CSCI 2270 Lecture Notes

3/15/19

Binary Heaps

* binary tree
* not a BST
* satisfies the heap ordering property
  + min-heap
    - the value of each node is greater than or equal to the value of its parent
  + max-heap
    - the value of each node is less than or equal to the value of its parent
  + parent/child relationship
  + has to be a complete tree
    - height between two branches must be less than or equal to 1
* Applications
  + sorting algorithm
    - if you put everything in a min heap, then remove each one by one, it will come out sorted
  + priority queues

Heap Implementation

* Store a tree in an array
* root of the tree is always in element 0
* for any element at index x, the left child index is 2x+1
* right child index is at 2x+2
* parent index is floor((x-1)/2) \*\*can just do integer division in c++, don’t need floor function\*\*

Min Heap Abstract Data Type

* public:
  + constructor
  + insert(value)
  + int peek()
  + printHeap()
  + extractMin()
* private
  + int \*heap //point to 0th element of the array (root)
  + int capacity; //optional
  + int currentSize; //keep track of # of elements
  + void minHeapify(index) //recursive method to fix heap
  + int parent(index){return (i-1)/2)}
  + int leftC(index){return 2i + 1}
  + int rightC(index){return 2i + 2}
  + void swap(int \*x, int \*y)

Min Heap Constructor

MinHeap(int c)

currentSize = 0;

capacity = c;

heap = new int[capacity];

Insert Function

* if currentSize == capacity
  + “full heap”
* else
  + append the value to the end of the array
  + if heap property violated (value is smaller than parent for example)
    - repair the heap
      * swap value with its parent until value is greater that or equal to its parent

Extract Min Function

* if currentSize <= 0
  + return error
* if(currentSize == 1)
  + return heap[0]
  + currentSize—
* else
  + x = heap[0]
  + move the last element of the array into the root position
  + call minHeapify(0)
  + return x