Data Structures with C++ : CS189

Lecture 13-1: Priority Queue and Heap

Recap

- A hash function takes a piece of data and turns it into a number
  - Number is consistent and ideally varying
- That number becomes the index in an unordered_map's internal vector
- Since a hash function's result is unreadable, it can accidentally be used as encryption
  - Not anymore, it's been broken

Queue Recap

- A Queue is an Adapter around a List
- An adapter does no work, all it does is reroute commands to the inner container
- The first thing put in a queue is the first one to come out
  - push_back, pop_front
  - Stack is push_back, pop_back
- You cannot access anything but the front item

Priorities

- You may write a program that wants the orderly behaviour of a queue, but also wants to be able to have some data jump to the head of the line
  - In a lab, there is a printer queue. First job submitted is the first printed.
  - Faculty have a higher priority, so when I submit a job it goes straight to the front of the line
- This is a priority_queue. A queue that takes a comparator argument to decide who goes first
  - If enough jobs cut, it is possible the last in the queue never gets to the front. Consider that when picking a container

STL's priority_queue

```
#include <queue>

priority_queue<int> IDs;

priority_queue< Student,
vector<Student>,
Student::ByAge >
ages;
```

- STL's PQ is similar to all the others
- If you give it a T that doesn't understand what < means, you need to provide a Functor for Comparing
- Without a functor you get < behavior with one stupid stupid change to what you'd think would happen
- < is the default, and gives you the *largest* data first
  - A "Maxheap"
  - Please tell me I'm reading that wrong
  - https://www.cplusplus.com/reference/queue/priority_queue/

How : List

Information hiding says you don't get to know how a PQ works. There are several ways it could be done

Linked List
Deque
Cache
Heap

- How about a plain old List?
- If you wanted to sort a list of T, you'd need a functor because T's might not have <
  - Same rule as Set
- Just replace < with Better* and that's it
  - It's O(n) though so we want to beat that
  - *Better means: less than, greater than, older, yellower, taller, more in debt…
- Loop through the list and Insertion Sort it

How : Deque /dek/

- A Deque is a container with the insert speed of List and the lookup speed of Vector
- It does that by putting ListNode pointers in the vector that point at the correct order
- Replace "correct" with "better" and you're done
    - But you are still O(n)

How : Cache

- Continuing the Printer example, maybe students all have the same 0 priority, but faculty have a priority based on years worked
- I don't want the addition of my print job to have to loop through all the students I know I don't care about in O(n)
  - I do care about you irl though
- Faculty get their own PQ and students get a normal queue
- Like how you can cut at MouseLand with a Smart Pass

How : Heap

- Welcome to 189, where care about code being fast as well as correct
- A heap gets O(log n) by behaving like a tree, but still decorating a vector.
- Picture a tree that is packed as tight as possible.
    - All data is as high as it can be, and as left as it can be
- If you started at the root, you could read all of the data left to right on each line like a book
    - And you could put that in a vector

Heap is not a Tree

- If you gave each row a letter, and each successive item an increasing number, you'd have:
  - The X is blank for reasons coming up
  - X, A1, B1, B2, C1, C2, C3, C4, D1, D2, D3…
- We are only going to *picture* that as a tree, because it helps understand adding and removing
  - We are also not left-right ordered like a Set is
  - We are top-down ordered though since we want the best at top
    - Every node is better than all the nodes beneath it

Traversing the "Tree"

- If you took the last slide and wrote in the vector indices you'll see some interesting patterns
- If a node wants to find its parent, you just divide the index by 2
- If you want your children, that's doubling for the left and then adding 1 for the right
  - Up: i / 2
  - Down: 2*i and 2*i + 1
  - This is why index 0 is skipped in that list above

Push

- Still not a tree. First person who says "rotate" loses 50 imaginary points
- Start by doing a push_back with the new data
- If we are better than our parent, we switch places
  - Stop if they are better than us, or we made it to the root
- We don't care about left-right ordering at all, so we don't need to bother the rest of the tr-err, vector
  - And that's the reason for log n.

Pop

- We know we want the root, but we can't leave a hole
  - Which value do we put there?
- The overall size of the vector is about to go down by one
  - If we did nothing the rightmost data would be lost so…
- Take the last item and put it in the root's spot
- Then do the Push idea backwards, with one addition to handle the forking
  - Compute my better child
  - If it is better than us, swap with it

Heap

- A Heap is a vector sorted like a tree
- You can also use Heap as a verb
    - You can heap a vector with make_heap
- Therefore:
    - priority_queue is the container, heap is how
    - vector is the container, array is how
    - list is the container, dynamic nodes is how
    - set is the container, a tree is how
- Remember, I try to only be a jerk about vocab if it is important. These are important

End