Discussion Week 8 - Priority Queues & (Binary) Heaps

Last updated Spring 2018

Topics Covered

- Priority queues
- Heaps (min/max heaps)
- Uses of heaps
- Remove an element from a heap
- Adding an element to a heap
- Representing a heap as an array

Worksheet

Following the slides, implement a binary heap that uses an array to store its elements.

Skeleton code has been given to you. The heap should afford the following methods in its API:

- `int Peek`, which returns the root if the heap is non-empty and throws an exception otherwise
- `void ExtractMax`, which removes and returns the root if the heap is non-empty and throws an exception otherwise
- `void AddElement(integer key)`, which adds the element to the heap if it is not full and throws an exception otherwise

Write unit tests for the following edge cases: - Checking if a newly created heap is empty - Checking that a heap that was recently added to is non-empty - Checking that a heap that has N elements added to it and is extracted from N times is empty - Checking for an exception on extracting from an empty heap - Checking for an exception on adding to a full heap - Adding an element and then immediately extracting it - Adding many elements and then extracting them back out. For this, use a `for` loop to add many elements (1000+), and a reverse `for` loop to remove them again.

If you're feeling adventurous, you can witness the speed of this heap implementation by writing a unit test that adds 2^20 elements and then extracts them back out. My implementation takes ~900ms to do this. Yours should be about the same.

Micro-quiz