Longest Kidney Donation Chain project

Introduction to Computer Science and Programming course

Ferdowsi University of Mashhad

Fall 2023

Dr. Ashkezari

Mohammad Ali Karbaschi

Problem

At the general hospital in Alpha City, there are n patients who need kidney donations. Each patient has a friend who is willing to donate. We only have information on the blood groups of each patient-friend pair.

The goal is to find the longest possible donation chain. The input consists of:

- n - the number of patient-friend pairs
- n lines with 2 numbers separated by spaces
  - The first number represents the patient's blood group
  - The second number represents the friend's blood group

The blood groups are encoded as:

- 100 - Blood group O-
- 101 - Blood group O+
- 200 - Blood group A-
- 201 - Blood group A+
- 300 - Blood group B-
- 301 - Blood group B+
- 230 - Blood group AB-
- 231 - Blood group AB+

With this input, we need to determine the longest possible chain of compatible kidney donations between patient-friend pairs. Maximizing chain length will allow the most transplants to take place.

Solution

This problem can be modeled as a directed graph, with patients representing nodes and potential donations representing edges. The goal is to find the longest path in the graph that satisfies the blood type compatibility constraints.

My approach is to generate all possible donation chains in a tree-like structure rooted at each patient node. I recursively explore paths, advancing to a new patient-friend pair only if the blood types are compatible for donation.

Once all possible chains are mapped out from each starting node, I can iterate through the chains and find the optimal ones with maximum length. This brute force approach examines the full space of options to guarantee finding the longest-length solutions.

The complexity is exponential since the chain options grow exponentially for each new patient added. While not the most efficient method for large datasets, for reasonably sized input this full enumeration approach provides an exact solution.

The key advantage of modeling the problem graph-theoretically is it allows bringing to bear graph algorithms for optimization and already-understood theoretical results. This simplified translating the real-world problem into a computational algorithm.

Implementation

The code begins by taking user input in the main driver logic and storing the data in a dictionary called Alpha_Hospital. Helper functions like Can_Donate() and Generator() are defined to check blood type compatibility and generate length 2 donation chains respectively.

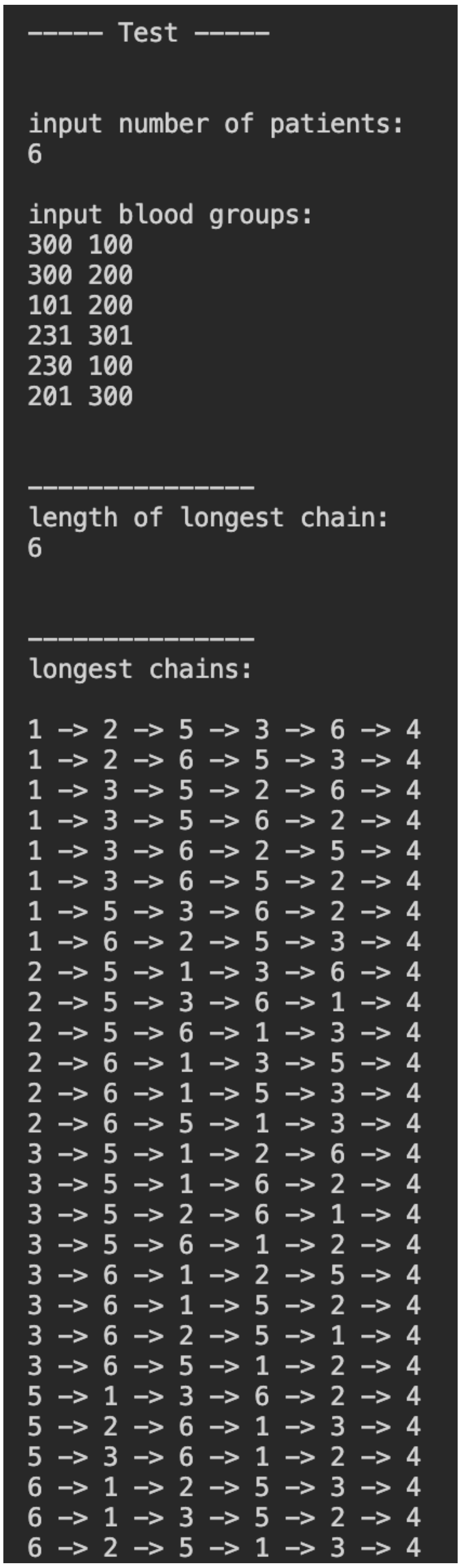
The Create_Tree() function contains the recursive depth-first search that builds out the tree of all possible donation chains from each starting node. It utilizes the next() function to explore valid new branches where blood types match between donor and recipient based on Can_Donate().

A key challenge was ensuring the donation chains were valid for both patients and donors when traversing the tree. Since the initial traversal used patient nodes, this meant the starting patient would not receive a donation. To address this, the Final_Checking() function does post-processing on the chains. It checks that the last donor can donate to the first patient, and if so, appends the starting node to close the chain.

Performance optimizations are also done like skipping over redundant subsequences. The maximum chain length is found by iteration and the optimal maximum length chains are extracted.

The complexity of Create_Tree() is exponential due to the recursive traversal, but Can_Donate() constraints bound it by pruning incompatible branches. The space complexity is linear in the length of input pairs n. While not the most efficient, this recursive brute force approach in Create_Tree() guarantees optimal chains.

Test



```
------ Test ------


input number of patients:
6

input blood groups:
300 100
300 200
101 200
231 301
230 100
201 300



---------------
length of longest chain:
6



---------------
longest chains:

1 -> 2 -> 5 -> 3 -> 6 -> 4
1 -> 2 -> 6 -> 5 -> 3 -> 4
1 -> 3 -> 5 -> 2 -> 6 -> 4
1 -> 3 -> 5 -> 6 -> 2 -> 4
1 -> 3 -> 6 -> 2 -> 5 -> 4
1 -> 3 -> 6 -> 5 -> 2 -> 4
1 -> 5 -> 3 -> 6 -> 2 -> 4
1 -> 6 -> 2 -> 5 -> 3 -> 4
2 -> 5 -> 1 -> 3 -> 6 -> 4
2 -> 5 -> 3 -> 6 -> 1 -> 4
2 -> 5 -> 6 -> 1 -> 3 -> 4
2 -> 6 -> 1 -> 3 -> 5 -> 4
2 -> 6 -> 1 -> 5 -> 3 -> 4
2 -> 6 -> 5 -> 1 -> 3 -> 4
3 -> 5 -> 1 -> 2 -> 6 -> 4
3 -> 5 -> 1 -> 6 -> 2 -> 4
3 -> 5 -> 2 -> 6 -> 1 -> 4
3 -> 5 -> 6 -> 1 -> 2 -> 4
3 -> 6 -> 1 -> 2 -> 5 -> 4
3 -> 6 -> 1 -> 5 -> 2 -> 4
3 -> 6 -> 2 -> 5 -> 1 -> 4
3 -> 6 -> 5 -> 1 -> 2 -> 4
5 -> 1 -> 3 -> 6 -> 2 -> 4
5 -> 2 -> 6 -> 1 -> 3 -> 4
5 -> 3 -> 6 -> 1 -> 2 -> 4
6 -> 1 -> 2 -> 5 -> 3 -> 4
6 -> 1 -> 3 -> 5 -> 2 -> 4
6 -> 2 -> 5 -> 1 -> 3 -> 4
```