2013147513 조영재

1. We say that is if there are positive constants and such that



for . Since is , there exists such that for . Next, we can get by multiplying to the both side. So, we can say is such that which has as coeffiient.



2. Let us write definition of big-O again, in the form of this question's literal.

for , for ,



By adding two inequality, we can get

for some and .



So we can call is .



3. My algorithm is

*First, sort an array in descending order.*

*Second, select first five entries. That are the five largest elements.*

In second step, selecting the first five entries is because that is not related to the size of . So this runtime it is negligible. To minimize runtime, I will use the fastest sorting algorithm, radix sort, and its runtime would be full runtime of my algorithm.



Runtime of radix sort is , means maximum length of number.



4. Input :　A set array.

Output : Print subsets of the input set.

Subset of a set can be obtained by using recursive algorithm. For example, let . Subsets of S are subsets of and elements which are added to each subset. Next, we can obtain subsets of in the same way. Eliminating one of element, , subsets of are the subsets of and elements which are added to each subset.

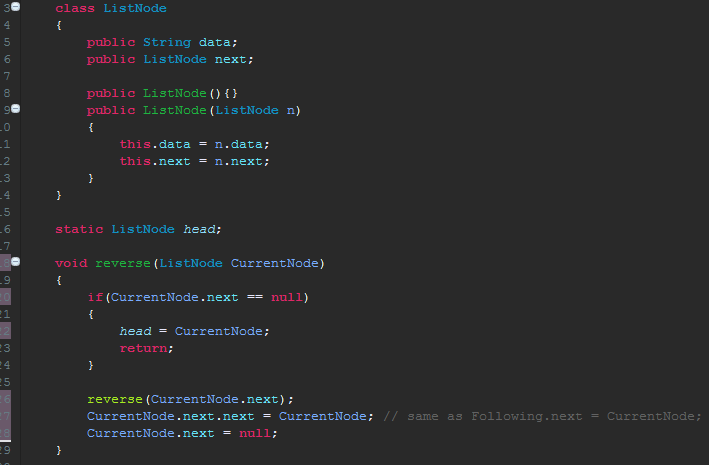


In short, subsets can be categorized by two group, one is subsets without a specific element of set , the other is subsets with a specific element of set . The former one can be found by using recursive method, the latter one can be found by adding specific element to the each obtained subsets in the former recursive method.





5. reverse() method



Input :　ListNode which pointed by head pointer.

Output : Reversed singly linked list.

1. First, check if this node points null. If so, execute first if statement which is exit condition. First if statement make head pointer point current node which points null point.

2. Second, if it doesn't satisfy exit condition, call recursive method whose input is CurrentNode.next, the next node of current node.

3. If returned by recursively called method, convert next node pointer to current node and make current node pointer to null.

6. Input : An array of comparable elements.

Output : An array of value



1. Use one counter variable, C, initialized as 0.

2. Compare one and the next entries from the first two entries to the last two entries. In comparing, increase counter by 1 until satisfies. If does not be satisfied,



3. If does not be satisfied, save counter value into a stack. and set counter as 0.



4. From that point, restart counting until is satisfied. If does not be satisfied again, compare counter value and previous counter value in the stack. Select a bigger one and save the value in the stack.



5. Number of operations of the best case will be times, and those of worst case will be times. So big-O will be .

