Graph

Tuesday, 24 January 2023 1:38 PM

Graph:
ArrayList representation
Matrix representation
https://www.geeksforgeeks.org/graph-and-its-representations/

https://towardsdatascience.com/10-graph-algorithms-visually-explained-e57faa1336f3

1. https://www.geeksforgeeks.org/program-to-count-number-of-connected-component
2. https://www.geeksforgeeks.org/program-to-count-number-of-connected-component
   a. https://www.geeksforgeeks.org/find-the-number-of-islands-using-dfs/
   b. https://www.geeksforgeeks.org/islands-in-a-graph-using-bfs/

3. Swim in rising water -> https://massivealgorithms.blogspot.com/2018/12/leetcode-77 water.html
   a. https://dev.to/seanpgallivan/solution-swim-in-rising-water-4gfe#:~:text=You%2 20a,the%20grid%20during%20your%20swim.
   - https://just4once.gitbooks.io/leetcode-notes/content/leetcode/binary-search/7 water.html
   - LeetCode 70 Problem 4 - Swim in Rising Water
   - Dijikrstra's algorithm, BFS traversal with priority queue to choose the least path.
   - Already travelled path can be set as -1, use boolean node to set travelled path a

- https://www.geeksforgeeks.org/single-source-shortest-path-between-two-cities/
- As far from land as possible
  - https://jimmylin1991.gitbook.io/practice-of-algorithm-problems/dfs-and-bfs/11 possible
  - https://www.pepcoding.com/resources/data-structures-and-algorithms-in-java- levelup/graphs/as_far_from_land_as_possible/topic
- Cheapest flight within k stops -> https://aaronice.gitbook.io/lintcode/graph_search/ch stops
  - https://leetcode.com/problems/cheapest-flights-within-k-stops/solutions/1155 Queue-Solution/
  - BFS logic:
    - Insert into queue with k-1 stops for all the adjacency nodes for the curren
  - DFS pruning logic:

s-in-an-undirected-graph/

s-in-an-undirected-graph/

78-swim-in-rising-

0can%20swim%20from%

778-swim-in-rising-

.

s true.

62.-as-far-from-land-as-

-

heapest-flights-within-k-

41/JavaPython-Priority-

t node

    - DFS pruning logic.

Minimum Spanning Tree

- https://www.tutorialandexample.com/minimum-spanning-tree
- Kruskal's algorithm - sort(edges), union-disjoint set strategy
    - https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-gre
- Prim's algorithm - select min weights and find connectivity
    - https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo

Topological sorting - https://www.geeksforgeeks.org/topological-sorting/

- Kahn's algorithm (BFS)
    - https://www.geeksforgeeks.org/topological-sorting-indegree-based-solution/

Dijikstra'a algorithm

- Single Source Shortest Path Tree(SPT)
- Based on prim's algorithm logic...but adding distances whereas prim's algorithm won't
- https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
- Return shortest path of each node from source vertex based on weights assigned
- Optimized one: use priority queue/min-heap and adjacency List representation
- BFS search
- https://www.geeksforgeeks.org/dijkstras-algorithm-for-adjacency-list-representation-
- https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-in-java-using-priori

- Difference between dijikstra's SST(Single Source Shortest Path) and prim's DST
    - https://www.baeldung.com/cs/prim-dijkstra-difference#:~:text=Dijkstra's%20algorith 20shortest,only%20works%20on%20undirected%20graphs
    - https://www.quora.com/Whats-the-difference-between-Prim-algorithm-and-Dijkstra-
- Bellman-Ford algo SST:
    - https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
    - It will be used to calculate distance of all vertices from source. Dist[u]
    - Using dynamic programming approach
    - Identify the shortest path for less vertices first and increase it gradually using
    - Bottom-up manner approach
    - O(VE) time complexity - outer loop iterate for (v-1) edges
    - Inner loop iterate all edges
    - Set A as source with 0 first
    - And gradually extend the calculation of short vertices to other vertices in each iteratio
    -

- Kruskal's algo MST

eedy-algo-2/

o-5/

t

-greedy-algo-8/
tyqueue/?ref=rp

m%20finds%20the%

-algorithm

on

- https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-a

Disjoint data structure

https://www.geeksforgeeks.org/disjoint-set-data-structures/
Job sequencing problem
Solution using disjoint set:
https://www.geeksforgeeks.org/job-sequencing-problem-using-disjoint-set/
Greedy about profits..so sort them the array in decreasing order according to profits
https://takeuforward.org/data-structure/job-sequencing-problem/

Spanning tree

https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-a

https://www.geeksforgeeks.org/shortest-path-for-directed-acyclic-graphs/

Floyd warshall algorithm - to find shortest path between any 2 nodes

- https://www.geeksforgeeks.org/finding-shortest-path-between-any-two-nodes-using-
ref=rp
- Youtube -> G-42. Floyd Warshall Algorithm
- O(V3) = 3 times nested for loop is required to identify shortest path between 2 nodes
- Iterate I and j to get different combinations of I and j
- Iterate K different paths to identify the shortest path between I and J.
- Calculate next[i][j] = next

Detect cycle using bfs in undirected graph
https://takeuforward.org/data-structure/detect-cycle-in-an-undirected-graph-using-bfs/

Bipartite graph:
G-18. Bipartite Graph | DFS | C++ | Java

lgo-2/

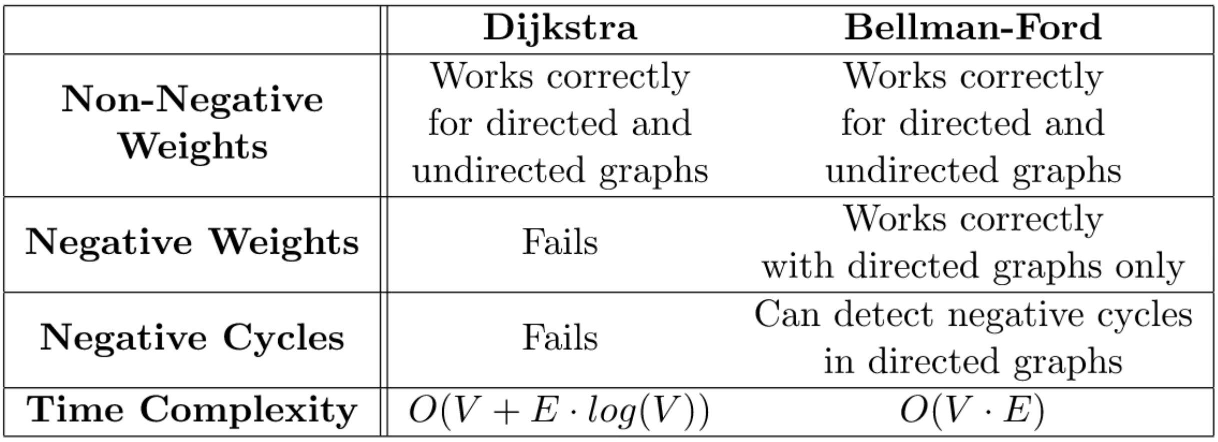
lgo-2/

-floyd-warshall-algorithm/?

Difference:

- https://www.tutorialandexample.com/dijkstras-vs-bellman-ford-algorithm
- https://www.geeksforgeeks.org/what-are-the-differences-between-bellman-fords-an
- https://www.baeldung.com/cs/dijkstra-vs-bellman-ford

Bellman Ford's Algorithm	**Dijkstra's algorithm**
When there is a negative weight edge, Bellman Ford's Algorithm detects the negative weight cycle.	When there is a negative weight ed may or may not work. However, it w in a negative weight cycle.
The vertices in the result contain information about the other vertices to which they are connected.	The vertices in the result contain all not just the vertices to which they a
It is simple to implement in a distributed manner.	It is difficult to implement in a distri
It takes longer to complete than Dijkstra's algorithm. It has an O time complexity (VE).	It takes less time to complete. O is t logV).
The algorithm is implemented using a Dynamic Programming approach.	The algorithm is implemented in a g



	Dijkstra	**Bellman-Ford**
Non-Negative Weights	Works correctly for directed and undirected graphs	Works correctly for directed and undirected graphs
Negative Weights	Fails	Works correctly with directed graphs only
Negative Cycles	Fails	Can detect negative cycles in directed graphs
Time Complexity	$O(V + E \cdot log(V))$	$O(V \cdot E)$

d-dijkstras-algorithms/

ge, Dijkstra's Algorithm vill not work if you are
network information, ire connected.
buted manner.
he time complexity (E
greedy manner.