Data Structure

HW2

Yijia (Tiffany) Gao

Sorting

Length of Array: N

Bubble Sort (Non-Recursive) MAX:

1. Declare new int i: 1
2. Declare new Boolean: 1
3. While: Check Boolean: N
4. Set Boolean to false: 1
5. For loop(N-1):
   1. If statement: 3(two access and one comparison)
   2. Swap: 5
   3. Flag: 1
6. Return: 1

TOTAL: 9N2 -8N+3------O(N2)

Bubble Sort(Recursive):

1. Method is called N-1 times
2. Check 1;
3. For loop is executed based on n: N-1 for the first time, N-2 for the second time; N-k for the kth time
   1. Comparison: 3
   2. Swap: 5
4. Return

Total:  **Σ**(n-k)\*8+k

O(N2)

Selection Sort:

1. First for loop: n time;
   1. Set minIndex: tri
   2. Second for loop: j=I, n-j times
      1. Comparison: 3
      2. Set minIndex: 1
      3. Swap: 5
2. Return 1;

Since the second loop is already counting the first loop in and is called based on the first loop, we could say that the function altogether called **Σ**(n-j) times

TOTAL: **Σ**(n-j)\*9+1=n2-(9/2)n

O(N2)

Insertion Sort:

1. First for loop: n-1
   1. Second for loop: j=I; called I times;
      1. Comparison: 3
      2. Swap: 5
2. Return 1

TOTAL: n\*n(8)+1=8n2+1;

O(N2)

Merge Sort:

1. Declare int array;
   1. Comparison: 2
   2. Assign value: 2
2. Else:
   1. mid: 2
   2. left halve: call: 1
   3. right halve: call: 1
   4. return Merge method

Merge:

1. declare three int; 3
2. declare new array: 1 (length: left+right)
3. while loop (first execute)------10
   1. comparison 3
   2. if: 3
   3. temp assign: 2
   4. leftFirst++/rightFirst++: 1
   5. k++; 1
4. (only one will execute) while loop------5
   1. comparison: 1
   2. temp assign: 2
   3. leftFirst++; 1
   4. k++:1
5. (only one will execute) while loop------5
   1. comparison: 1
   2. temp assign: 2
   3. rightFirst++: 1
   4. k++:1
6. return :1

Total layers of mergeSort is logN-----------since the array is divided into two parts

Since all procedures are linear in both mergeSort and merge method, we will simplify it by only calculating the number of method being called;

In Merge method: length of temp array each time is logN\*(N/logN), therefore==N

TOTAL:17N\*(8+logN)

O(NlogN)

Quick Sort:

1. if statement: 1
2. split method called: 1
3. Quicksort (left subarray): 1
4. Quicksort (right subarray): 1

Split

1. declare splitpoint: 1; mid: 2; splitvalue: 2; swap: 5; declare Boolean: 1; first++:1; change Boolean: 1
2. while loop:
   1. if: 2
      1. first++:1
      2. check side: 2
   2. change bool: 1
3. check bool: 1
4. while loop:
   1. if: 2
      1. last--:1
      2. check side: 2
   2. change bool: 1
5. if statement: 1
   1. swap: 5:
   2. first++: 1
   3. last--:1
6. while: 1:
7. swap: 5
8. return 1;

Since all procedures are linear in quick sort and split method, we will simplify it by only calculating the numbers of method being called;

Number of splits: logN times-------since we are dividing the array into two parts

In SPLIT method: we are comparing every element with the splitting value, therefore, the comparison will roughly be run N times for each logN layer, by which means logN parts of divided subarrays

TOTAL: O(NlogN)