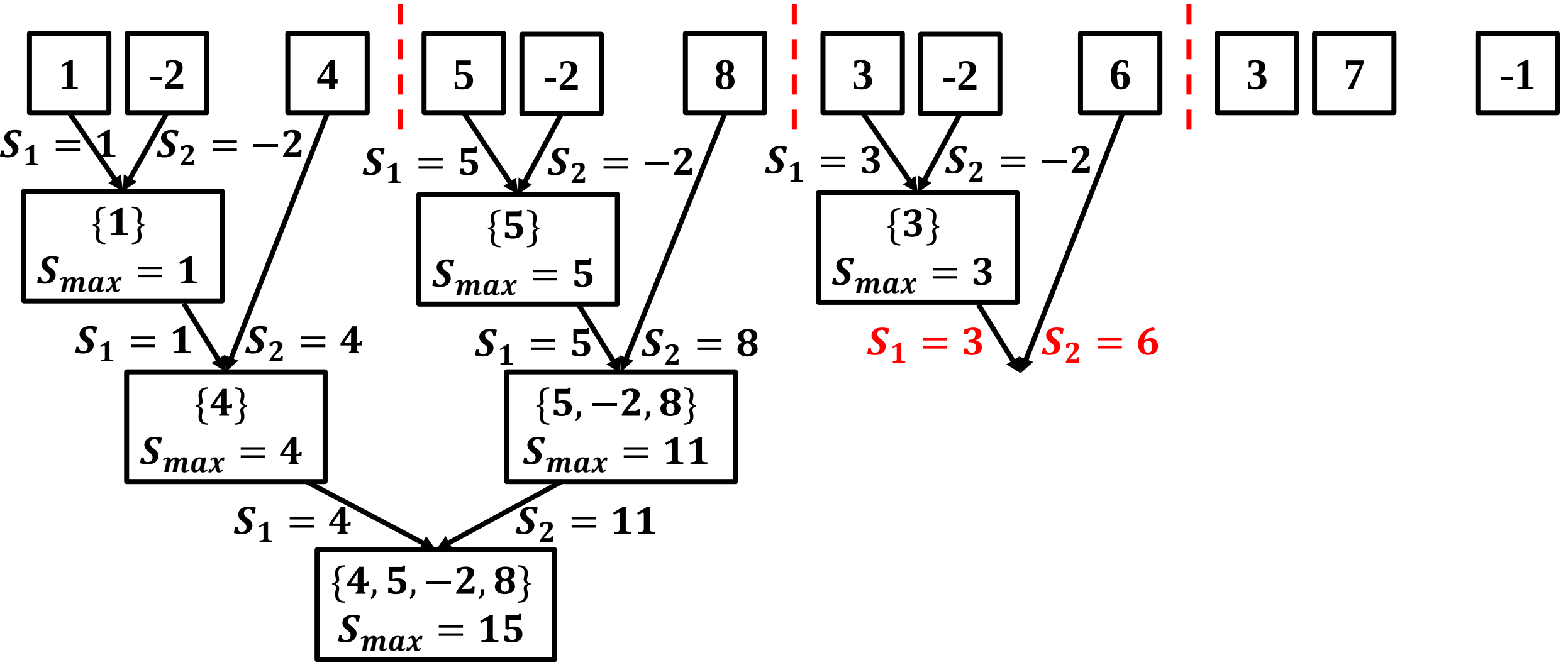
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3, -2\}$
$S_3 = 1$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

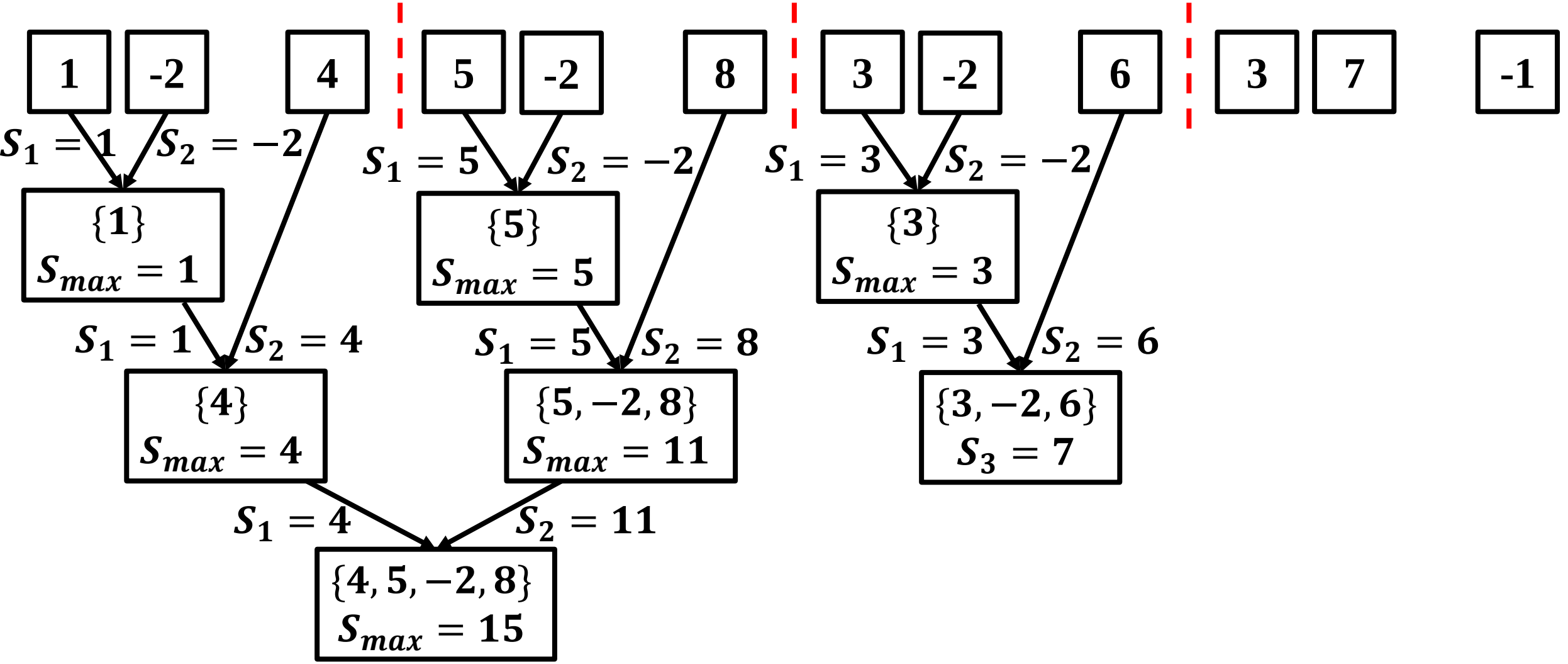
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

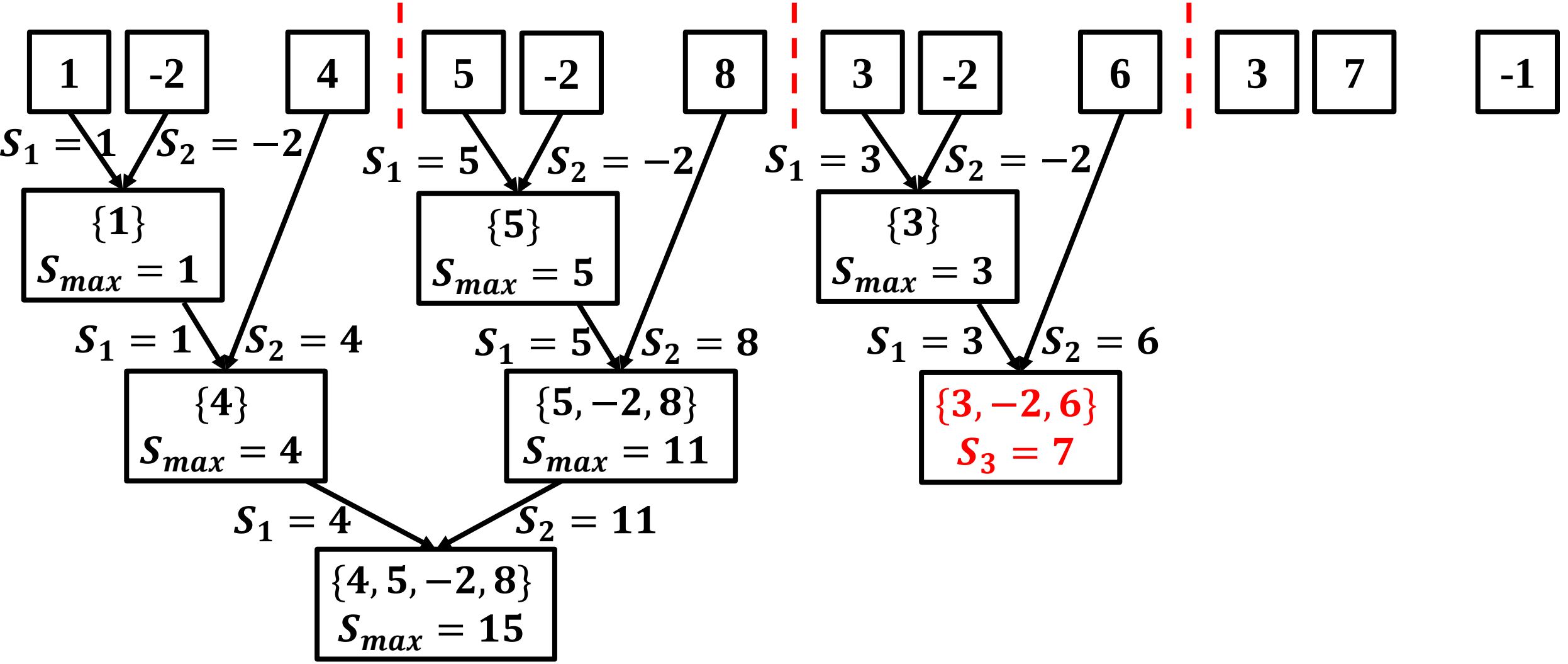
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

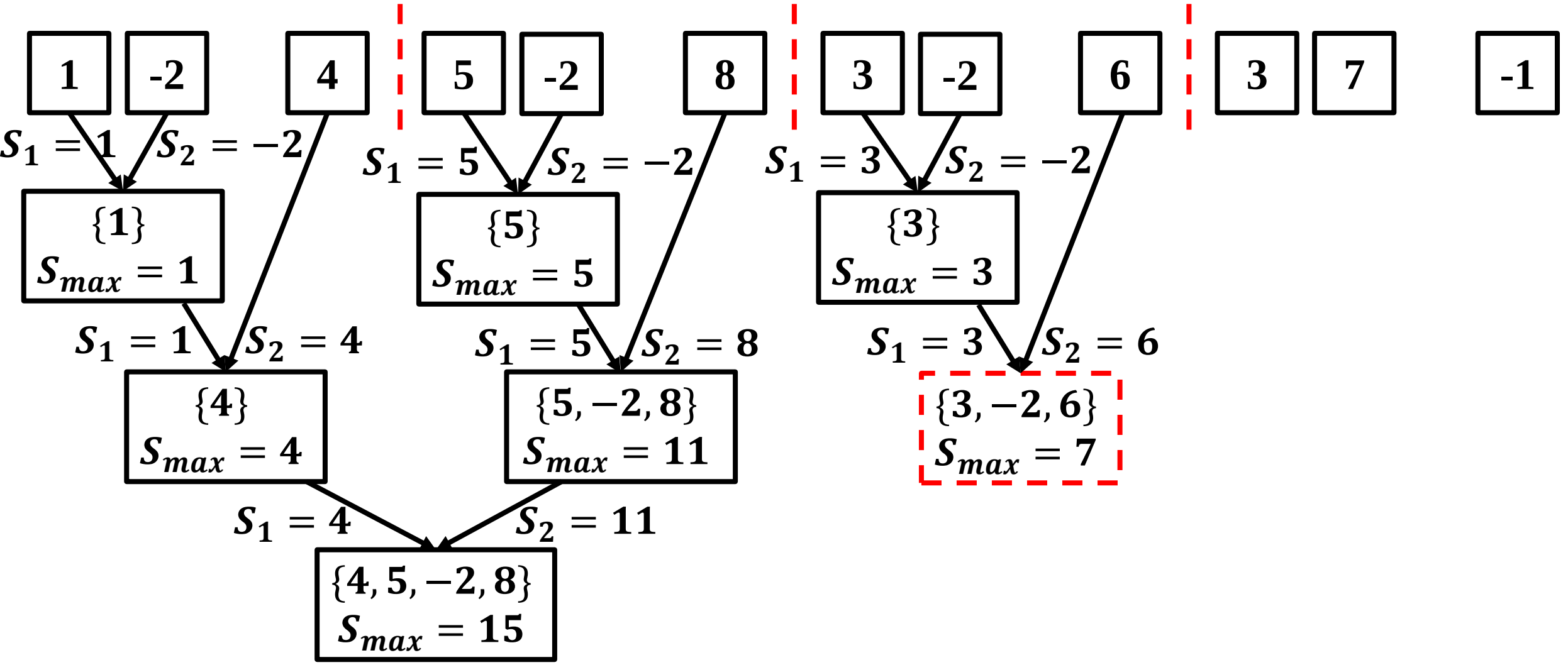
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_3 = 7$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

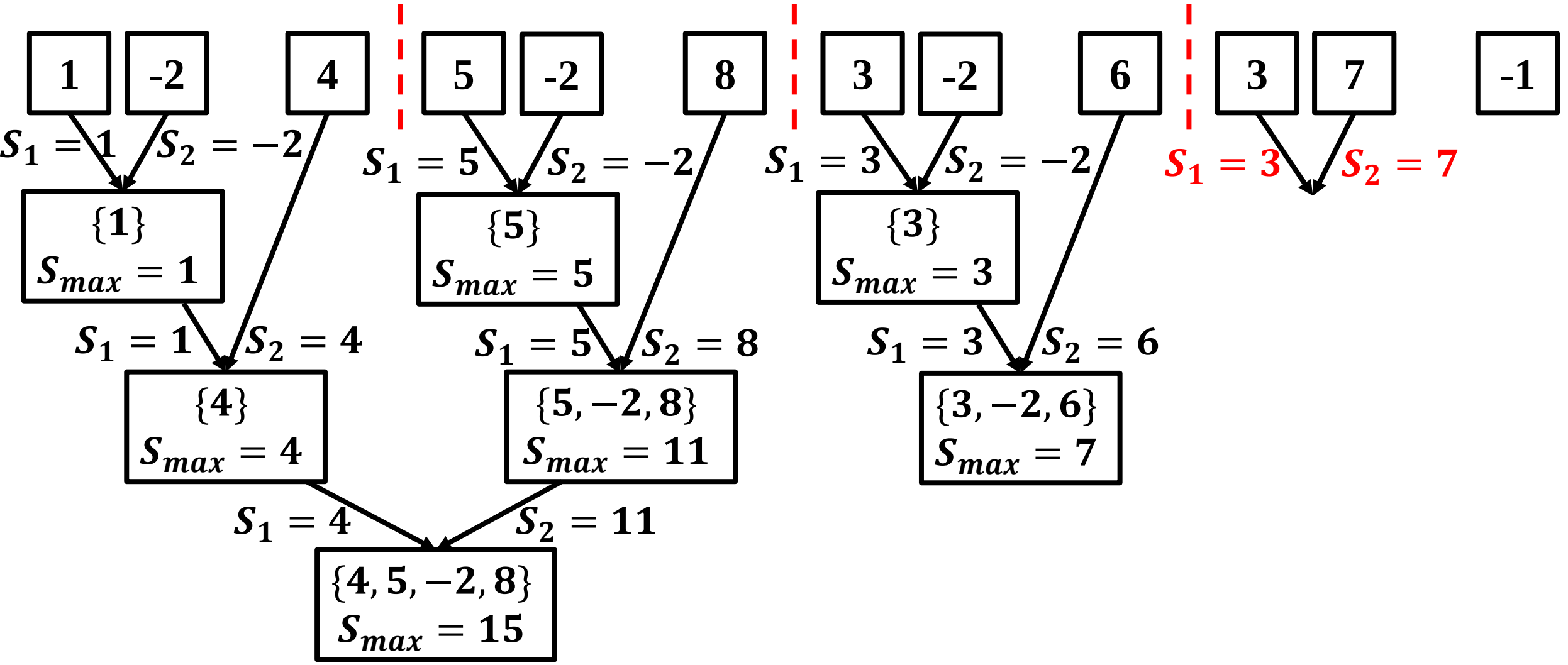
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_3 = 7$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

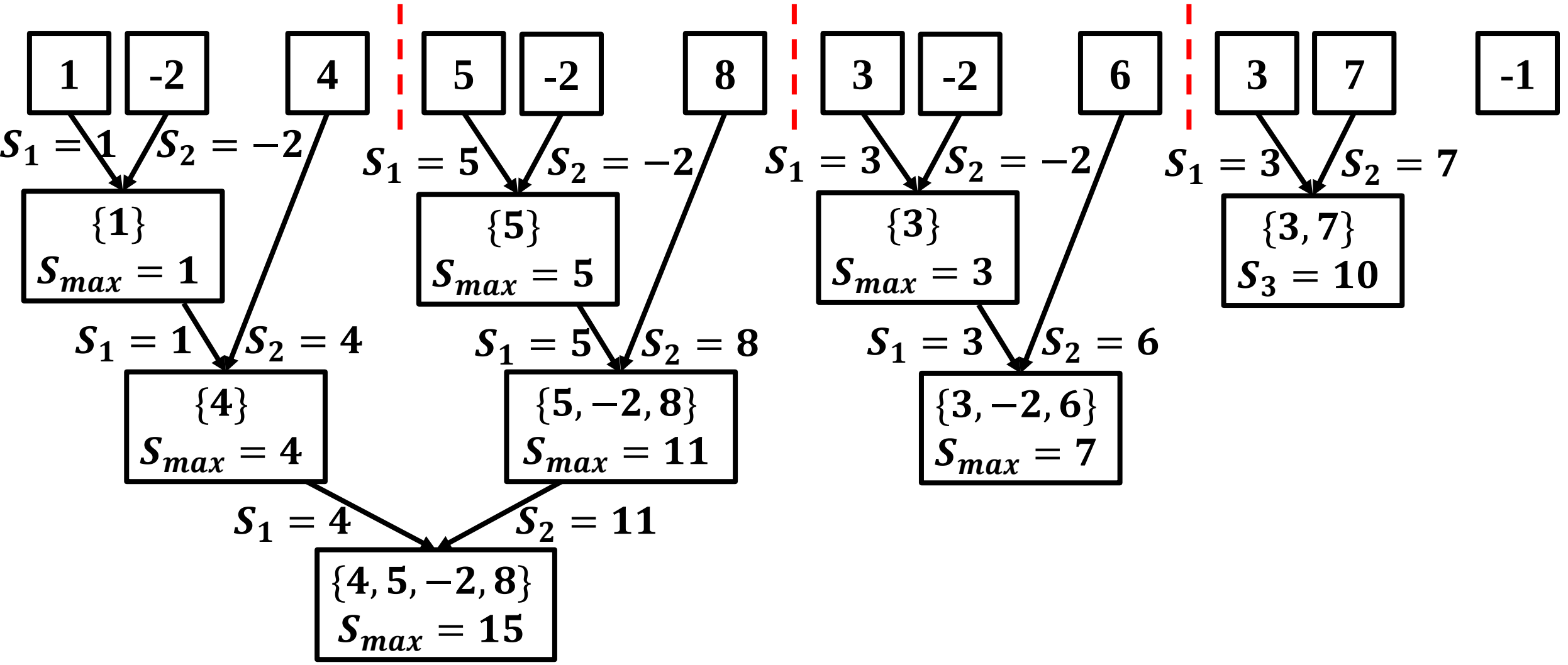
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

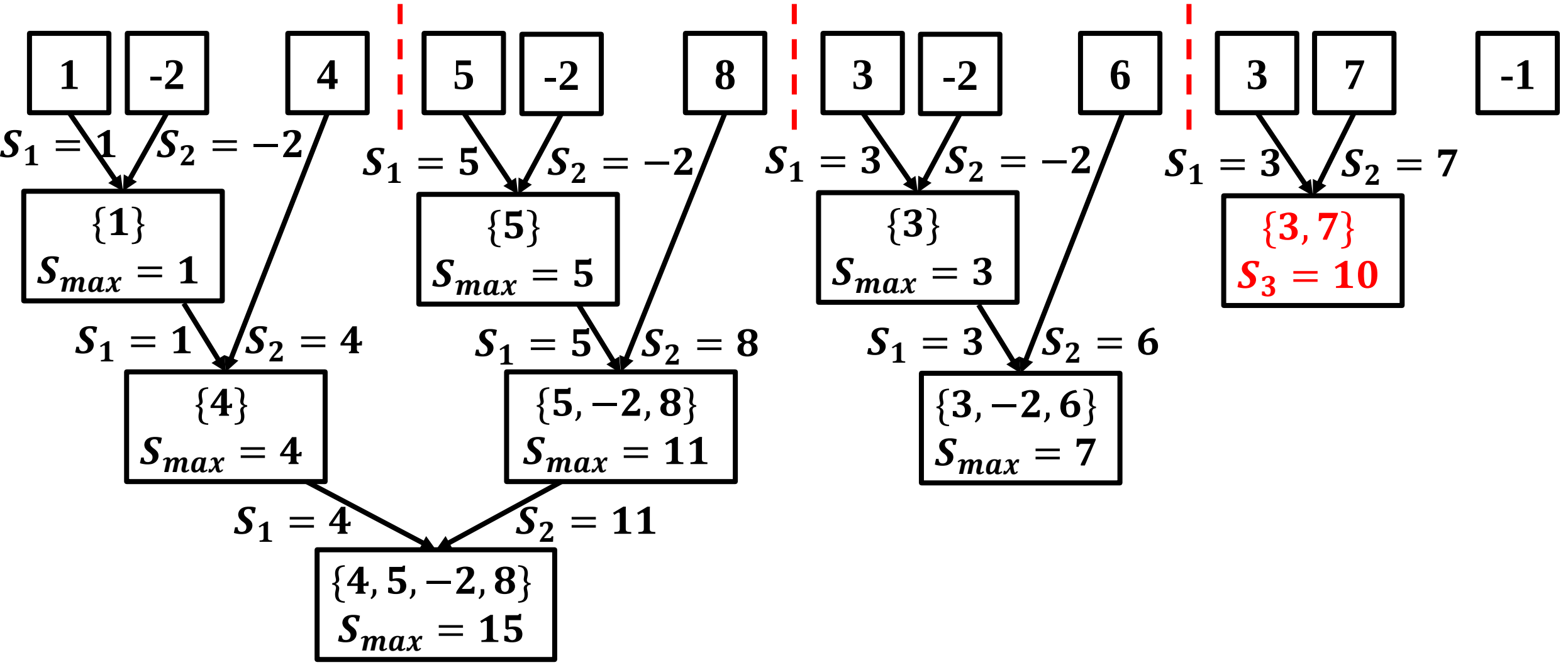
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

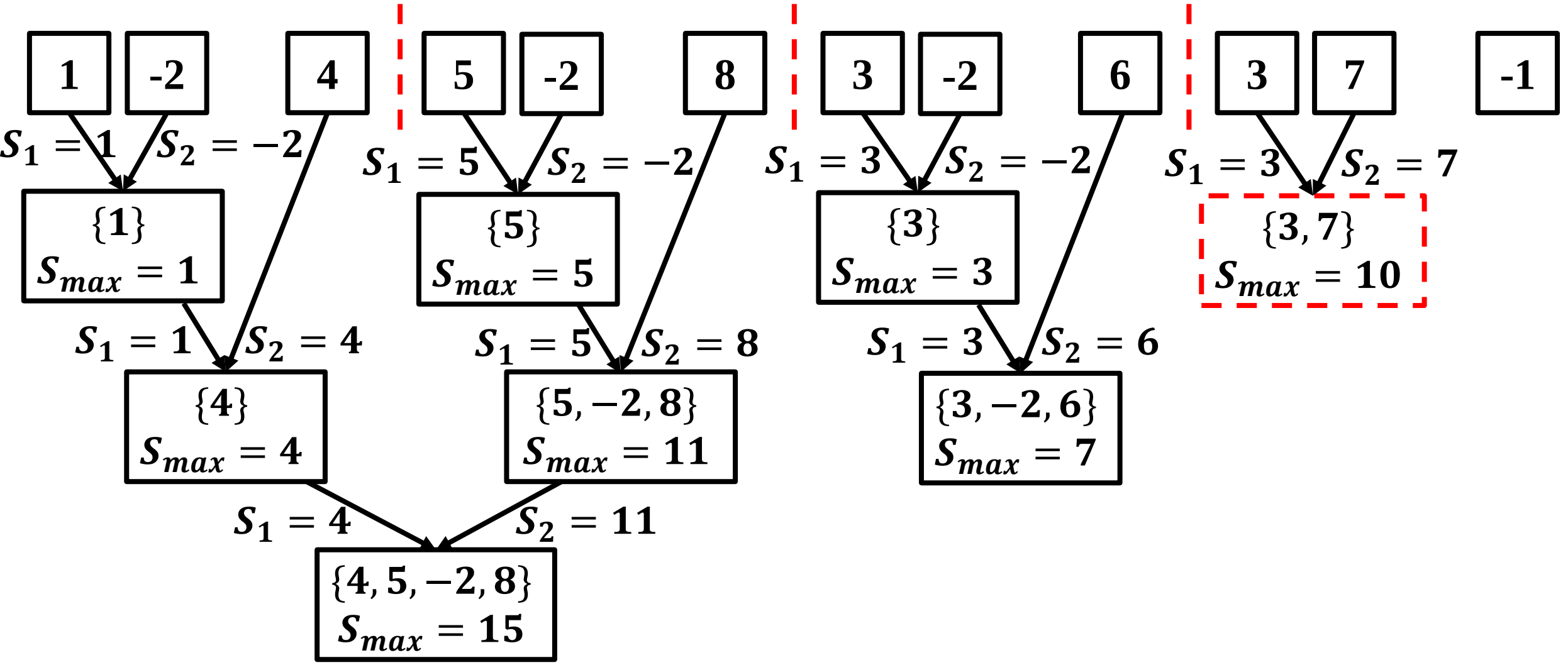
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3\}$
$S_{max} = 3$
$\{3, 7\}$
$S_3 = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$\{3, -2, 6\}$
$S_{max} = 7$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

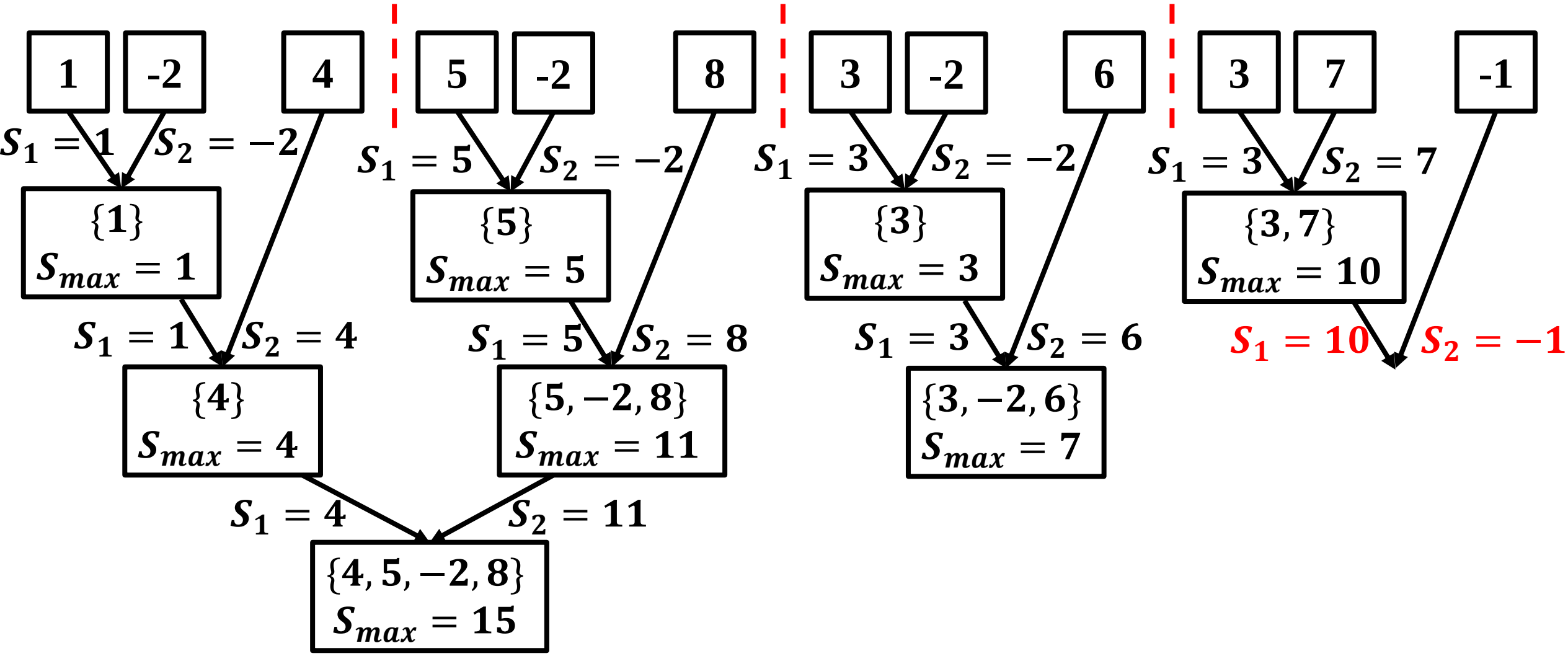
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3\}$
$S_{max} = 3$
$\{3, 7\}$
$S_3 = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$\{3, -2, 6\}$
$S_{max} = 7$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

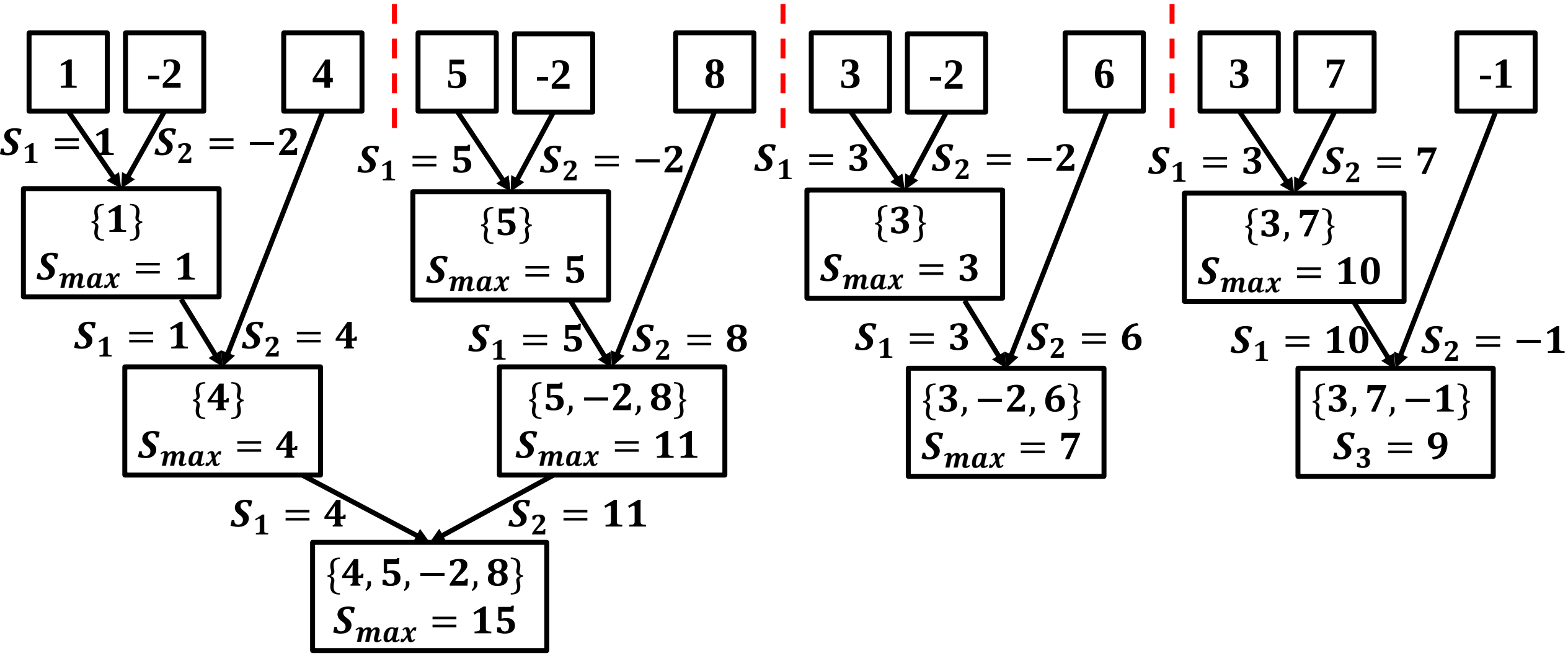
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
$S_1 = 4$
$S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

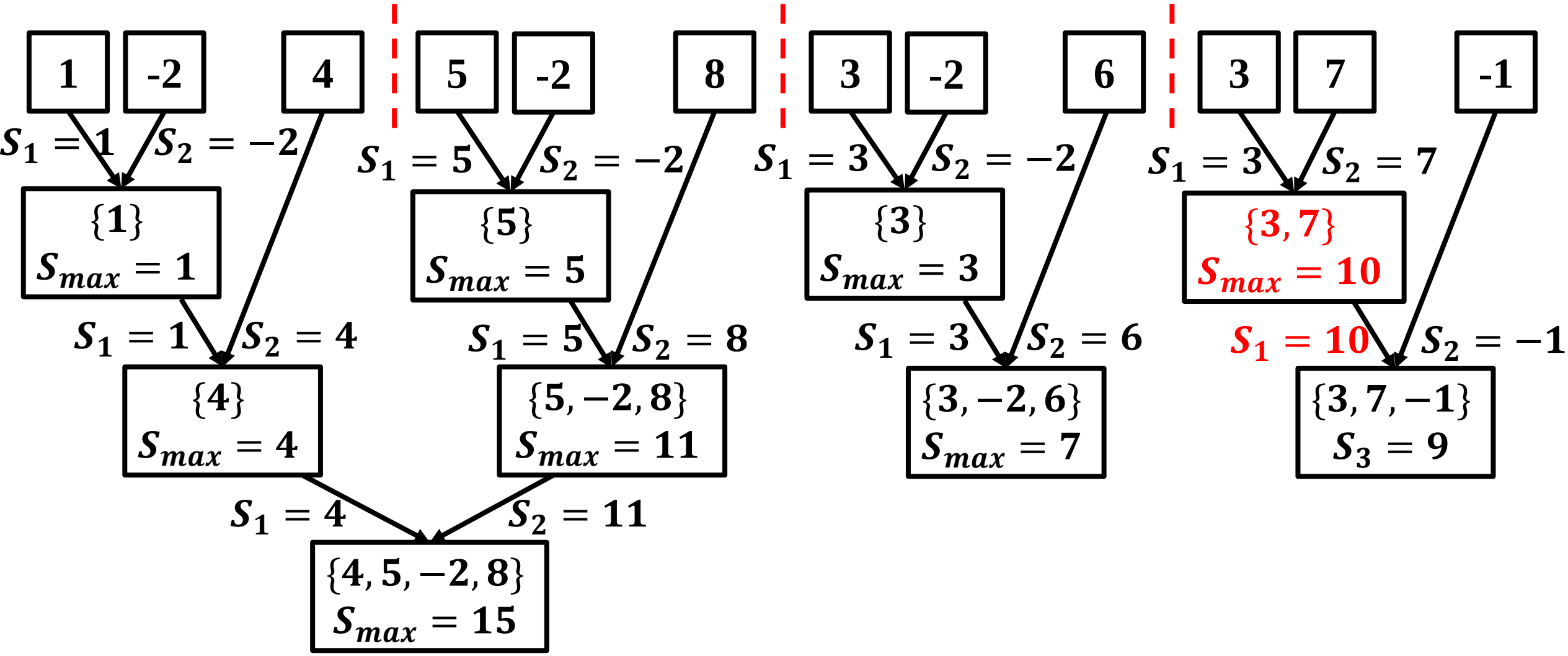
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3\}$
$S_{max} = 3$
$\{3, 7\}$
$S_{max} = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
$S_1 = 10$
$S_2 = -1$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$\{3, -2, 6\}$
$S_{max} = 7$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

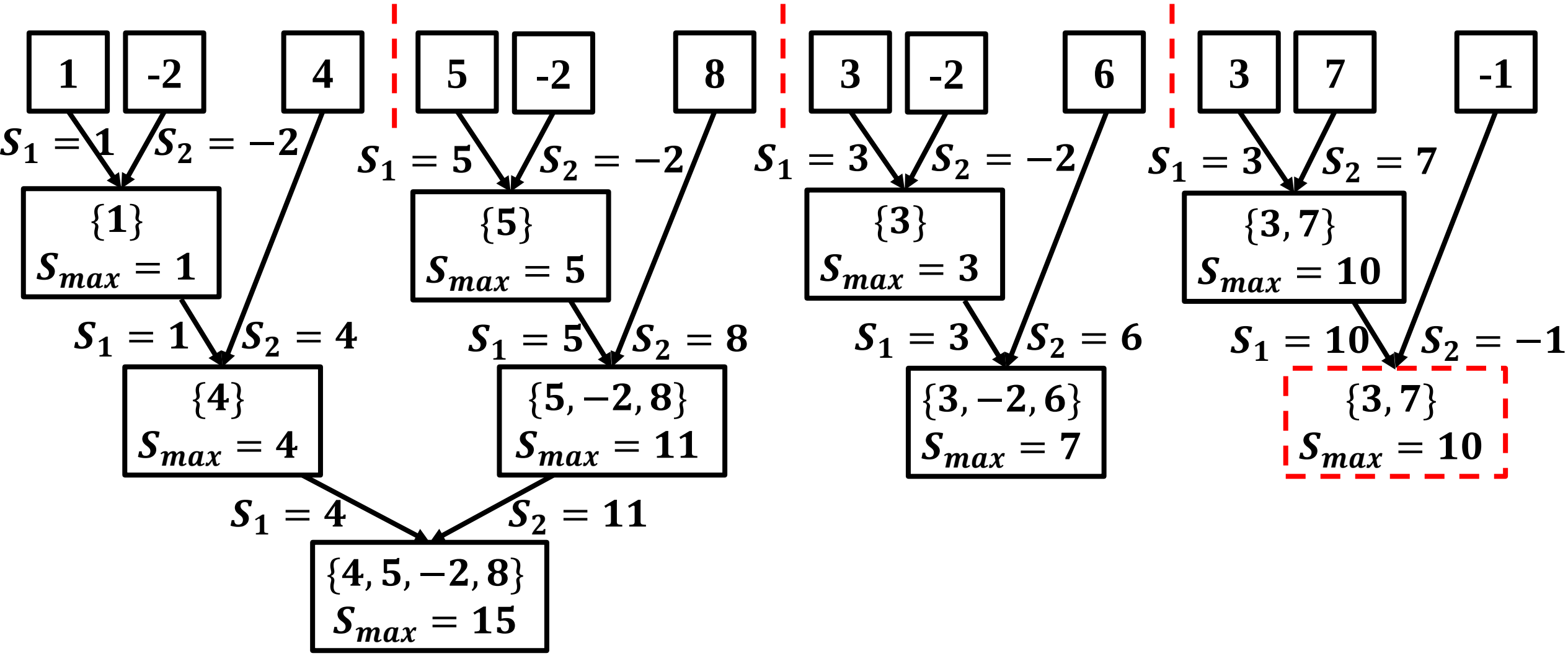
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3\}$
$S_{max} = 3$
$\{3, 7\}$
$S_{max} = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
$S_1 = 10$
$S_2 = -1$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$\{3, -2, 6\}$
$S_{max} = 7$
$\{3, 7, -1\}$
$S_3 = 9$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

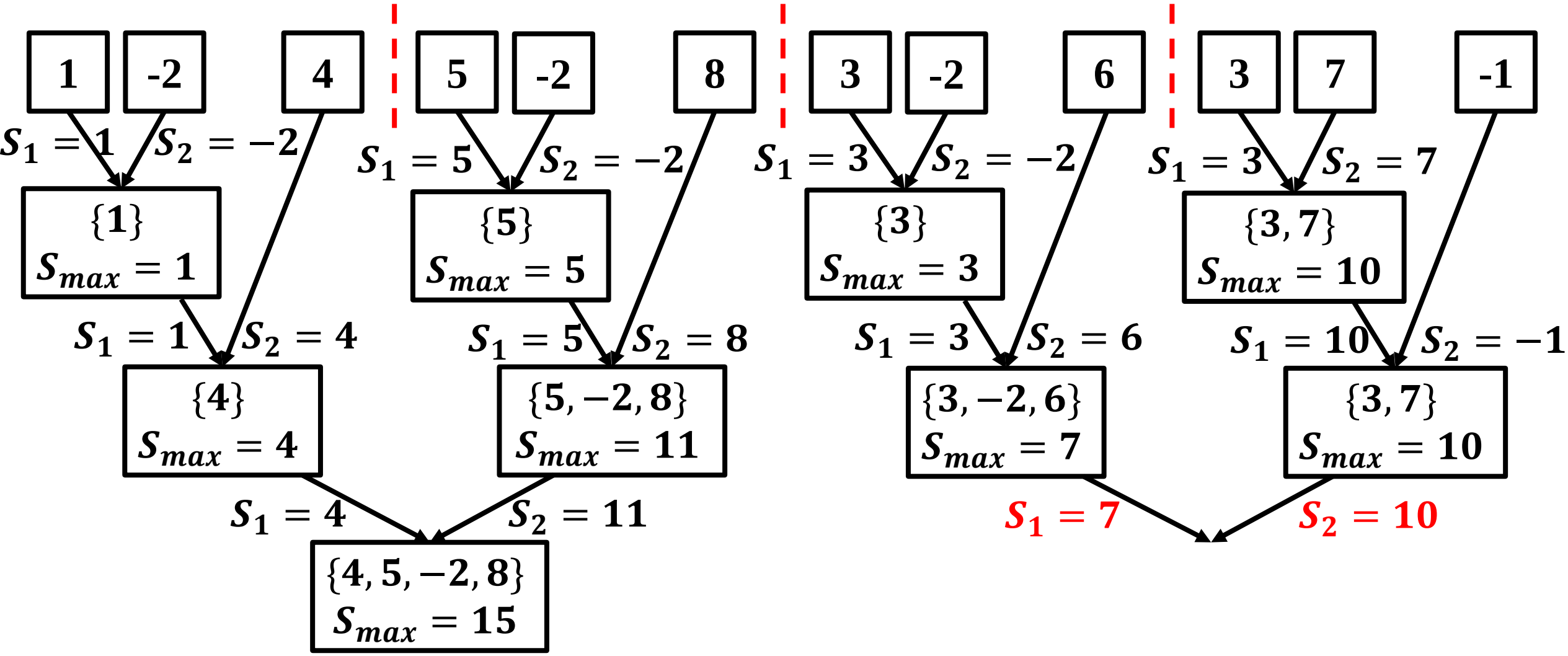
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
$\{1\}$
$S_{max} = 1$
$\{5\}$
$S_{max} = 5$
$\{3\}$
$S_{max} = 3$
$\{3, 7\}$
$S_{max} = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
$S_1 = 10$
$S_2 = -1$
$\{4\}$
$S_{max} = 4$
$\{5, -2, 8\}$
$S_{max} = 11$
$\{3, -2, 6\}$
$S_{max} = 7$
$\{3, 7, -1\}$
$S_3 = 9$
$S_1 = 4$
$S_2 = 11$
$\{4, 5, -2, 8\}$
$S_{max} = 15$

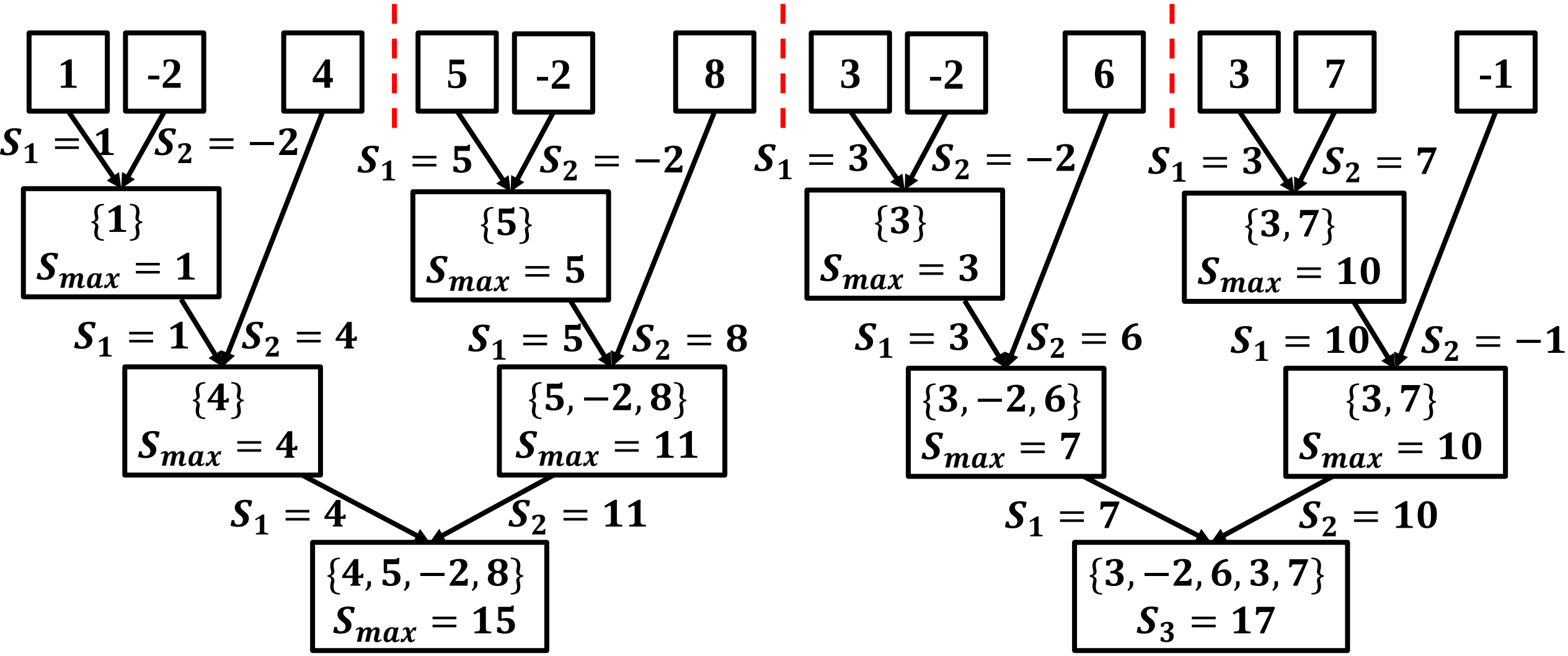
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
{4, 5, −2, 8}
$S_{max} = 15$

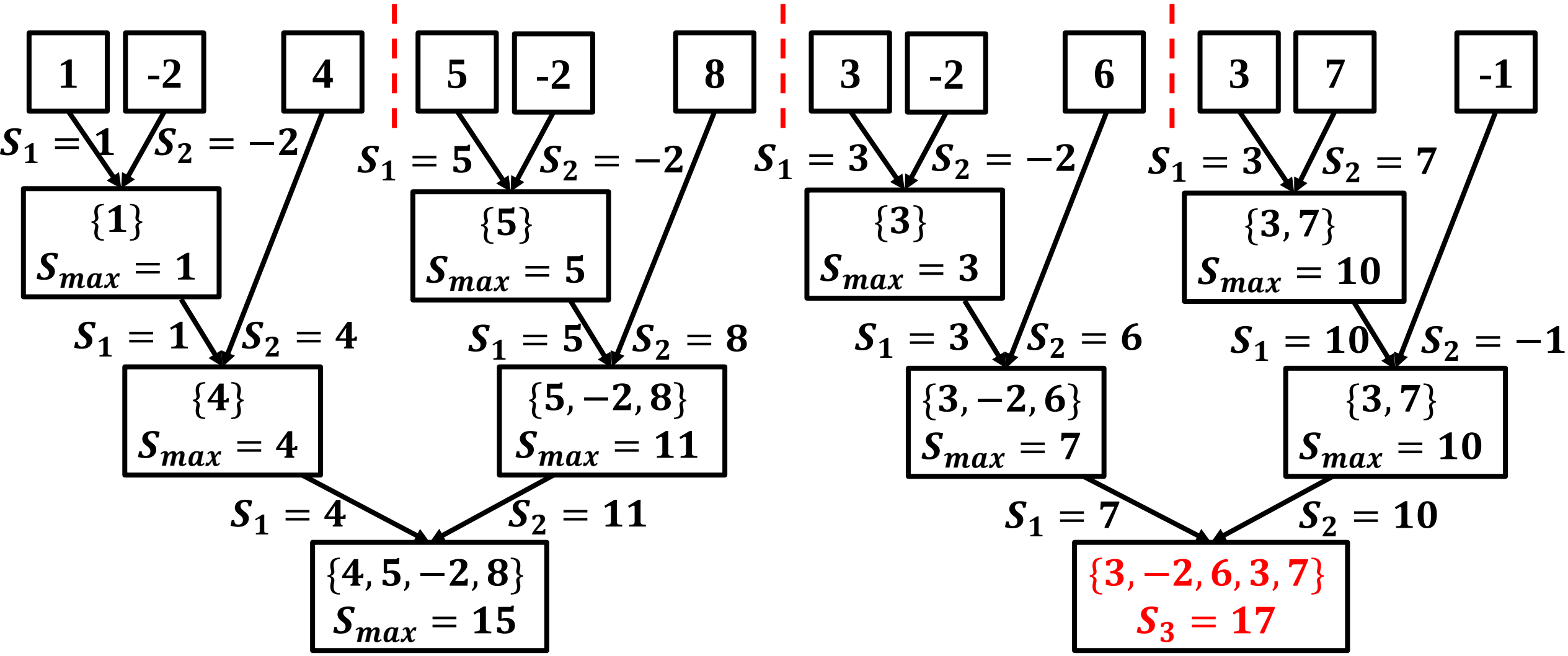
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$
$S_2 = -2$
$S_1 = 5$
$S_2 = -2$
$S_1 = 3$
$S_2 = -2$
$S_1 = 3$
$S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$
$S_2 = 4$
$S_1 = 5$
$S_2 = 8$
$S_1 = 3$
$S_2 = 6$
$S_1 = 10$
$S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$
$S_2 = 11$
$S_1 = 7$
$S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$

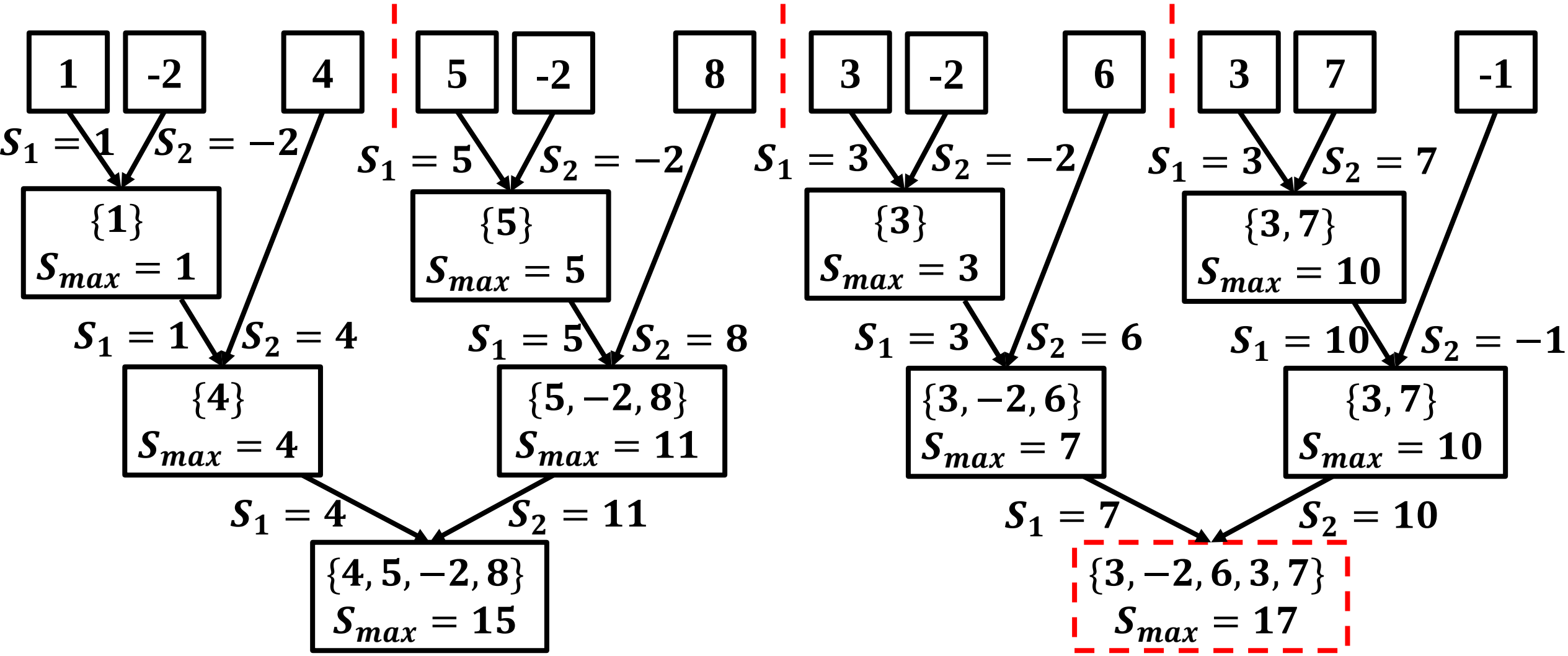
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_3 = 17$

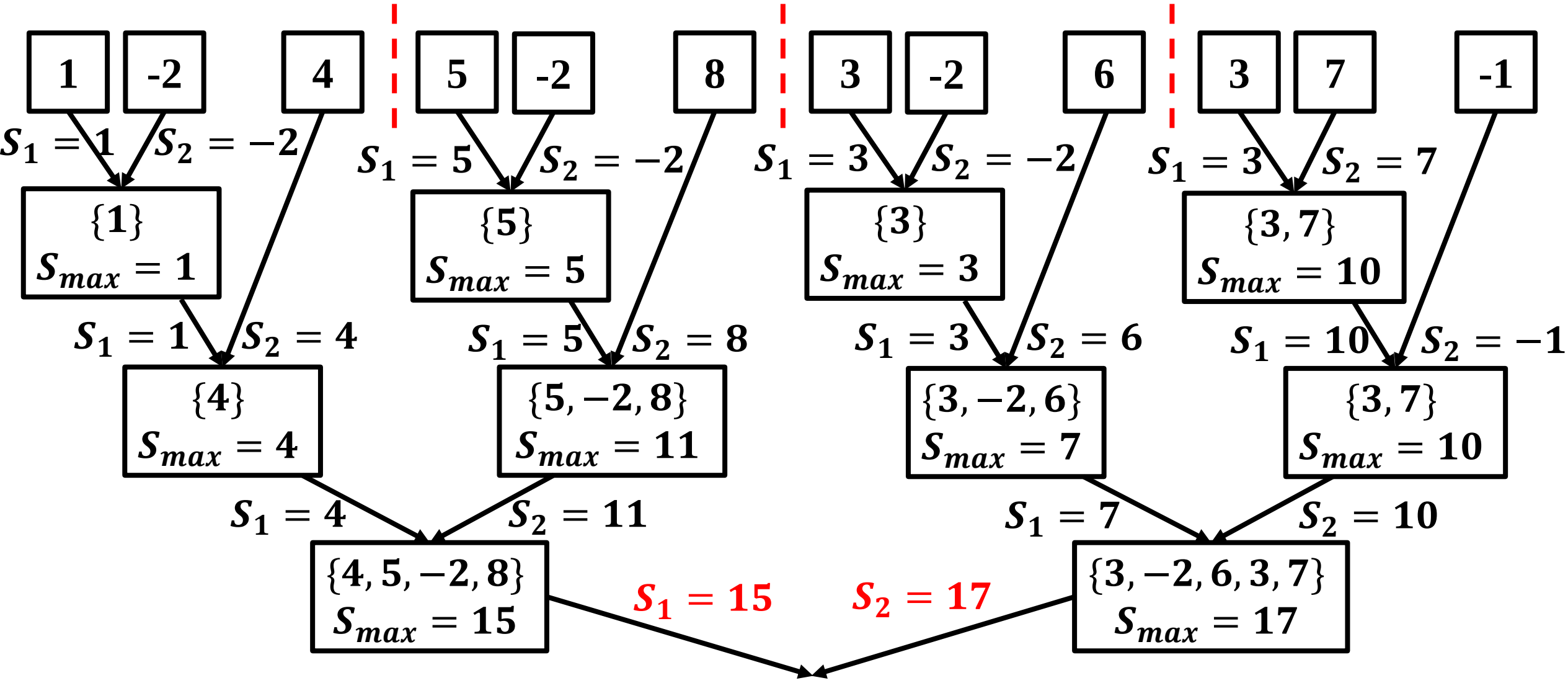
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_3 = 17$

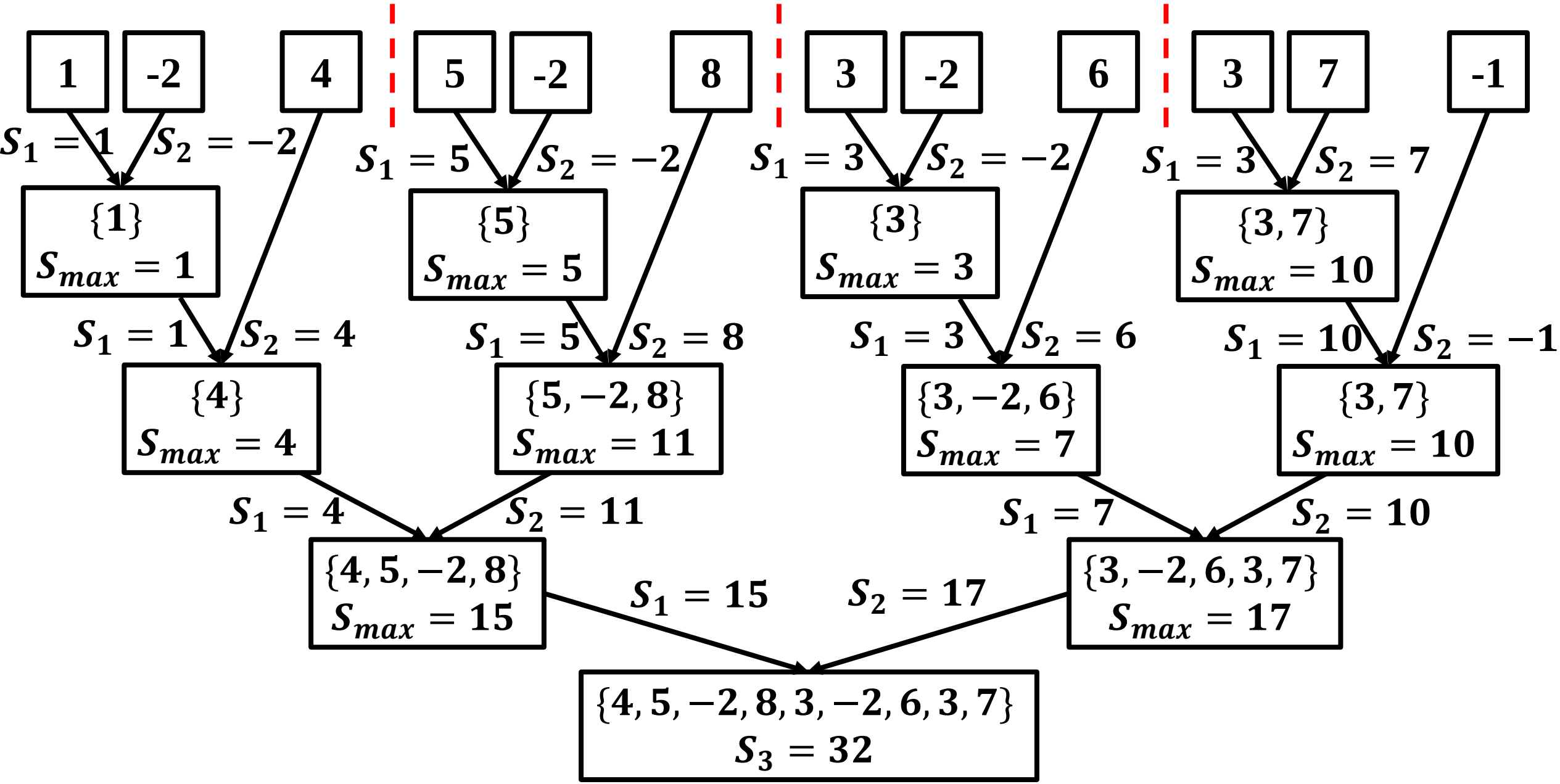
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_{max} = 17$

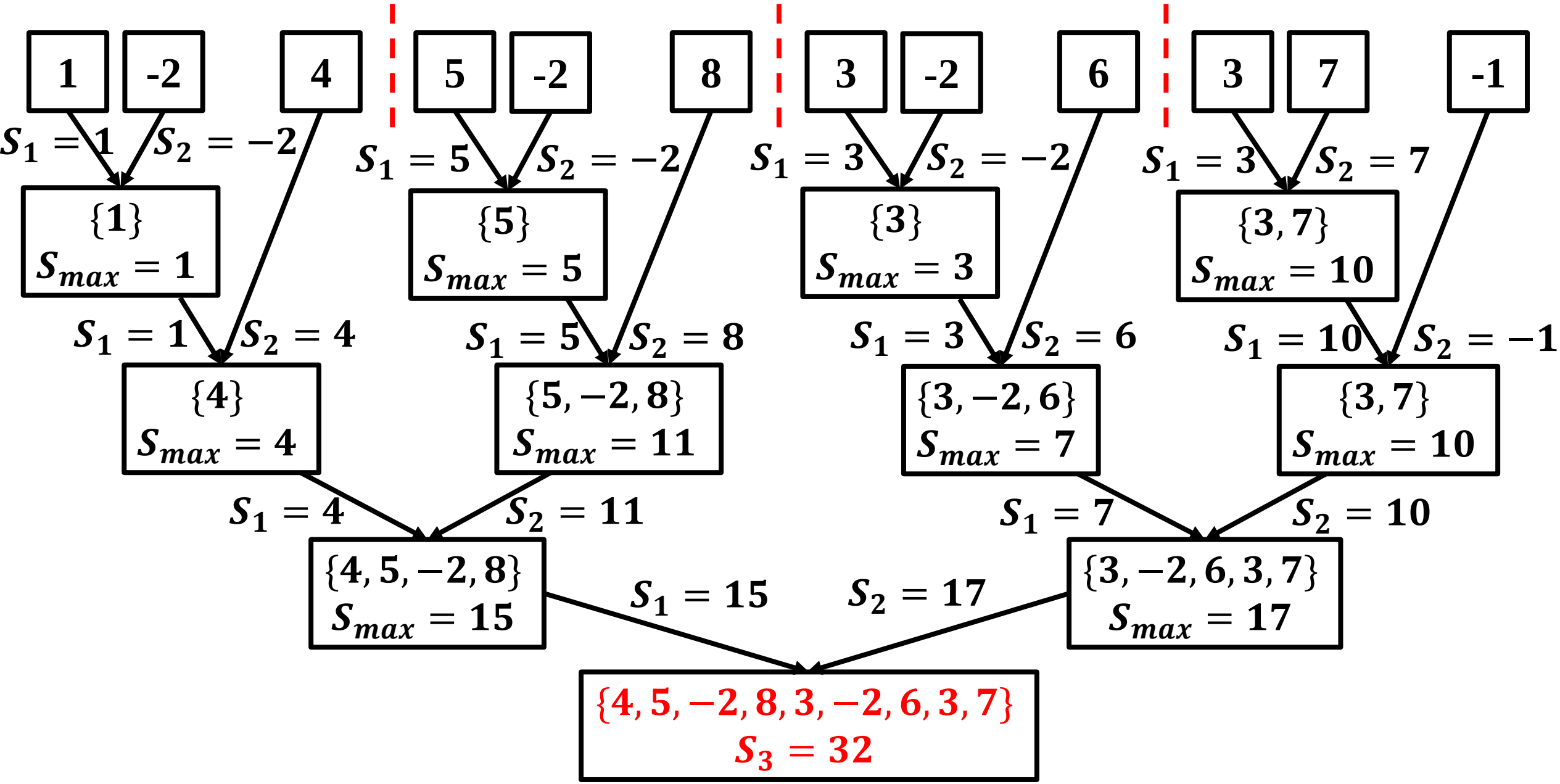
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_{max} = 17$
$S_1 = 15$
$S_2 = 17$

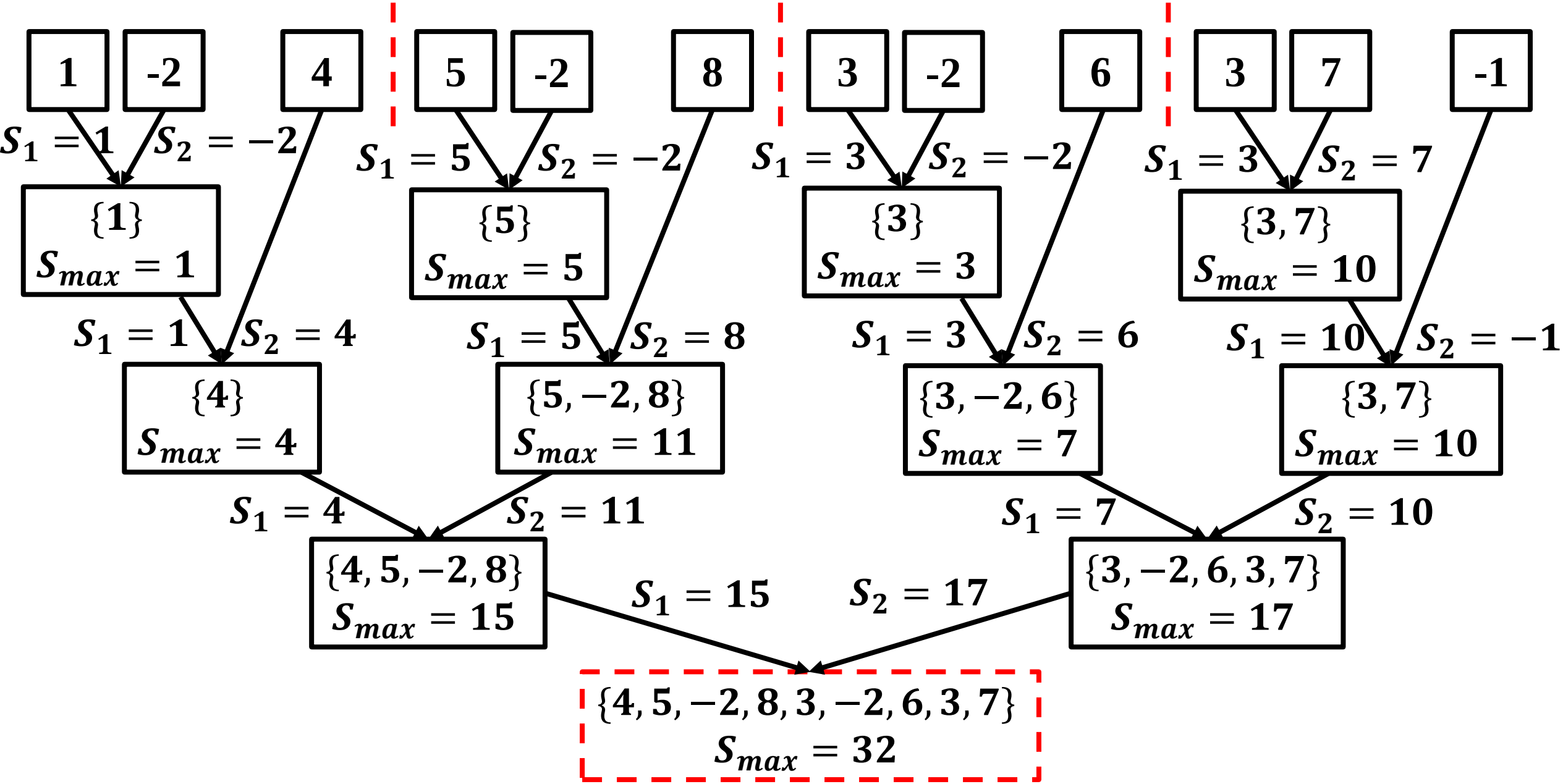
算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_{max} = 17$
$S_1 = 15$ $S_2 = 17$
{4, 5, −2, 8, 3, −2, 6, 3, 7}
$S_3 = 32$

算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_{max} = 17$
$S_1 = 15$ $S_2 = 17$
{4, 5, −2, 8, 3, −2, 6, 3, 7}
$S_3 = 32$

算法实例


1
-2
4
5
-2
8
3
-2
6
3
7
-1
$S_1 = 1$ $S_2 = -2$
$S_1 = 5$ $S_2 = -2$
$S_1 = 3$ $S_2 = -2$
$S_1 = 3$ $S_2 = 7$
{1}
$S_{max} = 1$
{5}
$S_{max} = 5$
{3}
$S_{max} = 3$
{3, 7}
$S_{max} = 10$
$S_1 = 1$ $S_2 = 4$
$S_1 = 5$ $S_2 = 8$
$S_1 = 3$ $S_2 = 6$
$S_1 = 10$ $S_2 = -1$
{4}
$S_{max} = 4$
{5, −2, 8}
$S_{max} = 11$
{3, −2, 6}
$S_{max} = 7$
{3, 7}
$S_{max} = 10$
$S_1 = 4$ $S_2 = 11$
$S_1 = 7$ $S_2 = 10$
{4, 5, −2, 8}
$S_{max} = 15$
{3, −2, 6, 3, 7}
$S_{max} = 17$
$S_1 = 15$ $S_2 = 17$
{4, 5, −2, 8, 3, −2, 6, 3, 7}
$S_{max} = 32$

伪代码

- **MaxSubArray($X, low, high$)**

输入: 数组X, 数组下标$low, high$
输出: 最大子数组之和S_{max}

```
if low = high then
    return X[low]
end
else
    mid ← ⌊(low+high)/2⌋
    S1 ← MaxSubArray(X,low,mid)
    S2 ← MaxSubArray(X,mid+1,high)
    S3 ← CrossingSubArray(X,low,mid,high)
    Smax ← max{S1, S2, S3}
    return Smax
end
```

递归终止条件

伪代码

- **MaxSubArray($X, low, high$)**

输入: 数组X, 数组下标$low, high$
输出: 最大子数组之和S_{max}
if $low = high$ then
  return $X[low]$
end
else
  $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
  $S_1 \leftarrow$ MaxSubArray(X,low,mid)
  $S_2 \leftarrow$ MaxSubArray(X,mid+1,high)
  $S_3 \leftarrow$ CrossingSubArray(X,low,mid,high)
  $S_{max} \leftarrow \max\{S_1, S_2, S_3\}$
  return S_{max}
end

拆分原问题

伪代码

- **MaxSubArray($X, low, high$)**

输入: 数组X, 数组下标$low, high$
输出: 最大子数组之和S_{max}
if $low = high$ then
  return $X[low]$
end
else
  $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
  $S_1 \leftarrow$ MaxSubArray(X,low,mid)
  $S_2 \leftarrow$ MaxSubArray(X,mid+1,high)
  $S_3 \leftarrow$ CrossingSubArray(X,low,mid,high)
  $S_{max} \leftarrow \max\{S_1, S_2, S_3\}$
  return S_{max}
end

求解子问题

伪代码

- **MaxSubArray($X, low, high$)**

输入: 数组X, 数组下标$low, high$
输出: 最大子数组之和S_{max}
if $low = high$ **then**
  return $X[low]$
end
else
  $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
  $S_1 \leftarrow$ MaxSubArray(X,low,mid)
  $S_2 \leftarrow$ MaxSubArray(X,mid+1,high)
  $S_3 \leftarrow$ CrossingSubArray(X,low,mid,high)
  $S_{max} \leftarrow \max\{S_1, S_2, S_3\}$
  return S_{max}
end

合并问题解

伪代码

- **MaxSubArray($X, low, high$)**

初始调用：MaxSubArray($X, 1, n$)

输入: 数组X, 数组下标$low, high$
输出: 最大子数组之和S_{max}
if $low = high$ **then**
  return $X[low]$
end
else
  $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
  $S_1 \leftarrow$ MaxSubArray(X,low,mid)
  $S_2 \leftarrow$ MaxSubArray(X,mid+1,high)
  $S_3 \leftarrow$ CrossingSubArray(X,low,mid,high)
  $S_{max} \leftarrow \max\{S_1, S_2, S_3\}$
  return S_{max}
end

时间复杂度分析

- 输入规模为：n

$$T(n)=\begin{cases}1, & n=1\\ 2\cdot T(n/2)+O(n), & n>1\end{cases}$$

输入: 数组X, 数组下标$low, high$
输出: 最大子数组之和S_{max}

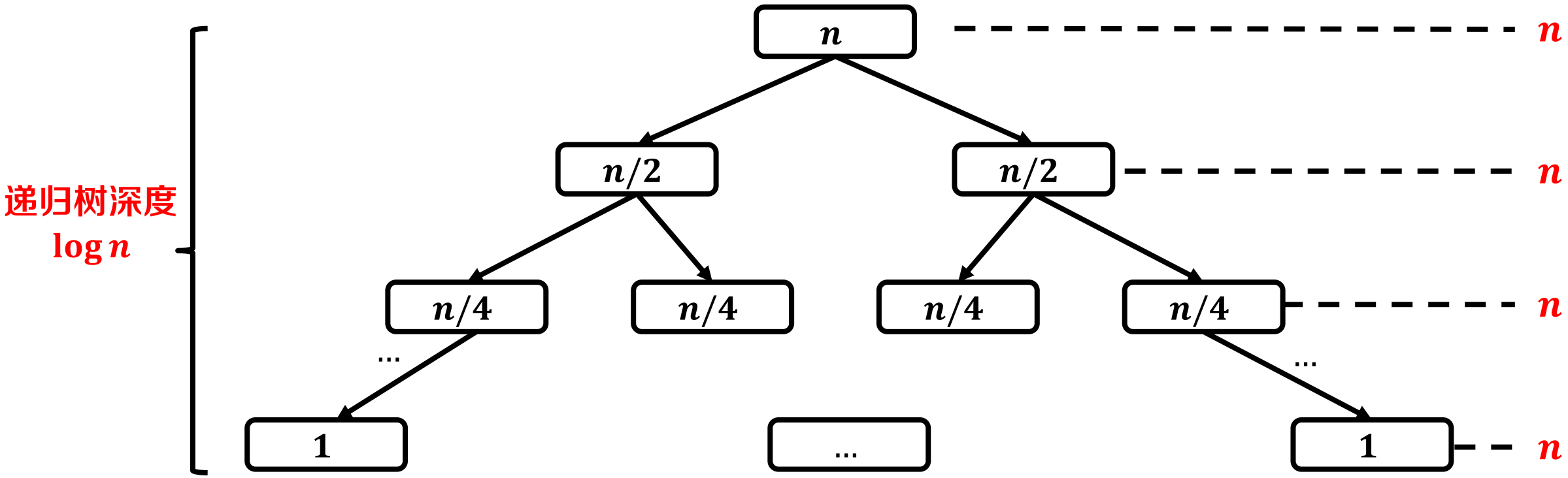
```
if low = high then
    return X[low]
end
else
    mid ← ⌊(low+high)/2⌋
    S1 ← MaxSubArray(X,low,mid)
    S2 ← MaxSubArray(X,mid+1,high)
    S3 ← CrossingSubArray(X,low,mid,high)
    Smax ← max{S1, S2, S3}
    return Smax
end
```

$2\cdot T(n/2)$

$O(n)$

时间复杂度分析

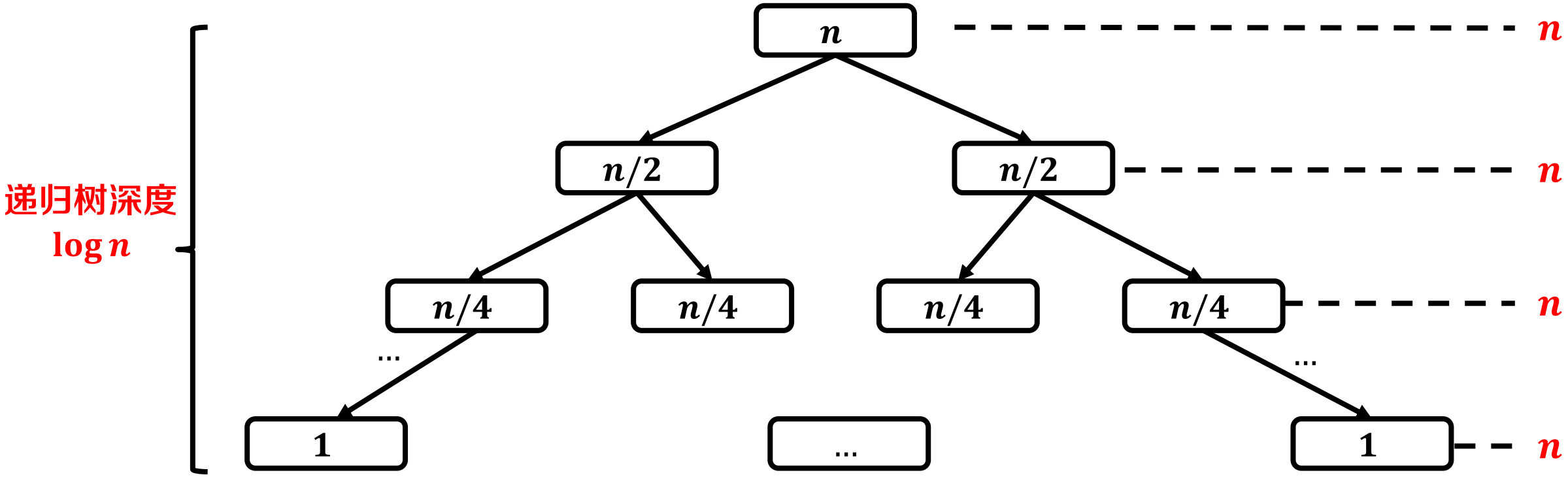
- 输入规模为：n
  - $T(n) = \begin{cases} 1, & n = 1 \\ 2 \cdot T(n/2) + O(n), & n > 1 \end{cases}$



时间复杂度分析

- 输入规模为：n
  - $$T(n)=\begin{cases}1, & n=1\\ 2\cdot T(n/2)+O(n), & n>1\end{cases}$$

$T(n)=n\log n$



小结

- 本节讲述了三种不同的算法，用于解决最大子数组和问题：

算法名称	时间复杂度
蛮力枚举	$O(n^3)$
优化枚举	$O(n^2)$
分而治之	$O(n\log n)$

小结

- 本节讲述了三种不同的算法，用于解决最大子数组和问题：

算法名称	时间复杂度
蛮力枚举	$O(n^3)$
优化枚举	$O(n^2)$
分而治之	$O(n\log n)$
?	$O(n)$

问题：是否可以设计一个时间复杂度为$O(n)$的算法?

小结

- 本节讲述了三种不同的算法，用于解决最大子数组和问题：

算法名称	时间复杂度
蛮力枚举	$O(n^3)$
优化枚举	$O(n^2)$
分而治之	$O(n\log n)$
动态规划	$O(n)$

问题：是否可以设计一个时间复杂度为$O(n)$的算法？

答案：动态规划！