Quicksort

The Divide-and-Conquer Approach

- ❑ Two Necessary Steps

1. Divide: Smaller problems are solved recursively.
2. Conquer: The solution t the original problem is formed from the solutions to the subproblems.

- ❑ The running time equation of divide and conquer approach:

$$T(n) = 2T(n/2) + O(n) \qquad \text{where } n \text{ is the input size.}$$

The solution to this equation is $O(n\log_2 n)$.

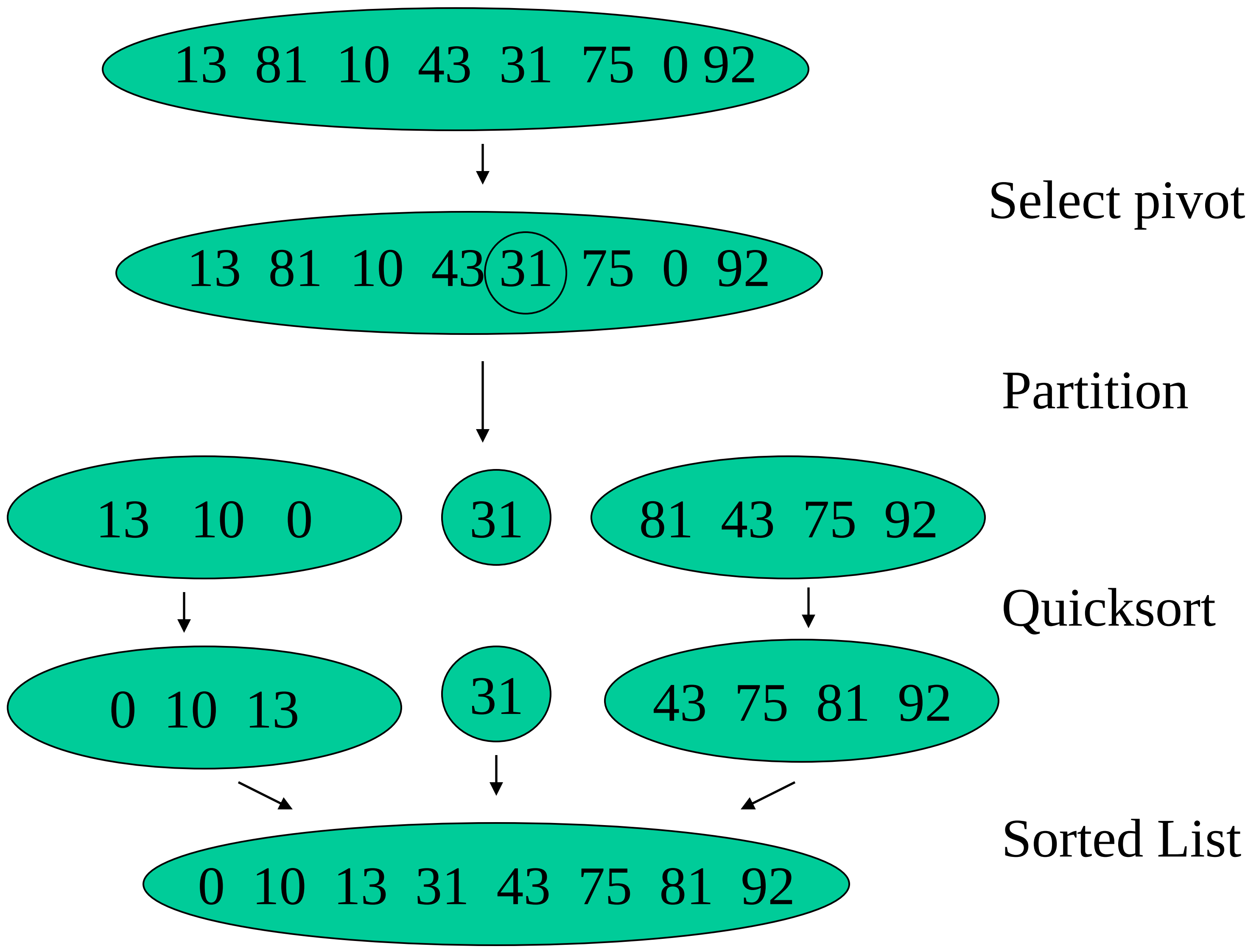
Quick Sort

- It is a comparison sort where elements are sorted in place.
- This sort uses the divide-and-conquer algorithm for sorting an array of n elements, S[1,2,…..n].
- It maintains the following 3 steps:

1. If the number of items in S is 0 or 1, then return.
2. Pick any element v in S. This is called pivot or partitioning element.
3. Partition the elements S-{v} into two disjoint sets S1 and S2 such that

   $S1=\{x \in S-\{v\} | x \leq v\}$ and $S2=\{x \in S-\{v\} | x \geq v\}$

Example:


13 81 10 43 31 75 0 92
Select pivot
13 81 10 43 31 75 0 92
Partition
13 10 0
31
81 43 75 92
Quicksort
0 10 13
31
43 75 81 92
Sorted List
0 10 13 31 43 75 81 92

Pivot Selection

1. Use the first element as pivot. If the input is random, it is acceptable. But if the input is presorted or in reverse order, pivot provides a poor partition.
2. Choose the pivot randomly. Although a random pivot is very unlikely provide poor partition but it is an expensive approach and does not reduce the average running time of rest of the algorithm.
3. Choose the median of three numbers as pivot. The best choice of pivot would be the median of array. Unfortunately, it is relatively hard to calculate and would slow down quicksort considerably.

Quicksort Algorithm

Quicksort(S, N)

```
1.   Qsort(S, 0, N-1)
2.   End
```

Median(S, Left, Right)

```
1. Center = (Left+Right)/2
2. If S[Left]>S[Center]
       Swap( S[Left],S[Center])
3. If S[Left]>S[Right]
       Swap( S[Left],S[Right])
4. If S[Center]>S[Right]
       Swap(S[Center], S[Right])
5. Swap(S[Center], S[Right-1]
6. Return S[Right-1]
```

Qsort(A, left, Right)

```
1.    If left <= Right
2.       pivot = Median(A, Left, Right)
3.       I = Left+1 and J = Right-2
4.       for( ;  ; )
5.          while(S[ I ] < pivot) I++;
6.          while(S[ J ] > pivot) J- - ;
7.          if I < J  Swap(S[ I ], S[ J ])
8.      Swap(S[ I ] ,  S[Right-1])    /*Restore pivot
9.      Qsort(A, Left, I-1)
10.     Qsort(A, I+1, Right)
```

Complexity of Quick Sort

1. Worst Case:

The worst case occurs if the input is presorted or in reverse order and if the 1st or the last element is chosen as pivot. In this case pivot provides a poor partition because all elements go into one partition and the other partition is empty. The worst case running time is $0(N^2)$.

2. Average Case:

The average running time of Quick sort is obtained by averaging the runtime of all possible subproblems' sizes. That means it occurs from the fact that on average each reduction step of the algorithm produces two subsets. So, the average running time is 0(NlogN).

3. Best Case:

The best case occurs when the partition always splits into two subsets with equal number of elements. So, the running time is 0(NlogN).

<u>END</u>