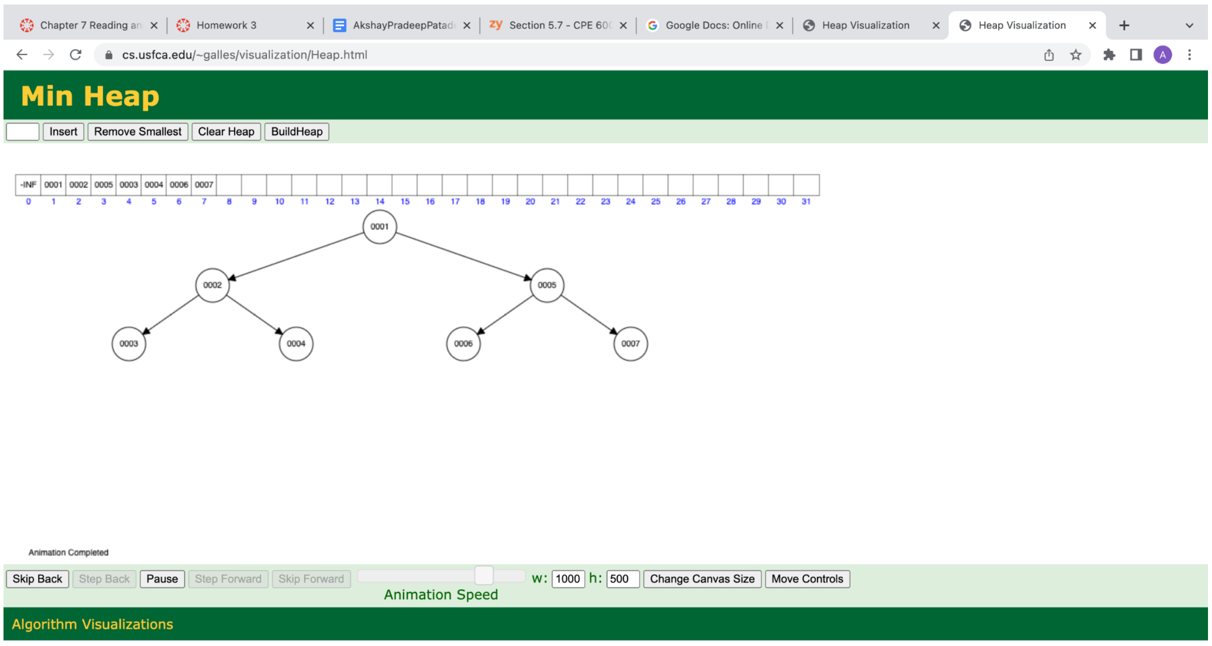
1. Is there a heap T storing seven distinct elements such that a preorder traversal of T yields the elements of T in sorted order? How about an inorder traversal? How about a postorder traversal?

Answer:

In a preorder traversal we traverse the root first then the left child and then the right child. We perform this operation recursively. We can create a min heap by inserting the elements in the below order: 1,2,5,3,4,6,7



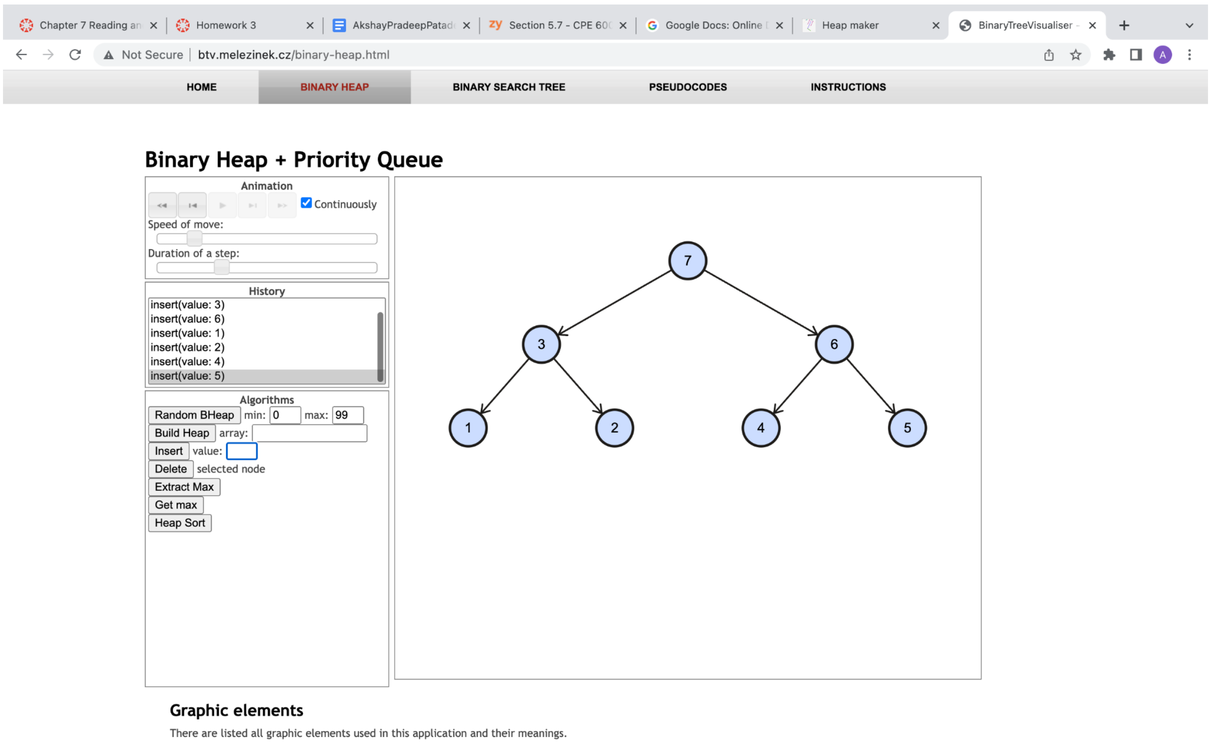
If we perform the pre Order traversal we get: 1,2,3,4,5,6,7

In an inorder traversal, the left child is accessed first, then root and finally the right child. We perform this operation recursively. If we take the above example and try to traverse the tree in an inorder fashion we get: 3,2,4,1,6,5,7 which is not in sorted order.

In a postorder traversal, the left child is accessed first, then the right child and then finally the root node. We perform this operation recursively. If we take the above examplne and try to traverse the tre in a post order we get:3,4,2,6,7,5,1 which is not in sorted order

For the max - heap tree shown below if we perform an inorder traversal we get 1,3,2,7,4,6,5 which is still not in sorted order.

Let us consider a max- heap tree: A max heap tree is a tree wherin the value of the internal nodes is greater than or equal to the values of the children in that node.



Binary Heap + Priority Queue

[✓ Continuously]

```
insert(value: 3)
insert(value: 6)
insert(value: 1)
insert(value: 2)
insert(value: 4)
insert(value: 5)
```

min: 0 max: 99
array:
value:
selected node

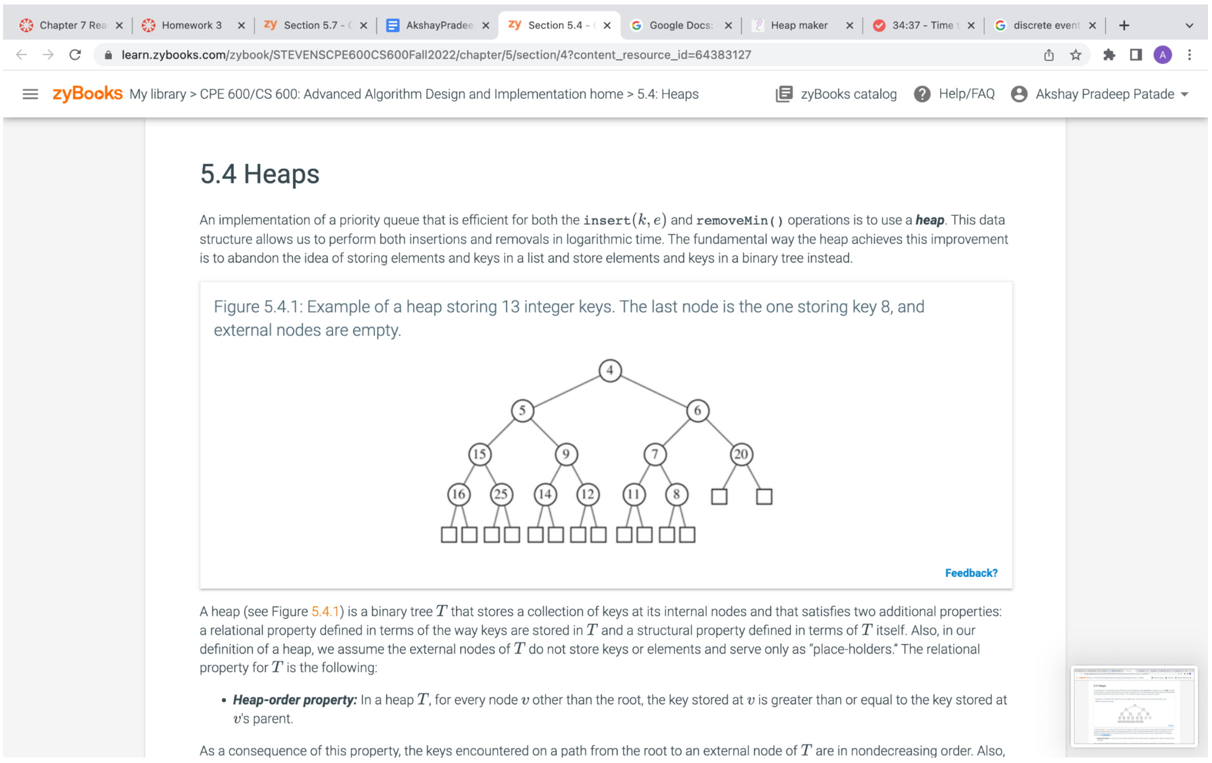
Graphic elements

There are listed all graphic elements used in this application and their meanings.

If we perform a post order traversal on a max- heap tree we get: 1,2,3,4,5,6,7 which is in sorted order.

2. Let T be a heap storing inkeys. Give an efficient algorithm for reporting all the keys in
Tthat are smaller than or equal to a given query key x(which is not necessarily in T
). For example, given the heap of Figure 5.4.1 and query key x=7, the algorithm should report 4, 5, 6, 7. Note that the keys do not need to be reported in sorted order. Ideally, your algorithm should run in O(k)time, where k is the number of keys reported.

Answer:



Chapter 7 Rea
Homework 3
Section 5.7 -
AkshayPradee
Section 5.4 -
Google Docs
Heap maker
34:37 - Time
discrete event
learn.zybooks.com/zybook/STEVENSCPE600CS600Fall2022/chapter/5/section/4?content_resource_id=64383127
zyBooks My library > CPE 600/CS 600: Advanced Algorithm Design and Implementation home > 5.4: Heaps
zyBooks catalog
Help/FAQ
Akshay Pradeep Patade

5.4 Heaps

An implementation of a priority queue that is efficient for both the `insert`(k, e) and `removeMin()` operations is to use a ***heap***. This data structure allows us to perform both insertions and removals in logarithmic time. The fundamental way the heap achieves this improvement is to abandon the idea of storing elements and keys in a list and store elements and keys in a binary tree instead.

Figure 5.4.1: Example of a heap storing 13 integer keys. The last node is the one storing key 8, and external nodes are empty.

A heap (see Figure 5.4.1) is a binary tree T that stores a collection of keys at its internal nodes and that satisfies two additional properties: a relational property defined in terms of the way keys are stored in T and a structural property defined in terms of T itself. Also, in our definition of a heap, we assume the external nodes of T do not store keys or elements and serve only as "place-holders." The relational property for T is the following:

- ***Heap-order property:*** In a heap T, for every node v other than the root, the key stored at v is greater than or equal to the key stored at v's parent.

As a consequence of this property, the keys encountered on a path from the root to an external node of T are in nondecreasing order. Also,

From the given figure 5.4.1 we can see that this heap is a min-heap tree wherein the value of every internal nodes is less than or equal to all of its children node. We will perform pre order traversal and we will check if the current node key is less than x if yes then we will print that key and recursively traverse the left and the right sub tree. If not we will come out of the recursion.

Algorithm: reportKeys(Node node):
Input: A min-heap tree T with n nodes
Ouput: Printing all the keys in the tree whose key value is <= x

```
if(node == null or node.key > x) then
        return;

if(node.key <= x)
        Print(node.key)

reportKeys(node.left);
reportKeys(node.right);
return;
```

Intitally we will call the function reportKeys() by passing the root node as a parameter.

The time complexity of the above algorithm is O(k) where k is the number of keys reported and the space complexity is O(n).

3. In a discrete event simulation, a physical system, such as a galaxy or solar system, is modeled as it changes over time based on simulated forces. The objects being

modeled define events that are scheduled to occur in the future. Each event, e, is associated with a time, t_e, in the future. To move the simulation forward, the event, e, that has smallest time, t_e, in the future needs to be processed. To process such an event, e is removed from the set of pending events, and a set of physical forces are calculated for this event, which can optionally create a constant number of new events for the future, each with its own event time. Describe a way to support event processing in a discrete event simulation in O(log n)time, where n is the number of events in the system.

Answer:
We are told to desing a system which will perform event processing in O(logn) time. We are given a future set of events each with its time t_e. We have toselect an event from a set of event where the processing time t_e is minimum.

We can perform a linear search to check the minimum event among the set of the events. But this operation takes O(n) time. We have to come up with a solution of O(logn).

The first thing which pops in our mind is to use min - heap tree. In a min-heap tree we can access the minimum element in O(1) and since there are a set of events which are going to occur in the future, we can insert those events in O(logn) because min-heap tree performs insertion in O(logn) time.

Steps:
1. We will be using array data structures to implement the min-heap data. Declare an array a with a max size lets say 1000 and declare a varaible named size = 0.
2. Initialize array a[0] as -∞.
3. Suppose a new element needs to be added we will increment the counter and set a[size] = value.
4. Now there will be a case where the min heap property is violated.Here we perform ‘heap up’ operation to check if the new element added in the heap does not violate min heap property. If it does then we will roll that element up until the min heap property is satisfied.
5. Suppose we need to remove an element from the heap. We will return the first element of the array a[1] and remove the last element of the array a[size] and replace it with a[1] and decrement the size counter.
6. Now there will be a case where the min heap property is violated. So we will perform ‘heap down’ operation. This operation moves the element downwards until the value of the current node is smaller than all of its children node.

4.
Answer:

Open addressing, like independent chaining, is a strategy for dealing with collisions. All items in Open Addressing are kept in the hash table itself. As a result, the table's size must always be more than or equal to the entire number of keys. Closed hashing is another name for this method. The entire operation is built upon probing.

In the given problem statement it is stated that if the value is present in the hashtable at location h(k) then it uses a formula to map the value to the new locations (h(k) + i * f(k)) mod N where N is the size of the hash table and i iterates from 1,2,3,.......

Lets say we have N = 8 and we have a hash table from 0 to 7 where h(21) = 1 This means that the key 21 is mapped to the hash table position 1

Now lets assume we have a set of keys S = {k_1,k_2,k_3,k_4,k_5….k_n}. For all the keys present in the set let’s assume it produces us the value h(k_i) = 1. Since we have a data already present in the table for index 1 we will use the hash function. (h(k) + i * f(k)) % mod n for i = 1, 2, 3…. lets say f(k) produces value 4 for every keys present in the set S.

So the funntion will produce only values 1 & 5 for every keys in the set S.

For Example let i = 1

Therefore,
(1 + 1 * 4) % 8 = 5

When i = 2
(1 + 2 * 4) % 8 = 1

When i = 3
(1 + 3 * 4) % 8 = 5

When i = 4
(1 + 4 * 4) % 8 = 1

Here we could see a pattern that the values are repeating. This collision will always occur if we do not consider N as a prime number.

5.
Suppose you are working in the information technology department for a large hospital. The people working at the front office are complaining that the software to discharge patients is taking too long to run. Indeed, on most days around noon there are long lines of people waiting to leave the hospital because of this slow software. After you ask if there are similar long lines of people waiting to be admitted to the hospital, you learn that the admissions process is quite fast in comparison. After studying the software for admissions and discharges, you notice that the set of patients currently admitted in the hospital is being maintained as a linked list, with new patients simply being added to the end of this list when they are admitted to the hospital. Describe a modification to this software that can allow both admissions and discharges to go fast. Characterize the running times of your solution for both admissions and discharges.

Answer:

From the given information we are able to know that the current software is implementing linked list to admit and descharge the patients.

We also know that the insertion operation is fast but the removal operation is slow. The removal operation is slow because we are doing a linear search in the linked list to find the patient which has a O(n) complexity. The insertion operation is fast because we are inserting the element at the end of the linked list which takes O(1) tiem. We can improve the removal operation by introducting the hash table.

Whenever a new patient comes to the hospital to get admitted, we will check whether that patient is present in the hash table or not. If not we will add that patient in the hash table and store the linked list id as the value of the key.

Whenver a patient wants to discharge, we will retrive his/ her record from the hash table. This will provide us the linked list index. If the index value is last then we will just remove the patient record from the linked list and hash table. If not then, then we will replace the current patient record with the last patient record present in the linked list, update the last patient record index in the hash table and remove the last record from the linked list.

6. A popular tool for visualizing the themes in a speech is to draw a word cluster diagram, where the unique words from the speech are drawn in a group, with each word's size being in proportion to the number of times it is used in the speech. Given a speech containing n total words, describe an efficient method for counting the number of times each word is used in that speech. You may assume that you have a parser that returns the n words in a given speech as a sequence of character strings in O(n) time. What is the running time of your method?

Answer:

We can use hash table to count the frequency of the words used in the speech.

Our hashtable will contains key as the word and the value will represent the number of times it is used in the speech.

Steps:
1. Declare a hashtable
2. Run a for loop to iterate over all the words.
3. Check if the given word is present in the hashtable. If yes then increment the hash table value for a particular word by 1. If not then add a new word in the hash table and set its value equal to 1.
4. Once all the words are iterate just print the hashtable.

The Time Complexity for the above Algorithm is O(n) because we are iterating over all the word one by once. The space Complexity is also O(n).