01-E3

绪论

迭代与递归：总和最大区段

这时，公共智慧的结果便产生理智与意志在社会体中的结合，也才有了各个部分的密切配合，以及最后全体的最大力量。

邓俊辉

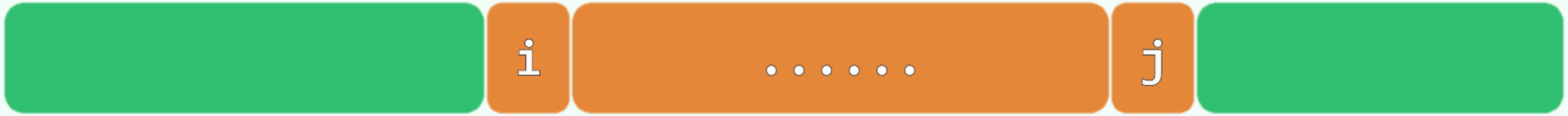
deng@tsinghua.edu.cn

问题 + 蛮力算法

❖ **从整数序列中，找出总和最大的区段（有多个时，短者、靠后者优先）**

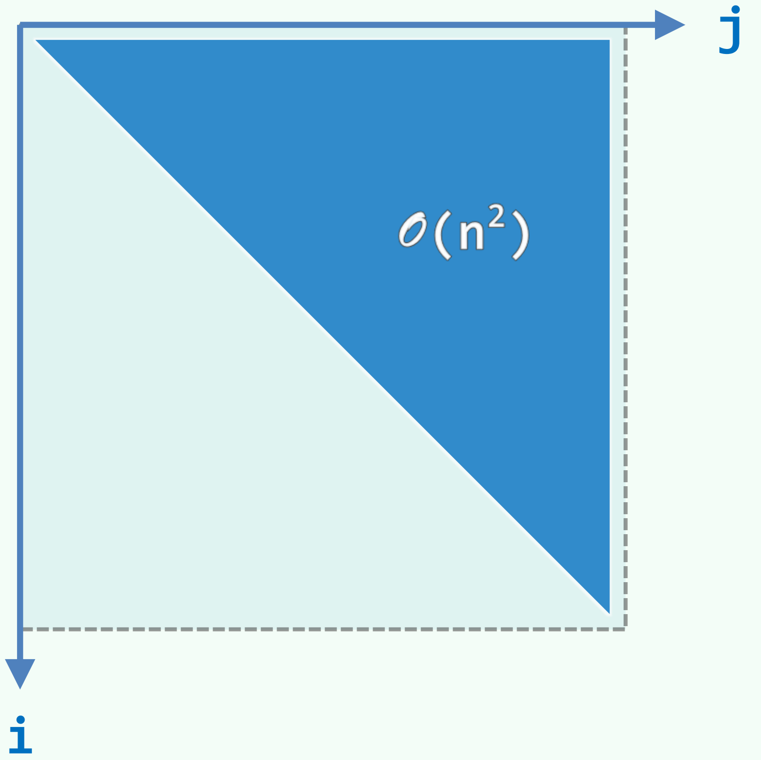
$$\mathcal{A}[0,19) = \{ 1,\ -2,\ \underline{7,\ 2,\ 6,\ -9,\ 5,\ 6,}\ -12,\ -8,\ \underline{13,\ 0,\ -3,\ 1,\ -2,\ 8,}\ 0,\ -5,\ 3 \}$$

```
int gs_BF( int A[], int n ) { //蛮力策略：O(n^3)
   int gs = A[0]; //当前已知的最大和
   for ( int i = 0; i < n; i++ )
      for ( int j = i; j < n; j++ ) { //枚举所有的O(n^2)个区段！
         int s = 0; for ( int k = i; k <= j; k++ ) s += A[k]; //用O(n)时间求和
         if ( gs < s ) gs = s; //择优、更新
      }
   return gs;
}
```



递增策略

```
int gs_IC( int A[], int n ) { //Incremental Strategy: O(n^2)
   int gs = A[0]; //当前已知的最大和
   for ( int i = 0; i < n; i++ ) { //枚举所有起始于i
      int s = 0;
      for ( int j = i; j < n; j++ ) { //终止于j的区间
         s += A[j]; //递增地得到其总和: O(1)
         if ( gs < s ) gs = s; //择优、更新
      }
   }
   return gs;
}
```



分而治之：前缀 + 后缀：$\mathcal{A}[lo,hi) \;=\; \mathcal{A}[lo,mi) \cup \mathcal{A}[mi,hi) \;=\; \mathcal{P} \cup \mathcal{S}$

❖ 借助递归，便可求得$\mathcal{P}$、$\mathcal{S}$内部的GS；而剩余的实质任务无非是...

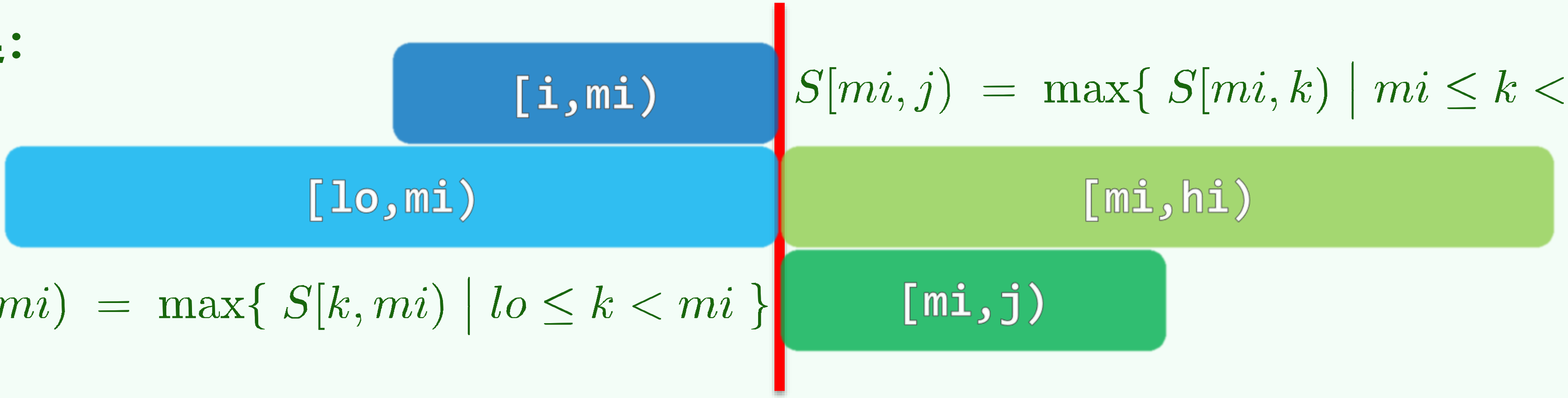
考察那些**跨越**切分线的区段...

❖ 沿着切分线，这类区段必被分割为**非空**的前缀、后缀：

$$\mathcal{A}[i,j) \;=\; \mathcal{A}[i,mi) + \mathcal{A}[mi,j),\ i < mi < j$$

而且满足：

$$S[mi,j) \;=\; \max\{\ S[mi,k) \mid mi \le k < hi\ \}$$

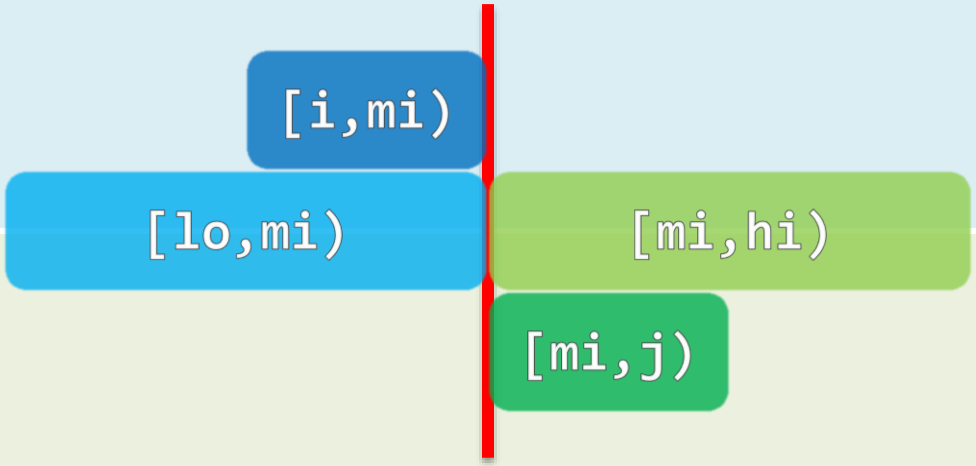


$$S[i,mi) \;=\; \max\{\ S[k,mi) \mid lo \le k < mi\ \}$$

❖ 可见，二者均可独立计算，且累计耗时不过 $\mathcal{O}(n)$；于是总体复杂度也优化为 $\mathcal{O}(n \cdot \log n)$

分治策略：实现

```
int gs_DC( int A[], int lo, int hi ) { //Divide-And-Conquer: O(n*logn)
   if ( hi - lo < 2 ) return A[lo]; //递归基
   int mi = (lo + hi) / 2; //在中点切分
   int gsL = A[mi-1], sL = 0, i = mi; //枚举
   while ( lo < i-- ) //所有[i, mi)类区段
      if ( gsL < (sL += A[i]) ) gsL = sL; //更新
   int gsR = A[mi], sR = 0, j = mi-1; //枚举
   while ( ++j < hi ) //所有[mi, j)类区段
      if ( gsR < (sR += A[j]) ) gsR = sR; //更新
   return max( gsL + gsR, max( gs_DC(A, lo, mi), gs_DC(A, mi, hi) ) ); //递归
}
```

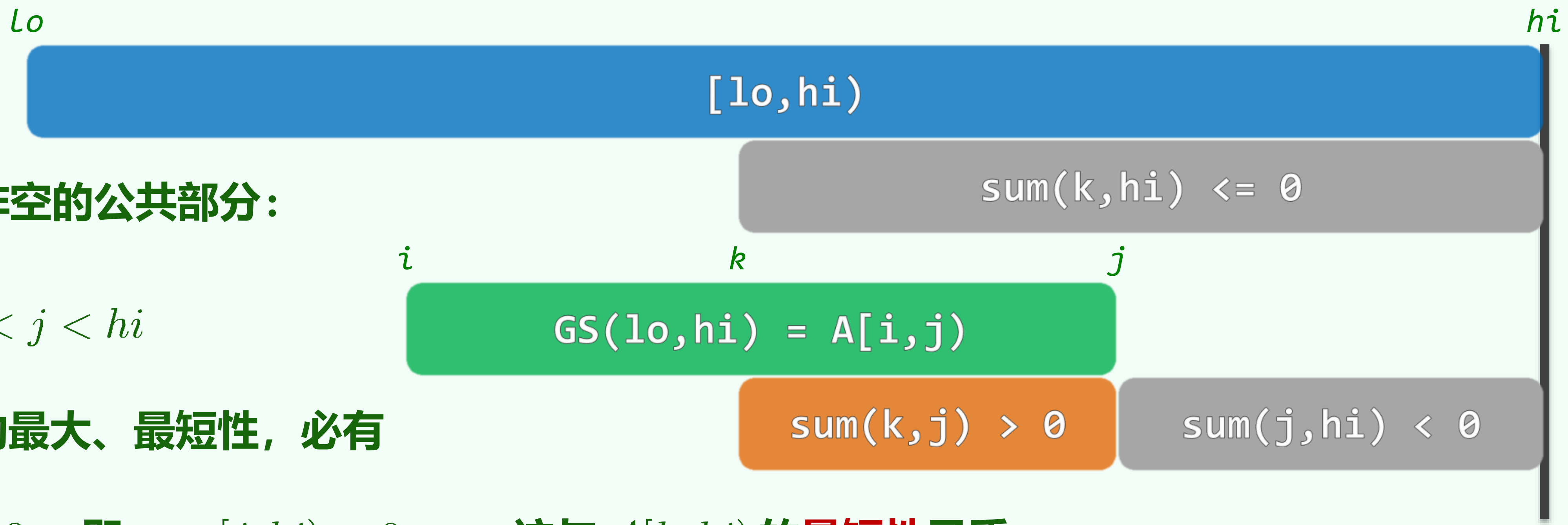


减治策略：最短的总和非正的后缀 ~ 总和最大区段

$$suffix(k) = \mathcal{A}[k,hi),\ k = \max\{\ lo \le i < hi \mid sum[i,hi) \le 0\ \} \qquad GS(lo,hi) = \mathcal{A}[i,j)$$

❖ 后者要么是前者的（真）**后缀**，要么与前者**无交**

❖ [反证]



假若二者确有非空的公共部分：

$$\mathcal{A}[k,j),\ k < j < hi$$

❖ 由 $GS[lo,hi)$ 的最大、最短性，必有

$sum[k,j) > 0$ ，即 $sum[j,hi) < 0$ ——这与 $\mathcal{A}[k,hi)$ 的**最短性**矛盾

❖ 基于以上事实，完全可以采用“减而治之”的策略，通过**一趟**线性扫描在**线性**时间内找出GS...

减治策略：实现

```
int gs_LS( int A[], int n ) { //Linear Scan: O(n)
   int gs = A[0], s = 0, i = n;
   while ( 0 < i-- ) { //在当前区间内
      s += A[i]; //递增地累计总和
      if ( gs < s ) gs = s; //并择优、更新
      if ( s <= 0 ) s = 0; //剪除负和后缀
   }
   return gs;
}
```