Approximation Notes – Alston Knight

I chose to build my longest-path approximation around a mix of randomization and greedy behavior mainly because I implemented the project in C first. Since C is fast and low-level, it made sense to take advantage of that speed by sampling many randomized greedy walks instead of trying to engineer a heavier or more complex deterministic approach. Random exploration helps escape bad local choices, while greedy selection gives the method a strong baseline direction.

Working in C also meant I had to re-create a few things that Python gives you for free. The most annoying part was building my own hashmap. Python's dict handles hashing, resizing, probing, memory, and collisions automatically, but in C I had to write all of that manually. My hashmap works, but it's not optimal — it uses simple linear probing, no rehashing step, and makes extra allocations for keys and values. It's fine for this assignment, but not something I'd use in a production system.

Overall, C's speed made randomization feasible and efficient, but implementing data structures by hand was definitely a trade-off.