

Student ID: _____ Student Name: _____

CS203 Data Structure and Algorithm Analysis
Quiz 1

Note 1: Write all your solutions in the question paper directly. You can ask additional answer paper if necessary

Note 2: If a question asks you to design an algorithm, full points will be given if your algorithm runs with optimal time complexity

Note 3: If a question asks you to design an algorithm, you should **first** describe your ideas in general words, **then** write the pseudocode, and **end** with time complexity analysis.

Problem 1 [25 points, 5 per points]

1) Which of the following function is not $O(n^{2.5})$ ()

- A. $\frac{n^{100}}{2^n}$ B. $(\log_2 n)^{98}$ C. $938593729n^2$ D. $n^{2.6}/\log^2 n$

2) Which of the following functions is $O(n \log \sqrt{n})$ ()

- A. $358 \cdot n \log_2 n$ B. $n^{1.2}/\log^5 n$ C. $(1.01)^n$ D. $n \cdot (\log_2 n)^{1.0003}$

3) There are 5 elements, which are pushed into the stack by the order e_1, e_2, e_3, e_4, e_5 . If e_3 is the first element be popped and e_4 is the second, in this case, please list out all the possible order of pop stack:

- A. e_3, e_4, e_5, e_2, e_1 B. e_3, e_4, e_2, e_5, e_1 C. e_3, e_4, e_2, e_1, e_5 D. e_3, e_4, e_5, e_1, e_2

4) Stack s is empty, use s to convert the infix expression $6/2 + (8*9 - 3*5)/3$ to equivalent postfix expression. In the process, when the 5 is scanned, what are the elements in the stack (from stack bottom to top)?

- A. $+(*-$ B. $+(-*$ C. $/+(*-*$ D. $/+--*$

5) Suppose $f(1) = 1; f(2) = 2; f(n) = 3 + f(n - 2)$. $f(n) =$ **$O(n)$** . (in terms of Big-O notation).

Problem 2, 25 points Let $S1$ be an unsorted array of n integers, and $S2$ is another sorted array of $\log_2 n$ integers (n is a power of 2, **$S2$ is in descending order**). Describe an algorithm to output the number of pairs (x, y) satisfying $x \in S1$, $y \in S2$, and $x \leq y$. Your algorithm must terminate in $O(n \log \log n)$ time. For example, if $S1 = \{10, 7, 12, 18\}$ and $S2 = \{15, 7\}$, then you should output 4 because 4 pairs satisfy the required conditions: $(10, 15), (7, 15), (12, 15), (7, 7)$.

Idea: (10 marks)

For every element $x \in S1$, perform binary search on $S2$ to find the number t_x of elements in $S2$ that are larger than or equal to x . Return $\sum_{x \in S1} t_x$.

Pseudocode: (10 marks)

Algorithm CountPairs($S1, S2$)

1. $n \leftarrow \text{len}(S1)$
2. $\text{sum} \leftarrow 0$ // the total number of pairs
3. **for** $i \leftarrow 0$ to $n-1$
4. $\text{sum} += \text{findPairs}(S1[i], S2)$
5. **return** sum

Algorithm findPairs($S1[i], S2$)

1. $\text{left} \leftarrow 0, \text{right} \leftarrow \text{len}(S2)$
2. **repeat**
3. $\text{mid} \leftarrow (\text{left} + \text{right}) / 2$
4. **if** $(S1[i] = S2[\text{mid}])$ **then**
5. **return** mid
6. **else if** $(S1[i] < S2[\text{mid}])$ **then**
7. $\text{left} \leftarrow \text{mid} + 1$
8. **else**
9. $\text{right} \leftarrow \text{mid} - 1$
10. **until** $\text{left} > \text{right}$
11. **if** $(\text{right} = 0)$ // all values are larger than $S1[i]$
12. **return** right
13. **if** $(\text{left} = \text{len}(S2))$ // all values are smaller than $S1[i]$
14. **return** left

Time complexity analysis: (5 marks)

There are $O(n)$ elements in $S1$, for each element, the binary search on $S2$ costs $O(\log \log n)$ time, thus, the total cost is therefore $O(n \log \log n)$.

Problem 3, 20 points Method B is a sorting algorithms, please answer questions. The start position of array **Arr** is 0. Method B:

```
void sortB(int Arr[], int low, int high){
    if(low < high){
        int pi = partition (Arr, low, high);
        print array Arr; // Line Output
        sortB( Arr, _low_, _pi-1_ );
        sortB( Arr, pi+1, high );
    }
}

int partition (int Arr[], int low, int high) {
    pivot = Arr[high];
    i = (low - 1)
    for (j = low; j <= high- 1; j++) {
        if (Arr[j] <= pivot) {
            i++;
            swap (_Arr[i]_, _Arr[j]_)
        }
    }
    swap (Arr[i + 1] , Arr[high])
    return (i+1);
}
```

- (1) Please complete the method B then it works as quick sort [6 points]
- (2) If the original sequence is “25, 40, 3, 55, 30, 26, 18, 45”, after invoking Method B, please write down the outputs (Line output) step by step. [14 points]

[25, 40, 3, 30, 26, 18, 45, 55]

[3, 18, 25, 30, 26, 40, 45, 55]

[3, 18, 25, 30, 26, 40, 45, 55]

[3, 18, 25, 26, 30, 40, 45, 55]

Problem 4, 30 points Design a function to check if a linked list is a palindrome.
For example:

Linked list **A**: 1->2->3 is not a palindrome, return No.

不知道长度的、判断循环的
：快慢指针

Linked list **B**: 1->2->3->2->1 is a palindrome, return Yes.

Idea: (15 marks)

We want to detect linked lists where the front half of the list is the reverse of the second half. How would we do that? By reversing the front half of the list. A stack can accomplish this.

We need to push the first half of the elements onto a stack. Since we do not know the length of the linked list, we can do like this way: we use a slow runner (go one step per iteration) and a fast runner (go two step per iteration). At each step in the loop, we push the data from the slow runner onto a stack. When the fast runner hits the end of the list, the slow runner will have reached the middle of the linked list. By this point, the stack will have all the elements from the front of the linked list, but in reverse order.

Pseudocode (10 marks)

```
Algorithm isPalindrome(LinkedListNode head) {  
1.  LinkedListNode fast ← head;  
2.  LinkedListNode slow ← head;  
3.  Stack s ← empty stack;  
4.  while (fast != null && fast->next != null) {  
5.      s.push(slow->data);  
6.      slow ← slow->next;  
7.      fast ← fast->next->next;  
8.  }  
9.  if (fast != null) { //Has odd number of elements, so skip the middle element  
10.     slow ← slow->next;  
11.  }  
12. while (slow != null) {  
13.     top = s.pop()->data;  
14.     if (top != slow->data) {  
15.         return false;  
16.     }  
17.     return true  
}
```

Time complexity analysis: (5 marks)

Since slow runner only pass the linked list once, thus the time complexity is $O(n)$, where n is the length of given linked list.