Studytonight – DS test 2 – Aditya Jain

(Queue)

1. Which one of the following is an application of Queue Data Structure?

a) When a resource is shared among multiple consumers.
b) When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes
c) Load Balancing
d) All of the above

2. How many stacks are needed to implement a queue. Consider the situation where no other data structure like arrays, linked list is available to you.

a) 1
b) 2
c) 3
d) 4

Explanation: A queue can be implemented using two stacks

3. How many queues are needed to implement a stack. Consider the situation where no other data structure like arrays, linked list is available to you.

a) 1
b) 2
c) 3
d) 4

Explanation: A stack can be implemented using two queues

4. A priority queue can be efficiently implemented using which of the following data structures? Assume that the number of insert and peek (operation to see the current highest priority item) and extraction (remove the highest priority item) operations are almost same.

a) Array
b) Linked List
c) Heap Data Structures like Binary Heap, Fibonacci Heap
d) None of the above

Explanation: http://en.wikipedia.org/wiki/Priority_queue

5. Which of the following is true about linked list implementation of queue?

a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end.
b) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from the beginning.
c) Both of the above
d) None of the above

Explanation: To keep the First In First Out order, a queue can be implemented using linked list in any of the given two ways.

6. Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are:

a) Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT

b) Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR
c) Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT
d) Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT

---

7. A Priority-Queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is given below: 10, 8, 5, 3, 2 Two new elements ”1‘ and ”7‘ are inserted in the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

a) 10, 8, 7, 5, 3, 2, 1
b) 10, 8, 7, 2, 3, 1, 5
c) 10, 8, 7, 1, 2, 3, 5
d) 10, 8, 7, 3, 2, 1, 5

8. Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue(Q){
  m = k
  while (Q is not empty and m  > 0) {
    Dequeue(Q)
    m = m - 1
  }
}
```

What is the worst case time complexity of a sequence of n MultiDequeue() operations on an initially

(A) $\Theta(n)$
(B) $\Theta(n+k)$
(C) $\Theta(nk)$
(D) $\Theta(n^2)$

a) A
b) B
c) C
d) D

Explanatin: Since the queue is empty initially, the condition of while loop never becomes true. So the time complexity is $\Theta(n)$.

9. Suppose implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?

a) A queue cannot be implemented using this stack.
b) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.
c) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.
d) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

Explanation: To DEQUEUE an item, simply POP. To ENQUEUE an item, we can do following 3 operations 1) REVERSE 2) PUSH 3) REVERSE

10. A queue is implemented using an array such that ENQUEUE and DEQUEUE operations are performed efficiently. Which one of the following statements is CORRECT (n refers to the number of items in the queue)?

a) Both operations can be performed in O(1) time

b) At most one operation can be performed in O(1) time but the worst case time for the other operation will be Ω(n)
c) The worst case time complexity for both operations will be Ω(n)
d) Worst case time complexity for both operations will be Ω(log n)

Explanation: We can use circular array to implement both in O(1) time.

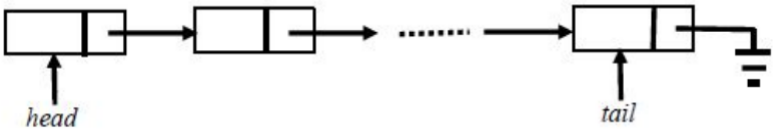
11. Consider the following pseudo code. Assume that IntQueue is an integer queue. What does the function fun do?

```
void fun(int n)
{
   IntQueue q = new IntQueue();
   q.enqueue(0);
   q.enqueue(1);
   for (int i = 0; i < n; i++)
   {
      int a = q.dequeue();
      int b = q.dequeue();
      q.enqueue(b);
      q.enqueue(a + b);
      ptint(a);
   }
}
```

a) Prints numbers from 0 to n-1
b) Prints numbers from n-1 to 0
c) Prints first n Fibonacci numbers
d) Prints first n Fibonacci numbers in reverse order.

Explanation: The function prints first n Fibonacci Numbers. Note that 0 and 1 are initially there in q. In every iteration of loop sum of the two queue items is enqueued and the front item is dequeued.

12. A queue is implemented using a non-circular singly linked list. The queue has a head pointer and a tail pointer, as shown in the figure. Let n denote the number of nodes in the queue. Let 'enqueue' be implemented by inserting a new node at the head, and 'dequeue' be implemented by deletion of a node from the tail.



Which one of the following is the time complexity of the most time-efficient implementation of 'enqueue' and 'dequeue, respectively, for this data structure?

a) Θ(1), Θ(1)

b) Θ(1), Θ(n)

c) Θ(n), Θ(1)

d) Θ(n), Θ(n)

Explanation: For Enqueue operation, performs in constant amount of time (i.e., Θ(1)), because it modifies only two pointers, i.e.,

---

Create a Node P.

P-->Data = Data

P-->Next = Head

Head = P

For Dequeue operation, we need address of second last node of single linked list to make NULL of its next pointer. Since we can not access its previous node in singly linked list, so need to traverse entire linked list to get second last node of linked list, i.e.,

```
temp = head;
 While( temp-Next-->Next != NULL){
       temp = temp-Next;
       }
temp-->next = NULL;
Tail = temp;
```

Since, we are traversing entire linked for each Dequeue, so time

complexity will be Θ(n). Option (B) is correct.

13. Suppose you are given an implementation of a queue of integers. The operations that can be performed on the queue are:
    i. isEmpty (Q) — returns true if the queue is empty, false otherwise.
    ii. delete (Q) — deletes the element at the front of the queue and returns its value.
    iii. insert (Q, i) — inserts the integer i at the rear of the queue.

Consider the following function:

```
 void f (queue Q) {
int i ;
if (!isEmpty(Q)) {
   i = delete(Q);
   f(Q);
   insert(Q, i);
  }
}
```

What operation is performed by the above function f ?

a) Leaves the queue Q unchanged

b) Reverses the order of the elements in the queue Q

c) Deletes the element at the front of the queue Q and inserts it

at the rear keeping the other elements in the same order

d) Empties the queue Q

Explanation: As it is recursive call, and removing from front while inserting from end, that means last element will be deleted at last and will be inserted 1st in the new queue. And like that it will continue till first call executes insert(Q,i) function.
So, the queue will be in reverse.

14. An implementation of a queue Q, using two stacks S1 and S2, is given below:

```
void insert(Q, x) {
  push (S1, x);
}

void delete(Q){
  if(stack-empty(S2)) then
    if(stack-empty(S1)) then {
       print("Q is empty");
       return;
    }
    else while (!(stack-empty(S1))){
       x=pop(S1);
       push(S2,x);
    }
  x=pop(S2);
```

}

Let n insert and m (<=n) delete operations be performed in an arbitrary order on an empty queue Q. Let x and y be the number of push and pop operations performed respectively in the process. Which one of the following is true for all m and n?

a) n+m <= x < 2n and 2m <= y <= n+m

b) n+m <= x < 2n and 2m<= y <= 2n

c) 2m <= x < 2n and 2m <= y <= n+m

d) 2m <= x <2n and 2m <= y <= 2n

Explanation: The order in which insert and delete operations are performed matters here. The best case: Insert and delete operations are performed alternatively. In every delete operation, 2 pop and 1 push operations are performed. So, total m+ n push (n push for insert() and m push for delete()) operations and 2m pop operations are performed. The worst case: First n elements are inserted and then m elements are deleted. In first delete operation, n + 1 pop operations and n push operation are performed. Other than first, in all delete operations, 1 pop operation is performed. So, total m + n pop operations and 2n push operations are performed (n push for insert() and n push for delete())

15. Following is C like pseudo code of a function that takes a Queue as an argument, and uses a stack S to do processing.

```
void fun(Queue *Q)
{
   Stack S;  // Say it creates an empty stack S

   // Run while Q is not empty
```

```
   while (!isEmpty(Q))
   {
      // deQueue an item from Q and push the dequeued item to S
      push(&S, deQueue(Q));
   }

   // Run while Stack S is not empty
   while (!isEmpty(&S))
   {
    // Pop an item from S and enqueue the poppped item to Q
    enQueue(Q, pop(&S));
   }
}
```

What does the above function do in general?

a) Removes the last from Q

b) Keeps the Q same as it was before the call

c) Makes Q empty

d) Reverses the Q