ECE374 SP23 HW7

Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

Problem 5

You are given a directed graph $G = (V, E)$ with possibly negative weighted edges.

(a) Suppose you know that the shortest path between any two vertices is guaranteed to have **at most** k edges. Give an algorithm that finds the shortest path between two vertices u and v in $O(k(n+m))$ time.

(b) Suppose you know the shortest path from u to v contains **exactly** k edges. Give an algorithm that finds the shortest path between two vertices u and v in $O(k(n+m))$ time. The path need not be simple.

Solution

(a)

Intuition. Assume the path runs from u to v. We run the Bellman-Ford algorithm for k rounds, and then backtrack to find the distance of v from u.

To be more specific, we use a modified version of BFS to traverse all the nodes within k steps from u.

Pseudocode.

- Create a BFS queue, a set visited, and a dist array initialized with ∞ for all vertices except u
- Add u to queue and mark it as visited
- Initialize an iteration counter
- While queue is not empty AND counter is less than k
  - Dequeue a vertex w from queue
  - For each vertex x ***adjacent to w (Modification to B-F)***
    - $\text{dist}(x) = \min(\text{dist}(x), \text{dist}(w) + \text{weight}(w, x))$
    - If $\text{dist}(x)$ is updated, make w the parent of x
      - If x is not visited, mark x as visited and add x to queue.
  - Increment the counter
- Among all visited nodes, find the node v and its *distance* from u.
- Starting at v, backtrack the parents until we reach u, adding nodes along the way to the *path* in reverse order.

Time complexity of this modified BFS algorithm is $O(k(n+m))$, as it iterates at most k times, and each iteration processes all n vertices and m edges in the worst case.

(b)

Same as part A. We find the distance and path of $u \to v$ after exactly k rounds of Bellman-Ford.