

最大字段和问题:

问题描述:

给定由 n 个整数组成的序列 a_1, a_2, \dots, a_n , 求该序列字段和的最大值, 当所有整数均为负值时定义其最大字段和为 0。数学描述为:

$$\text{MAX_SUBSTRING} = \text{MAX} \{ 0, \max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a[k] \}$$

例子: 给定 $\{-2, 11, -4, 13, -5, -2\}$,

最大字段和为 $11 + (-4) + 13 = 20$

常规做法如下:

$\text{max_substring} = \max\{0, \max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a_k\}$

常规做法: 我们从公式中可以看到三重循环。

- ① i 的循环, 即由 $1 \rightarrow n$.
- ② j 的循环, 即由 $i \rightarrow n$.
- ③ k 的循环, 即由 $i \rightarrow j$.

所以利用三重循环即可对问题进行解决, 时间复杂度为 n^3 .

对常规做法进行修改:

对常规做法进行改进: 从常规做法中, 我们知道共有三重循环, 而在最后一重循环其实是没有必要的, 即 k 的循环由 $i \rightarrow j$.

```
for i in 1 to n:
  for j in i to n:
    for k in i to j:
```

解释: 比如计算 $\{-2, 11, -4, 13, -5, -2\}$

我们需要计算当 $i=0$ 时所有子序列和。

当 j 由 $0 \rightarrow 3$ 时, 其实 $i=0, j=2$ 的总和

是计算 $i=0, j=3$ 的基础, 不需重复, 只需要得到

当前的最大值即可。

由上的理由, 常规做法只需两重循环, 时间复杂度为 n^2 .

分治法分析:

分治法求解:

将最大子段和 $a[1:n]$ 分为 $a[1:n/2]$ 和 $a[n/2+1:n]$ 两部分, 则可能有下列三种情况:

1) $a[1:n]$ 的最大子段和 = $a[1:n/2]$ 的最大子段和

2) $a[1:n]$ 的最大子段和 = $a[n/2+1:n]$ 的最大子段和

3) $a[1:n]$ 的最大子段和 = $\sum_{k=i}^j a_k$, $1 \leq i \leq n/2$, $n/2+1 \leq j \leq n$

此时, 可分别计算 $a[1:n/2]$ 和 $a[n/2+1:n]$ 的最大子段和.

$$S_1 = \max_{1 \leq i \leq n/2} \sum_{k=i}^{n/2} a_k, \quad S_2 = \max_{n/2+1 \leq i \leq n} \sum_{k=i}^n a_k$$

$$\text{max-substring} = S_1 + S_2$$

复杂性分析:

$$T(n) = \begin{cases} O(1) & n \leq c \\ 2T\left(\frac{n}{2}\right) + O(n) & n > c \end{cases}$$

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

动态规划分析: (算法复杂度为 $O(n)$)

$$\text{max-substring} = \max\{0, \max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a_k\}$$

$$\max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a_k = \max_{1 \leq j \leq n} \max_{1 \leq i \leq j} \sum_{k=i}^j a_k \quad \text{令 } b_j = \max_{1 \leq i \leq j} \sum_{k=i}^j a_k$$

$$\Rightarrow \max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a_k = \max_{1 \leq j \leq n} b_j$$

$$\text{因为 } b_j = \max_{1 \leq i \leq j} \sum_{k=i}^j a_k \Rightarrow b_{j+1} = \max_{1 \leq i \leq j+1} \sum_{k=i}^{j+1} a_k$$

$$= \max_{1 \leq i \leq j} \sum_{k=i}^j a_k + a_{j+1}$$

$$= b_j + a_{j+1}$$

$$\text{令 } b_j > 0 \text{ 时 } b_{j+1} = b_j + a_{j+1}$$

$$b_j = 0 \text{ 时 } b_{j+1} = a_{j+1}$$

$$\text{因为: } \max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a_k = \max_{1 \leq j \leq n} b_j$$

$$\Rightarrow b_j = \max\{b_{j-1} + a_j, a_j\}$$

状态转移方程