

第九讲 排序（上）

浙江大学 陈 越

9.3 堆排序

选择排序

```
void Selection_Sort ( ElementType A[], int N )
{   for ( i = 0; i < N; i ++ ) {
        MinPosition = ScanForMin( A, i, N-1 );
        /* 从A[i]到A[N-1]中找最小元, 并将其位置赋给MinPosition */
        Swap( A[i], A[MinPosition] );
        /* 将未排序部分的最小元换到有序部分的最后位置 */
    }
}
```

无论如何: $T = \Theta(N^2)$

如何快速找到最小元? ? ?
最小堆

堆排序

堆排序是选择排序的一种改进

堆排序不是稳定的

■ 算法1

```
void Heap_Sort ( ElementType A[], int N )
{
    BuildHeap(A); /* O(N) */
    for ( i=0; i<N; i++ )
        TmpA[i] = DeleteMin(A); /* O(logN) */
    for ( i=0; i<N; i++ ) /* O(N) */
        A[i] = TmpA[i];
}
```

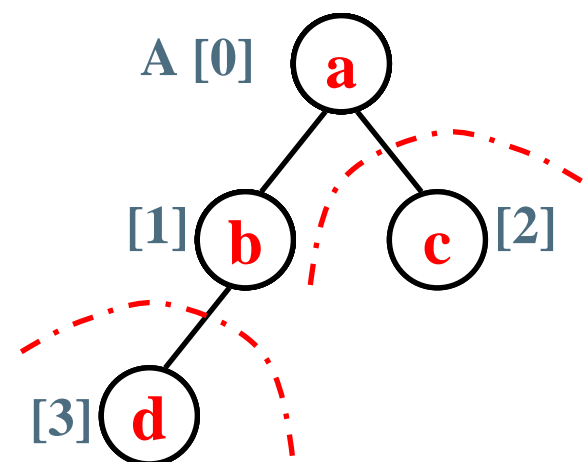
$$T(N) = O(N \log N)$$



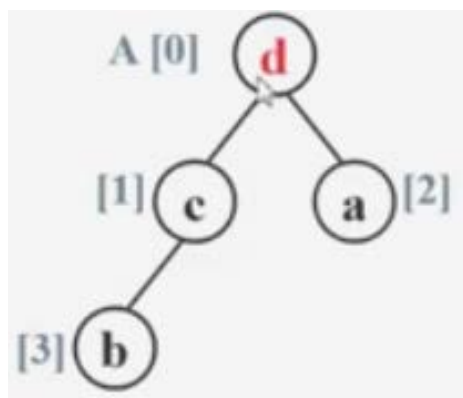
需要额外 $O(N)$ 空间，并且复制元素需要时间。

堆排序

■ 算法2



调整为最大堆:然后不断的将第一个元素与最后一个元素交换(不断的调整为最大堆)



```
void Heap_Sort ( ElementType A[], int N )
{
    for ( i=N/2-1; i>=0; i-- ) /* BuildHeap */
        PercDown( A, i, N ); /* i对应根节点所在的位置, N对应当前这个堆里一共有多少个元素 */
    for ( i=N-1; i>0; i-- ) {
        Swap( &A[0], &A[i] ); /* DeleteMax */
        PercDown( A, 0, i ); /* 把剩下的元素继续调整成最大堆, 调整的时候是以0为根节点, i是当前最大堆的元素的个数 */
    }
}
```

- 定理：堆排序处理 N 个不同元素的随机排列的平均比较次数是 $2N \log N - O(N \log \log N)$ 。
- 虽然堆排序给出最佳平均时间复杂度，但实际效果不如用Sedgewick增量序列的希尔排序。

堆排序里面的元素是从0下标开始记得,与最大堆从1下标开始记有区别

在堆排序中，元素下标从0开始。则对于下标为 i 的元素，其左、右孩子的下标分别为： $2i+1$, $2i+2$