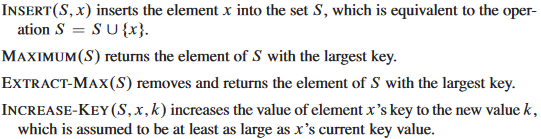
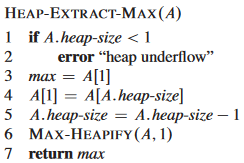
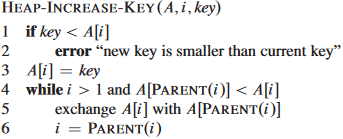
**Priority queues:**

Heap arrays with functionality for extracting max/min values, as well as editing values in specific places. See the image below. 

Maximum can be made by just returning the first entry in the set.

Extract-Max can be made by setting value1 to the first entry, then setting the first entry to become the last entry, remove the last entry, perform Max-Heapify (iirc the heap sort function. Check 06 to be sure.) with the array and the constant number 1 as parameters. Return value1.

Heap-Increase-Key makes sure the new key isn’t smaller than the old key. Thereafter, the designated key replaces the value in spot I, followed by a while-loop which then ensures the new value doesn’t have parents with smaller values.

Max-Heap-Insert just takes a key, makes a new entry with the absolute lowest value possible, and then runs Heap-Increase-Key on it with the key as a parameter.

**Linked lists**

Arrays where rather than having the elements stored in the same place in the memory, they are scattered about and have internal pointers to the next entry in the array. Normally, linked lists only points to the next entry, but there also exist doubly linked lists where entries also have pointers to the preceding entry. These lists are said to be sorted if the keys (entry values) appear in order from lowest to highest, with the lowest being the head and the highest being the tail. They can also be circular, which means the head has a pointer to the tail and vice versa.

In order to search for values in a list, you have to iterate through the entire thing and see if each entry has a key matching the one you’re looking for. In cases where the keys are known to be sorted, you can do time saving b.s. by taking the mid entry and doing a size comparison with the key you’re looking for, and repeat for other sides if it be smaller or bigger than the surrounding entries.

Lists are so fucking simple I do not feel like reading more about them. Basically, if you want to do operations, fuck with the pointers a bit and it will work out.

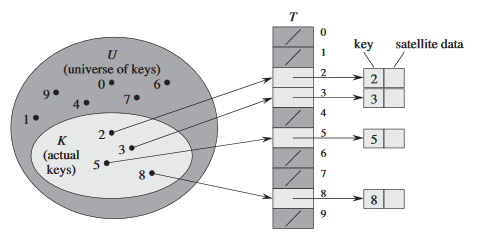
**Hash tables:**

Figure 1 U is the size of the table, while K represents the slots that have been calculated from keys.

A big array. Rather than having entries stored one after another, the table use an entry’s key to calculate a position within the table where it can be placed. When you need to look up the entry, simply enter it’s key and it will calculate where that entry is supposed to be.