Aniket Gupta
2022073

60

Rush

PAGE NO.:

DATE : / /

DSA HW-5

Q1.

```
struct node* insert_pos(struct node *head,
                        struct node *n, int pos) {
    if (pos == 0) {
        (*n).next = head;
        return n;
    }

    struct node *tmp = head;
    int i = 0;

    while (i != pos-1 && tmp) {
        tmp = (*tmp).next;
        i = i+1;
    }

    if (tmp != NULL) {
        (*n).next = (*tmp).next;
        (*tmp).next = n;
    }

    return head;
}
```

5

P.T.O

Q3 struct node * rev_list (struct node *

```
if (head == NULL || head ->
    return head;

struct node * reversed_list_head = rev_list(
head -> next -> next = head;
head -> next = NULL;
return
```

Q3. struct node * reversed_list (struct node *head) {

```
if (head == NULL || head -> next == NULL)
    return head;

struct node * reversed_list_head = reversed_list (head -> next);
head -> next -> next = head;
head -> next = NULL;
return reversed_list_head;
```

Q4. struct delete_info delete_min_max (struct node * head) {

```
struct delete_info data;
data.head = NULL;
data.min = NULL;
data.max = NULL;

if (head == NULL)
    return data;

struct node * min_prev = NULL;
struct node * max_prev = NULL;
struct node * min_node = head;
struct node * max_node = head;
struct node * prev = NULL;
struct node * curr = head;
```

}

```
while (curr != NULL) {
    if (curr -> val < min_node -> val) {
        min_node = curr;
        min_prev = prev;
    }
    if (curr -> val > max_node -> val) {
        max_node = curr;
        max_prev = prev;
    }
    prev = curr;
    curr = curr -> next;
}

curr = NULL;
prev = NULL;

if (head -> next == NULL) {
    data.head = NULL;
    data.min = head;
    data.max = head;
}

else {
    if (max_node == head) {
        min_prev -> next = min_node -> next;
        head = head -> next;
    }

    else if (min_node == head) {
        if (max_prev == min_node) {
            max_prev -> next = max_node -> next
            head = head -> next;
        }
```

```
        else {
            head = head -> next;
            max_prev -> next = max_node -> next;
        }
    }

    else {
        if (max_prev == min_node) {
            max_prev -> next = max_node -> next;
            min_prev -> next = min_node -> next;
        }

        else {
            min_prev -> next = min_node -> next;
            max_prev -> next = max_node -> next;
        }
    }

    data.head = head; data.min = min_node; data.max = max_node;
}

return data;
}
```

(25)

outputs :

→ 1 5 3 2 4 : 3 2 4
→ 5 1 3 2 4 : 3 2 4
→ 4 1 5 2 3 : 4 2 3
→ 4 5 1 2 3 : 4 2 3
→ 4 3 3 2 1 : 4 3 2
→ 5 4 3 2 1 : 4 3 2

struct node * cur = head;

Rysh
PAGE NO.:
DATE: / /

2. Initially, size of array is N with N elements. We delete each element one by one, until, $\frac{N}{4}$ elements remain. Now, we create a new array of size $\frac{N}{2}$ and copy $\frac{N}{4}$ elements from old array to new array, free the old array and delete an element in new array. This marks the completion of first phase. We continue this process for K phases so that, at the end, we are left with only one element in the array

Phase	no. of elements copied
I	N/4
II	N/8
III	N/16
:	:
K	$N/2^{K+1}$

we stop this when $\frac{N}{2^{K+1}} = 1$, $\Rightarrow K = \log_2 N - 1$

Allocation Overhead

Let cost of allocating a new array and freeing the old array be 1. Since there are K allocations therefore, allocation overhead = $K = \log_2 N - 1$

Deletion Overhead

Let cost of deleting an element at a given index in the array be one. Since there are N-1 deletions,

$\therefore$ deletion overhead $= N-1$

Copy Overhead

Let the cost of copying one element from old array to new array be one

$\therefore$ copy overhead $= \frac{N}{4} + \frac{N}{8} + \frac{N}{16} + \dots \frac{N}{2^{K+1}}$

$= \frac{N \cdot 2^{K-1}}{2^{K+1}} + \frac{N \cdot 2^{K-2}}{2^{K+1}} + \dots \frac{N \cdot 2^0}{2^{K+1}}$

$= \frac{N}{2^{K+1}} \left(2^0 + 2^1 + \dots + 2^{K-2} + 2^{K-1} \right)$

$= 2^K - 1$

now, $N = 2^{K+1} = 2 \cdot 2^K$, $\therefore 2^K = N/2$

$\therefore$ copy overhead $= \frac{N}{2} - 1$

$\therefore$ total overhead = allocation overhead + deletion overhead + copy overhead

$= \log_2 N - 1 + N - 1 + \frac{N}{2} - 1$

$= O(N)$

$\therefore$ average (amortized) cost of deletion is $O(1)$