1 Abstract

In the world of data structures there are two major contenders. On the array side of this battle is the heap. Using its random access arsenal of functions and methods, the heap hits the world with a force of a thousand thunderbolts. In the other corner, there is a data structure that fights the linked battle. Using its infantry-sized collection of linked nodes, the binary tree packs a punch that heaps can never hope to match. Two contenders, but in the end there can only be one. So, which data structure did we use? Ha, like all clever people we decided (with help from professors and such) to cheat. Instead of waiting for the never-ending battle to finish, we just implemented a heap-tree, also known as a he-ree or, in its ear pleasing form, the treap. Now is the time to implement the worlds greatest program with the worlds greatest data structure! On to the Treap DVD Library!

2 Keywords

1. Binary Tree - A data structure. It uses nodes with right and left children pointers to maintain its branch structure. It is traversed in sorted order following the order: left, the self, the right.

2. Heap - A data structure. Generally it is an array that simulates an almost complete binary tree in a linear way. Each level is completely filled, except possibly the lowest which is filled up to a point going left to right.

3. Treap - A data structure. A Binary Tree that maintains the properties of a Heap. The Binary Tree characteristics are satisfied by sorting the Keys, and the Heap characteristics are satisfied by sorting the Priorities.

4. Key - The value of the object contained in the node. It is used to determine the order of traversal in the Binary Tree.
5. **Priority** - The value of the node that determines how close to the root of the tree the node should be. It is the value used to make the tree a Heap, and is present in order to maintain reasonable balance within the trees different branches.

3 **Introduction**

In order to build our DVD library, first we need to implement a new type of binary tree node that contains a priority as well as a key. Then we must implement a treap which uses the new nodes to construct its tree/heap structure. The treap will be the core of the entire library. It will have insertion, deletion, and searching capabilities for public access, as well as a few internal methods assisting the those public functions. It will also provide services so that different ordered traversals of the treap can be performed.

The DVD Library will be a collection of treaps that will allow for ordering of the DVDs and the members. DVD and member classes will be made. The member class will also contain a treap for checkout of DVDs from the Library. The DVD Library will support these major actions: available and checked out sorted DVD listings, sorted member listings, new DVD and member insertion, DVD check out, DVD check in, and member query. Once those actions are implemented and working, it will be a fully functional DVD Library. There will even be checks in place to enforce the 5 day rental deadline, or at least identify and track offenders.

4 **The Treap**

A treap, as was stated in the keywords section, is a binary tree and a heap at the same time. There are important reasons for this, and there are many important properties and functions which will make the treap possible.

The main reason to maintain the properties of a heap in this tree is to keep it sufficiently balanced to have \(\lg(n)\) insertion, deletion, and search time. Even if the elements are inserted into the treap so that one side contains many more and the opposite size, the readjustment according to priority will push the sides back to an equal or almost equal length in most case. It is virtually guaranteed for treaps of a large size. If there are more elements, there are more random priorities, and so there will be greater chance for an even distribution of nodes.

In order to do this, there are two important steps to the insertion process. First, the element to be inserted is placed at the appropriate node for its key. This is a basic process for a binary tree. The tree is traversed through left and right nodes by comparing the value of the key at each node to the insertion element. When a null node is found \((\text{tnull})\), then the new element can be placed at that spot as a leaf. The second part involves adjusting the tree so that the elements are not only in binary tree order by
key, but also heap order by priority. Rotations are used within the tree to achieve heap ordering. This is possible, because rotations change the structure of the tree but do not change the result of an in order traversal. Each rotation, whether left or right, moves the newly inserted node up the tree one level until it reaches the correct spot for its priority.

5 The DVD Library

Now that the data structure of choice is implemented, we move on to the actual content and purpose of the project. We now have the tools necessary to make a DVD Library that can check out and check in DVD to all who are willing to become member. Our Library includes two treaps for DVDs and one treap for members. It will include two additional classes to supplement the treap and treap node classes. The DVD class and the member class will be used as the template type for the treaps. The DVDs have two treaps; one for DVDs that are currently available for rental and one for DVDs that are currently checked out. The third treap keeps track of all current members.

The all three treaps are actively used in the check in and check out processes. When a DVD is check out, a member is identified as the renter and a DVD is determined on which the transaction will take place. The member must be found in the member treap to confirm that he is indeed a member. Not only that, but an actual member must be used for the DVD to be correctly checked back in again. Each member has an internal treap which contains all the DVDs he is currently borrowing. Therefore, the new check out must be inserted into it. The DVD is then removed from the available treap and inserted into the checked out treap.

When the DVD is checked back in, all three treaps are used again. The returned DVD must identify its owner. The owner can then delete that DVD from his treap. The DVD is removed from the check out treap and inserted into the available treap. We also must check to see if the returned DVD was returned on time, or after the due date.

Other simple actions that our DVD Library include insertion and query. When a shipment of new releases arrives at our store, they need to be added to the Library so that they can be rented. The same goes for people. Say that someone wants to rent DVDs for the first time from our store? They need to be added to be given a member number and then they should be added to our member list so that we can check DVDs out to them, and keep track of their rentals. Finally, if a member walks into our store, it would be nice to be able to pull up their info and see the DVDs they currently have checked out. This can be done with out system as well.
6 Future Additions

Our DVD Library was quite a success. The way it is now, it could be used for a small collection of DVDs. If it was ever to be used for a large scale commercial operation, however, there would need to be a few additions. First, we would enhance the DVD class so that there is more information about each title. There could be fields for short descriptions of the movie, UPC code, director, major actors, release date, etc. We would also need to make modifications to allow for multiple copies of any given DVD title. Most DVD rental places have multiple copies of the more popular and new release films. The coolest idea for an addition to this project is the ability to take inputs not only from the keyboard, but also from a bar code scanner. Not only would it be cool, but it would also make it easier to use for the person working the desk. Instead of typing in the title, UPC, member name, or member number, the worker would make a selection for desired action (check in/out), scan the members card and the DVDs, and instantly the process would be done.

7 Conclusions

We are very satisfied with the results of this project. The treap class that we implemented is very stable. It is more stable than almost any data structure that we implemented this semester. Almost all the problems that we ran into when designing and coding this DVD Library came not while making the tNode and treap classes, but instead while trying to design the specific in order traversal schemes for specific DVD and member treaps. We also had problems designing the interface for the DVD Library. There were issues with searching the members by name and by member number, the later being more difficult. Finally, we were greatly disappointed that no GUI was made for this design. We spent quite a bit of time trying to get one working. We were hoping for a three-tab or three-window set up that would allow for parallel searching of the three treaps: available movies, checked out movies, and members. It became clear that with the time left, such a plan was out of the question, so we did not convert the text interface into a GUI one. As was said in the Future Additions section, it was quite a success.

8 References


9 Biographical Information

<table>
<thead>
<tr>
<th>James Stephen Moiani, a lifelong resident of suburban South Philly, aspires to be a nerd when he grows up. Current hobbies include doing more homework in a night then a room full of business majors can do in a semester, video games, PC building, and marching in the Band of the Fighting Irish. He is on pace to graduate with a degree in Computer Engineering from Notre Dame in December 2004.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael Stephen Crocker was born in Corvallis, Oregon, at 6am, December 6th, 1982. This information is not one-hundred percent verifiable, and thus allows him to maintain the belief that he is not human but instead a pan-dimensional being who’s physical and temporal manifestations merely suggest a human person. Michael is currently studying Computer Engineering, and a little bit of film, at the University of Notre Dame. Sweet irony!</td>
</tr>
</tbody>
</table>

Table 1: The Designers