动态规划篇:最大子数组问题II

童咏昕

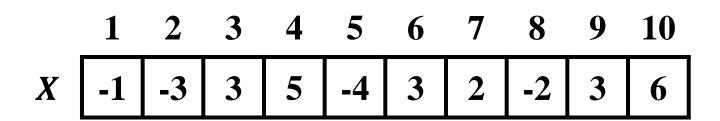
北京航空航天大学 计算机学院

中国大学MOOC北航《算法设计与分析》



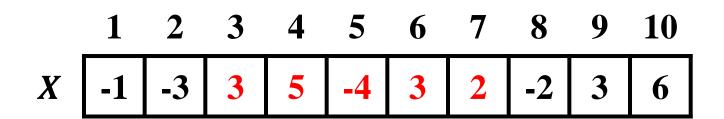
										10
X	-1	-3	3	5	-4	3	2	-2	3	6





• 子数组为数组*X*中连续的一段序列





• 子数组X[3..7]



- 子数组X[3..7]
 - 求和为: 3+5-4+3+2=9



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



- 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..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]



- 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



- 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?



- 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



• 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 ($l \le r$)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}

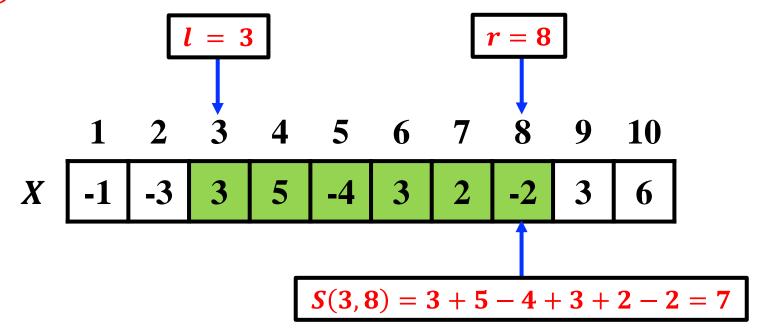


- 数组X[1..n],其所有的下标 $l, r(l \le r)$ 组合分为以下两种情况
 - 当l = r时,一共 $C_n^1 = n$ 种组合
 - 当l < r时,一共 C_n^2 种组合
- 枚举 $n + C_n^2$ 种下标l, r组合,求出最大子数组之和

蛮力枚举



- l = 3, r = 8
- 计算S(3,8):



蛮力枚举

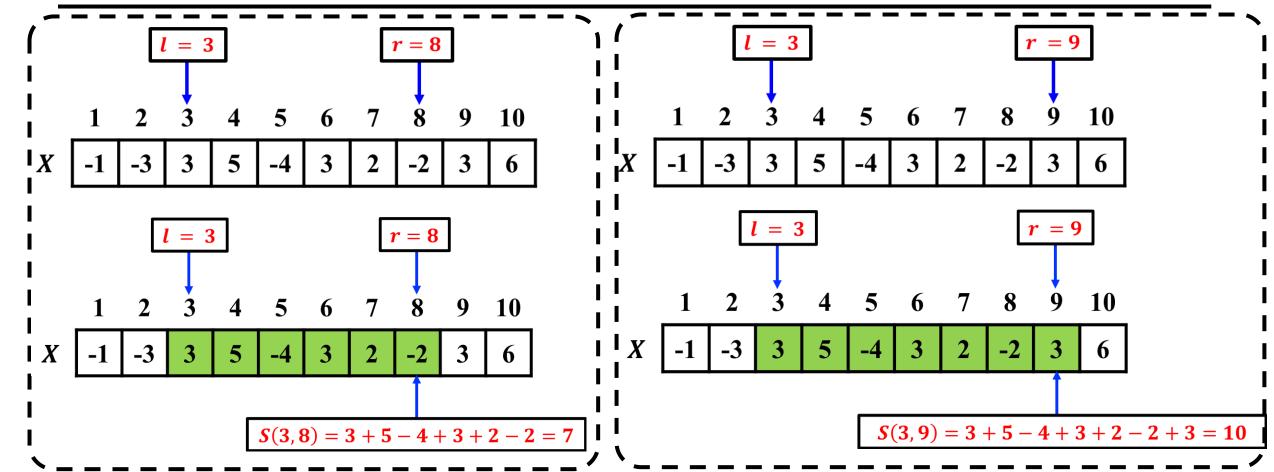


• 枚举 $n + C_n^2$ 种可能的区间 $[l,r](l \le r)$,求解最大子数组之和 S_{max}

```
输入: 数组X[1..n]
输出: 最大子数组之和S_{max}
S_{max} \leftarrow -\infty
for l \leftarrow 1 to n do
    for r \leftarrow l \ to \ n \ do
        S(l,r) \leftarrow 0
       for i \leftarrow l \ to \ r \ do
         S(l,r) \leftarrow S(l,r) + X[i]
        end
       S_{max} \leftarrow \max\{S_{max}, S(l, r)\}
    end
end
                                                      时间复杂度: O(n^3)
return S_{max}
```

从蛮力枚举到优化枚举



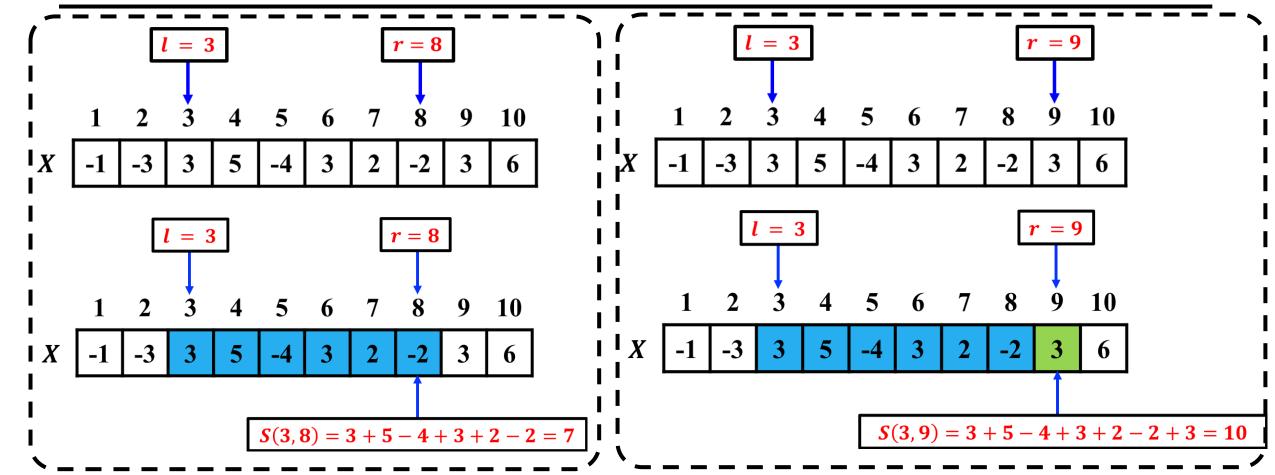


计算S(3,8)循环6次

计算S(3,9)循环7次

从蛮力枚举到优化枚举



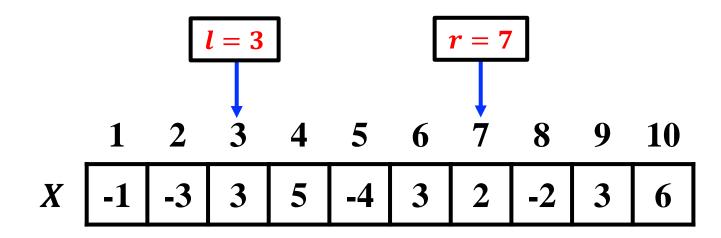


计算S(3,8)循环6次

计算S(3,9)循环7次

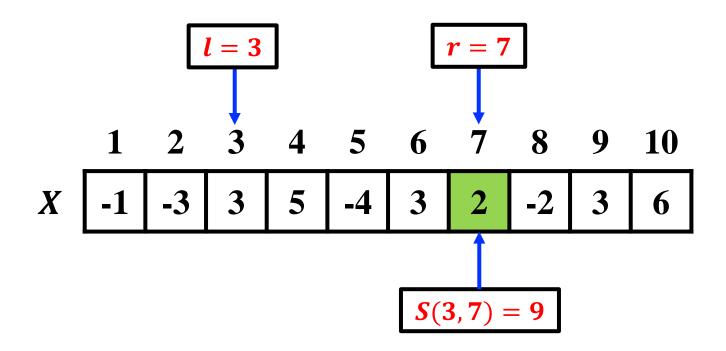


• l = 3, r = 7



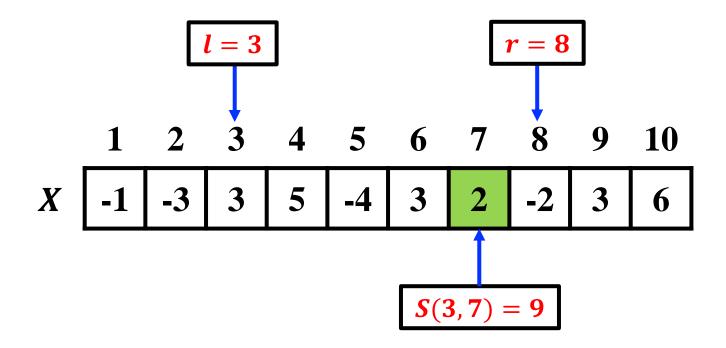


• l = 3, r = 7, S(3,7) = 9



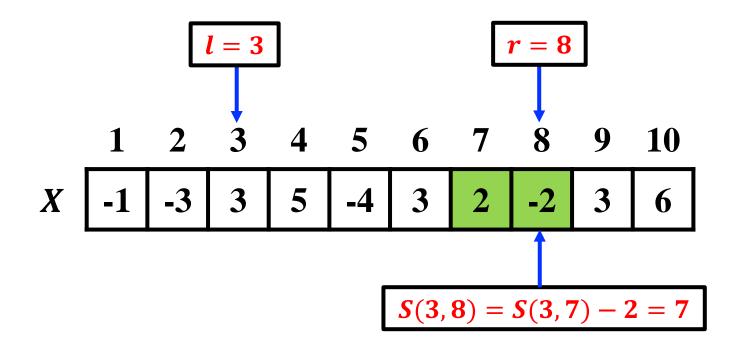


• l = 3, r = 8, S(3,8) = ?



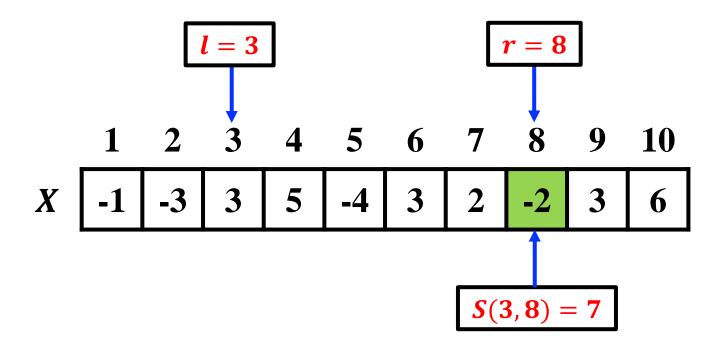


• l = 3, r = 8, S(3,8) = 7



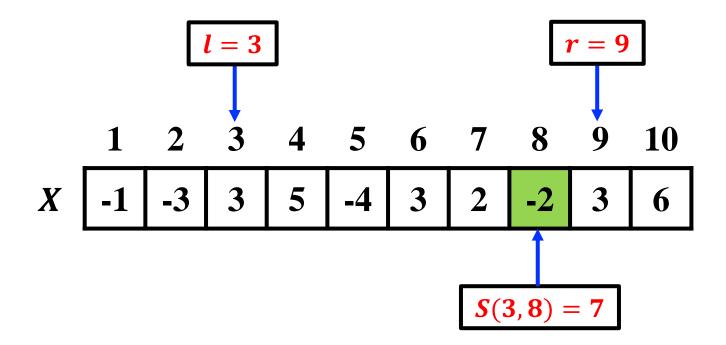


• l = 3, r = 8, S(3,8) = 7



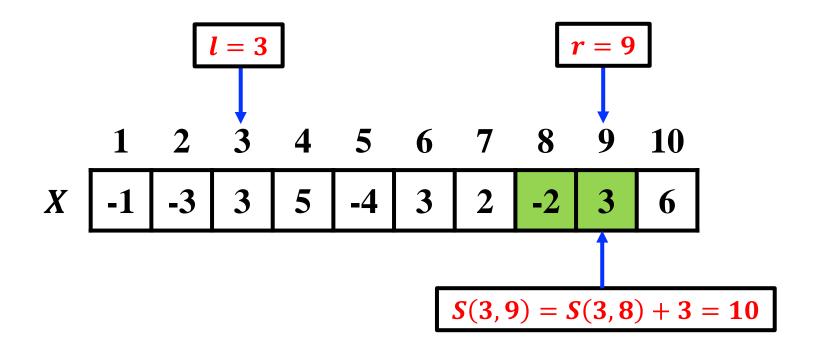


• l = 3, r = 9, S(3, 9) = ?





• l = 3, r = 9, S(3, 9) = ?



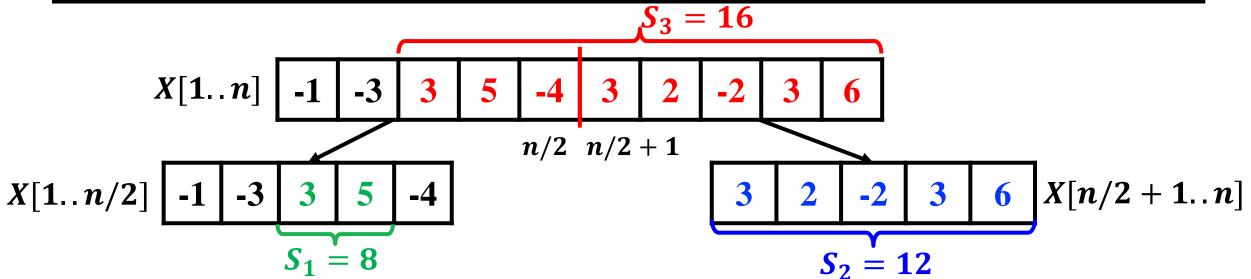


• 核心思想: $S(l,r) = \sum_{i=l}^{r} X[i] = S(l,r-1) + X[r]$

```
输入: 数组X[1..n]
输出: 最大子数组之和S_{max}
S_{max} \leftarrow -\infty
for l \leftarrow 1 to n do
    S \leftarrow 0
    for r \leftarrow l \ to \ n \ do
    S \leftarrow S + X[r]
S_{max} \leftarrow \max\{S_{max}, S\}
    end
end
                                                              时间复杂度: O(n^2)
return S_{max}
```

分而治之





• 将数组X[1..n]分为X[1..n/2]和X[n/2 + 1..n]

分解原问题

- 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 $S_{max} = max\{S_1, S_2, S_3\}$

解决子问题

合井问题解

分而治之

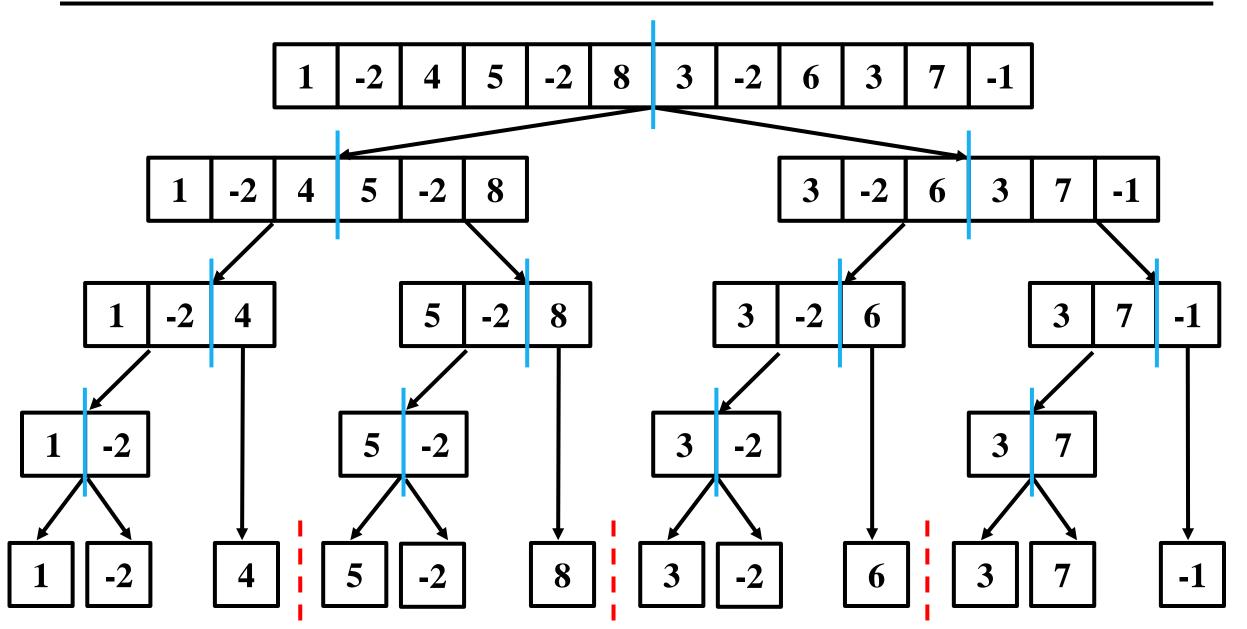


• 时间复杂度分析

```
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 \text{MaxSubArray}(X,\text{low,mid})
    S_2 \leftarrow \text{MaxSubArray}(X,\text{mid}+1,\text{high})
    S_3 \leftarrow \text{CrossingSubArray}(X,\text{low,mid,high})
    S_{max} \leftarrow \max\{S_1, S_2, S_3\}
    return S_{max}
end
```

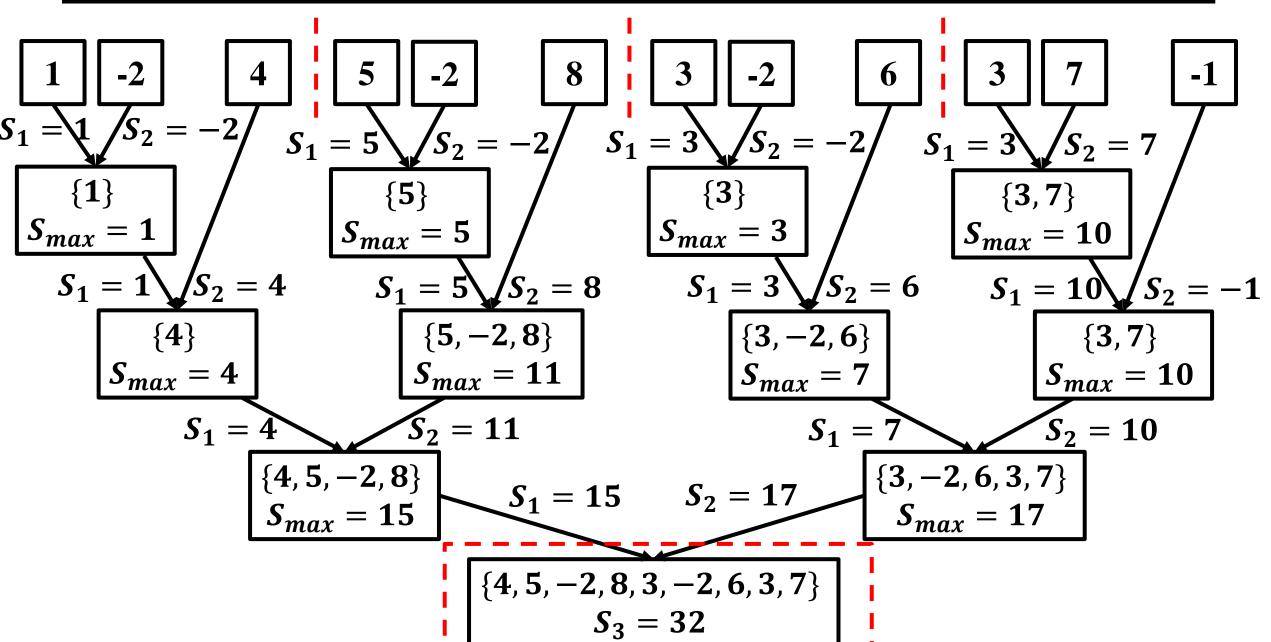
时间复杂度: $O(n \log n)$





算法实例







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)

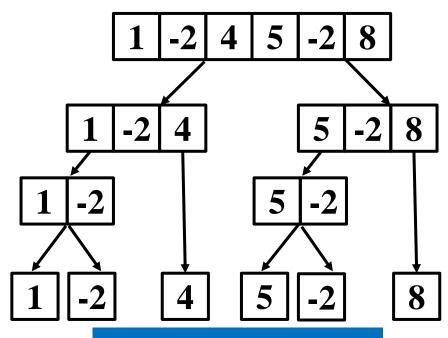
问题:是否可以设计一个时间复杂度为O(n)的算法?



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)



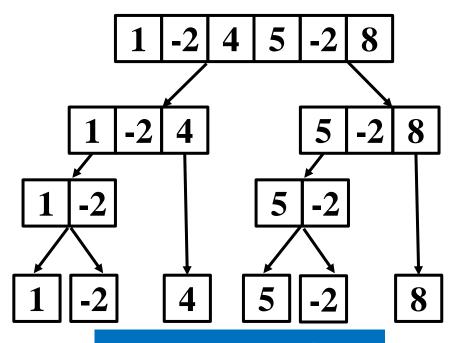
算法名称	时间复杂度
蛮力枚举	$O(n^3)$
优化枚举	$O(n^2)$
分而治之	$O(n \log n)$
动态规划	O(n)



子问题相互独立

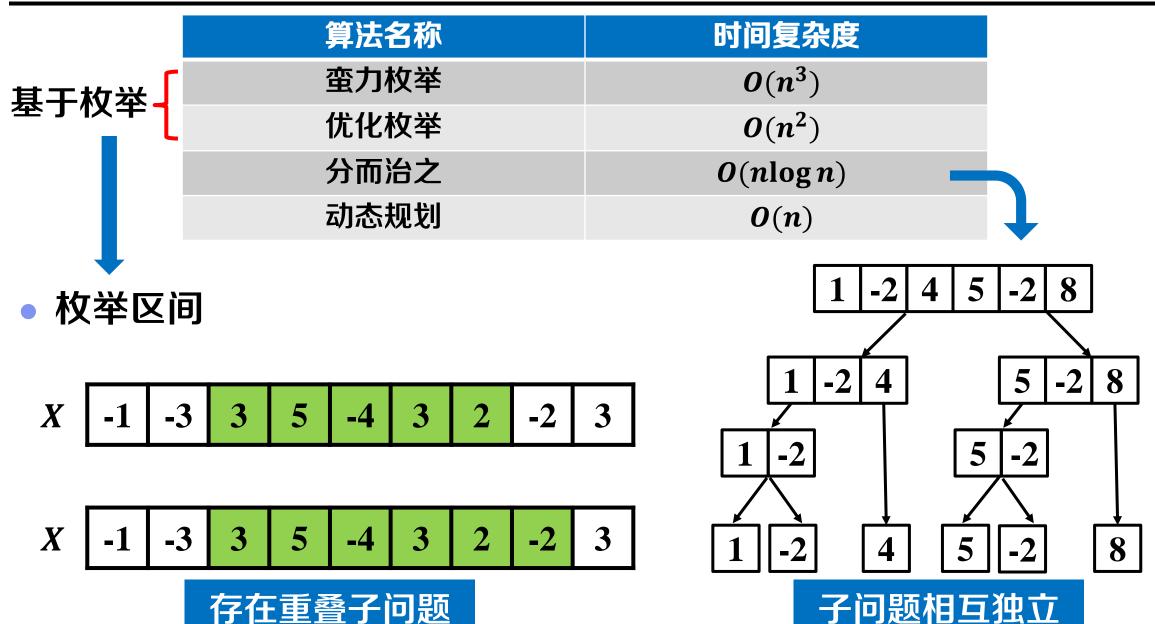


算法名称	时间复杂度
蛮力枚举	$O(n^3)$
优化枚举	$O(n^2)$
分而治之	$O(n \log n)$
动态规划	O(n)



子问题相互独立





枚举过程分析



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

枚举过程分析



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



当前最大值= 4, S(i,j) = 4



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



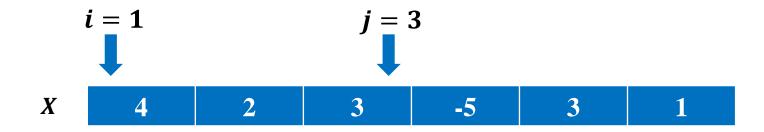
当前最大值= 6, S(i,j) = 6



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



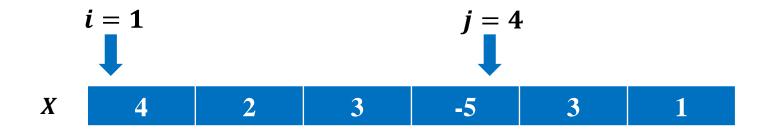
当前最大值= 9, S(i,j) = 9



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



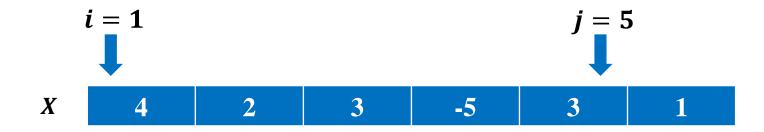
当前最大值= 9, S(i,j) = 4



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



当前最大值= 9, S(i,j) = 7



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



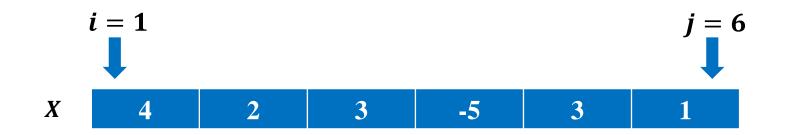
当前最大值= 9, S(i, j) = 8



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾



当前最大值=9

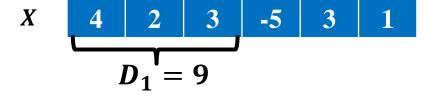
 D_i : 以X[i]开头的最大子数组



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置*j*作为区间结尾





当前最大值=9

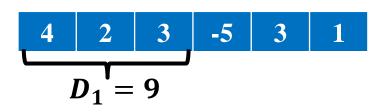


• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

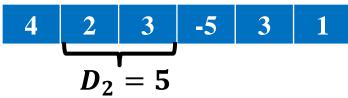
• 第2层: 枚举位置j作为区间结尾





X

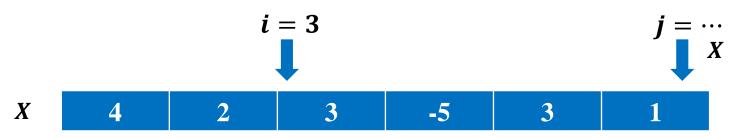
X

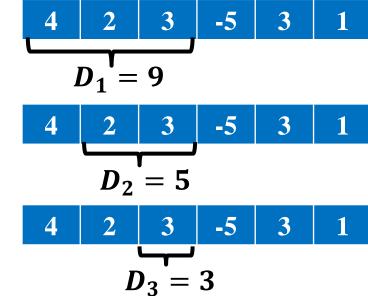




• 两层枚举

- 第1层: 枚举位置*i*作为区间开头
- 第2层: 枚举位置j作为区间结尾





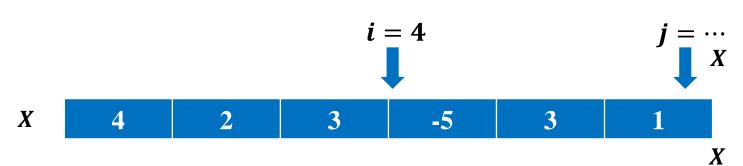
X

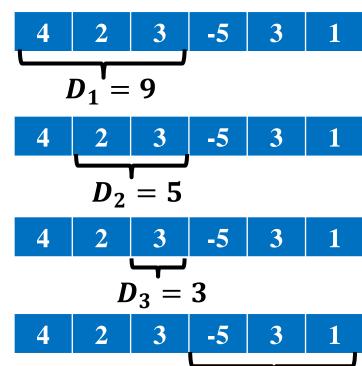
X



• 两层枚举

- 第1层: 枚举位置*i*作为区间开头
- 第2层: 枚举位置*j*作为区间结尾





X

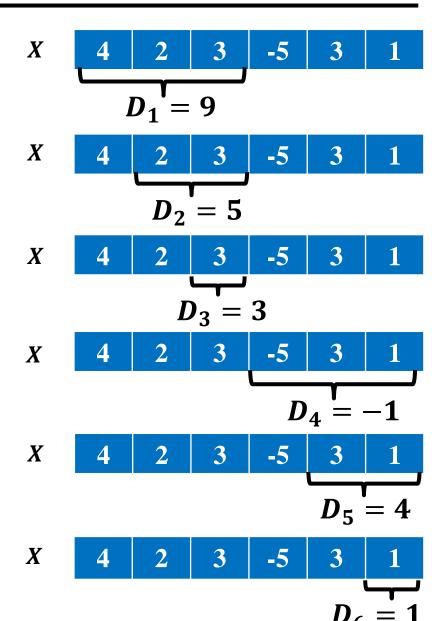
X



• 两层枚举

• 第1层: 枚举位置*i*作为区间开头

• 第2层: 枚举位置j作为区间结尾





• 两层枚举

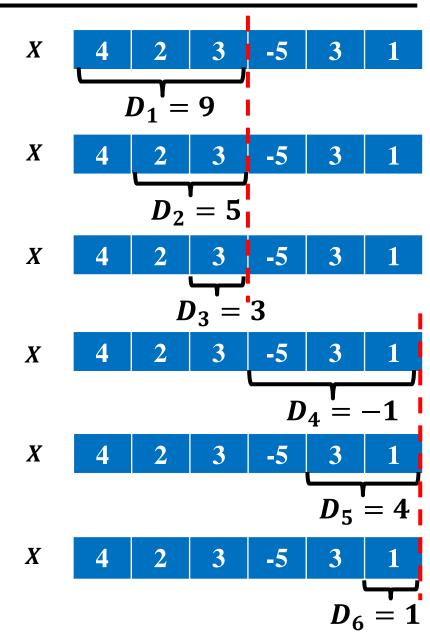
• 第1层: 枚举位置i作为区间开头

• 第2层: 枚举位置*j*作为区间结尾

• 观察 D_i

结尾相同: D₁, D₂, D₃

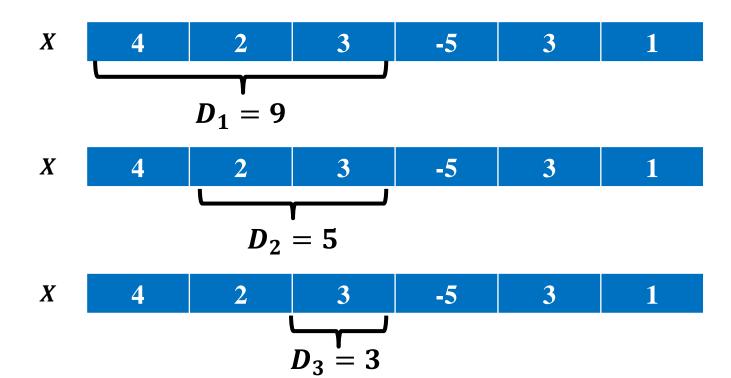
结尾不同: D₃, D₄

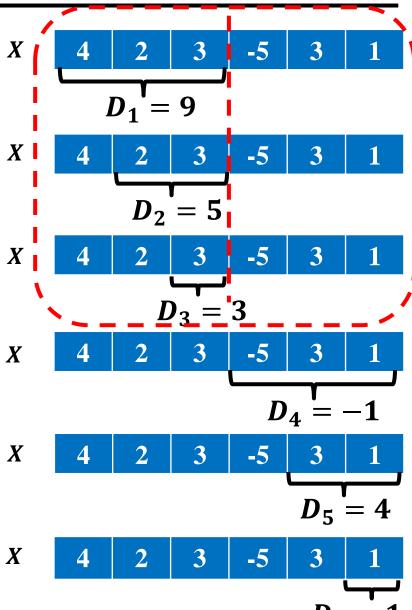




• 结尾位置相同

 $lackbox{0}$ D_1, D_2, D_3

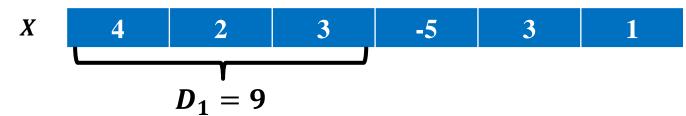


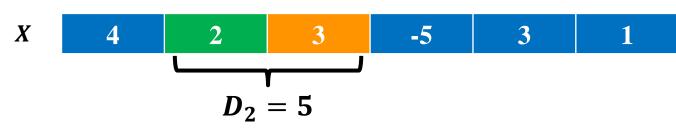


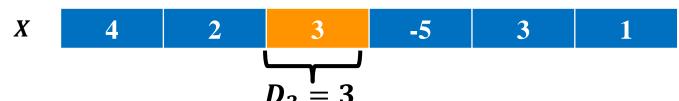


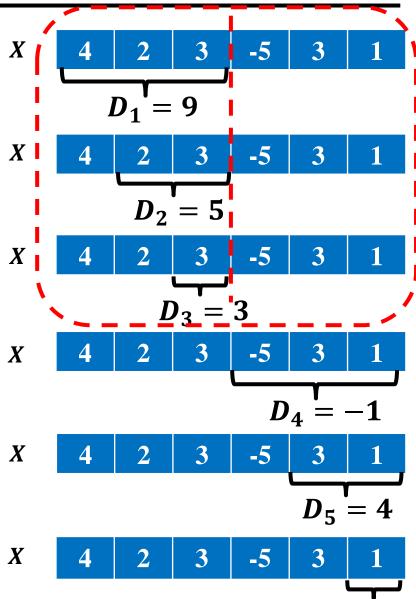
• 结尾位置相同

- $lackbox{0}{} D_1, D_2, D_3$
 - o $D_2 = X[2] + D_3$







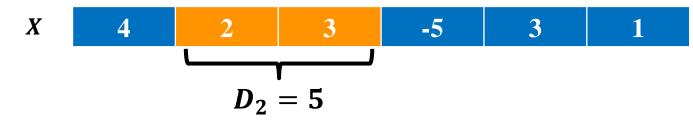


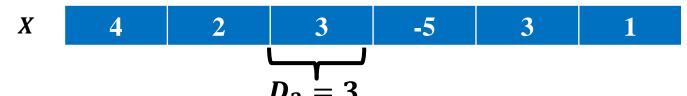


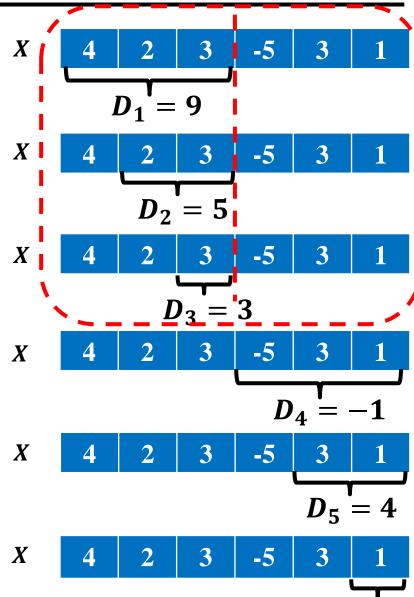
• 结尾位置相同

- $lackbox{0}{} D_1, D_2, D_3$
 - o $D_2 = X[2] + D_3$
 - $D_1 = X[1] + D_2$



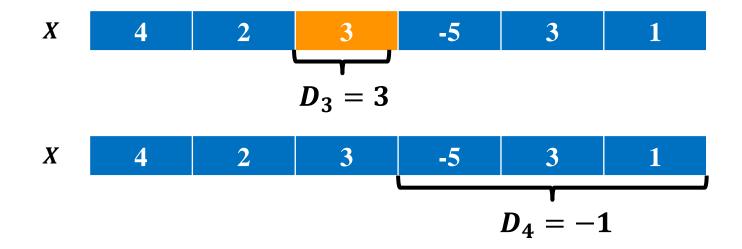


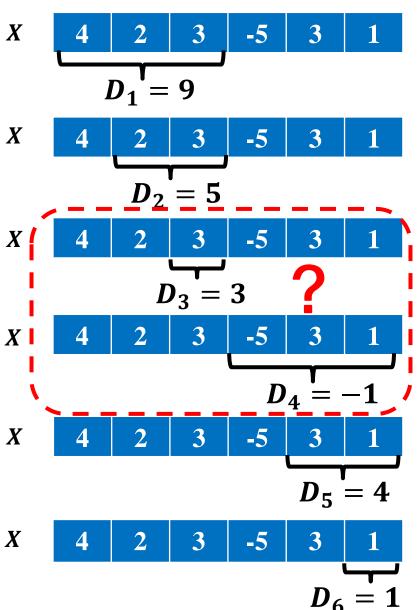






- 结尾位置不同
 - D_3, D_4
 - o $D_3 = X[3]$

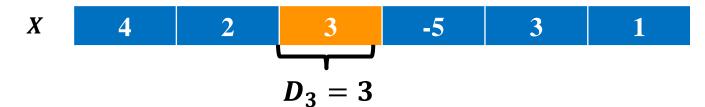


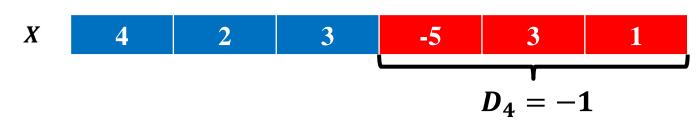


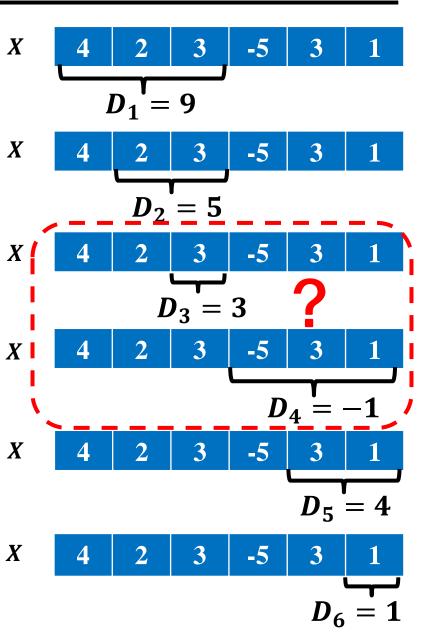


• 结尾位置不同

- $lackbox{0}{} D_3, D_4$
 - o $D_3 = X[3]$
 - $D_4 < 0$







规律描述



• D_i : 以X[i]开头的最大子数组和

• 情况1: $D_{i+1} > 0$ 时 X[1] ... X[i] X[i+1] ... X[n] D_{i+1} ... $D_{i} = X[i] + D_{i+1}$

规律描述



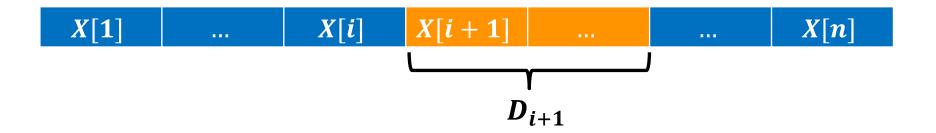
• D_i : 以X[i]开头的最大子数组和

• 情况1: $D_{i+1} > 0$ 时 X[1] ... X[i] X[i+1] ... X[n] D_{i+1} ... $D_{i} = X[i] + D_{i+1}$

• 情况2: $D_{i+1} \leq 0$ 时 X[1] ... X[i] X[i+1] ... X[n] $D_i = X[i]$

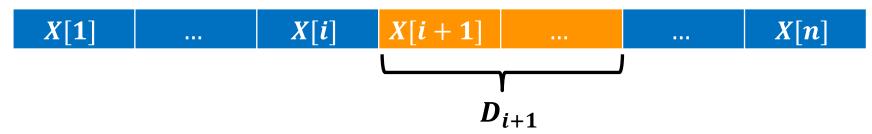


• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} > 0$)

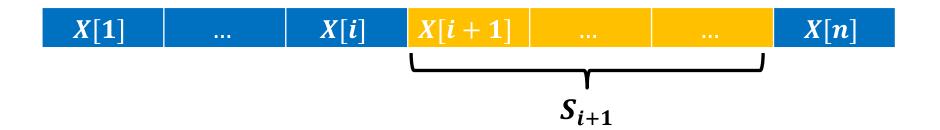




• D_{i+1} : 以X[i+1]开头的最大子数组和 $(D_{i+1} > 0)$

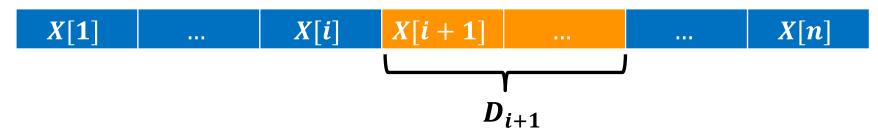


• S_{i+1} : 以X[i+1]开头的任一子数组和

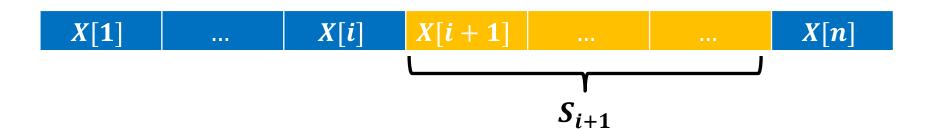




• D_{i+1} : 以X[i+1]开头的最大子数组和 $(D_{i+1} > 0)$

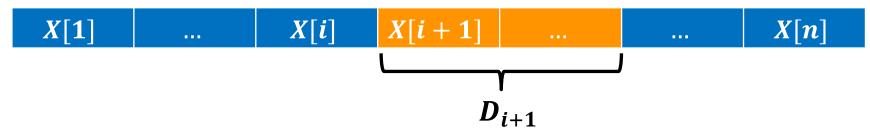


• S_{i+1} : 以X[i+1]开头的任一子数组和

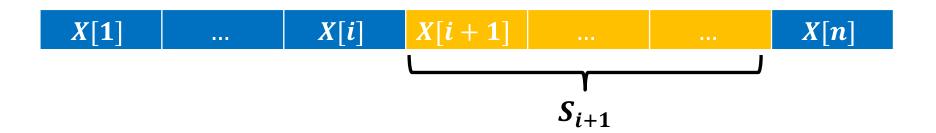




• D_{i+1} : 以X[i+1]开头的最大子数组和 $(D_{i+1} > 0)$

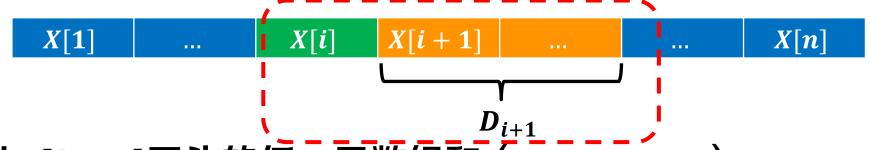


• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$

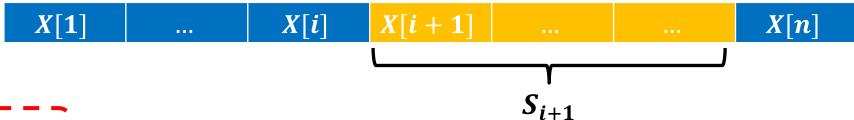




• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} > 0$)



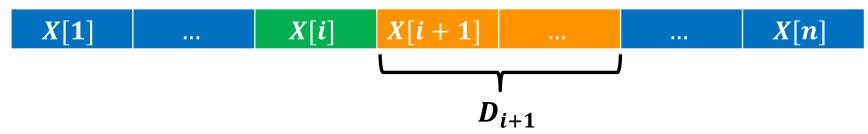
• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



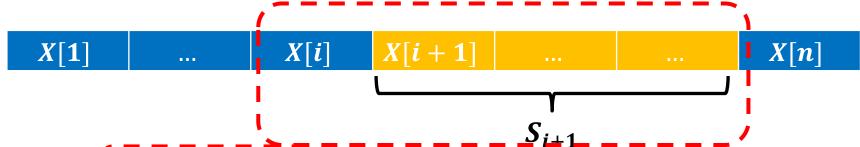
 $\bullet X[i] + D_{i+1}$



• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} > 0$)



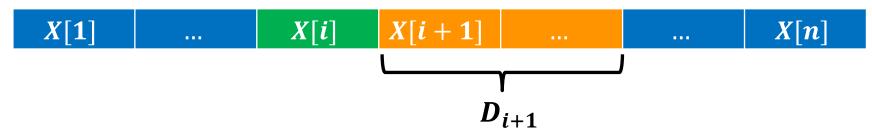
• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



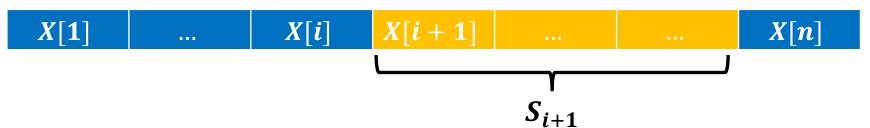
•
$$X[i] + D_{i+1} \quad X[i] + S_{i+1}$$



• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} > 0$)



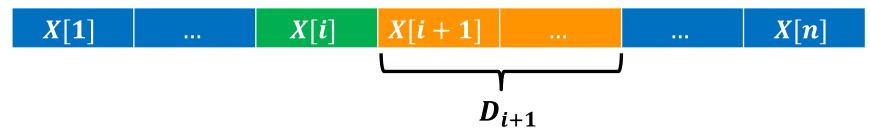
• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



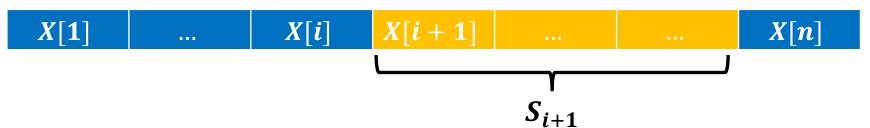
• $X[i] + D_{i+1} \ge X[i] + S_{i+1}$



• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} > 0$)



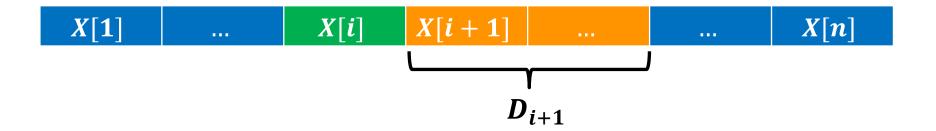
• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



- $X[i] + D_{i+1} \ge X[i] + S_{i+1}$
- $D_i = X[i] + D_{i+1}$

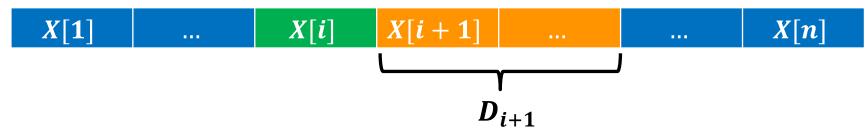


• D_{i+1} : 以X[i+1]开头的最大子数组和 $(D_{i+1} \leq 0)$

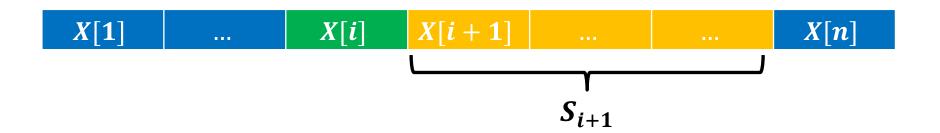




• D_{i+1} : 以X[i+1]开头的最大子数组和 $(D_{i+1} \leq 0)$

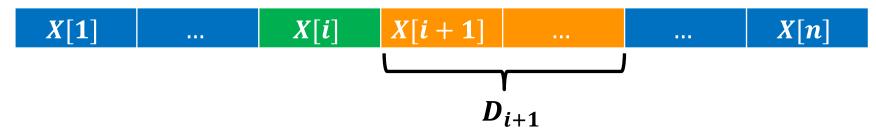


• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$

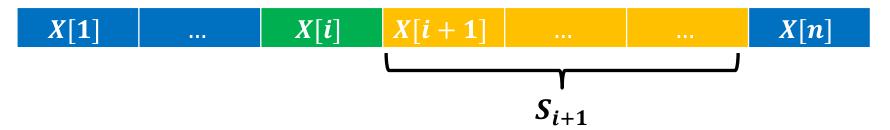




• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} \leq 0$)



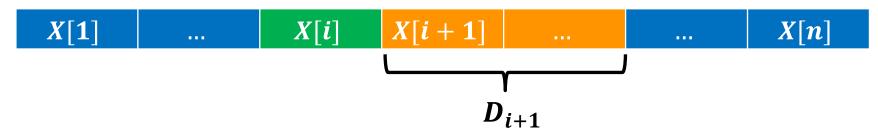
• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



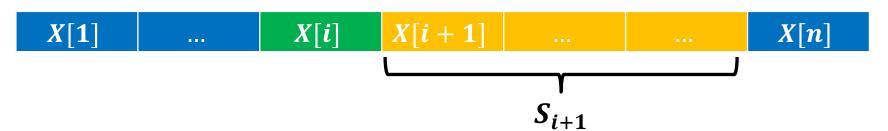
• $X[i] + S_{i+1} \le X[i] + D_{i+1}$



• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} \leq 0$)



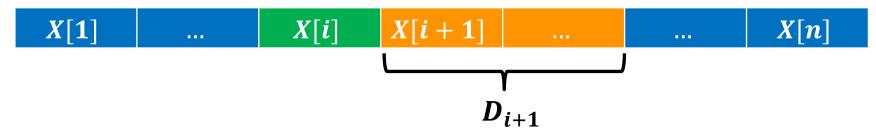
• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



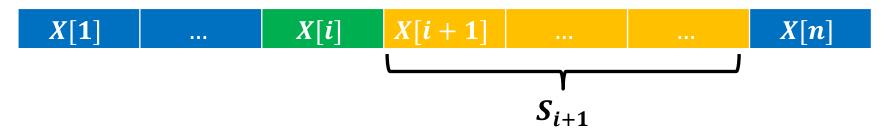
• $X[i] + S_{i+1} \le X[i] + D_{i+1} \le X[i]$



• D_{i+1} : 以X[i+1]开头的最大子数组和 ($D_{i+1} \leq 0$)



• S_{i+1} : 以X[i+1]开头的任一子数组和 $(S_{i+1} \leq D_{i+1})$



- $X[i] + S_{i+1} \le X[i] + D_{i+1} \le X[i]$
- $ullet D_i = X[i]$

问题结构分析



• 给出问题表示

• D[i]: 以X[i]开头的最大子数组和

• 明确原始问题

 $S_{max} = \max_{1 \le i \le n} \{D[i]\}$

问题结构分析



递推关系建立



自底向上计算



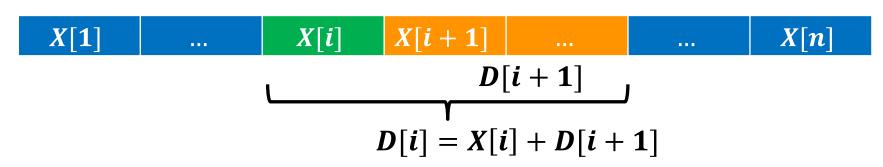
最优方案追踪

递推关系建立:分析最优(子)结构



• D[i]: 以X[i]开头的最大子数组和

情况1: D[i+1] > 0



• 情况2: $D[i+1] \leq 0$



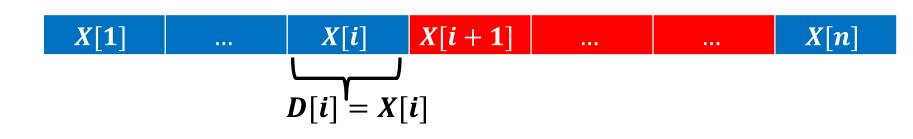




自底向上计算



最优方案追踪



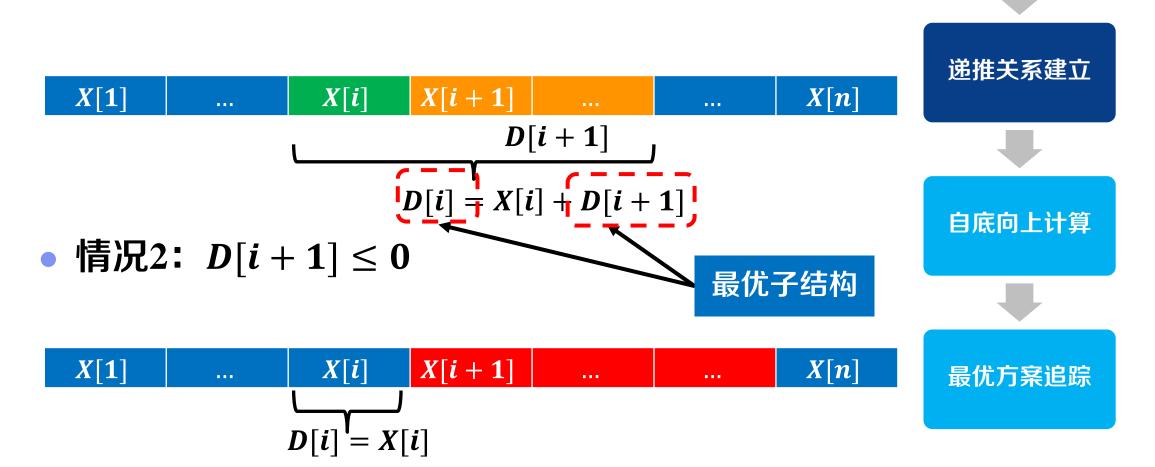
递推关系建立:分析最优(子)结构



问题结构分析

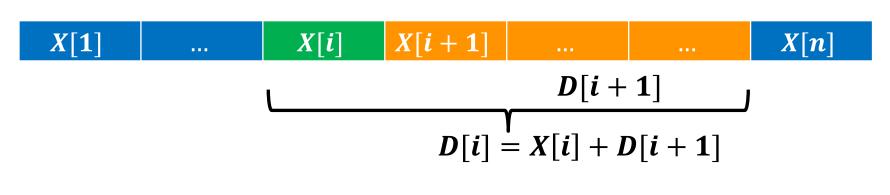
• D[i]: 以X[i]开头的最大子数组和

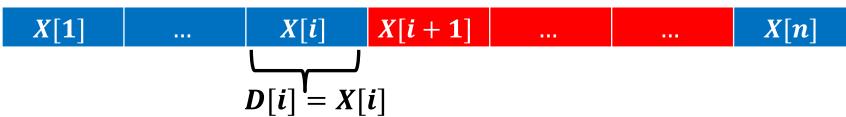
情况1: D[i+1] > 0



递推关系建立: 构造递推公式







•
$$D[i] = \begin{cases} X[i] + D[i+1], & if D[i+1] > 0 \\ X[i], & if D[i+1] \le 0 \end{cases}$$

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



自底向上计算



最优方案追踪

已知

	1	2	•••	i	<i>i</i> +1	•••	n-1	n
X	X[1]	X[2]		X[i]				X[n]



- 初始化
 - D[n] = X[n]

已知

	1	2	•••	i	<i>i</i> +1	•••	n-1	n
X	X[1]	X[2]		X[i]				X[n]
	1	2	•••	i	<i>i</i> +1	•••	n-1	$\frac{1}{n}$
D								X[n]

问题结构分析



递推关系建立



自底向上计算





- 初始化
 - D[n] = X[n]
- 递推公式

•
$$D[i] = \begin{cases} X[i] + D[i+1], & if D[i+1] > 0 \\ X[i], & if D[i+1] \le 0 \end{cases}$$

已知

	1	2	•••	i	<i>i</i> +1	•••	n-1	n
X	X[1]	X[2]		X[i]				X[n]
	1	2	•••	i	<i>i</i> +1	•••	n-1	\boldsymbol{n}
D								X[n]

问题结构分析



递推关系建立



自底向上计算





- 初始化
 - D[n] = X[n]
- 递推公式

•
$$D[i] = \begin{cases} X[i] + D[i+1], & if D[i+1] > 0 \\ X[i], & if D[i+1] \le 0 \end{cases}$$

已知

	1	2	•••	i	<i>i</i> +1	•••	n-1	n
X	X[1]	<i>X</i> [2]		X[i]				X[n]
	1	2	•••	i	<i>i</i> +1	•••	n-1	\boldsymbol{n}
D					D[i+1]			X[n]

问题结构分析



递推关系建立



自底向上计算





- 初始化
 - D[n] = X[n]
- 递推公式

•
$$D[i] = \begin{cases} X[i] + D[i+1], & if \ D[i+1] > 0 \\ X[i], & if \ D[i+1] \le 0 \end{cases}$$

已知

	1	2	•••	i	<i>i</i> +1	•••	n-1	n
X	X[1]	<i>X</i> [2]		X[i]				X[n]
	1	2	•••	i	<i>i</i> +1	•••	n-1	n
D				D[i]	D[i+1]			X[n]

问题结构分析



递推关系建立



自底向上计算



自底向上计算: 依次求解问题



- 初始化
 - D[n] = X[n]
- 递推公式

•
$$D[i] = \begin{cases} X[i] + D[i+1], & if \ D[i+1] > 0 \\ X[i], & if \ D[i+1] \le 0 \end{cases}$$

已知

	1	2	•••	i	<i>i</i> +1	•••	n-1	n
X	<i>X</i> [1]	<i>X</i> [2]		X[i]				X[n]
	1	2	•••	i	<i>i</i> +1	•••	n-1	n
D	+							X[n]

自底向上计算

问题结构分析



递推关系建立



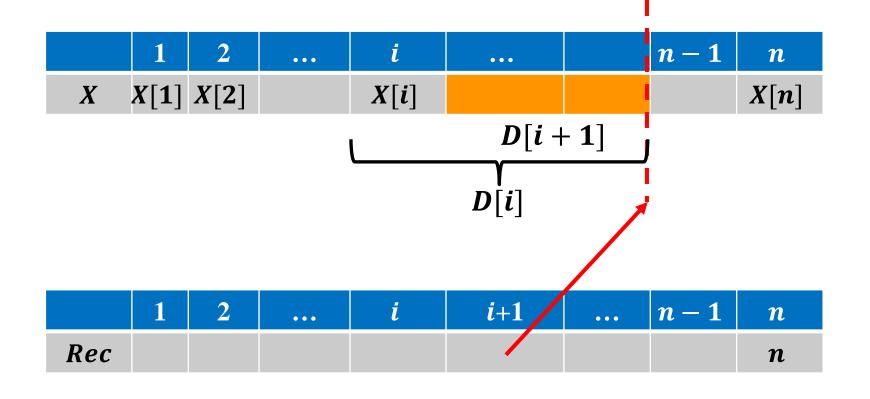
自底向上计算



最优方案追踪: 记录决策过程



- 构造追踪数组Rec[1..n]
 - 情况1: 结尾相同
 - o Rec[i] = Rec[i+1]



问题结构分析



递推关系建立



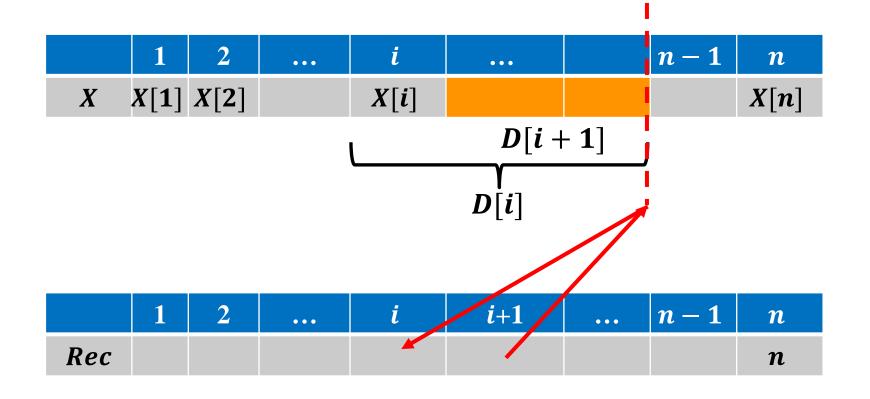
自底向上计算



最优方案追踪: 记录决策过程



- 构造追踪数组Rec[1..n]
 - 情况1: 结尾相同
 - o Rec[i] = Rec[i+1]



问题结构分析



递推关系建立



自底向上计算



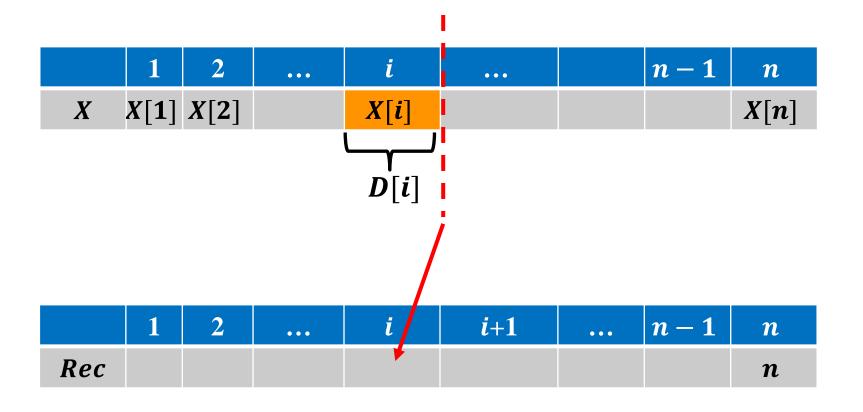
最优方案追踪: 记录决策过程



构造追踪数组Rec[1..n]

• 情况2: 结尾不同

o Rec[i] = i



问题结构分析



自底向上计算



最优方案追踪: 输出最优方案



• 从子问题中查找最优解

	1	2	•••	•••	i	n-1	n
X	X[1]	X[2]			X[i]		X[n]

_		1	2		i	n-1	n
已求得	D	D[1]	D[2]		D[i]		D[n]

	1	2	•••	i	•••	n-1	n
Rec							n

问题结构分析



递推关系建立



自底向上计算



最优方案追踪:输出最优方案



• 从子问题中查找最优解

	1	2	•••	•••	i		n-1	n
X	<i>X</i> [1]	<i>X</i> [2]			X[i]			X[n]
	1	2			i		n-1	n
D	D [1]	D [2]			D[i]			D[n]
						最优	た解 しんしん かいかい かいかい かいかい かいかい かいかい かいかい かいかい	
	1	2	•••		i	•••	n-1	n
Rec								n

问题结构分析



递推关系建立



自底向上计算

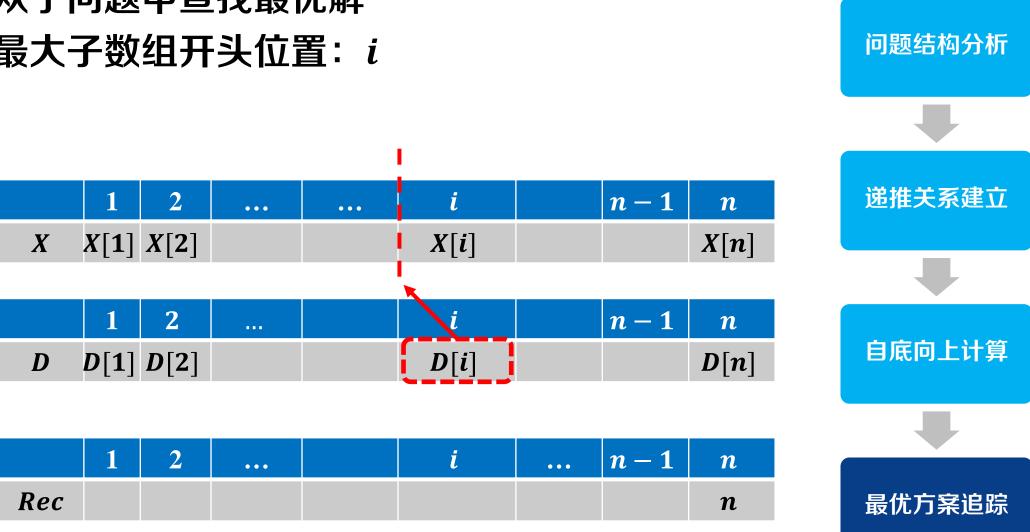


最优方案追踪:输出最优方案



• 从子问题中查找最优解

• 最大子数组开头位置: i



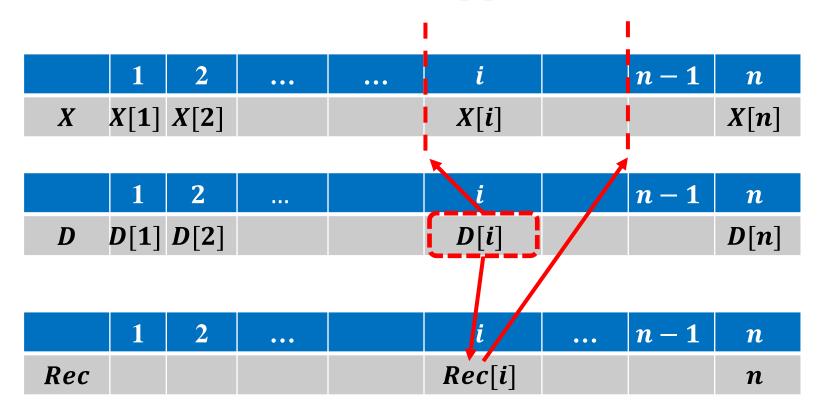
最优方案追踪:输出最优方案



• 从子问题中查找最优解

• 最大子数组开头位置: i

最大子数组结尾位置: Rec[i]



问题结构分析

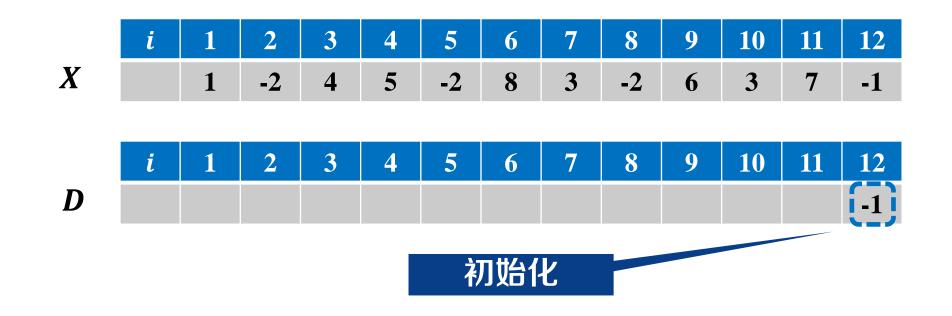
递推关系建立

自底向上计算



	i	1	2	3	4	5	6	7	8	9	10	11	12
X													
	i	1	2	3	4	5	6	7	8	9	10	11	12
D													





 i
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12

 Rec
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...
 ...

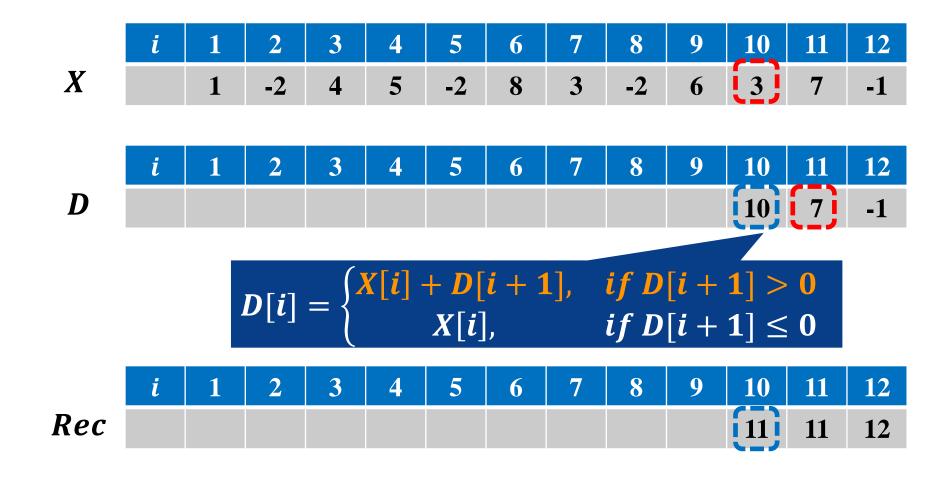


	i	1	2	3	4	5	6	7	8	9	10	11	12
X		1	-2	4	5	-2	8	3	-2	6	3	7	-1
	i	1	2	3	4	5	6	7	8	9	10	11	12
D												7	$\begin{bmatrix} -1 \end{bmatrix}$
			,	D[i]	_ {\lambda}	K[i] -	+ D [a	i+1	_], i	if D	[i +]	1] >	0
)[i]	_ (X[i]	,	i	if D	[i +]	1] ≤	0
	i	1	2	3	4	5	6	7	8	9	10	11	12
Rec													12

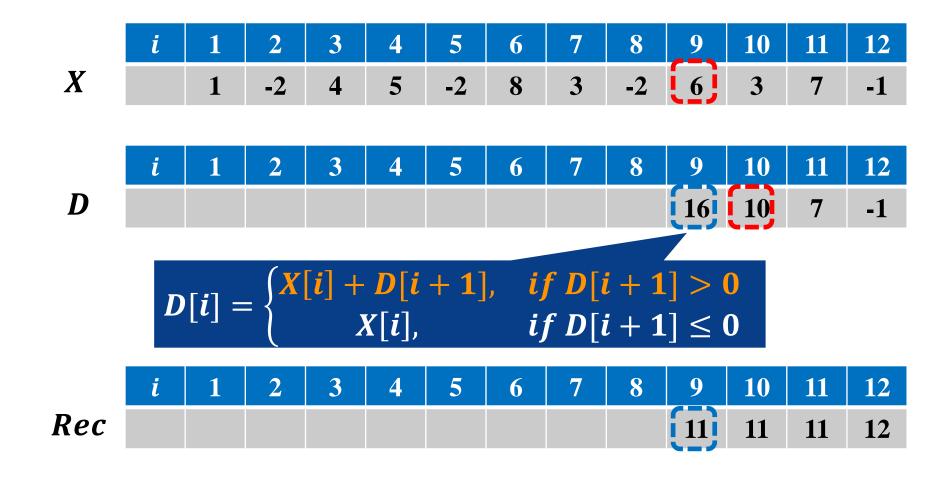


1
; -1
12
-1
> 0 ≤ 0
≤ 0
12
12
1 1

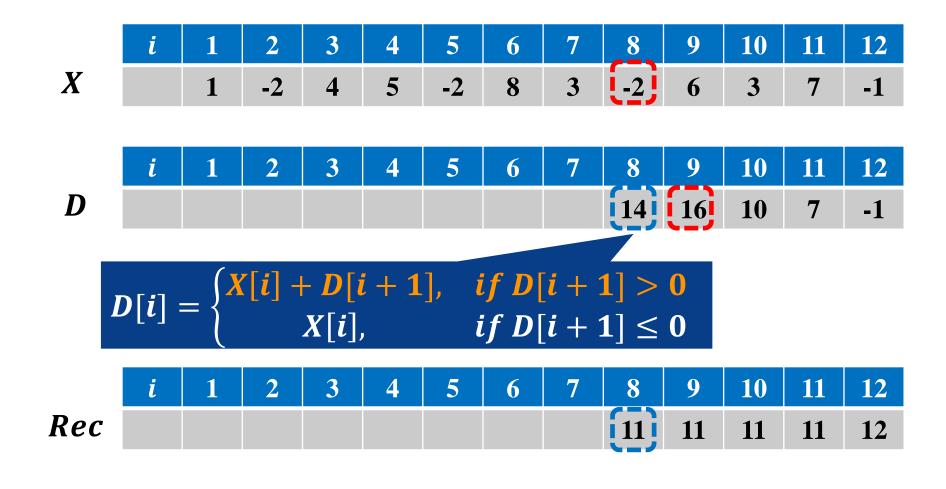




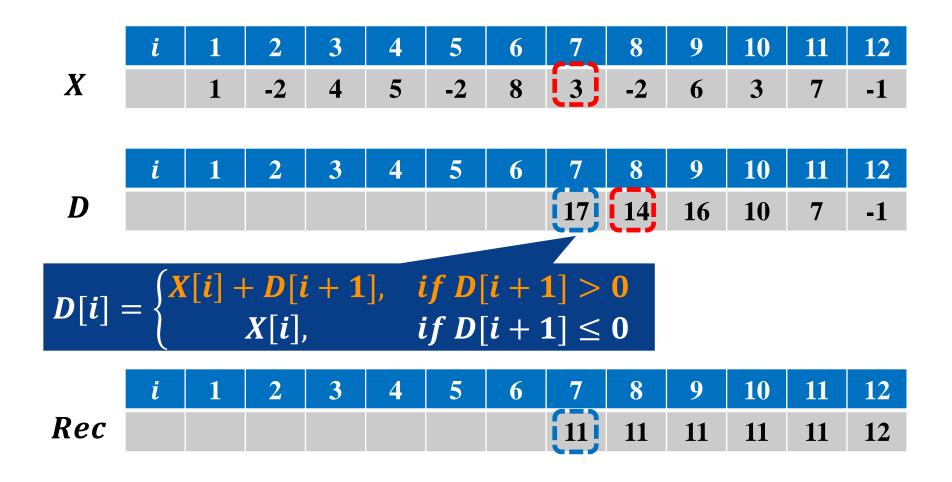




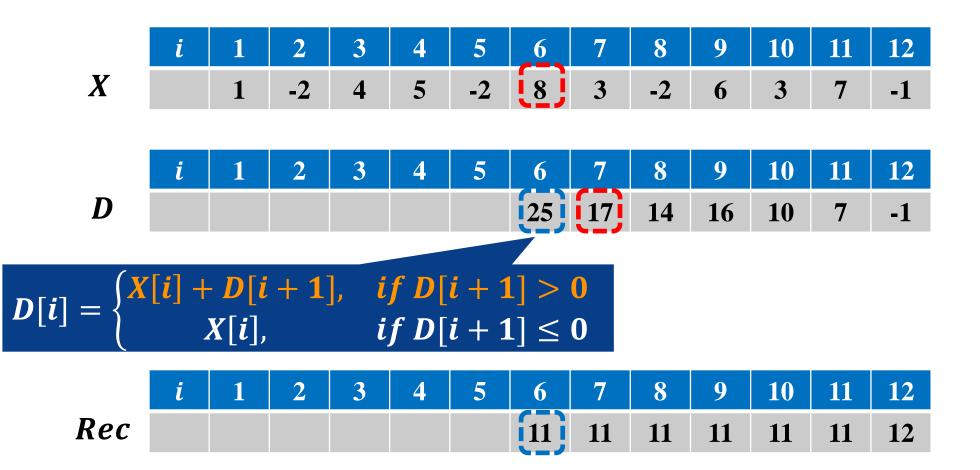














	i	1	2	3	4	5	6	7	8	9	10	11	12
X		1	-2	4	5	-2	8	3	-2	6	3	7	-1
	i	1	2	3	4	5	6	7	8	9	10	11	12
D						23	25	17	14	16	10	7	-1

$$D[i] = egin{cases} X[i] + D[i+1], & if D[i+1] > 0 \ X[i], & if D[i+1] \leq 0 \end{cases}$$

 i
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12

 Rec
 1
 1
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11
 11



10 3	7 -1
	, –
10	11 12
10	7 -1
	10 10

$$D[i] = \begin{cases} X[i] + D[i+1], & if D[i+1] > 0 \\ X[i], & if D[i+1] \leq 0 \end{cases}$$

Rec



	i	1	2	3	4	5	6	7	8	9	10	11	12
X		1	-2	4	5	-2	8	3	-2	6	3	7	-1
	i	1	2	3	4	5	6	7	8	9	10	11	12
D				32	28	23	25	17	14	16	10	7	-1
			D[i]	$=\left\{ \begin{array}{c} \lambda \end{array} \right.$	K[i] -	+ D [i+1	L],	if D	[i +	1] >	0	
			[י]			X[i]	,		if D	[i +	1] ≤	0	
	i	1	2	3	4	5	6	7	8	9	10	11	12
Rec				11	11	11	11	11	11	11	11	11	12



i	1	2	3	4	5	6	7	8	9	10	11	12
	1	-2	4	5	-2	8	3	-2	6	3	7	-1
i	1	2	3	4	5	6	7	8	9	10	11	12
		30	32	28	23	25	17	14	16	10	7	-1
	ווּות		X[i]	+ D	[i +]	1],	if D	[i +	1] >	> 0		
	$D[\iota]$	_ (X[i]],		if D	0[i +				
i	1	2	3	4	5	6	7	8	9	10	11	12
		11	11	11	11	11	11	11	11	11	11	12
	i	i 1 D[i]	$ \begin{array}{c cccc} i & 1 & 2 \\ \hline & 30 \end{array} $ $ D[i] = \begin{cases} i & 1 & 2 \end{cases} $	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$								



i	1	2	3	4	5	6	7	8	9	10	11	12
	1	-2	4	5	-2	8	3	-2	6	3	7	-1
i	1	2	3	4	5	6	7	8	9	10	11	12
	31	30	32	28	23	25	17	14	16	10	7	-1
[נ]ת	ı _ ∫	X[i]	+D[i+1],			if I	D[i +	· 1] :				
$D[\iota]$	—		X[i]	i],		if I	D[i +	- 1] :				
i	1	2	3	4	5	6	7	8	9	10_	11_	12
	11	11	11	11	11	11	11	11	11	11	11	12
	i	$i \qquad 1$ 31 $D[i] = \begin{cases}$	i 1 2 $31 30$ $i 1 2$ $i 1 2$	$ \begin{array}{c cccc} i & 1 & 2 & 3 \\ \hline 31 & 30 & 32 \\ \end{array} $ $ D[i] = \begin{cases} X[i] + D \\ X[i] \end{cases} $	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$							



	i	1	2	3	4	5	6	7	8	9	10	11	12
X		1	-2	4	5	-2	8	3	-2	6	3	7	-1
	i	1	2	3		5	6	7	8	9	10	11	12
\boldsymbol{D}		31	30	32	28	23	25	17	14	16	10	7	-1
			占	曼优角	屛								

Rec

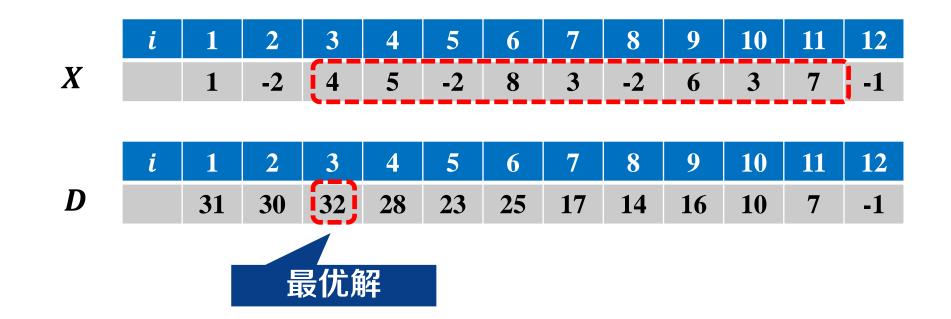


	i	1	2	3	4	5	6	7	8	9	10	11	12						
\boldsymbol{X}		1	-2	4	5	-2	8	3	-2	6	3	7	-1						
	i	1	2	3 32	4	5	6	7	8	9	10	11	12						
\boldsymbol{D}		31	30	32	28	23	25	17	14	16	10	7	-1						
			盲	是优角	军		最优解												

11 11 Rec

终止位置





终止位置

$$S = \{4, 5, -2, 8, 3, -2, 6, 3, 7\}$$



```
输入: 数组X, 数组长度n
输出: 最大子数组和S_{max},子数组起止位置l, r
新建一维数组D[1..n]和Rec[1..n]
//初始化
D[n] \leftarrow X[n]
                                初始化
Rec[n] \leftarrow n
7/动态规划
for i \leftarrow n-1 to 1 do
    if D[i+1] > 0 then
       D[i] \leftarrow X[i] + D[i+1]
       Rec[i] \leftarrow Rec[i+1]
    end
    else
       D[i] \leftarrow X[i]
       Rec[i] \leftarrow i
    end
end
```



```
输入: 数组X, 数组长度n
输出: 最大子数组和S_{max},子数组起止位置l, r
新建一维数组D[1..n]和Rec[1..n]
//初始化
D[n] \leftarrow X[n]
Rec[n] \leftarrow n
for i \leftarrow n-1 to 1 do
                                               自底向上计算
   if D[i+1] > 0 then
     D[i] \leftarrow X[i] + D[i+1]
      Rec[i] \leftarrow Rec[i+1]
   end
   else
       D[i] \leftarrow X[i]
      Rec[i] \leftarrow i
   end
end
```



```
输入: 数组X, 数组长度n
输出: 最大子数组和S_{max},子数组起止位置l, r
新建一维数组D[1..n]和Rec[1..n]
//初始化
D[n] \leftarrow X[n]
Rec[n] \leftarrow n
//动态规划
for i \leftarrow n-1 to 1 do
  if D[i+1] > 0 then
D[i] \leftarrow X[i] + D[i+1]
                                            情况1: D[i+1] > 0
      Rec[i] \leftarrow Rec[i+1]
   end
   else
       D[i] \leftarrow X[i]
       Rec[i] \leftarrow i
   end
end
```



```
输入: 数组X, 数组长度n
输出: 最大子数组和S_{max},子数组起止位置l, r
新建一维数组D[1..n]和Rec[1..n]
//初始化
D[n] \leftarrow X[n]
Rec[n] \leftarrow n
//动态规划
for i \leftarrow n-1 to 1 do
   if D[i+1] \ge 0 then
     D[i] \leftarrow X[i] + D[i+1]
Rec[i] \leftarrow Rec[i+1]
                                           记录子问题结果与决策
   end
   else
       D[i] \leftarrow X[i]
       Rec[i] \leftarrow i
   end
\mathbf{end}
```



```
输入: 数组X, 数组长度n
输出: 最大子数组和S_{max},子数组起止位置l, r
新建一维数组D[1..n]和Rec[1..n]
//初始化
D[n] \leftarrow X[n]
Rec[n] \leftarrow n
//动态规划
for i \leftarrow n-1 to 1 do
   if D[i+1] > 0 then
      D[i] \leftarrow X[i] + D[i+1]
      Rec[i] \leftarrow Rec[i+1]
   end
  else
                                         情况2: D[i+1] \leq 0
      D[i] \leftarrow X[i]
     Rec[i] \leftarrow i
  \mathbf{end}
end
```



```
//查找解
S_{max} \leftarrow D[1]
for i \leftarrow 2 to n do

| if S_{max} < D[i] then
| S_{max} \leftarrow D[i]
| l \leftarrow i
| r \leftarrow Rec[i]
| end
end
return S_{max}, l, r
```

时间复杂度分析



```
输入: 数组X, 数组长度n
输出: 最大子数组和S_{max},子数组起止位置l, r
新建一维数组D[1..n]和Rec[1..n]
//初始化
D[n] \leftarrow X[n]
Rec[n] \leftarrow n
//动态规划
for i \leftarrow n-1 to 1 do
   if D[i+1] > 0 then
      D[i] \leftarrow X[i] + D[i+1]
     Rec[i] \leftarrow Rec[i+1]
   end
   else
                                                                              O(n)
      D[i] \leftarrow X[i]
      Rec[i] \leftarrow i
   end
                                              时间复杂度: O(n)
\mathbf{end}
```