ReviewSession1: StableMatching & ShortestPath

What is Stable Matching?

- **Definition:** A matching between two sets (e.g., men and women) where there is no pair (m, w) who would both rather be matched with each other than with their current partners.
- **Context:** Commonly illustrated using the "Stable Marriage Problem."
- **Why It Matters:** Real-world applications: College admissions, job placements, organ donation matching, etc.
- Ensures fairness and efficiency in allocation processes.

- **Basic Terminology:**
- **Participants:** Two disjoint sets (e.g., men and women).
- **Preference Lists:** Each participant ranks all members of the opposite set in order of preference.
- **Matching:** A set of pairs where each participant is paired with exactly one partner.

- **Stable Matching Criteria:**
- **Blocking Pair:** A man and a woman who prefer each other over their current partners.
- **Stability:** A matching with no blocking pairs.

The Gale-Shapley Algorithm (Deferred Acceptance)

- **Algorithm Overview:**
- **Idea:** Iteratively build a matching by having one set (say, men) propose to members of the other set (women), and the women tentatively accept the best proposal.
- **Steps:**
  - **Initialization:** All men and women are free.
  - **Proposals:** Each free man proposes to the highest-ranked woman on his list that he has not yet proposed to.
  - **Tentative Acceptance:** Each woman considers all proposals she receives and tentatively accepts the one she prefers most (she may “trade up” if a better proposal comes later).
  - **Repeat:** Continue until every man is matched.

- while there exists a free man m who hasn't proposed to every woman:
- w = highest-ranked woman on m's list to whom he has not yet proposed
- if w is free:
- match m and w
- else if w prefers m to her current partner m':
- match m and w, and m' becomes free
- else:
- m remains free (or moves to his next preference)

- **Properties:**

- **Stability:** Guarantees no blocking pairs.
- **Optimality:** The resulting matching is optimal for the proposing side (e.g., men-optimal if men propose).

Analysis and Properties of Gale-Shapley

- **Correctness and Stability:**
- **Proof Idea:** No blocking pairs exist because any potential pair would have been matched earlier in the process.
- **Time Complexity:** $O(n^2)$ in the worst-case scenario for n participants in each set.

- **Optimality vs. Pessimality:**
- **Proposer-Optimal:** The proposing side gets the best possible partner among all stable matchings.
- **Receiver-Pessimal:** The receiving side gets the worst acceptable partner among all stable matchings.

Question 1

Find an instance of stable matching problem where there are multiple solutions and point out the solution that G-S algorithm will return. (Assume men proposing).

	1st	2nd
M1	W1	W2
M2	W2	W1
W1	M2	M1
W2	M1	M2

Table: Table caption.

1. (M1,W1), (M2,W2)
2. (M1,W2), (M2,W1)

Question 2

Find an instance of statble matching problem of size n, such that G-S algorithm terminates in $O(n)$ iteration.

Simply assign each man with different most preferred woman. E.g. m_i prefers w_i the most. In this case G-S algorithm will run exactly n iterations as each man will propose to different woman.

Question 3

If every man has identical preference list, how many iteration does it take for G-S algorithm to terminate, give the precise answer in n.

With out lose of generality, let's assume that every man's preference list is exactly $(w_1, w_2, ..., w_n)$. G-S algorithm returns a stable matching $S = \{(m'_1, w_1), (m'_2, w_2), ..., (m'_n, w_n)\}$. Since every man has the same preference list. m'_i must have proposed exactly i times. Then the total number of iteration is $\sum_{i=1}^{n} i = \frac{(n+1)n}{2}$.

Question 4

Design an algorithm that determine whether an instance of stable matching has only one solution. Your algorithm should run in $O(n^2)$ time.

Run G-S algorithm twice, one with men proposing and the other with women proposing. The solution is unique if and only if the two runs return the same result.

Run G-S algorithm twice, one with men proposing and the other with women proposing. The solution is unique if and only if the two runs return the same result.

- ←: If two runs has the same result, this means that everyone's best and worst valid partners are the same, thus everyone can be matched with only one partner and the stable matching is unique.
- →: Trivially true.

Shortest Path

- **What is the Shortest Path Problem?**
  - **Definition:** The challenge of finding the minimum cost or distance between nodes in a weighted graph.
  - **Importance:** Used in networking (routing packets), navigation (GPS systems), and even in resource management (supply chains).
- **Key Applications:**
  - Road maps and navigation systems.
  - Internet routing protocols.
  - Robotics and game AI.
- **Graph Terminology Recap:**
  - **Vertices (Nodes):** The individual points (e.g., cities, routers).
  - **Edges:** Connections between nodes with assigned weights (e.g., distances, travel time).
  - **Directed vs. Undirected Graphs:** In directed graphs, edges have a direction; in undirected, they do not.
  - **Edge Weights:** Can be positive or negative (with caution on negative cycles).

Dijkstra's Algorithm

- **Overview:**
- **Purpose:** Finds the shortest path from a single source to all other nodes in a graph with non-negative weights.
- **Mechanism:** Uses a greedy strategy to always expand the nearest unvisited node.

- **Key Concepts:**
- **Priority Queue:** Ensures the next closest vertex is always chosen.
- **Distance Relaxation:** Update distances to neighboring nodes if a shorter path is found.

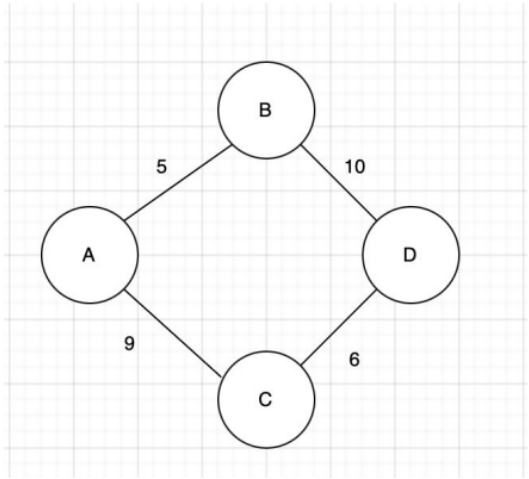
Dijkstra’s Algorithm

- **Pseudocode (Simplified):**
- Initialize distances to all vertices as infinity, except the source (0).
- Use a min-heap or priority queue to repeatedly select the vertex with the smallest known distance.
- For each neighbor, update the distance if a shorter path is found.

True or False ?

If all edges in a connected undirected graph have distinct positive weights, the shortest path between any two vertices is unique.

FALSE



True or False ?

Dijkstra's algorithm may not terminate if the graph contains negative-weight edges, i.e. the iterations may continue forever.

FALSE

- Even in graph with negative weight edges, Dijkstra's Algorithm will terminate.
- But it is possible that the shortest path found using Dijkstra's Algorithm may not be correct.

Solve the following Question

You want to go from node s to node t in a connected undirected graph G(V,E) with positive edge costs. You would also like to stop at node u if it is possible to do so without increasing the cost of your path by more than a factor of k.

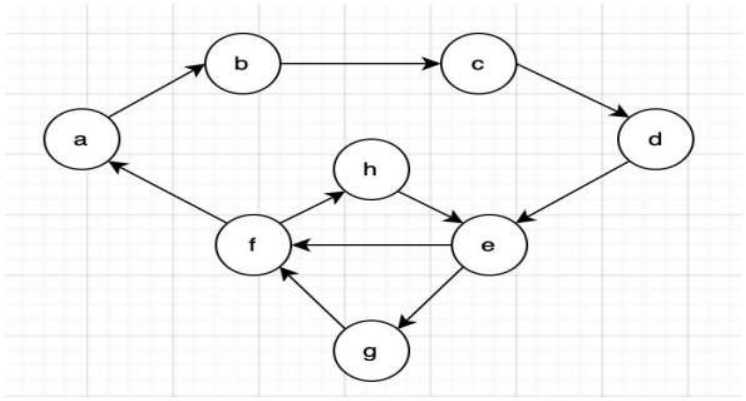
Design an efficient Algorithm to find the optimal path of going from node s to node t considering your preference of stopping at node u if possible.

Solution

- Run Dijkstra's Algorithm from node s to find the shortest path to node u [d(su)] and node t [d(st)]
- Also run Dijkstra's Algorithm from node u to find shortest path to node t [d(ut)]
- Compare the value of d(su) +d(ut) with d(st). If it is less than a factor of k then the path will contain node u else it will not
- Since we are running Dijkstra's Algorithm twice, total run time of the algorithm is O(|E| log |V|)

Solve the following Question

Consider a positively weighted directed graph. Design the most efficient algorithm you can for finding a minimum weight simple cycle in the graph. Be sure to prove that your algorithm is correct and find its running time.



Solution

Consider a minimum weight simple cycle and consider any edge $t \rightarrow s$ within it. The portion of the cycle from s to t is a simple path and it must be of minimum cost: if it were not then one could concatenate the better path from s to t with the edge $t \rightarrow s$ and obtain a cycle of lesser cost, a contradiction.

So we have shown that it certainly suffices to compute the shortest path from s to each vertex t and then check the cost of the cycle this path makes with the edge $t \rightarrow s$ (if it exists). By checking over all possible sources s and all edges to s we are guaranteed of seeing a minimum cycle at least once.

Solution

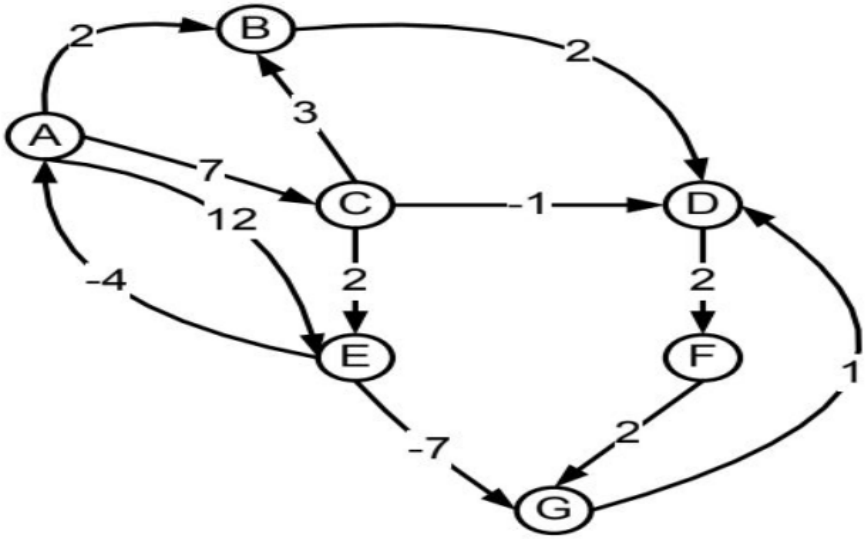
PSEUDO CODE :

```
best ← ∞
For each vertex s do
    {
      Run Disjkstra's algorithm from s
      For each edge t→s do
            best ← min(best, weight(t → s) + dist[t])
    }
```

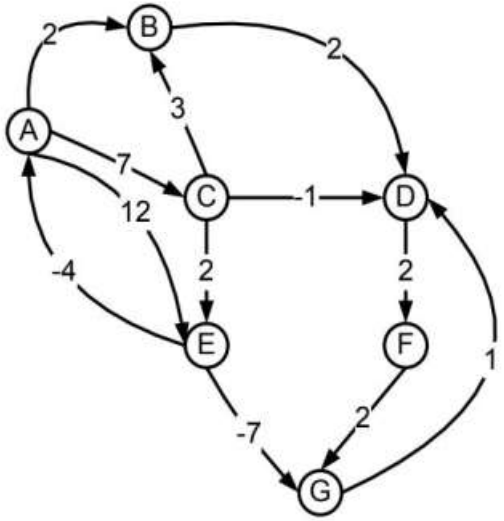
The algorithm clearly takes O(VElogV) time if one uses the heap implementation of Dijkstra,

Solve the following Question

Consider the following graph. Find the shortest path from Node A to all other nodes using Dijkstra's Algorithm.



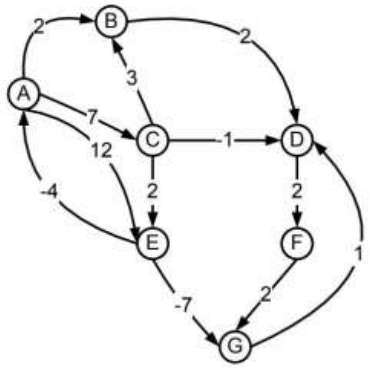
Solution



Nodes	Distance	Path
A	0	A
B	2	A-B
D	4	A-B-D
F	6	A-B-D-F
C	7	A-C
G	8	A-B-D-F-G
E	9	A-C-E

Solve the following Question

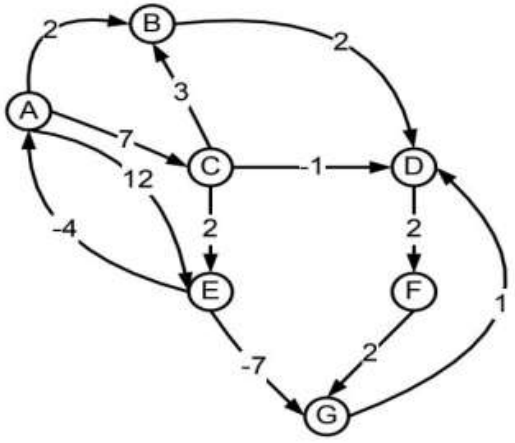
Dijkstra's algorithm found the wrong path to some of the vertices. For just the vertices where the wrong path was computed, indicate both the path that was computed and the correct path.



Nodes	Distance	Path
A	0	A
B	2	A-B
D	4	A-B-D
F	6	A-B-D-F
C	7	A-C
G	8	A-B-D-F-G
E	9	A-C-E

Solution

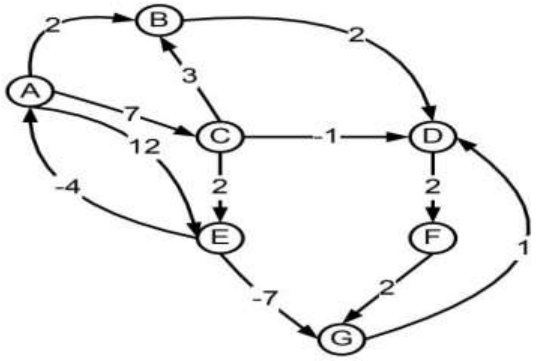
Computed path to G is A,B,D,F,G but shortest path is A,C,E,G (2).
Computed path to D is A,B,D but shortest path is A,C,E,G,D (3).
Computed path to F is A,B,D,F but shortest path is A,C,E,G,D,F (5).



Nodes	Distance	Path
A	0	A
B	2	A-B
D	4	A-B-D
F	6	A-B-D-F
C	7	A-C
G	8	A-B-D-F-G
E	9	A-C-E

Solve the following Question

List the minimum number of edges that could be removed from the graph so that Dijkstra' algorithm would happen to compute correct answers for all vertices in the remaining graph.



Solution

Removing the edge from E to G will allow Dijkstra's algorithm to compute correct shortest path for all vertices in the remaining graph.