Project 3, Standard Template Library (STL)

Bryce Boe

2013/11/25

CS24, Fall 2013

Outline

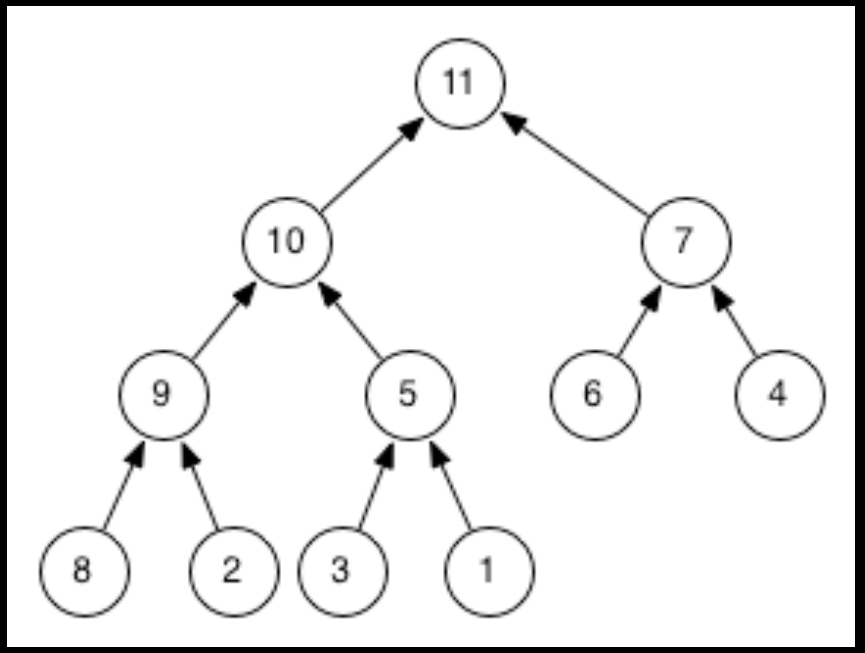
- Priority Queue Review
- Project 2 Solution
- Project 3 Overview
- When should I?
- C++ Standard Template Library (STL)

Priority Queue

- Queue-like ADT where each item is assigned a priority, and the highest priority item is always dequeued
- Implementations
  - 1 **queue** per priority (fixed number of priorities)
  - Build on top of a **List** with a sorted insert operation
  - **Heap**

Heap

- A **complete** tree where each node's parent has a higher or equal priority



Project 2 Solution

- <In class look at my solution>

PROJECT 3

Project 3 Overview

- Concept: Hospital emergency room log
- Logs when (1) patients arrive, (2) patients are visited by a doctor, and (3) when patients leave

Input File

Priorities
01 Cough
10 Bleeding
Doctors
Dr. Doctor
Patient Arrivals
08:00 “Patient A” “Head ache” 25
08:00 “Patient B” “Bleeding” 60

Output File

08:00 Patient A arrives

08:00 Patient B arrives

08:00 Dr. Doctor visits Patient B about Bleeding

09:00 Patient B departs

09:00 Dr. Doctor visits Patient A about Head ache

09:25 Patient A departs

Data Structures to Use

- Queue
  - Patient arrivals (1)
  - Available doctors (2a)
- Priority Queue
  - Patients who are waiting to see a doctor (2b)
  - Patients who are currently seeing a doctor (3)
- Hash Table
  - Mapping of symptom to priority

Suggested Schedule

- By Friday (11/29) – complete the parsing of the input
  - Store arrivals in an arrival queue
  - And store priority mapping in a hash table (might want to complete lab 10 first)
- By next Monday – pass the 1 doctor tests
- By next Wednesday – pass most of the multiple doctor tests
- By the deadline – handle the corner cases for 100%

WHEN SHOULD I?

When should I use recursion / iteration?

- Use whatever you are more comfortable with
- Consider:
  - Recursion usually results in less code (arguably means less development time, fewer bugs)
  - Recursion requires extra memory
    - Good: <= O(log(n))
    - Acceptable: O(n) – I prefer an O(1) iterative solution
    - Bad: > O(n)

When should I use an array or a linked implementation?

- Tradeoffs
  - Space
    - Arrays often have wasted space (holes)
    - Linked nodes require a constant factor more memory per node
  - Locality of reference (memory caching)
    - Arrays are in contiguous chunks of memory thus have tremendous caching performance gains
    - Whereas linked nodes may require fetching multiple pages from memory

STANDARD TEMPLATE LIBRARY (STL)

C++ Standard Template Library

- algorithm
- vector
- queue
- unordered_map (lab 10)
  - Also known as a hash table
  - Expected insert / contains / remove: O(1)
  - Trades space for size

HAPPY THANKSGIVING!