CS203 Data Structure and Algorithm Analysis   Mid-term 2024Fall

Student ID: 12312710 Student Name: [illegible]   94

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]

27

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 ____ 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 ~~[b,c,d,g]~~ [b,c,d,g,a,f,e,h]

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(logn), if array is monotonical; ϕ, if array isn't monotonical, we c

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 2,3,4,5,7,8. (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.prev=q ; 2. q.next=p.next ;
3. p.next=q ; 4. q.prev=p ;

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

5↑ 4↑ ↑ 6/2 7

Part II. Short answer question [24 marks]

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

23

	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	Null	0
1	"b"	0
2	"ba"	1
3	"baba"	2
4	"baba"	3
5	"babac"	0
6	"babaca"	1
7	"babacab"	2

"babacaba" ...3

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, ; the radix is **26** and the prime number to take mod is **15131**)

① for "abc", $(0\times26^3+1\times26^2+2\times26) \bmod 15131 = 728$

② for "bcd", $((728-26^3\cdot0)\times26+26\times3) \bmod 15131 = 3875$

③ for "cde" $((3875-26^3\cdot1)\times26+26\times4) \bmod 15131 = 7022$

④ for "def" $((7022-26^3\cdot2)\times26+26\times5) \bmod 15131 = 10169$.

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 4 6 5~~ 1 2 3 5 6 4
5. 1 2 3 4 5 6
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

n't use binary.

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 = new Node.
    head.val = k
} else { node temp=head
    while(temp.next!=null){
     temp= temp.next
    }
    temp.next=new node
    temp.next.val=k
    temp.next.prev=temp
}
```

```
dequeue(){
    if(head == null) return -1;
    int x = head.val;
    //your answer
    head = head.next
    head.prev = null
    return x;
}
```

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

}

+4

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.)

+10

① mergesort the A array

② for A[0], find i that A[i] > m - A[0] & A[i-1] ≤ m - A[0] in subarray A[1...n] by using binary search. {answer += n-i+1, if find; answer += 0, if not find.

③ for A[j], find i that ... in subarray A[j+1...n], answer add as same prin

Analasis: mergesort = O(nlogn)

each binary search: O(log n), n times of binary search: O(nlogn)

∴ whole time complexity is O(nlogn)

2. **[12 marks]** Given a sequence **A** of **N** integers (**1≤ A[i] ≤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 **i**th 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 ...)*

+10

- this idea will use the binary search of the answer.

Step 1: since $1\le A[i]\le 100000$, the answer is bounded in [N, 100000N]

Step 2: Let left=N, right=100000N, mid=(left+right)/2.

then, sum A[0],A[1]..A[i] until ~~sum~~ $\sum_{i=0}^{m} A[i] \le mid$, $\sum_{i=0}^{m+1} A[i] > mid$. count++,

do this count until finish the A[N]. if count > ~~M~~ M. then left = mid+1

if count ≤ M, then right = mid.

...

Step n: do until left ≥ right, then the answer is "left". for each cut, O(n)

for binary search O(log n)

∴ the whole is O(n log n)

+12

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**

- this idea will use the feature of ~~Linked list : easy to delete~~ ~~stack queue~~ monotonically decrease stack.

① create a node array=N, N[0] = New Node(index 0, data A[0])

N[1]= New Node(index 1, data A[1]) ....and so on.

② push N[0] to the stack ③ ~~push~~ if N[1].data > N[0].data.

then we find N[0].index's smallest k is N[1].index

then pop N[0], push N[1]. if not, just push N[1]

④ push N[i] to the stack,

before pushing, pop the element smaller than N[i] to make the stack MONOTONically decreasing

+9

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 $\lfloor \frac{i}{2} \rfloor$-th smallest number of the array). Design an algorithm to find **median[]** array(**0 ≤i< N**).

for the element popped, i is the wanted "k"/(cost not popped now)

for each element is only push&pop once, the time complexity is O(n)

- this idea is similar to insertion sort:

Step 1: A[0], itself is the Median. median[0] = A[0]

Step 2: A[0].A[1], Compare two and put bigger one on the right. median[1] = left one.

Step 3: {A[0], A[1]}, insert A[2] into it to make array monotonically increase.

median[2] = middle one

...

Step i+1: {A[a], A[b], A[c]...A[j]} insert A[i] into it.

monotonically increasing. the median[i] = $A[\lfloor \frac{i}{2} \rfloor]$

ble.

for each insert, it cost O(n) to insert an element into a increasing array., O(1) to get median

and there's (n-1) times of insertions. ∴ the whole algorism's time complexity is $O(n^2)$