UMCPC - Unimelb Competitive Programming Club

Week 4 Thursday Tutorial

Bellman-Ford Algorithm

- **What is a graph?**
    - (From MAST30011 Graph Theory)
    - A finite graph *G* consists of:
        - A finite set of *vertices* (*nodes*), denoted V(G)
        - A set E(G) of 2-elements subsets of V(G) called *edges*
    - An edge {u, v} is denoted uv for short
    - Note that uv = vu denotes the same edge - (only on undirected graph)
    - If e = uv is an edge of a graph then
        - u and v are adjacent
        - e joins u and v
        - e is incident with u and v
- **Drawings of a graph**
    - Usually with circle denoting nodes, and line denoting edges
    - Shape, size and position of such dots and lines are irrelevant
- **Shortest path problem**
    - Path - A walk of the graph where no vertices are repeated
    - Single-source shortest path (SSSP)
        - Computer shortest path from one source to any nodes in the graph
        - **Breadth-first search (BFS)** - Only works on unweighted graph
        - Dijkstra's algorithm (will be taught in COMP20003/COMP20007)
        - **Bellman-Ford Algorithm**
    - All Pairs Shortest Path
        - Floyd-Warshall Algorithm (will be taught in COMP20003/COMP20007)
    - Negative Cycle
        - A cycle where the sum of the edges < 0
        - No *cheapest* path is defined - You can loop in the cycle infinitely
- **Bellman-Ford - Algorithm**
    - (Run the algorithm on an example graph on board)
        - Example Graph

          Use graph from Algorithms in a Nutshell p.162
    - (Show the algorithm in pseudo code from textbooks/wikipedia)

        - *dist[]* array and *pred[]* array
        - Initialization: all *dist[]* = inf, *dist[s]* = 0, all pred[] = -1
        - ***Initialization***
        - ***Relaxations***
        - ***Negative Cycle detection***
- **Bellman-Ford - Correctness**
    - After ***ith* iteration**, all nodes with ***i* edges in their shortest path solution** will have their correct distances computed.
    - Every **shortest path** can only at maximum ***n - 1* edges long** (or unless there is a negative cycle, in which case the shortest is undefined)
    - From above, you computer all shortest path in a graph by a maximum of ***n - 1*** iterations
- **Bellman-Ford - Complexity**
    - O(V*E)
        - All edge relaxation in one iteration
        - Total of V-1 iterations performed on the graph
        - Hence (V-1)*E edges relaxation, O(V*E)
    - **Early termination**
        - Can terminate with all values of *dist[]* remain unchanged after one full relaxation.
- **Bellman-Ford v.s. Dijkstra**
    - Time Complexity
        - **Dijkstra (with Binary Heap) -** O((E+V)logV) - More practical in competitions
        - **Dijkstra (with Fibonacci Heap)** - O(E+VlogV) - Best compelxity
        - **Bellman-Ford** - O(V*E)
    - **Class of inputs**
        - **Dijkstra -** *Positive* graph only
        - **Bellman-Ford** - Tolerant to *negative edges,* capability of detecting negative cycles
- **Problems**
    - https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=499
    - https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8&category=714&page=show_problem&problem=5572

- Rough estimation of input complexity for the solution, unsolvable by pure Bellman-Ford