Set representation

In A* algorithm, we have the open set and closed sets.

There are three main operations we perform on the OPEN set: the main loop repeatedly finds the best node and removes it; the neighbor visiting will check whether a node is in the set; and the neighbor visiting will insert new nodes. Insertion and remove-best are operations typical of a priority queue.

In addition, *there's a fourth operation*, which is relatively rare but still needs to be implemented. If the node being examined is already in the OPEN set (which happens frequently), and if its `f` value is better than the one already in the OPEN set (which is rare), then the value in the OPEN set must be adjustment. The adjustment operation involves removing the node (which is not the best `f`) and re-inserting it.

Recommendation

Using a binary heap. In c++, we can use the priority_queue class, which doesn't have increase-priority, or boost's mutable priority queue, which does.

If you have more than 10000 elements in open set, then consider more complicated structures such as a busketing system.

Using bucket approaches or hashtable for membership.

Other structure need to consider:

- **Splay trees**

With splay trees, membership, insertion, remove-best, and increase-priority are all expected O(log F), worst case O(F).

- **HOT queues**
- **Pairing heaps**
- **Soft heaps**
- **Sequence heaps**

Hybrid representations

Hybrid representations#

To get the best performance, you will want a hybrid data structure. For my A* code, I used an indexed array for O(1) membership test and a binary heap for O(log F) insertion and O(log F) remove-best. For increase-priority, I used the indexed array for an O(1) test whether I really needed to perform the change in priority (by storing the `g` value in the indexed array), and then in those rare cases that I did need to increase priority, I used the O(F) increase-priority on the binary heap. You can also use the indexed array to store the location in the heap of each node; this would give you O(log F) for increase-priority.

In my newer A* tutorial, I use a hash table for membership and a binary heap for priorities. I further simplified by combining OPEN and CLOSED into the same set, which I call Visited.