南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

2018-2019 Academic Year
Fall Semester
Midterm Exam Paper

**Course Name: CS203 Data Structure and Algorithm Analysis**

**Department: Department of Computer Science and Engineering**

**Exam Duration: 2 hours**

## Part I. Filling-blank question [40 marks]

1. What is the time complexity of deleting a node **p** from a linked list? O(1) [2 marks]

2. The order of pushing six integers into the stack is 6 5 4 3 2 1. Which of the following pop-out sequence is impossible? C [2 marks]

   A. 5 4 3 6 1 2 B. 4 5 3 1 2 6 C. 3 4 6 5 2 1 D. 2 3 4 1 5 6

3. Suppose we implement a ring queue by an array A with size **m**, where **rear** and **front** are the index of the rear, and the front of queue, respectively. Given the value pair of **rear** and **queue**, how many elements are in the queue? A [2 marks]

   A. (rear-front+m) %m B. rear-front+1 C. rear-front-1 D. rear-front

4. What is the time complexity of the following method A .[2 marks]

```
method(n)
{
    x=2;
    while ( x < n/2 )
        x = 2*x;
}
```

   A. O(logn) B. O(n) C. O(nlogn) D. $(n^2)$

5. The average time complexity of quick sort algorithm is **1.** $O(n^2)$ .
   The worst-case time complexity of quick sort is **2.** O(nlogn) .[2 marks]

6. Given an array **A** with size **m**, suppose there are n (n<m) integers store in **A[0], A[1], A[2],…, A[n-1]**. If we insert integer **k** into **A[i] (i<n)**, we need to move

n-i integers. [2 marks]

7. In order to solve the problem of the speed mismatch between computer and printer, we set a print data buffer. The computer writes data into the buffer in sequence, and the printer prints the data in the same sequence. The data structure of the buffer should be Queue.[2 marks]

8. Suppose we want to push **n** numbers {**1, 2, 3, …, n-1, n**} into a stack in sequence and we can execute pop out at any time. If the second number we popped out is 3, how many possible values the third popped number could be?_________[2 marks]

9. Suppose **p** is a node of Doubly Linked List **L**, **p->prev** point to the previous node of **p** and **p->next** point to the next node of **p**. How to delete node **p** from **L** : [2marks]

**1.** p <- p->prev ;

**2.** p->next <- p->next->next ;

10. Suppose **A** is an array with 1536 sorted numbers (in ascending order). If we use binary search to find a number **k** (which does not exist in **A**), we will do ________ times of comparisons at most. [2 marks]

11. Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced). C [2 marks]

A. Quick Sort B.Merge Sort C.Insertion Sort D.Bubble Sort

12. The postfix expression of (4 - 3 * 2 ) * (6 - 5 - 1) is 432*-65-1-* .[2 marks]

13. To a string **P**="acabac", please fill in the following transition function table of Finite State Automata (For one cell, its column means the current state and its row means the current input. The number in the cell means the next state.) [6 marks]

| State | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| a | 1 | 1 | 3 | **1.** ___ | 5 | **2.** ___ | **3.** ___ |
| b | 0 | 0 | 0 | 4 | **4.** ___ | 0 | 0 |
| c | **5.** ___ | 2 | 0 | **6.** ___ | 0 | 6 | 0 |

14. Method A and method B are two implementations of binary search algorithm. Both of A and B can be used to decide whether an integer **k** exists in an ascending size-n array **Arr** or not. Please complete method A and B. [10 marks]

**Method A:**

```
Binarysearch(n, Arr, k)
{
    return A(n, Arr, k);
}
A(n, Arr, k)
{
    left ← 0;
    right ← 1.______________;
    mid;
    if( k < Arr[0] || k > Arr[n-1] )
        return false;
    while( 2.________________ )
    {
        mid ← 3.________________;
        if( 4.________________ )
            return true;
        else if( Arr[mid] > k )
            5.________________;
        else
            right ← mid - 1;
    }
    return false;
}
```

**Method B:**

```
Binarysearch(n, Arr,k)
{
    left ← 0;
    right ← 6.____________________;
    if( k < Arr[0] || k > Arr[n-1] )
        return false;
    return B( Arr, k, left, right);
}

B(Arr, k, left, right)
{
    if( 7.__________________ )
    {
        mid ← 8.________________;
        if( Arr[mid] == k )
            return true;
        else if( Arr[mid] > k )
            9.__________________;
        else
            right ← mid - 1;
    }
    else
        return false;
    return 10.__________________;
}
```

## Part II. Algorithm Design [60 marks]

**Note:** For each question in this part, Please design a correct algorithm for the given problem and write down the pseudocode of your algorithm [75%].

(1) You can only use the provided variables.

(2) You are NOT allowed to declare any new variables or methods.

Please analyze the time complexity of each function in your algorithm [25%].
We only give full marks to those solutions with optimal time complexity.

1. We want to design a queue by using two stacks. Please implement the “enqueue” and “dequeue” functions. You can and only can use pop(), push(x), isEmpty() in Stack. The time complexities of the three provided function are O(1). [12 marks]

```
Stack S1;
Stack S2;

enqueue ( k )
{

    ①

}

dequeue()
{

    ②

    return S2.pop();
}
```

2. Suppose **L** is a linked list, each node in **L** contains a value **v** and a **next** pointer, which points to its next node. Suppose pointer **head** is pointing to the head of **L.** Given you two pointers **p1** and **p2**, and linked list **L** with **head** pointer, please design an algorithm to reverse **L**. Please fill the method below. [12 marks]

```
 reverse( node*head, node*p1, node*p2 )
 {










        return head;                //return the head of reversed L
 }
```

3. Suppose E is an integer array with n integers {E[0], E[1], E[2],…, E[n-1]}. We define OE Sort as: given an array E, the even numbers in it are in the first part of sorted array E, and the odd numbers in it are in the second part of sorted array E.
   Please design an algorithm to sort E with OE Sort. [12 marks]

```
sort( int E[], int length, int i, int j, int k )
{












       return E;
}
```

4. Given a String **S** and its length **n**, design an algorithm to calculate how many prefixes of **S** are also a suffix of **S**. [12 marks]

```
calpresuffix(char S[], int next[], int n, int count, int i, int j)
{

        return count;
}
```

5. Given n words and the length of **i**-th word is **W[i]**. We want to print these **n** words in **m** lines. Suppose **Lmax** is the length of the longest in these **m** lines. Please design an algorithm to print these n words into m lines with a minimum **Lmax**. [12 marks]

   Please note that:

   (1) One word cannot be printed in two lines
   (2) No whitespace between two words
   (3) **tlength** is the total length of n words, i.e., **tlength = W[0] + W[1] + … + W[n-1]**
   (4) Return value is **Lmax**

```
longestline( int i, int j, int k, Boolean T, int left, int right, int mid, int Lmax, int m,
int n, int tlength, int W[] )
{









    return Lmax;
}
```