1

2123475 7061

Masters of Computer Applications

MCAC-201: Data Structures

Unique Paper Code: 223401201
Semester II
August-2022
Year of Admission: 2021

Duration: 3hours   Max Marks: 70

Note: Attempt all questions. Write neatly and absolutely to the point. Assumptions (if any) should be clearly stated.

1. A priority queue is a data structure similar to a queue, where each element has a priority associated with it. We can denote these priorities with the help of a priority number; where lower priority number signifies highest priority.
When an element is inserted (*enqueue*) it is placed in the queue but when an element is removed (*dequeue*), the element with highest priority gets removed. For example if the queue contains keys $(10, 2), (30, 1)$ and $(40, 4)$, where (a, b) denote key a with priority value b, then next *dequeue*() will return the key 30.

   (a) Consider the implementation of priority queue where, a sorted array is used as underlying data structure. Give an efficient implementation for the following member functions

   - *enqueue*(*Ele*, *pri_Ele*) (Element *Ele* with priority *pri_Ele* is inserted to the queue)
   - *dequeue*() (Element with lowest priority number is removed from the queue and returned)

   What will be the time complexity for your implementation of *enqueue*(*Ele*, *pri_Ele*) and *dequeue*() operations? Justify briefly. (3+1=4)

   (b) If the priority queue is implemented using a Red-Black Tree as underlying data structure, what will be the time complexity to implementation the functions *enqueue*(*Ele*, *pri_Ele*) and *dequeue*()? Justify briefly. (3)

   (c) If we implement priority queue using Binary Heap as underlying data structure, what will be the time complexity to implementation of *enqueue*(*Ele*, *pri_Ele*) and *dequeue*() operations? Justify briefly. (2)

   (d) What is the advantage of using Binary Heap over Red-Black Tree to implement priority queue? (1)

2. Given a singly linked list, with only head pointer that contains n integers. Implement a linear time function *rearrange*(*node***head*) that takes no more than constant extra

~~(10, 2)~~ → ~~(30, 1)~~ → ~~(40, 4)~~
~~dequeue~~

space to rearrange the list contents such that, all negative numbers are placed before all positive numbers. (6)

3. Consider a undirected graph $G(V, E)$, write a function that returns the maximum degree of any vertex.

   (a) Assuming that the graph is implemented using Incident Matrix. (2.5)

   (b) Assuming that the graph is implemented using Adjacency List. (2.5)

   (c) What is the running time of your functions, in each case? (1)

4. Consider the following class, that maintains 2 stacks that grow from the opposite ends of a single array.

```
class MyStacks {
      int Array[100]; // single array that is used to maintain 2 stacks
      int top_stack1; // points to the index of the top element on stack 1
      int top_stack2; // points to the index of the top element on stack 2
      public:
      MyStack(){
      top_stack1=-1;
      top_stack2=100;
      }
      void push(int Ele, int stack_num); /* Push 'Ele' on stack1 or
      stack2 based on 'stack_num'= 1 or 2 respectively. */

      int pop(int stack_num); /* returns the element popped from
      stack1 or stack2 based on 'stack_num'= 1 or 2 respectively. */

      bool isEmpty(int stack_num); /* Returns if stack1 or
      stack2 is empty or not based on 'stack_num'= 1 or 2 respectively. */

      bool isFull(); /* Returns if the stacks are full or not */
};
```

   (a) Give the implementation for the functions 'push', 'pop', 'isEmpty' and 'isFull'. (2+2+1+1=6)

   (b) Give the running time for each of the above functions. (2)

   (c) What is the advantage of the above implementation, compared to having 2 arrays each of size 50 to implement individual stacks? (2)

5. What is the best case and worst case time to search a value from the following data structures storing n keys? Also state the scenario that will act as the best and the worst case respectively.

(a) Binary Heap (1+1=2)

(b) Binary Search Tree (1+1=2)

(c) Array (1+1=2)

(d) R-B Tree (1+1=2)

(e) Stack (1+1=2)

6. Answer the following questions

(a) Consider the structure of perfect squares and discuss whether the function $H(x) = x^2\%10$ where x is a non-negative integer, is a good hashing function? Justify briefly. (3)

(b) Consider the hash function $H(x)$ where $H(x)$ is the floor of the average of the digits in the decimal representation of the x where x is a non-negative integer. For example: for $x = 372$, $H(x)$ is the floor of the average of 3, 7 and $\underset{2}{4}$, thus $H(x) = 4$. Is $H(x)$ a good hashing function? Justify briefly. (3)

(c) Assuming hash function $H(x) = x\%10$, give a possible sequence of inserts, that will lead to the above hash map. Open addressing with linear probing is used for collision resolution. (3)

Index	Key
0	
1	
2	32
3	112
4	54
5	43
6	72
7	46
8	
9	

(d) Give one advantage of chaining over open addressing, for collision resolution. (1)

7. Construct a suffix tree for the text *CARPENTERS*. Show the steps, without explanation. (6)

8. Give an algorithm to construct a Binary Search Tree from a given pre-order traversal. What is the time and space complexity of your algorithm? (5+1=6)

9. Answer the following questions briefly.

   (a) Give an example of a Red Black tree, that is not an A.V.L. tree. (2)

   (b) How many different A.V.L. trees are possible for the set of keys $\{1, 2, 3, 4\}$. Show without explaining. (2)

   (c) What is the maximum and the minimum number of nodes possible in a Binary Search Tree with height h? (2)

Best wishes

$n \cdot \log_2 n - 1$

$2^{(n+1)} - 1 \quad = \log n + n - 1$