1. A police department wants to maintain a database of up to 1,800 license-plate numbers of people who receive frequent tickets so that it can be determined very quickly whether or not a given license plate is in the database. Speed of response is very important; efficient use of memory is also important, but not as important as the speed of response. Which of the following data structures would be most appropriate for this task?

Please explain or show how you decided what the best answer is below:

A. a sorted linked list

B. a sorted array with 1,800 entries

C. a hash table using open addressing with 1,800 entries

D. a hash table using open addressing with 3,600 entries

E. a hash table using open addressing with 10,000 entries

A hash table is nothing but a data structure that implements an array of abstract data types, that can map the given keys to values.

Based on the below requirements:

Speed of response is very important.

efficient use of memory is also important

We require a hash table using open addressing with 3,600 entries

2. An array of 7 integers is being sorted by the heapsort algorithm. After the initial

phase of the algorithm (constructing the heap), which of the following is a possible

ordering for the array?

A.85 78 45 51 53 47 49

B.85 49 78 45 47 51 53

C.85 78 49 45 47 51 53

D. 45 85 78 53 51 49 47

E. 85 51 78 53 49 47 45

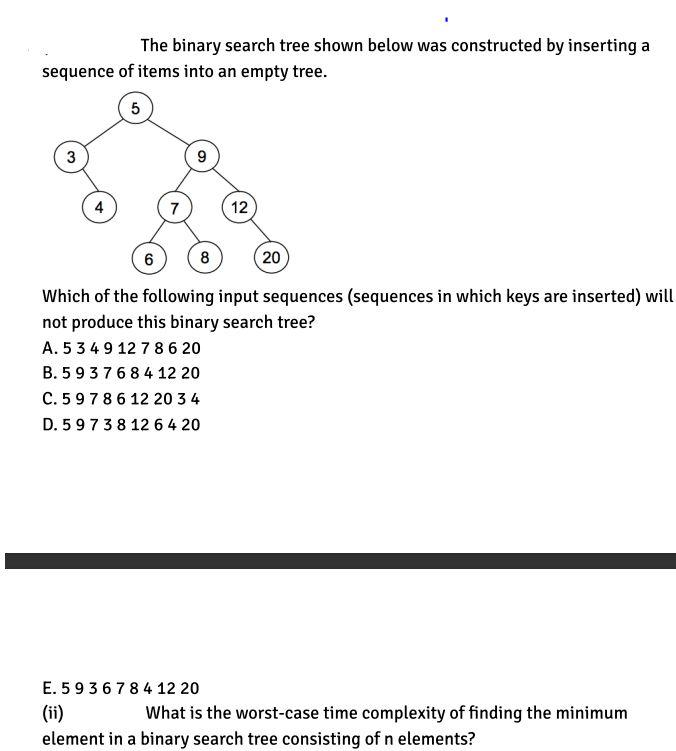
After the initial phase of the algorithm (constructing the heap) the possible ordering for array is

B. 85 49 78 45 47 51 53

The ordering is root then left and right then vice versa for the rest

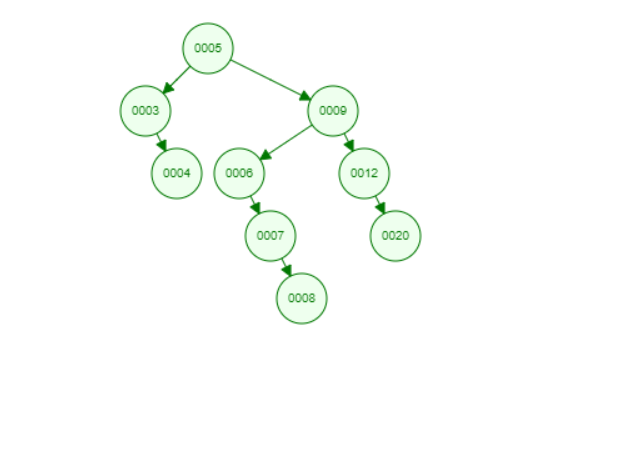
Explanation:- Since in HeapSort after the initial state we have either a max heap or min-heap. max-heap has properties that the value of the root node is greater than or equal to either of its children. while min heap has properties that value of the parent node to be less than that of the child node.

All the rest 4 options (A,C,D,E) do not satisfied the condition of either of max heap or min heap hence option B is correct.



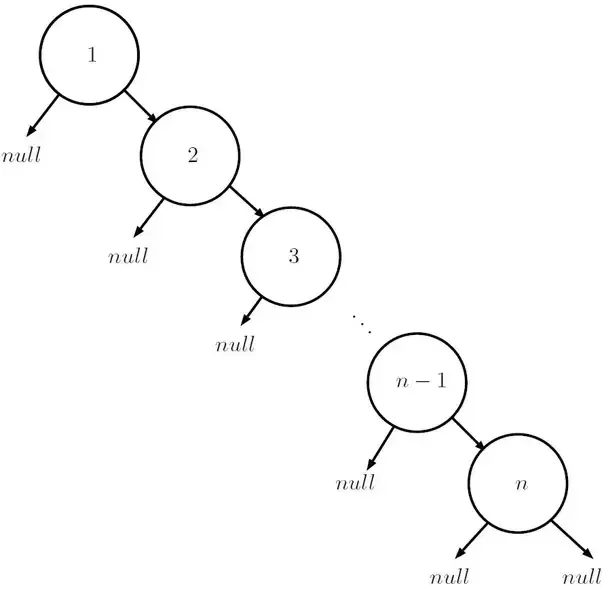
(i) E. 5 9 3 6 7 8 4 12 20

this is so because this will give us a binary tree as:



(ii) The worst case time-complexity of finding minimum element is: O(n)

Reason: this will happen when the tree either has only left children or only right children.



22. (9 points) You are given an empty hash table of size 7 that uses open addressing. The

following sequence of keys is to be inserted: 15, 17, 8, 23, 3, 5. Insert these keys using

each of the following approaches. If overflow occurs, say so, and indicate the element

that causes the overflow.

(a) h(x) = x % 7; linear probing

(b) h(x) = x % 7; quadratic probing

(c) h₁(x) = x % 7; double hashing with h₂(x) = x / 7+1 (using integer division)

a. Linear probing: Using linear probing when a collision occurs then the next available free slot in the hash table is checked sequentially . If a free slot is found then the key is placed in that free slot.

Given keys are 15,17,8,23,3,5.

given function hashing function is h(x)=x%7.

h(15)=15%7=1.

h(17)=17%7=3

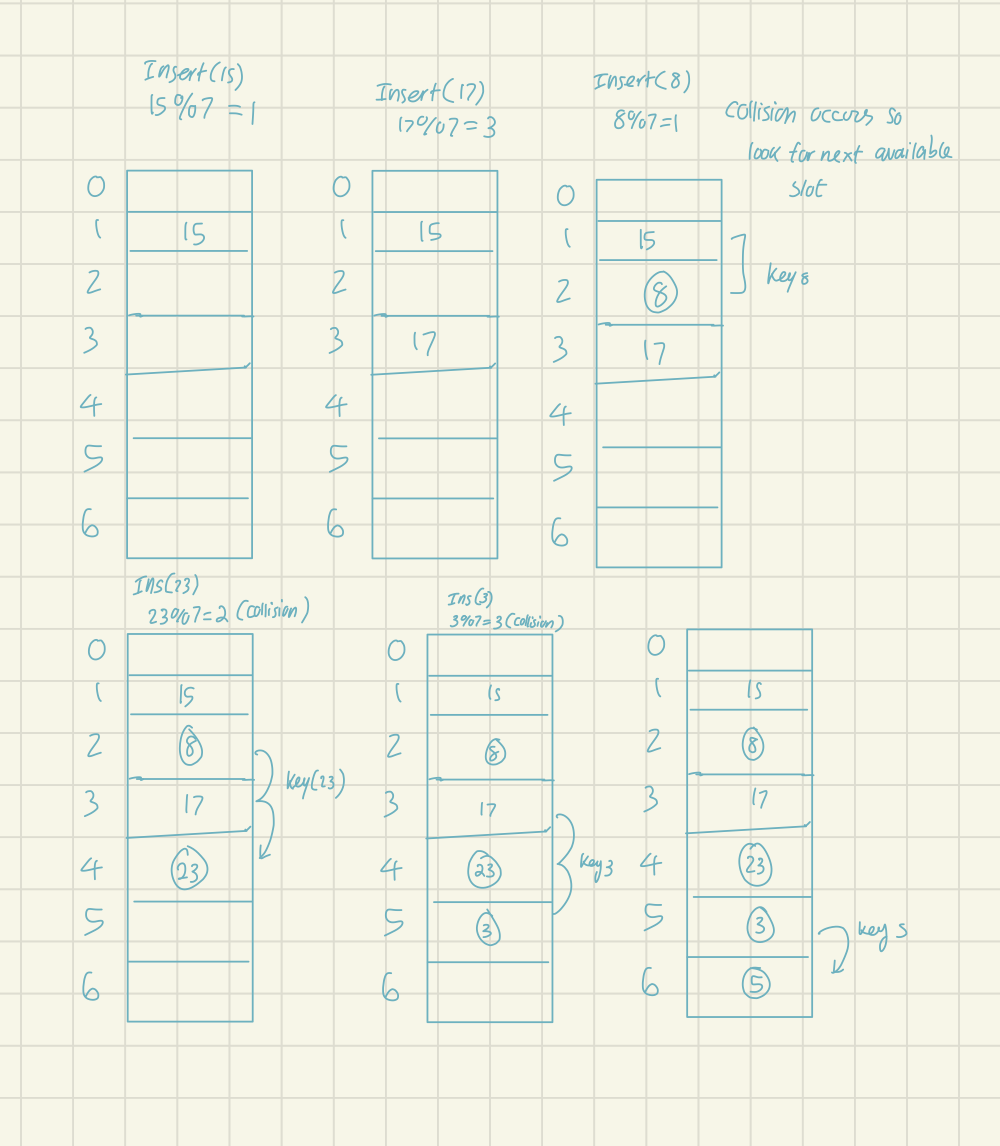
h(8)=8%7=1(collision occurred so the next available slot is 2. So key 8 is mapped to index 2 in the hash table.

h(23)=23%7=2(collision occurred so next available slot is 4. So key 23 is mapped to index 4.

h(3)=3%7=3(collision occurred so the next available slot is 5. So key 3 is mapped to index 5

h(5)=5%7=5(collision occurred so the next available slot is 6. So key 5 is mapped to index 6.

**The total number of collisions is 4 . Collisions occurred when inserting keys 8,23,3 and 5.**

****

B) Quadratic Probing

: Using quadratic probing when a collision occurs the next available slot is checked in the hash table by adding terms like +1,+4,+9+25....

given function hashing function is h(x)=x%7.

h(15)=15%7=1.

h(17)=17%7=3

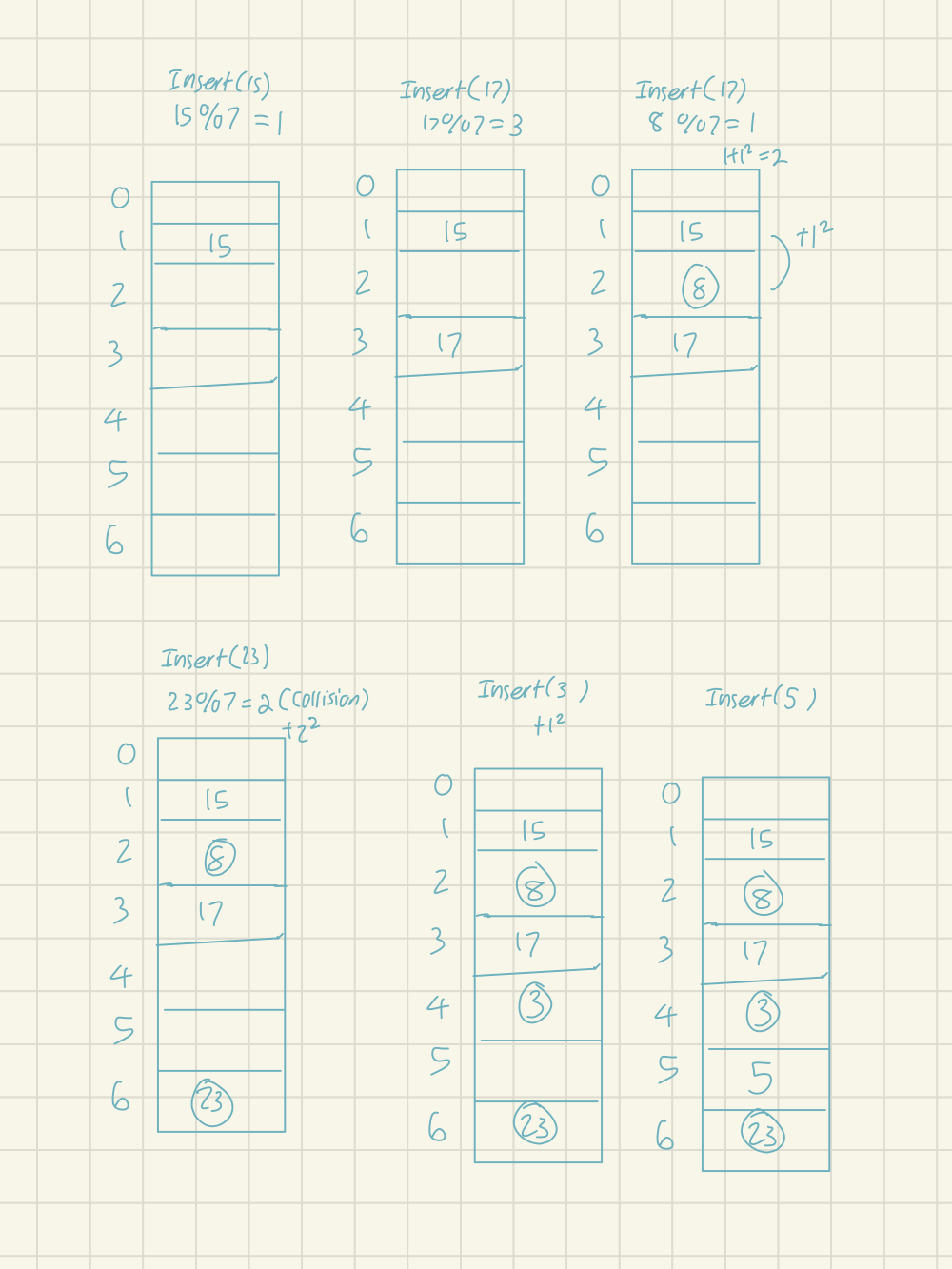
h(8)=8%7=1(so 1+1 i.e., 2 so 8 is mapped to index 2)

h(23)=23%7=2( similarly 23 is mapped to index 6)

h(3)=3%7=3(3 is mapped to index 4)

h(5)=5%7=5.

**using quadratic probing total number of collisions are 3 . Collision occurred when inserting keys 8, 3, 23..**



C. double hashing: Using double hashing technique when a collision occurs for the first hash function then we apply second hash function to get the free slot in the hash table.

given h1(x)=x%7 and h2(x)=x/7 + 1.

given keys are 15,17,8,23,3,5

given function hashing function is h(x)=x%7.

h(15)=15%7=1.

h(17)=17%7=3

h(8)=8%7=1(collision occurred. So apply second hash function h2(8)=2. so key 8 is occupied at index 2( since 1+1) )

h(23)=23%7=2(collision occurred so h2(23)=4. so key 23 is occupied at index 6 (since 2+4=6)

h(3)=3%7=3(collision occurred so h2(3)=1 so key 3 is occupied at index 4 ( since 3+1=4)

h(5)=5%7=5.

**total number of collisions using double hashing is 3 . Collision occurred when inserting keys 3,8,5**

**Double hashing can be done using :**

**(hash1(key) + i \* hash2(key)) % TABLE\_SIZE**

**Here hash1() and hash2() are hash functions and TABLE\_SIZE**

**is size of hash table.**

**(We repeat by increasing i when collision occurs)**

