Algorithms for Programming Contests SS20 - Week 11

Mikhail Raskin, Tobias Meggendorfer, Stefan Jaax,
A.R.Balasubramanian, Christoph Welzel

July 7, 2020

A: Contact List - Sample Solution

- Build a trie from all possible contact names.
- Recursively visit the trie, starting at the root node.
- If a node marks a word end and has children, it has to be renamed.
- Count nodes that match this criterion.

B: J&J's - Sample Solution

- Toss J&Js into n glasses standing in a line.
- i-query: Add v J&Js to all glasses between ℓ and r inclusively.
- q-query: Return the number of J&Js in glass a.

B: J&J's - Sample Solution

- Operations needed: range update & point query
- Solution:
  - Use a segment tree with lazy propagation.
  - i-query: Add value v to interval $[\ell, r]$ using lazy propagation.
  - q-query: Sum up interval $[a, a]$.
  - Caveat: Without lazy propagation adding J&Js to every glass takes $\mathcal{O}(n \log(n))$.

C: Broken Tetris - Sample Solution

- Given a broken Tetris game which only considers rectangular pieces, calculate the maximal height at each stage of the game.

C: Broken Tetris - Sample Solution

- Build a segment tree over the playing field.
- Use $(\mathbb{N}, \texttt{max}, 0)$ as monoid.
- For a new piece $w \times h$ inserted at position p
  - Determine the maximal height h^* on interval $[p, p + w - 1]$.
  - Set the maximum on range $[p, p + w - 1]$ to $h^* + h$.
- Lazy propagation is avoidable as pieces have width ≤ 100.
  (might depend on the language)

D: Treehouse - Sample Solution

Problem

Given a rooted tree, find the shortest path to visit a sequence of nodes

Solution

Let `level`(x) be the distance from x to the root.
Then, the shortest path between nodes x,y in a tree is given by

$$\texttt{level}(x) + \texttt{level}(y) - 2 * \texttt{level}(\texttt{lca}(x, y))$$

- Precompute the `level` values.
- Compute LCA in logarithmic time per query, i.e. by using Segment Trees.

E: Ghost - Sample Solution

- Given a dictionary of words, find a strategy to win the n-th round of *Ghost*

E: Ghost - Sample Solution

- Main idea: Build a trie from the dictionary.
- Minimax algorithm:
  - Starting from the leaves, mark nodes if Lea can win or lose if she starts.
- Improvement: Ignore words where a prefix is already in the trie.

E: Ghost - Sample Solution

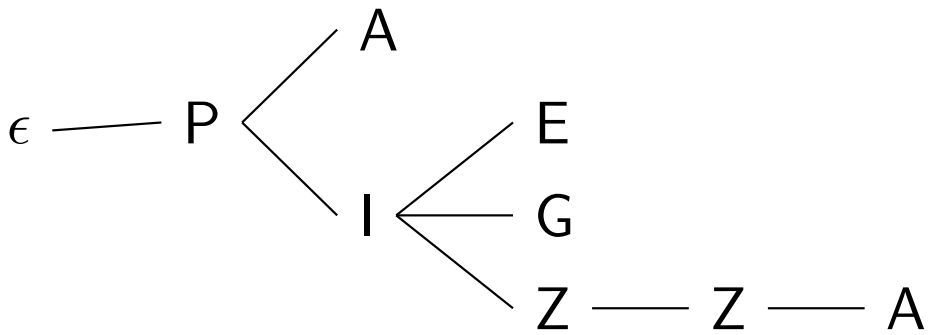
- Consider scenario 1:
  - Lea begins in round 1, the winner of a round begins the next.
  - Then Lea can win the last round iff either
    - she can win from the root node or
    - she can lose from the root node and n (the amount of played rounds) is even.
- Consider scenario 2:
  - Lea begins, but the loser of a round begins the next.
  - Then Lea can win the last round iff either
    - she can win and lose from the root node or
    - she can win from the root node and n (the amount of played rounds) is odd.
- Scenarios 3 and 4 are dual to 1 and 2.

E: Ghost - Sample Solution

- A node can be won if it
  - is a leaf node or
  - has a child that can not be won.
- A node can be lost if it
  - is not a leaf node and
  - has a child that can not be lost.
- Check for all nodes if they can be won or lost.
- Check what happens:
  - If root can be won and lost, Lea can force any outcome.
  - If root can not be won, but can be lost, Lea can win if n is even.
  - If root can be lost, but can not be won, Bea can win if n is odd.
  - If root can not be won and not be lost, Bea can force any outcome.

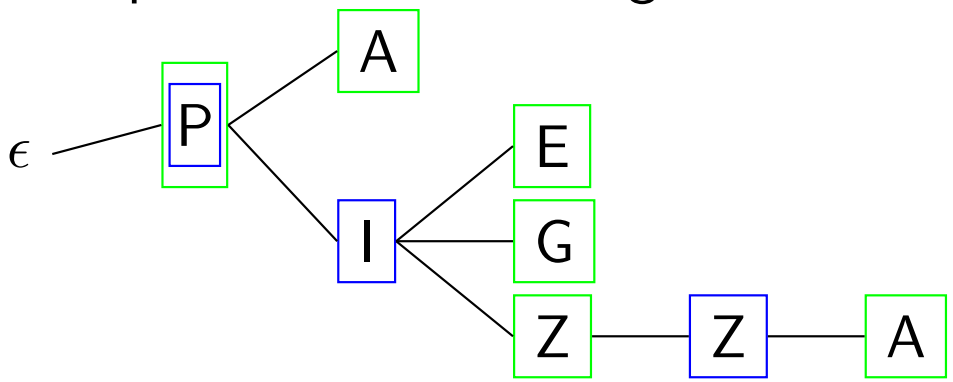
E: Ghost - Sample Solution

- Example trie:



E: Ghost - Sample Solution

- Example trie with labelling:



- Green labels: Can be won.
- Blue labels: Can be lost.
- Here, Bea can force any outcome.

E: Ghost - Sample Solution

Common Difficulties

- Organizing the trie through individual structs for each node has some overhead and bad cache efficiency.
- One of the words in the dictionary might be the prefix of another word. In this case, the game ends already when the shorter word is completed.

F: Catmon Go - Sample Solution

- Given spawn, despawn and catch events
- Find the sum of all Catmon in range of a catch event

F: Catmon Go - Sample Solution

- Build a segment tree over the locations, keeping track of the number of Catmon in a certain range.
- Spawn event: $+1$ at location a.
- Despawn event: -1 at location a if number is greater than 0. First recurse down the tree and then propagate change upwards (if there is any).
- Catch event: Sum in interval $[l, r]$, then set all numbers in this interval to 0. Again propagate change from full intervals upwards and use lazy propagation to later apply change to lower intervals.
- Caveat: Use fast input

F: Catmon Go - Sample Solution

- Without segment tree:
  - Keep a multiset of all Catmons ordered by their positions.
  - Insert on spawn and remove on despawn events.
  - Remove every single Catmon in the range on catch events one after another.
  - Since every Catmon in this range must have been inserted at some point, the amortized runtime of catch events is bounded.