最大子段和

最大子段和

问题：给定n个数(可以为负数)的序列

$$(a_1, a_2, \ldots , a_n)$$

求 $\max\{0, \max\limits_{1\le i\le j\le n}\sum_{k=i}^{j} a_k\}$

实例

$$(-2, 11, -4, 13, -5, -2)$$

解：最大子段和为 $a_2+a_3+a_4=20$

算法

算法1：对所有的 (i, j) 对，顺序求和 $a_i + \ldots + a_j$ 并比较出最大的和

算法2：分治策略，将数组分成左右两半，分别计算左边的最大和、右边的最大和、跨边界的最大和，然后比较其中最大者

算法3：动态规划

算法1

算法 Enumerate

输入：数组 $A[1..n]$，

输出：*sum, first, last*

```
sum ← 0
for i ←1 to n do
    for j ← i to n do
        thissum ← 0
        for k ← i to j do
            thissum ← thissum +A[k]
        if thissum > sum
        then sum ← thissum
            first ← i
            last ← j
```

和的边界*i-j*

找到更大和

时间复杂度：$O(n^3)$

算法2 分治策略

将序列分成左右两半，中间分点*center*

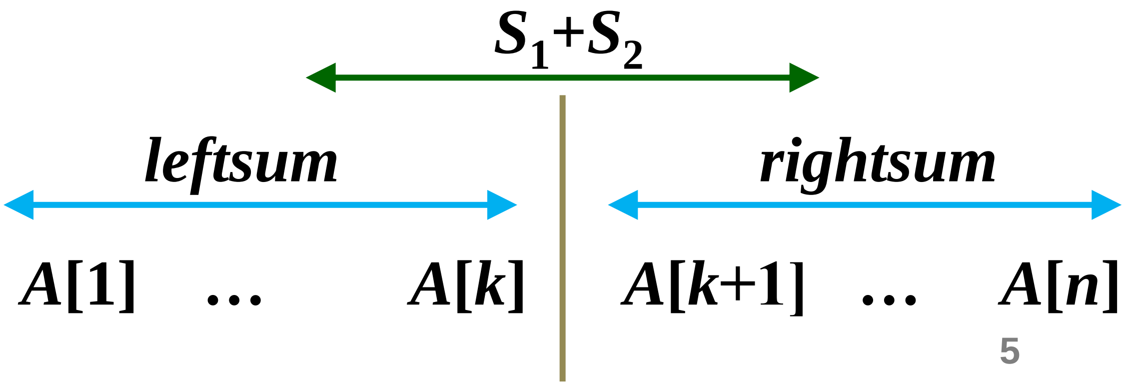
递归计算左段最大子段和 *leftsum*

递归计算右段最大子段和 *rightsum*

center 到a_1的最大和 S_1，k=*center*

center +1 到 a_n 的最大和 S_2

max { *leftsum*, *rightsum*, S_1+S_2 }



伪码

算法 **MaxSubSum** (*A*, *left*, *right*）

输入：数组 *A*, *left*, *right* (左,右边界)

输出：最大子段和*sum*及子段边界

1．if $|A|=1$ then 输出元素(值为负输出0)

2．*center*←⌊(*left*+*right*)/2⌋

3．*leftsum*←**MaxSubSum**(*A*, *left*,*center*)

4．*righsum*←**MaxSubSum**(*A*, *center*+1, *right*)

5．$S_1 \leftarrow A_1[center]$ //从*center*向左

6．$S_2 \leftarrow A_2[center+1]$ //从*center*+1向右

7．$sum \leftarrow S_1+S_2$

8．if *leftsum*>*sum* then *sum*←*leftsum*

9．if *rightsum*>*sum* then *sum*←*rightsum*

时间复杂度

计算 从$k = center$ 开始到 a_1方向的最大和，每次加1个元素，得到

$A[k]$, $A[k]+A[k-1]$, $A[k]+A[k-1]+A[k-2]$, … , $A[k]+\ldots+A[1]$

比较上述的最大和，时间为$O(n)$，

右半边也是$O(n)$

$$\boxed{\begin{aligned} T(n) &= 2T(n/2) + O(n) \\ T(c) &= O(1) \end{aligned}}$$

$$T(n) = O(n \log n)$$

算法3：动态规划

子问题界定：前边界为 1，后边界 i，$C[i]$ 是 $A[1..i]$中必须包含元素 $A[i]$ 的向前连续延伸的最大子段和

$$\boxed{C[i]=\max_{1\le k\le i}\{\sum_{j=k}^{i}A[j]\}}$$

$A[1]\ A[2]\ A[3]\ \ \ldots\ \ \ A[i\text{-}1]\ \ A[i]\ \ \ldots\ A[n]$

最大子段和 $C[i]$

优化函数的递推方程

递推方程:

$C[i]= \max\{C[i-1]+A[i], A[i]\}$

$i = 2, \ldots, n$

$C[1]=A[1]$ 若$A[1]>0$

$C[1]=0$ 否则

解: $\mathrm{OPT}(A) = \max\limits_{1\le i\le n}\{C[i]\}$

伪码

算法 MaxSum (A, n)

输入：数组A

输出：最大子段和sum, 子段最后位置c

```
sum ← 0
b ← 0
for i ← 1 to n do
    if b > 0
    then b ← b + A[i]
    else b ← A[i]
    if b > sum
    then sum ← b
         c ← i
return sum , c
```

子问题
后边界i

找到更
大的和

时间复杂度：$O(n)$, 空间复杂度：$O(n)$

小结

- 几个算法：蛮力, 分治, 动态规划
- 动态归划算法：
  子问题界定
  列优化函数的递推方程和边界条件
  (不一定是原问题的优化函数)
  自底向上计算，设计备忘录 (表格)
  如何根据动态规划的解找原问题的解
  时间复杂度估计