Pragmatic Programming

Session 10 - Topological Sorting and Treaps

Max Nilsson

February 2nd

Help Max Nilsson Finish His Screenplay!

- Max Nilsson has finally finished writing all of his scenes of his fantasy screenplay.
- All scenes have been numbered $n \in [0, 10^5]$ and all he needs to do is to fit them together into one big chronological document.
- Max Nilsson has a list of $\approx 10^6$ links of the form

  ```
  u -> v
  ```

  which indicates that scene `u` must come before scene `v` in the screenplay (for instance, Kahn the king of termites must be introduced before he is betrayed by the termites, but the betrayal should of course not occur before he meets the wizard of thunder).
- The solution needs not to be unique and we know that the links does not form any cycles.
- The list of links can be found in the GitLab (when unzipped it is 14.5 MB). Please help him solve his problem.

Treap

- A blend of the **binary search tree** and the **heap**. Each node in the tree contains a key (sorted as in a binary search tree) and a random priority (structured as in a heap). The resulting shape is a randomized binary search tree with high probability of logarithmic height.
- Searching is the same as in a binary search tree in $\mathcal{O}(\log n)$, but insertion/deletion is different.
- **Insertion:** Create a new node with assigned key and random priority, and search in the treap for it. If it does not exist, insert it. While there is a contradiction of the heap structure, perform the standard right/left shifts of binary search trees until we reach the root. This is done in $\mathcal{O}(\log n)$.
- **Deletion:** Change the priority of the node to be deleted to be $-\infty$. Then we can perform suitable shifts until the node is a leaf of the tree, and then we can delete it. In total $\mathcal{O}(\log n)$.

Why Do We Care?

- There is a nice geometric interpretation of treaps, which motivate that the shape of a treap *forgets about its history*. What I mean by this is that inserting items in different orders always result in the same balanced tree shape (as long as the priorities are deterministic), unlike an AVL/RedBlack tree. This could have applications in security software engineering.
- A defining characteristic of a treap is that it easily solves the problem of a global split operation: given a key x and a treap t give me two smaller treaps t_1, t_2 such that

$$\mathsf{keys}(t_1) < x < \mathsf{keys}(t_2).$$

- This is messy in a normal balanced binary search tree and impossible (?) to solve in $\mathcal{O}(\log n)$. With a treap it is one line of code ...
- ... and that is to insert a node with key x and priority ∞.
- With similar ideas we can perform a union ($\cup$), intersection ($\cap$) of treaps (which can represent sets) giving efficient set operations. If two sets A, B with sizes n, m ($m \leq n$) respectively, then representing them with treaps computes $A \cap B, A \cup B, A \setminus B$ in $\mathcal{O}(m \log \frac{n}{m})$.

Commercial for my Upcoming Study Circle

Week 1	Binary heap and Treap
Week 2	Binomial heap and Fibonacci heap
Week 3	Min-max heap and Brodal heap
Week 4	Randomized meldable heap and Leaf heap
Week 5	Weak heap and Radix heap
Week 6	Symmetry heap and Soft heap
Week 7	Leftist heap and Skew heap
Week 8	d-ary heap and Pairing heap
Week 9	Beap and K-D Heap
Week 10	2–3 heap and B-heap

It will be **3 hp** and go during the summer.

(Bonus challenge: find the heap above that I made up)

This Week

- Skim read section 7.4 and 15.3 in Laaksonen[1].
- Solve half of this weeks 8 problems (truncated).

[1] Antti Laaksonen. *Guide to competitive programming*. Springer, 2020.