# CSCA48 Winter 2015

## Week 8 - Priority Queues and Heaps

Anna Bretscher

March 2, 2015

# Priority Queue

A *priority queue* is a variation on a standard *queue*. It has:

- A set of elements
- Each element has a priority
- The operations are:
  - enqueue(x,p) or insert(x, p): Insert an *element x* in the set, with *priority value p*.
  - is_empty(): Return whether the priority queue is empty.
  - extract_min(): Remove and return an *element x* with the smallest *priority value p*.

# Applications of Priority Queues

**Check**: If your priority queue represents tasks, does extract_min() return a task that is most important or least important?

**A:** The item with minimum priority is most important. Think of a top 10 list. The best or most important item on the list is #1.

**Q:** Priority queues are very useful–what might some of their *applications* be?

- Job scheduling in operating systems
- Printer queues
- Event-driven simulation algorithms
- Greedy algorithms

# Implementing Priority Queues

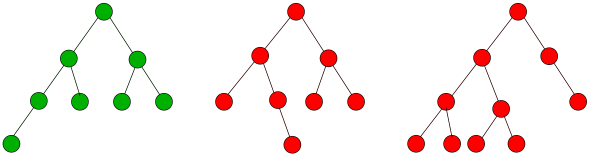
**Q:** What are several possible *data structures* for implementing *priority queues?*

1. Unsorted list
2. Sorted List
3. Binary Tree
4. Heap

# HEAPS FOR PRIORITY QUEUES

A *heap* is a *binary tree T* of elements with *priorities* such that the following *heap properties* hold:

- *T* is complete: Every level of the tree is full except perhaps the bottom one, which fills up from left to right.



Complete    Not complete    Not complete

# Heaps for Priority Queues

A *heap* is a *binary tree T* of elements with *priorities* such that the following *heap properties* hold:

- Priority Property: For each *node x* in T, let p(x) be the priority of x. Then

$$p(x) < p(\text{left-child}) \text{ and } p(x) \leq p(\text{right-child})$$

# HEAPS

We can conclude a few immediate *facts* about *heaps* from the definition:

- What can we say about the *root*?
  $\rightarrow$ It contains the *minimum* element.
- What must be true about every *subtree of a heap*?
  $\rightarrow$ It is also a *heap*.
- If a heap contains *n nodes*, what is its *height h*?
  $\rightarrow$ Since heaps are *complete*, a *heap* contains $\Theta(\log n)$.

# Implementing Heaps

Traditionally, a heap is implemented by using

- An array (or list)

  or

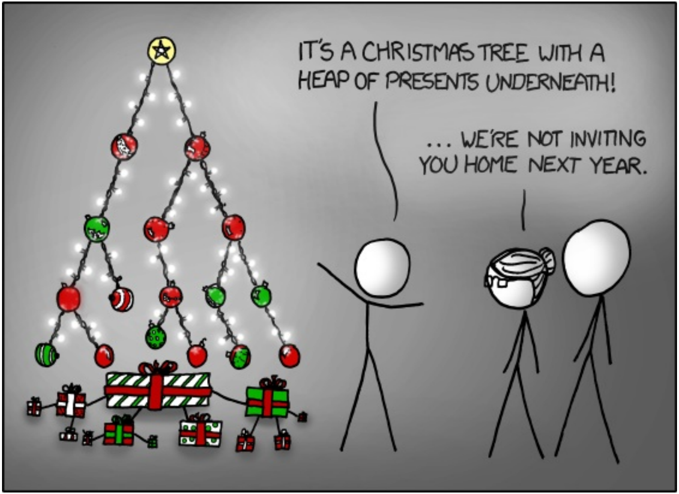
- A binary tree (rare)

# Implementing a Heap With a BST

The operations we need for a `heap` class are:

- `__init__`
- `insert(x, p)`
- `extract_min()`
- `__str__`

Let's write them!


IT'S A CHRISTMAS TREE WITH A HEAP OF PRESENTS UNDERNEATH!
... WE'RE NOT INVITING YOU HOME NEXT YEAR.