讲解code要点：

A**** 做题要求:

A complete answer will include the following:

1. Document your assumptions
2. Explain your approach and how you intend to solve the problem
3. Provide code comments where applicable
4. Explain the big-O run time complexity of your solution. Justify your answer.
5. Identify any additional data structures you used and justify why you used them.
6. Only provide your best answer to each part of the question.

```
// selection sort an array a[] with size n.
00 void SelectionSort(int a[], int n){
01    int global, temp;
02    for (int i = 0; i < n-1; i++) {//outer loop: how many iterations
03          global = i;
04          for (int j = i + 1; j < n; j++) {//inner loop: find the
global min from the rest elements.
05                if (a[j] < a[global]) {
06                   //record the index of the smallest element.
07                   global = j;
08                }
09          }
10          // swap the global (a[index]) min with a[i];
11          temp = a[i];
12          a[i] = a[global];
13          a[global]=temp;
14    }
15 }
```

Discussion:

什么是面试中一个类型的题?

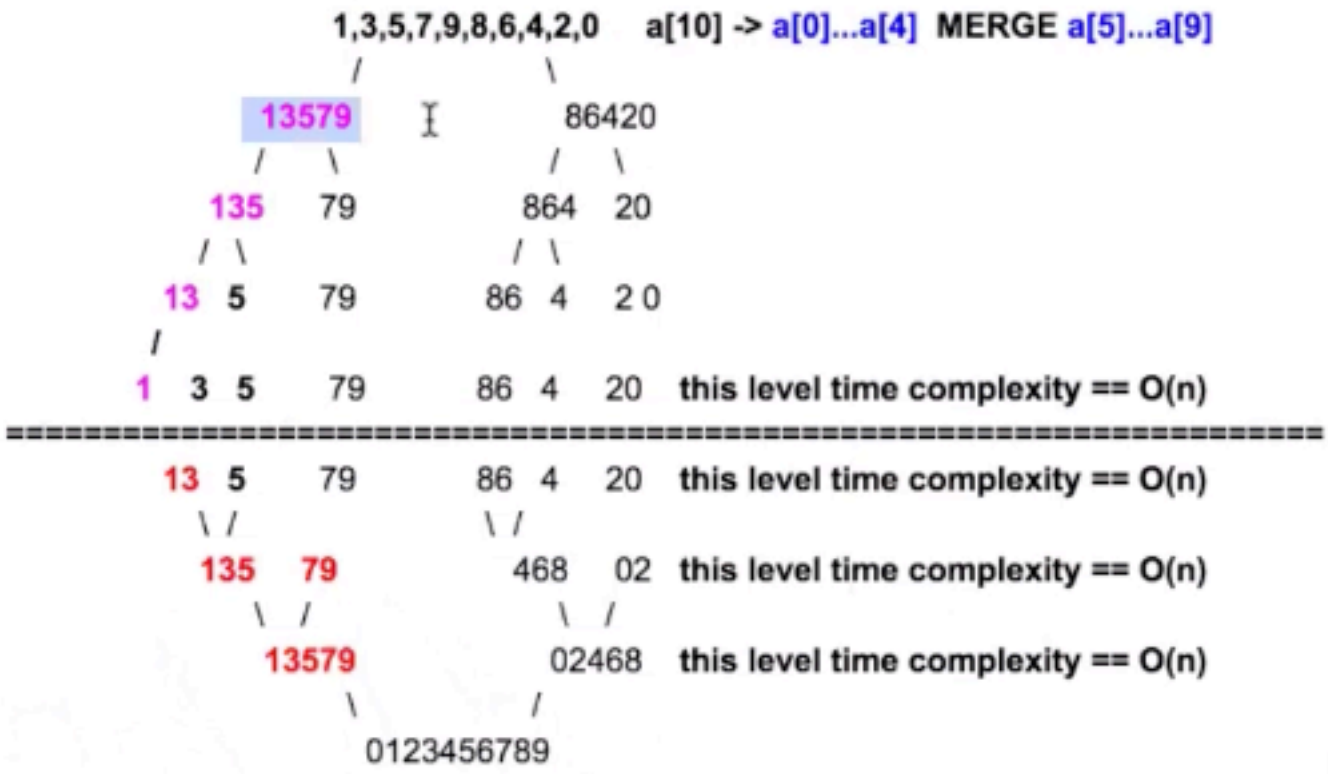
1. Given an array stored in Stack1, how to sort the numbers by using additional two stacks (will be discussed later in stack class)
2. Follow up, what if only 1 additional stack can be used?

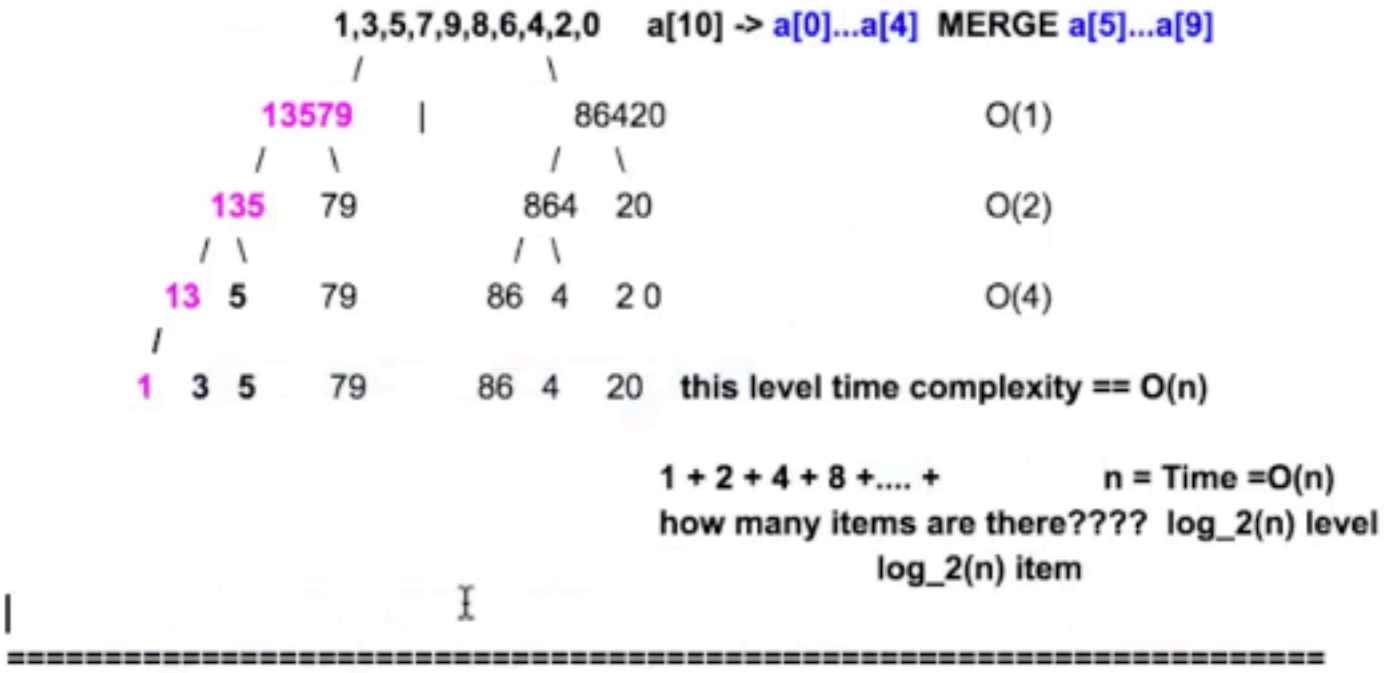
Stack1|| 3 2 4 ←

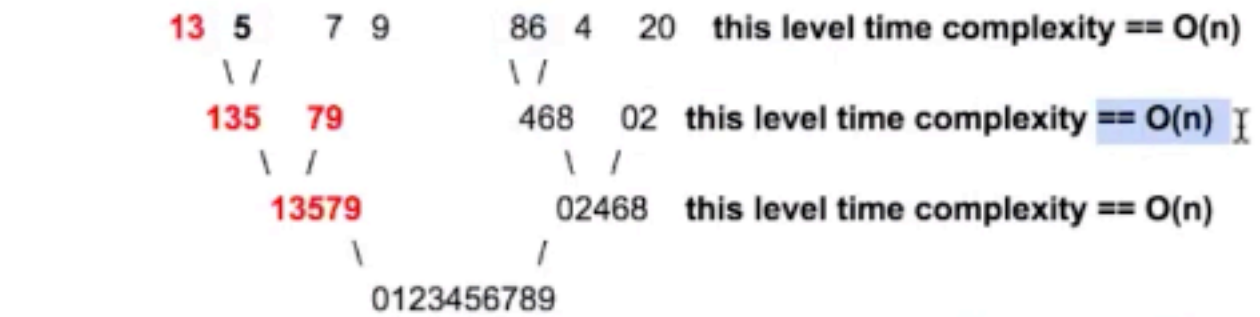
Stack2||

Stack3(final solution)|| 1

global_min = 1


1,3,5,7,9,8,6,4,2,0 a[10] -> a[0]...a[4] MERGE a[5]...a[9]
13579 86420
135 79 864 20
13 5 79 86 4 2 0
1 3 5 79 86 4 20 this level time complexity == O(n)
13 5 79 86 4 20 this level time complexity == O(n)
135 79 468 02 this level time complexity == O(n)
13579 02468 this level time complexity == O(n)
0123456789


1,3,5,7,9,8,6,4,2,0 a[10] -> a[0]...a[4] MERGE a[5]...a[9]
13579 | 86420 O(1)
135 79 864 20 O(2)
13 5 79 86 4 2 0 O(4)
1 3 5 79 86 4 20 this level time complexity == O(n)
1 + 2 + 4 + 8 +.... + n = Time =O(n)
how many items are there???? log_2(n) level
log_2(n) item



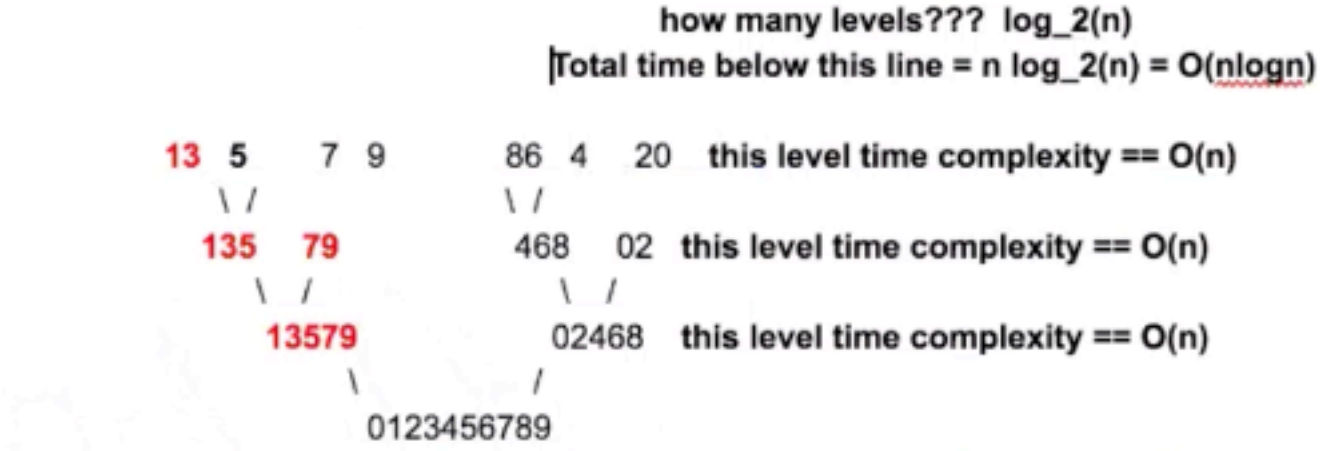

```
                                                          0          9
00 vector<int> mergesort (vector<int>& a, int left, int right) {
01    vector<int> solution;                  // store the final solution
02    if (left == right) {                   // base case
03          solution.push_back(array[left]);
04          return solution;
05    }
06    int mid = left + (right - left) / 2;      // mid is == 4

07    vector<int> solu_left = mergeSort(a, left, mid); //left:0 mid:4
08    vector<int> solu_right = mergeSort(a, mid + 1, right); //5    9
09    solution = merge(solu_left, solu_right); // shui xiao yi shui
10    return solution;
11 }
```

==========

how many levels??? log_2(n)

Total time below this line = n log_2(n) = O(nlogn)



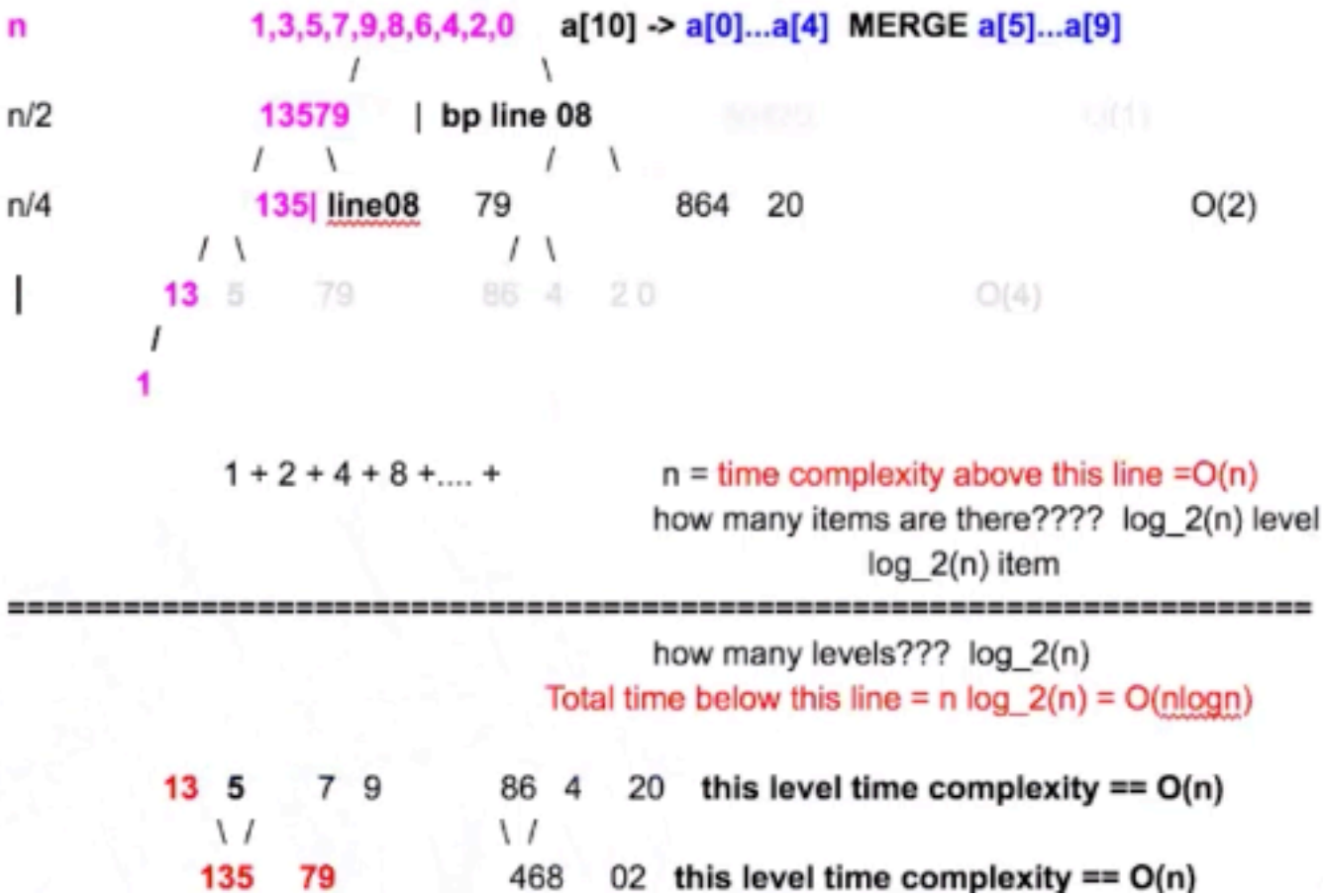
0 9

```
00 vector<int> mergesort (vector<int>& a, int left, int right) {
01    vector<int> solution;                  // store the final solution
02    if (left == right) {                   // base case
03          solution.push_back(array[left]);
04          return solution;
05    }
06    int mid = left + (right - left) / 2;      // mid is == 4

07    vector<int> solu_left = mergeSort(a, left, mid); //left:0 mid:4
```

```
                                                    0          9
00 vector<int> mergesort (vector<int>& a, int left, int right) {
01    vector<int> solution;                  // store the final solution
02    if (left == right) {                   // base case
03          solution.push_back(array[left]);
04          return solution;
05    }
06    int mid = left + (right - left) / 2;      // mid is == 4
07    vector<int> solu_left = mergeSort(a, left, mid); //left:0 mid:4
break point…….     // stop and store the local information.
```

Call_stack is a globally_accessible data structure that stores the local information of each level of recursion function call, such that, when we get back to this level, i still remember all local information in my history.


n 1,3,5,7,9,8,6,4,2,0 a[10] -> a[0]...a[4] MERGE a[5]...a[9]
n/2 13579 | bp line 08
n/4 135| line08 79 864 20 O(2)
13 5 79 86 4 2 0 O(4)
1
1 + 2 + 4 + 8 +.... + n = time complexity above this line =O(n)
how many items are there???? log_2(n) level
log_2(n) item
how many levels??? log_2(n)
Total time below this line = n log_2(n) = O(nlogn)
13 5 7 9 86 4 20 this level time complexity == O(n)
135 79 468 02 this level time complexity == O(n)