A Priority Queue, Reading

A regular queue, as we learned in a previous module, is FIFO -- first in, first out. The oldest, longest-waiting value is the one that would be returned in a peek operation or removed in a pop. In a *priority* queue, it's the *most important* value -- the one prioritized ahead of all the other values -- that gets the top stop. It's a queue with "cuts".

Apple Computer headquarters in Cupertino, California, has a cafeteria. Employees queue up in lines, awaiting their turn to be served at the various food counters. When the CEO, Tim Cook, shows up for lunch, do you think he goes to the back of the line? Probably not -- his time is more important than others'. He probably goes to the *front* of the line to be the next to be served. Same for CEO Satya Nadella at Microsoft's Redmond, Washington, headquarters cafeteria. (Although in fairness to these guys, all this is pure speculation, and for all we know, they take their turns like anyone else -- but it serves as a pretty good illustration.)

One application of a priority queue in programming is in "simulations", supporting real-life engineering design and analysis, or real-time gaming. Priority queues are used to manage "event queues" where events in the simulation are scheduled (like airplanes arriving near an airport and being assigned landing instructions). The highest priority event is at the front of the queue, so as the simulation looks at the next scheduled event or processes and removes it, peek and pop work on that event.

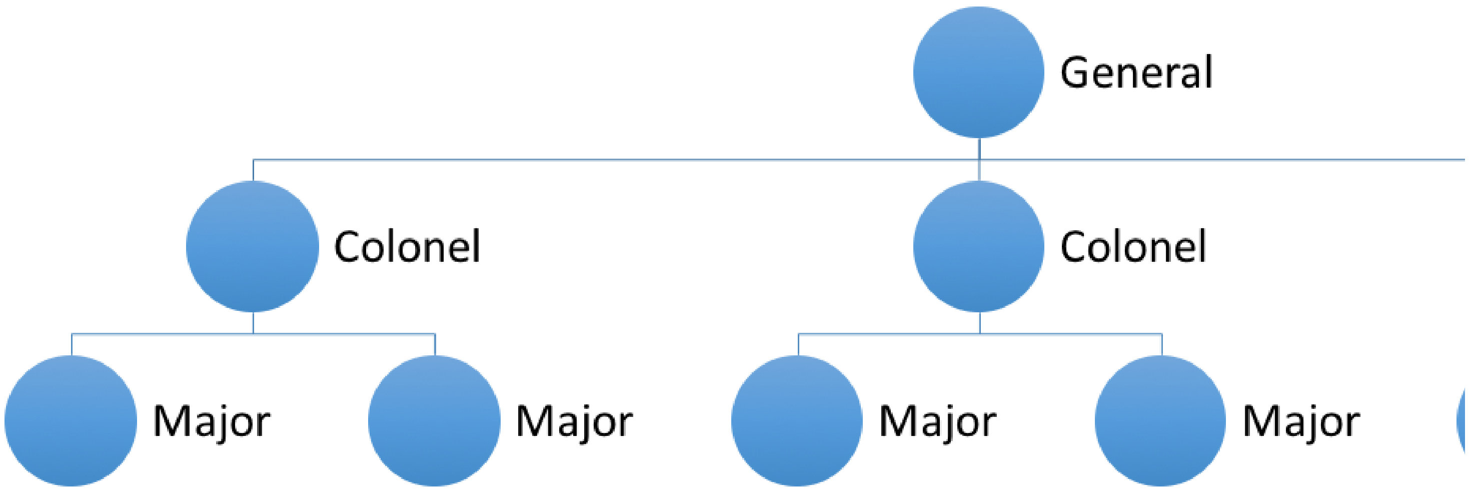
Modeling A Priority Queue

One option for peek and pop is to traverse the data array to find the highest priority value -- right away that should tell us "O(n)" and hopefully with all we've learned so far in this course, we'd look for a better answer.

The other option is for the push operation to seek a place in the data array to insert the value so that the array is always in order, highest-to-lowest priority. But that makes push O(n), so that's not ideal, either. We *still* need a better idea.

There's Got To Be A Better Way

Sometimes a good place to look for efficient ways to do things is in common everyday human experience. In this case, consider a military hierarchy, and how retirements and promotions and various arrivals and departures from the organization take place. Here's an example:



If the general retires, there are 3 colonels ready to replace him. When the highest-qualified colonel gets promoted, *his* higher-qualified major gets promoted, too. That "pop" operation took 3 steps. And if we were to include the lieutenants who report to the majors, effectively *doubling* the size of the data structure, that would add just one step to the pop operation. Let's look at the big oh of that operation:

$$\Delta f = 1 \qquad \Delta n = n \qquad \frac{\Delta f}{\Delta n} = \frac{1}{n} \qquad \frac{df}{dn} = \frac{1}{n} \qquad f = \int \frac{dn}{n} = constant \cdot \log(n)$$

Just to be sure, delta-f is the change in the number of operations involved in a pop -- one. Delta-n is the change in the data structure size, and since we're (approximately) doubling n, we're changing it by n. The rest is calculus. The big oh is O(log n), and recall from our study of big oh in an earlier module, O(log n) is closer to O(1) than it is to O(n). It's not our ideal O(1), but it's *really close*, and it's way better then O(n). There *is* a better way!

Push

But how about push -- is that any better than O(n), because if it's not, we're actually worse off with this new approach. Say a soldier gets transferred from another unit. How will he find his rightful place in his new organization?

The human thinking process would probably not work this way, but considering that a computer is going to have to take this over soon, let's think like a computer. An easy place to add a new value is as a subordinate to one of the lowest in the hierarchy -- a major in the diagram above. If the new arrival is also a major, his priority (based on seniority in the organization) would be lower the major to which he got attached, and he'd remain there. But it he was a colonel, he'd *swap* positions in the hierarchy. This would continue up the chain of command. As with pop, adding another layer of lieutenants under the majors would add one promotion operation while doubling the data structure size, and that too would be O(log n). Whew!