第一讲 基本概念

浙江大学 陈 越

浙江大学计算机科学与技术学院

1.3 应用实例：
最大子列和问题

给定N个整数的序列$\{A_1, A_2, \ldots, A_N\}$，求函数 $f(i,j)=\max\{0,\sum_{k=i}^{j}A_k\}$ 的最大值。

算法1

```
int MaxSubseqSum1( int A[], int N )
{ int ThisSum, MaxSum = 0;
  int i, j, k;
  for( i = 0; i < N; i++ ) { /* i是子列左端位置 */
    for( j = i; j < N; j++ ) { /* j是子列右端位置 */
      ThisSum = 0;  /* ThisSum是从A[i]到A[j]的子列和 */
      for( k = i; k <= j; k++ )
        ThisSum += A[k];
      if( ThisSum > MaxSum ) /* 如果刚得到的这个子列和更大 */
        MaxSum = ThisSum;    /* 则更新结果 */
    } /* j循环结束 */
  } /* i循环结束 */
  return MaxSum;
}
```

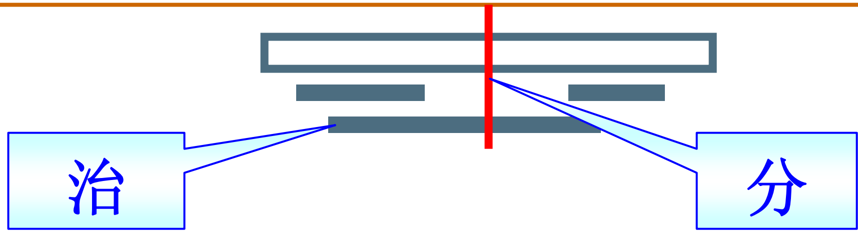
$T(N) = O(N^3)$

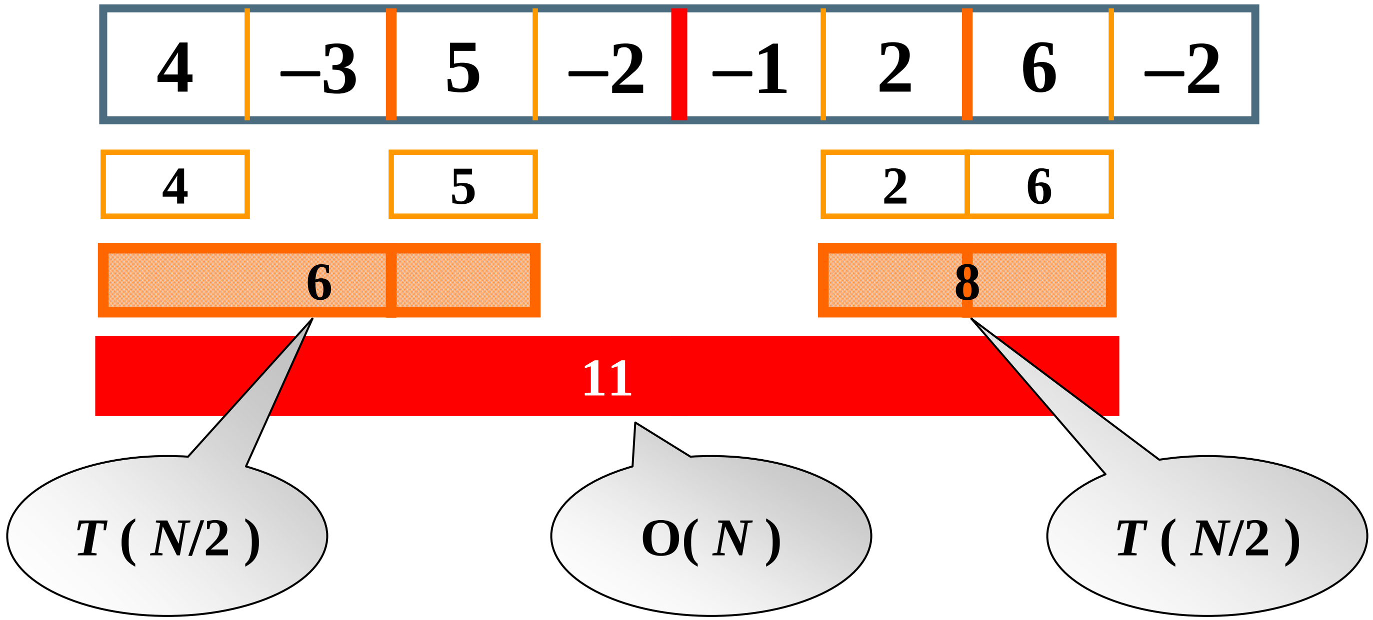
算法2

```
int MaxSubseqSum2( int A[], int N )
{ int ThisSum, MaxSum = 0;
  int i, j;
  for( i = 0; i < N; i++ ) { /* i是子列左端位置 */
    ThisSum = 0;  /* ThisSum是从A[i]到A[j]的子列和 */
    for( j = i; j < N; j++ ) { /* j是子列右端位置 */
      ThisSum += A[j];
      /*对于相同的i，不同的j，只要在j-1次循环的基础上累加1项即可*/
      if( ThisSum > MaxSum ) /* 如果刚得到的这个子列和更大 */
        MaxSum = ThisSum;    /* 则更新结果 */
    } /* j循环结束 */
  } /* i循环结束 */
  return MaxSum;
}
```

$T(N) = O(N^2)$

算法3：分而治之





$$T(N) = 2\,T(N/2) + c\,N\,, \qquad T(1) = O(1)$$

$$= 2\,[2\,T(N/2^2) + c\,N/2] + c\,N$$

$$= 2^k\,O(1) + c\,k\,N \qquad \text{其中}\ \ N/2^k = 1$$

$$= O(N \log N)$$

算法4：在线处理

```
int MaxSubseqSum4( int A[], int N )
{ int ThisSum, MaxSum;
  int i;
  ThisSum = MaxSum = 0;
  for( i = 0; i < N; i++ ) {
    ThisSum += A[i]; /* 向右累加 */
    if( ThisSum > MaxSum )
      MaxSum = ThisSum; /* 发现更大和则更新当前结果 */
    else if( ThisSum < 0 ) /* 如果当前子列和为负 */
      ThisSum = 0; /* 则不可能使后面的部分和增大，抛弃之 */
  }
  return MaxSum;
}
```

$$T(N) = O(N)$$

"在线"的意思是指每输入一个数据就进行即时处理，在任何一个地方中止输入，算法都能正确给出当前的解。

运行时间比较

（秒）

算　法		1	2	3	4
时间复杂度		$O(N^3)$	$O(N^2)$	$O(N \log N)$	$O(N)$
输入规模	N =10	0.00103	0.00045	0.00066	0.00034
	N =100	0.47015	0.01112	0.00486	0.00063
	N =1,000	448.77	1.1233	0.05843	0.00333
	N =10,000	NA	111.13	0.68631	0.03042
	N =100,000	NA	NA	8.0113	0.29832