CS203 Data Structure and Algorithm Analysis   Mid-term 2024Fall

Student ID:_________ Student Name:___________

Lab:☐ Tue.5-6 Tang ☐ Tue.5-6 Wang ☐ Tue.5-6 Shen

☐ Wed.3-4 Wang ☐ Wed.3-4 Shen ☐Wed.5-6 Wang ☐ Wed.5-6 Shen

Part I. Filling-blank question [30 marks, 3 marks for each question]

1. Given an array **A** with size **n**, suppose there are **n** integers store in **A[0], A[1], A[2],…, A[n-1]**. If we delete a continuous interval from **A[i]** to **A[j] (0<i<j<n)**, how many integers are left in A ? ______n-(j-i+1)_________ .

2. What is the time complexity of the following method _____O(logn)________.

```
public static void A(n) {
        int i=1;
        while(i<n) i=3*i;
}
```

3. Suppose that we use an array **A [0, … , m]** to store the elements of a circular queue. there are __B___ elements in this queue. (If **front** == **rear**, the queue is empty.)

   A. m-1 B. (rear-front+m+1) mod (m+1) C. (rear-front+m) mod m D. None of above

4. Given following pseudocode of mergesort:

```
Merge-Sort(A[], L, R) {
    if (L==R) return;
    mid=(L+R)/2;
    Merge-Sort(A, L, mid);
    Merge-Sort(A, mid + 1, R);
    Merge(A, L, mid, R);
}
```

   Sort the array A : **g, b, d, c, a, f, e, h** in the smallest lexicographical(字典序) using the above merge sort by **Merge-Sort(A, 0, 7)**, after the third time of return from calling function **Merge**, the array will be ____bcdgafeh______.

5. Next array of "aab" is 0,1,0. The next array of "cddeedccd" is ___0,0,0,0,0,0,1,1,2___.

6. The result of postfix expression "**9 3 1 - 3 × + 10 2 ÷ +** " is ____20_____.

7. There is an array with **n** integers, if you want to check whether an integer **K** is in this array use binary search, the time complexity is __O(nlogn)______.

8. There is a monotonically(单调) decreasing stack to be maintained, if the push-in sequence is **7 3 2 5 4 8**, the pop-out order is ________2 3 4 5 7_________. (All elements should be pop-out at the end)

9. In a doubly linked list, what is the operation of inserting a node **q** after node **p**?

   1.__________p.next.pre = q______; 2.____________q.next=p.next_____________;

   3.___________q.pre = p__________; 4.____________p.next = q_____________;

10. The push-in sequence of a stack is 7 6 2 5 4. Which of the following pop-out sequence(s) **is/are** possible? ___ABD___.

    A. 2 4 5 6 7 B. 7 6 5 4 2 C. 6 7 4 2 5 D. 6 2 7 4 5

Part II. Short answer question [24 marks]

1. **[6 marks]** Given a search pattern string "**ababacaba**" (index starts at 0), please finish the following tables of a FSA constructed for string matching:

	0	1	2	3	4	5	6	7	8
a	1	1	3	1	5	1	7	1	9
b	0	2	0	4	0	4	0	8	0
c	0	0	0	0	0	6	0	0	0

j	Pattern[1...j]	X
0		0
1	b	0
2	ba	1
3	bab	2
4	baba	3
5	babac	0
6	babaca	1
7	babacab	2

2. **[6 marks]** Given a String A = "abcdef", use rabin-karp to calculate the hash number **p** of all the sub-strings of A with length 3. Write down the calculation process. (Suppose the number **p** of "a", "b", "c", "d", "e", f is 0, 1, 2, 3, 4, 5, 6; the radix is **26** and the prime number to take mod is **15131**)

   substr: "abc","bcd","cde","def"
   R2= 26*26=676
   "abc":"a" 0×26=0->"ab" (0+1)×26=26 ->"abc" (26+2)%15131=28
   "bcd":((28-0×R2)×26+3)%15131=731
   "cde":((731-1×R2)×26+4)%15131=1434
   "def":((1434-2×R2)×26+5)%15131=2137

3. **[6 marks]** Given a sequence [2,6,3,5,1,4], use selection sort and insertion sort to sort it. Write down the whole sequence after each traversal.

   selection sort:
   1. 2 6 3 5 1 4(original array)
   2. 1 6 3 5 2 4
   3. _1 2 3 5 6 4_
   4. _1 2 3 5 6 4_
   5. _1 2 3 4 6 5_
   6. 1 2 3 4 5 6

   insertion sort:
   1. 2 6 3 5 1 4(original array)
   2. 2 6 3 5 1 4
   3. _2 3 6 5 1 4_
   4. _2 3 5 6 1 4_
   5. _1 2 3 5 6 4_
   6. 1 2 3 4 5 6

4. **[6 marks]** Design a queue by using a circular doubly linked list with only one pointer **head**. Please implement the "enqueue" , "dequeue" and "size" functions and analyze the time complexity of these three functions. If head is **null**, the queue is empty. (Code or pseudocode is OK)

```
node{
    int val;
    node prev, next;
}
Linkedlist{
    node head;
    enqueue (int k){
    //your answer
    if(head=null){
        head.next = head;
        head.pre = head;
    }
    else{
        node n = new node();
        n.val = k;
        head.pre.next=n;
        n.pre=head.pre;
        n.next=head;
        head.pre=n;
    }
    }

    dequeue(){
        if(head == null) return -1;
        int x = head.val;
        //your answer
        if(head.size()==1)head=null;
        node tmp = head;
        head = head.next;
        tmp.next.pre = tmp.pre;
        tmp.pre.next = tmp.next;

        return x;
    }

    size(){
        if(head == null) return 0;
        //your answer
        if(head.next==head)return 1;
        node iter = head;
        iter = iter.next;
        int cnt = 1;
        while(iter!=head){
            iter=iter.next;
            cnt++;
        }return cnt;
    }
}
```

Part III. Algorithm [46 marks]

Note: Describe the algorithm in words and analyze the complexity of the algorithms in this part. Don't write code or pseudocode.

1. **[10 marks]** Given an array A with **n** integers. Design an algorithm to calculate how many pairs **<i,j>** satisfy that **a[i]+a[j]>m** && **i < j**. (**m** is a given constant integer.)

Because sorting does not affect the total number of pairs, mergesort A in ascending order. Traverse from A[1] to A[n], for each A[i], use binary search to find the A[j] with the smallest j satisfied a[j]>m-a[i] in A[i] to A[n] and the pairs for A[i] is (n-j+1). Sum up the pairs for each A[i] is the answer.

mergesort O(nlogn)

Traverse and binarysearch and sum up O(n*(logn+1))=O(nlogn)

Time complexity: O(nlogn)

2. **[12 marks]** Given a sequence **A** of **N** integers ($1\le A[i] \le 100000$). Now we need to divide the sequence **A** into **M** continuous sub-intervals(子区间) (without any intersection between sub-intervals, sub-interval can have 0 element) and the **M** sub-intervals combined form the entire **A** sequence. For the **ith** sub-sequence, the sum of all its elements is denoted as **sum[i]**(0<=i<m). Design an algorithm to find the minimum value of $\max_{0\le i<m} sum[i]$ among all possible values. *(For example, A={1,2,3,4,5}, M=3, A can be divide to {1,(2,3),(4,5)} or {(1,2,3),(4,5),()} or {(1,2),(3,4),(5)} or ...)*

Given a value ans, design check() to verify whether A[i] can be divided into M consecutive subintervals such that the maximum subinterval sum is equal to ans.

Check(): Traverses A[i] from A[1] to A[n] and use cnt to record the number of subintervals. Starting from the first subinterval, A[i] is added to the sum of the current subinterval. If the sum exceeds ans, create a new subinterval, update the new subinterval sum. cnt is incremented. After A is traversed, if cnt > M, it means that ans cannot be reached return false, and if cnt <= M, then ans can be reached,return true.

Since if ans can be reached, then all values greater than ans can be reached, and values less than the smallest ans cannot be reached, we can use the result of check to perform binary search on the answers in [1,the maximum possible ans]. Mid is the average of the left and right bounds. If check(mid) is true, then the answer is in[left bound, mid], If check(mid) is false, then the answer is in [mid+1,right bound]. Perform binary search on the new answer interval,until the left and right bounds are reduced to a unique answer, and the value is the desired answer.

check(): O(n)

binary search: O(log(sum of A[i]))

Time complexity: O(n*log(sum of A[i]))

3. **[12 marks]** Given a sequence **A** of **N** integers. Design an algorithm, for each **i** in the sequence, find the smallest **k** such that **A[k] > A[i]** and **k > i**.

Using a descending monotonic stack, traverse the array A, for each A[i], if A[i] is greater than the top of the stack, then i is the corresponding k of the stack top, the stack top is popped, and the comparison between the stack top and A[i] is repeated until the stack top is greater than or equal to A[i], then push A[i] into the stack. After traversing A, the remaining elements in the stack have no k.

For each element, it will be only popped and pushed once by the stack, so the overall complexity is O(n).

4. **[12 marks]** Given a sequence **A** of **N** integers. For each **i** in the sequence, **median[i]** is the Median(中位数) of **{A[0], A[1], A[2],..., A[i]}**(Median is the $\left\lfloor\frac{i}{2}\right\rfloor$-th smallest number of the array). Design an algorithm to find **median[]** array($0 \le i< N$).

Create an array of nodes by A, and store A[i] as the val of the nodes, and copy the array A into an array B. Sort the array B based on the val of the nodes. Link the sorted B array into a doubly-linked list.

Traverse i from N-1 to 0. When i=N-1, median[i] is B[i/2].val, and node mid is B[i/2]. Before i--, compare A[i].val and mid.val, and if A[i].val<mid.val, consider median[i] is mid.val or mid.prev.val, update mid; otherwise, consider mid.next, update mid. Determine the new mid and delete A[i] from linked-list, mid.val is the value of median[i-1]. Repeat the above

process to obtain median[].
Sort: O(nlogn)
Traverse and delete:O(n)
Total: O(n)