Homework 6

M1522.000900 Data Structure (2019 Fall)

2013-12815 Dongjoo Lee

Question 1.

(1) [0, 1, 2, 4, 5, 7, 8]
(2) Θ(n+k)
(3) Θ(n+k)
(4) Yes

Question 2.

Suppose there were 2^k size of input array. When applying ordinary merge sort, array splits into half k times to be remained 1-size array.
By this, at every steps, we can drive relation between N number of subarray with size M like below.

<0th step>	N=2^0, M=2^k	⋯ (2^0) number of subarrays with size 2^k
<1st step>	N=2^1, M=2^(k-1)	⋯ (2^1) number of subarrays with size 2^(k-1)
……		
<(k-1)th step>	N=2^(k-1), M=2^1	⋯ (2^0) number of subarrays with size 2^k
<kth step>	N=2^k, M=2^0	⋯ (2^k) number of subarrays with size 2^0

When suppose that we do calls even with the 1-size array, then, ΣN be the calls to the merge sort function. So, <u>ΣN = 2^(k+1) -1 = (size of array) * 2 - 1 ⋯ (1)</u>

By the way, when applying optimized merge sort, then, we do calls insertion sort instead of merge sort from the moment (M/2) ≤ Threshold. The steps that would be substituted be the last part above.

Back to the question, merge sort calls means sum of number of subarrays above, so after merge sort is over there remain 2^m sub-arrays(∵ (1)).

∴ (n+1) or 2^m calls will there be to insertion sort.

Question 3.

When size of array to be sorted is n, then, there will be $\log_{(99/100)}(n)$ sublists. At each level, there is needed $\Theta(n)$ for visiting all indexes. So, the total time complexity is,

$T(n) = \Theta(n \cdot \log_{(99/100)}(n)) = \Theta(n \cdot \log(n) / \log(99/100)) = \Theta(n \cdot \log(n))$. Q.E.D.

Question 4.

0(A)	→	TAB	→	BAR	→	EAR	→	TAR						
1(B)	/													
2(C)	/													
3(D)	/													
4(E)	→	SEA	→	TEA										
5(F)	/													
6(G)	/													
7(H)	/													
8(I)	→	DIG	→	BIG										
9(J)	/													
10(K)	/													
11(L)	/													
12(M)	/													
13(N)	/													
14(O)	→	MOB	→	DOG	→	COW	→	ROW	→	NOW	→	BOX	→	FOX
15(P)	/													
16(Q)	/													
17(R)	/													
18(S)	/													
19(T)	/													
20(U)	/	RUG												
21(V)	/													
22(W)	/													
23(X)	/													
24(Y)	/													
25(Z)	/													

Question 5.

(1) (1,5), (2,5), (3,4), (3,5), (4,5)

(2) Array from largest to smallest inverce-sorted. A' = [n, n−1, ⋯ , 2, 1]

$(n-1)+(n-2)+\cdots+1 = \Sigma(k=1 \text{ to } n-1)\ (k) = \frac{1}{2}n(n-1)$.

(3) Let's assume that input array has size of n. Insertion sort's best case is pre-sorted input array. In this case, swap occurs 0 time and comparison occurs (n−1) times. On the other hand, by using result of (2), worst case is invert-sorted input array. In this case, swap occurs $\frac{1}{2}n(n-1)$ times and comparison occurs same times, $\frac{1}{2}n(n-1)$.

In terms of the number of inversions, let this by I. And let the number of comparisons C. I is equals to the number of swaps that would be performed. By using the relation between swaps and comparisons above, we can result the relation between C and I.

$$\therefore I \le C \le I+(n-1)$$