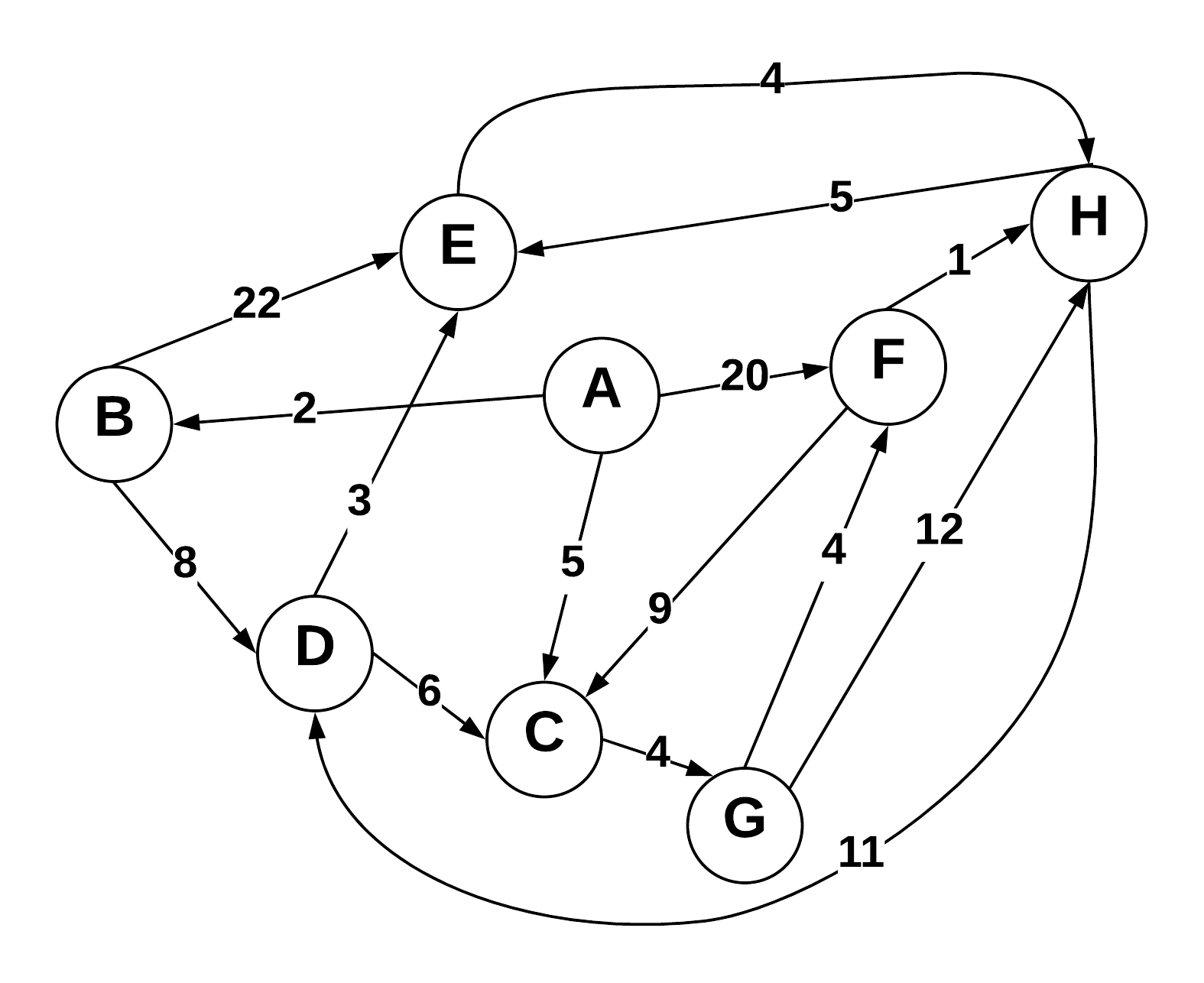
**Test 3**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Use pencil only.
* Write answers on the paper provided.
* Write clearly and legibly

|  |  |  |
| --- | --- | --- |
| Question | Possible | Score |
| 1 | 10 |  |
| 2 | 10 |  |
| 3 | 10 |  |
| 4 | 10 |  |
| 5 | 10 |  |
| 6 | 10 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Total: |  |  |

**1) Dijkstra’s Algorithm**



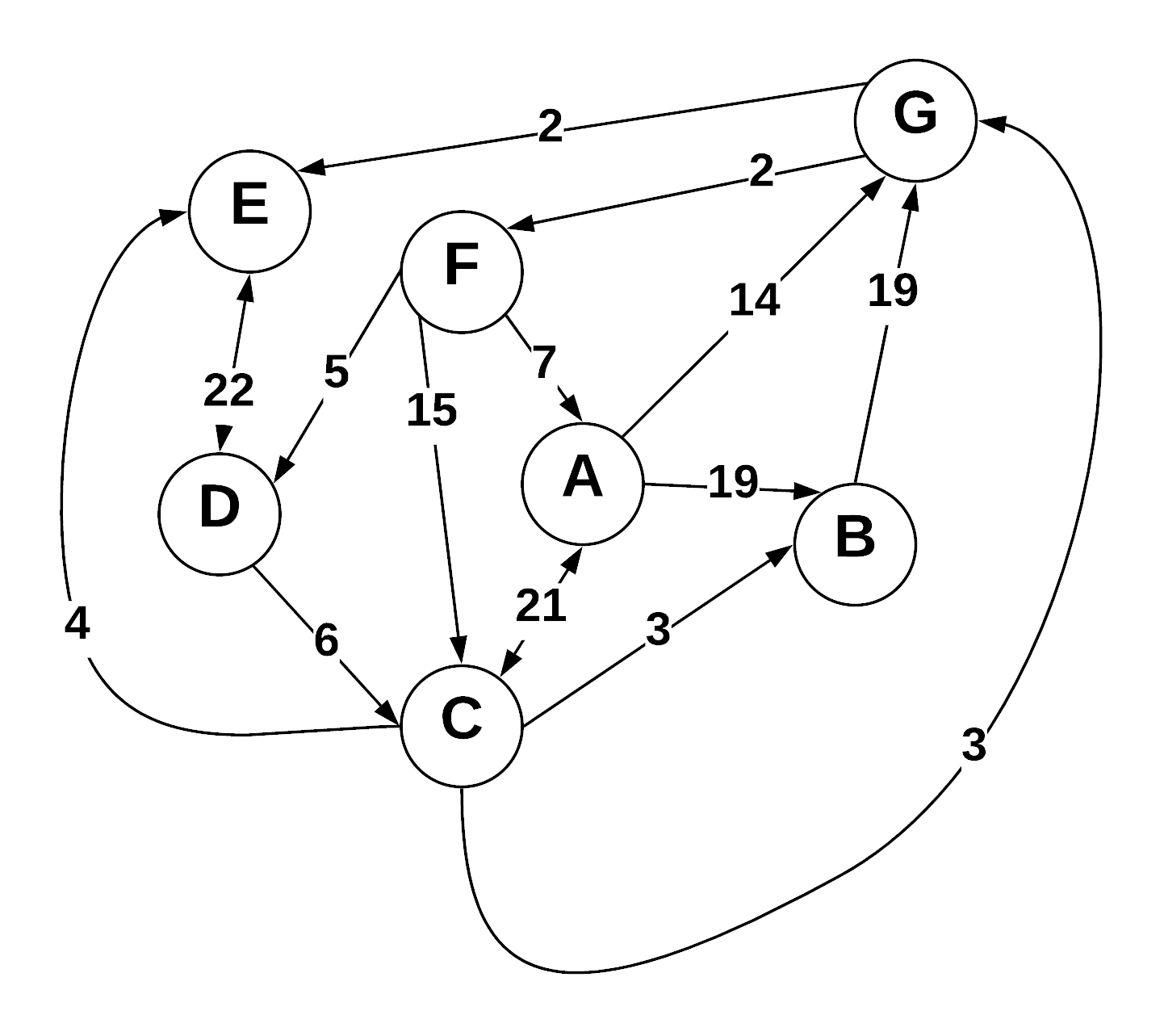
Use Dijkstra’s algorithm to compute the shortest paths from vertex A to every other vertex. Show your work in the space provided below. As the algorithm proceeds, cross out old values and write in new ones, from left to right in each cell. If during your algorithm two unvisited vertices have the same distance, use alphabetical order to determine which one is selected first. Also list the vertices in the order which Dijkstra's algorithm marks them as discovered.

Vertices in Order of Discovery:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Vertex | Known | Cost | Previous |
| A |  |  |  |
| B |  |  |  |
| C |  |  |  |
| D |  |  |  |
| E |  |  |  |
| F |  |  |  |
| G |  |  |  |
| H |  |  |  |

**2) Prims Algorithm**



Step through Prim’s algorithm to calculate a minimum spanning tree starting from vertex *G.* Show your steps in the table below. As the algorithm proceeds, cross out old values and write in new ones, from left to right in each cell. If during your algorithm two unvisited vertices have the same distance, use alphabetical order to determine which one is selected first. Also list the vertices in the order which Prims algorithm discovers them.

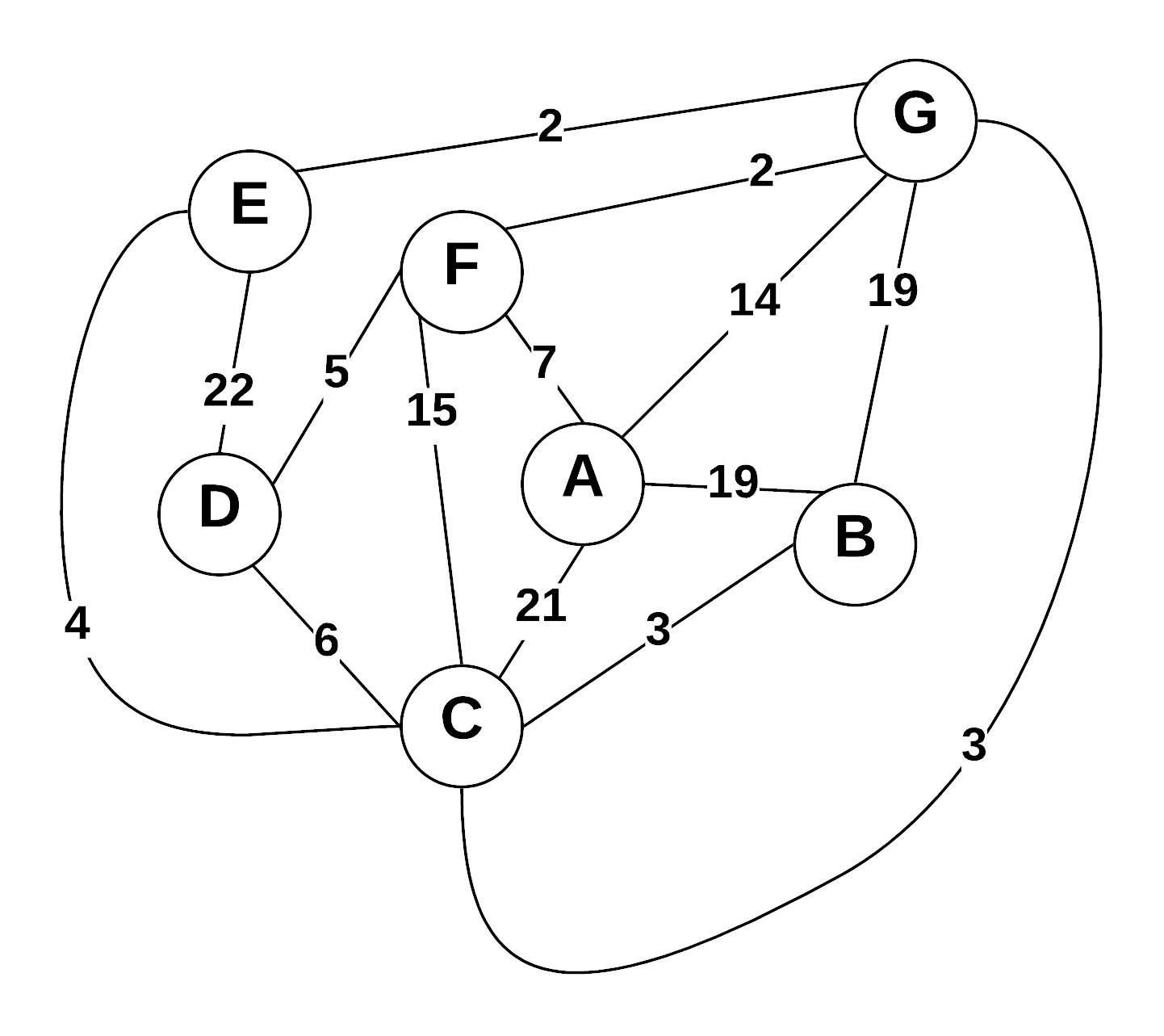
Vertices in Order of Discovery:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G |  |  |  |  |  |  |  |  |

* S = Vertices in spanning tree
* U = !S (vertices not in S)
* Cut = edges going across cut listed alphabetically: (A B) , (C D) , etc.

|  |  |  |
| --- | --- | --- |
| *S (spanning tree)* | *U* | Cut (alphabetize) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**3) Kruskel’s Algorithm**



Use Kruskal’s algorithm to calculate a minimum spanning tree of the graph. Show your steps in the table below, including the disjoint sets at each iteration. If you can select two edges with the same weight, select the edge that would come alphabetically last (e.g., select E—F before B—C. Also, select A—F before A—B).

* Edge Added: put edges added to MST marked as (A B), (E G), etc.
* Edge Cost: weight of edge added
* Running cost is total weight of spanning tree at the point another edge is added.
* Disjoint sets starts as: (A) (B) (C) (D) (E) (F) (G) , and as edges are added => (A) (B C) (D) (E) (F) (G)

|  |  |  |  |
| --- | --- | --- | --- |
| Edge Added | Edge Cost | Running Cost | Disjoint Sets |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**4) Prims Vs Kruskels**

* Explain why Prim’s algorithm is better for dense graphs, while Kruskal’s algorithm is better for sparse graphs.
* What data structures are used when implementing each algorithm?
* What are the run times for each algorithm?

**5) Greedy Algorithms**

1. Define “Greedy Algorithm”
2. Give an example of a greedy algorithm with explanation of its greediness and performance.
3. Can greedy algorithms produce “optimal” solutions? Short explanation.

**6) A-Star**

* Describe how the A-Star algorithm works using a list of steps.
* What makes A-Star different from Dijkstra’s algorithm?