1.绪论

迭代与递归

Max2

邓俊辉

deng@tsinghua.edu.cn

迭代1

- 从数组区间A[lo, hi)中找出最大的两个整数A[x1]和A[x2] //A[x1] ≥ A[x2]

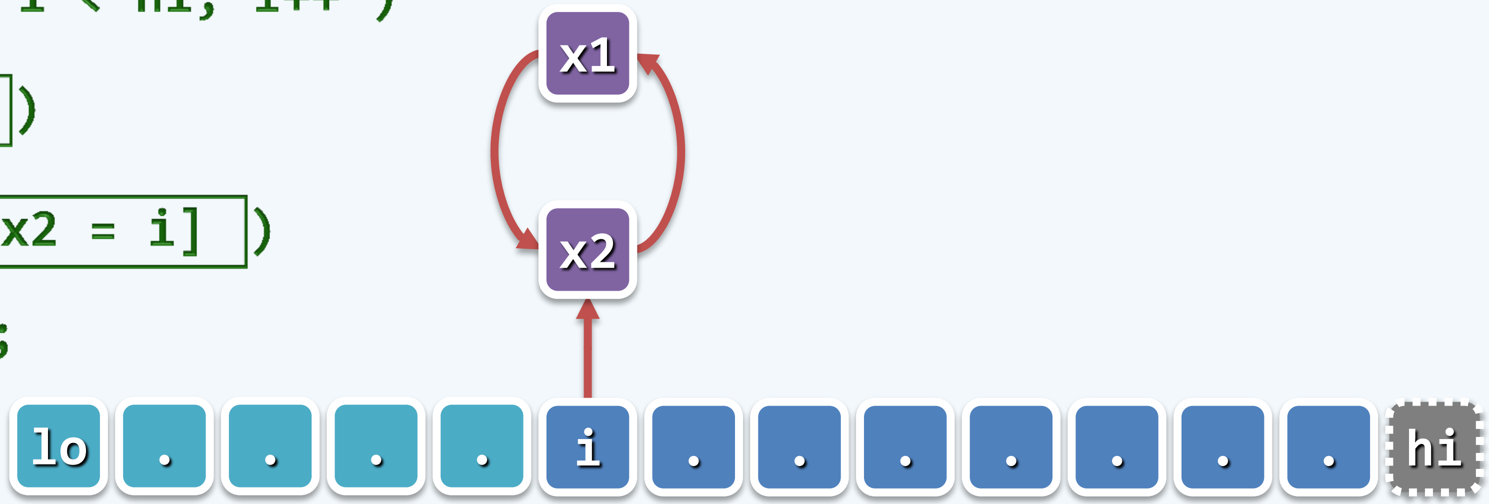
  元素比较的次数，要求尽可能地少

lo . . . . x1 . . . . hi

```
void max2(int A[], int lo, int hi, int & x1, int & x2) {  // 1 < n = hi - lo
   for ( x1 = lo, int i = lo + 1; i < hi; i++ ) //扫描A[lo, hi)，找出A[x1]
      if ( A[x1] < A[i] ) x1 = i; // hi - lo - 1  =  n - 1
   for ( x2 = lo, int i = lo + 1; i < x1; i++ ) //扫描A[lo, x1)
      if ( A[x2] < A[i] ) x2 = i; // x1 - lo - 1
   for ( int i = x1 + 1; i < hi; i++ ) //再扫描A(x1, hi)，找出A[x2]
      if ( A[x2] < A[i] ) x2 = i; // hi - x1 - 1
} //无论如何，比较次数总是Θ(2n - 3)
```

迭代2

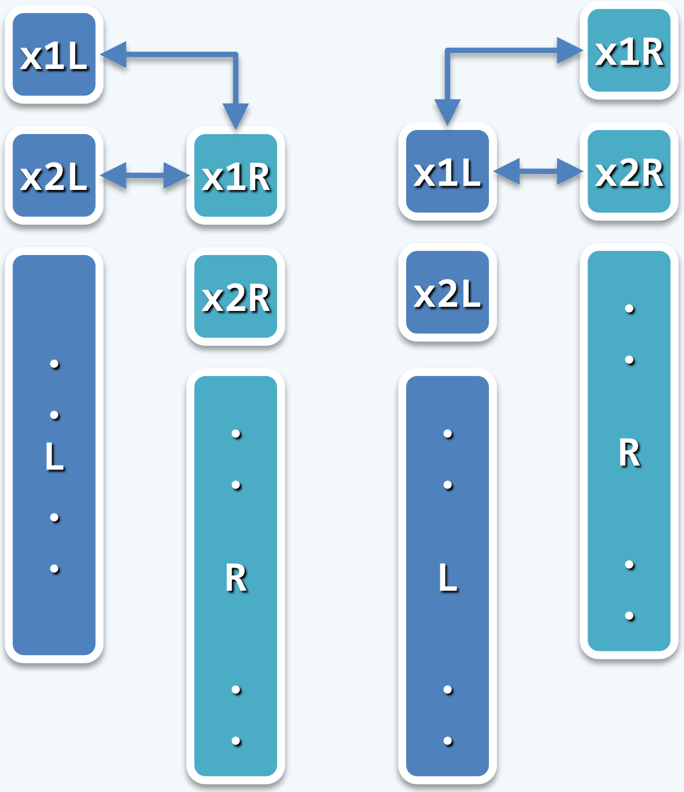
```
void max2( int A[], int lo, int hi, int & x1, int & x 2) {  // 1 < n = hi-lo
    if ( A[x1 = lo] < A[x2 = lo + 1] ) swap(x1, x2);
    for ( int i = lo + 2; i < hi; i++ )
        if ( A[x2] < A[i] )
            if ( A[x1] < A[x2 = i] )
                swap(x1, x2);
}
```



- 最好情况，$1 + (n - 2)*\boxed{1} = n - 1$
- 最坏情况，$1 + (n - 2)*\boxed{2} = 2n - 3$
- 比较次数可否进一步减少呢？分而治之！

递归 + 分治

```
❖ void max2( int A[], int lo, int hi, int & x1, int & x2 ) {
      if ( hi <= lo + 3 ) { trivial( A, lo, hi, x1, x2 ); return; } //T(3) <= 3
      int mi = (lo + hi)/2; //divide
      int x1L, x2L; max2( A, lo, mi, x1L, x2L );
      int x1R, x2R; max2( A, mi, hi, x1R, x2R );
      if ( A[x1L] > A[x1R] ) {
         x1 = x1L; x2 = A[x2L] > A[x1R] ? x2L : x1R;
      } else {
         x1 = x1R; x2 = A[x1L] > A[x2R] ? x1L : x2R;
      } //1 + 1  =  2
  } //T(n)  =  2*T(n/2) + 2  <=  5n/3 - 2
```



；借助数据结构，还可进一步优化（第10章）

Master Theorem

❖ [AHU-74], p64, Theorem 2.1

Recurrence	Solution	Examples
T(n) = T(n-1) + 1	$\mathcal{O}(n)$	向量求和之线性递归版
T(n) = T(n-1) + n	$\mathcal{O}(n^2)$	列表起泡排序之线性递归版
T(n) = 2*T(n-1) + 1	$\mathcal{O}(2^n)$	Hanoi塔、Fibonacci数
T(n) = 2*T(n-1) + n	$\mathcal{O}(2^n)$	
T(n) = T(n/2) + 1	$\mathcal{O}(\log n)$	向量的二分查找
T(n) = T(n/2) + n	$\mathcal{O}(n)$	列表的二分查找
T(n) = 2*T(n/2) + 1	$\mathcal{O}(n)$	向量求和之二分递归版
T(n) = 2*T(n/2) + n	$\mathcal{O}(n\log n)$	归并排序