Assignment3

NetID = lh2537

2021年10月3日 22:26

Q1.

a) step1:

1	2	3	4	5	6	7	8
19	5	9	52	26	35	61	28

step2

1	2	3	4	5	6	7	8
5	19	9	52	26	35	61	28

step3

1	2	3	4	5	6	7	8
5	9	19	52	26	35	61	28

step4

1	2	3	4	5	6	7	8
5	9	19	52	26	35	61	28

step5

1	2	3	4	5	6	7	8
5	9	19	26	52	35	61	28

step6

1	2	3	4	5	6	7	8
5	9	19	26	35	52	61	28

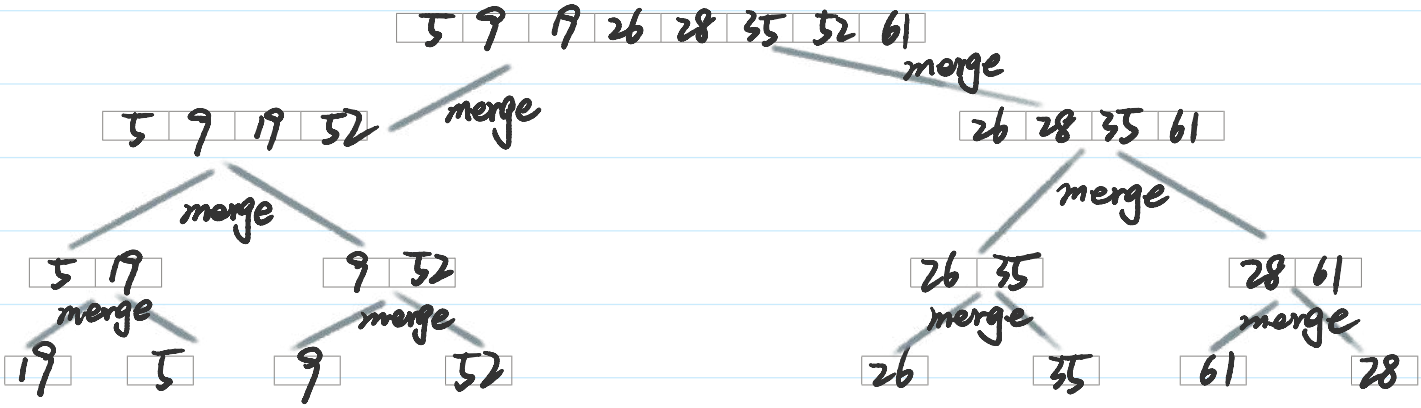
step7

1	2	3	4	5	6	7	8
5	9	19	26	35	52	61	28

step8

1	2	3	4	5	6	7	8
5	9	19	26	28	35	52	61

b)



Q2.

a) The pseudocode:

```
1 For i = 1 to A.length - 1:
2     minPos = i
3     For j = i+1 to A.length:
4         if A[j] < A[i]:
5             minPos = j
6     Swap(A[i], A[minPos])
```

b)
Loop invariant: At the start of each iteration in line 1:
A[1]...A[i-1] is sorted. Besides, A[1]...A[i-1] is smaller than any elements in A[i], A[i+1]....A[A.length]

c)
After n-1 iteration, A[1], A[2], ..., A[n-1] consists of the smallest n-1 element with sorted order. Since A[n-1] < A[n], A[n] is the largest element in the array

d) Best-case: $\Theta(n^2)$ Worst-case: $\Theta(n^2)$
Running Time: $\Theta(n^2)$
For each iteration, the program will go through the rest array to find the smallest number in the rest, the running time will be the same for all cases no matter best or worst.

Q3.
The algorithm should be like:
① Divide the array in three equal length halves: (left half & mid half & right half)
② For the following cases, return their maximum subarray respectively:
case 1: The maximum subarray is in the left half. (recursion)
case 2: The maximum subarray is in the right half. (recursion)
case 3: The maximum subarray is in the middle half (recursion)
case 4: The maximum subarray is in the left + middle half.
case 5: The maximum subarray is in the middle + right half.
case 6: The maximum subarray is in the left + middle + right half
Case 1 & case 2 & case 3 can simply solved by recursion. As for case 5: it can

be solved directly in linear moving time: start from $(mid_{end}+1)$ and add elements iteratively from index $(mid_{end}+1)$ to the end of the array. Find the index position = $Right_{pos}$ maximize Sum_{right}. Use the same method to find $Left_{pos}$. Case 5 is Done. Case 4 and Case 5 are the same. As for Case 6 start from mid+1 and go through until the end, start from $(mid_{end}+1)$ to the end of the array. As for the left, start from $(mid_{head}+1)$ to the start of the array. Find the index position so case 6 is done

Finally, find the maximum subarray among case 1, case 2, case 4, case 5, and case 6.

Q4.

```
MOVE(n, start, mid, end)
if n==1:
    PRINT(start, end)
else:
    MOVE(n-1, start, end, Mid)
    MOVE( 1, Start, Mid, end)
    MOVE(n-1, Mid, start, end)
```

Q5.

```
BinarySeparate(A)
LargeList = []
SmallList = []

For i = 1 to A.length:
    if i mod 2 == 0:
        num1 = A[i]
    else:
        num2 = A[i]
        LargeList.append(max(num1, num2))
        SmallList.append(min(num1, num2))
```

```
    num1 = -∞
    num2 = -∞
if num1 != -∞:
    LargeList.append(num1)
max = FindMax(LargeList)
min = FindMin(SmallList)
return max, min

FindMax(A)
if A.length == 1:
    return A[1], A[1]
else:
    Mid = ⌊A.length/2⌋
    max1 = FindMax(A[Mid+1], ..., A.length])
    max2 = FindMax(A[0, ..., Mid])
    max = max(max1, max2)
    return max

FindMin(A)
if A.length == 1:
    return A[1], A[1]
else:
    Mid = ⌊A.length/2⌋
    min1 = FindMin(A[Mid+1], ..., A.length])
    min2 = FindMin(A[0, ..., Mid])
    min = max(min1, min2)
    return min
```

① To separate array and divide into large and small List, it takes $N/2$ comparasion. As for the FindMin and FindMax function

$$T(n) = \sum_{i=0}^{\lg n - 1} 2^i = n - 1 \leq n.$$

② So that: FindMin and FindMax take comparation $\leq N$

① + ② $\leq \frac{3N}{2}$