Today: Single-source shortest paths. 24.{0,1,2,3}.
Next class: Homework 4 due. All-pairs shortest paths. 25.{0,1,2}.
Reminders: Read material *before and after* class. Use the class newsgroup. *Quiz* soon.

1. List the members of your group below. Underline your name.

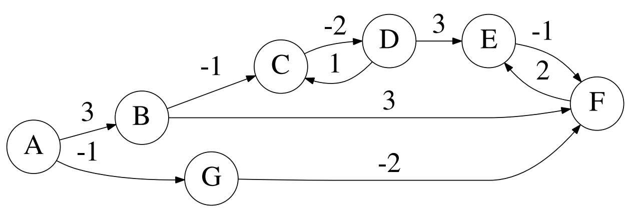
2. *Quick self-check.* (Use textbook conventions.)

   (a) If there is no path from A to B, the *shortest-path weight* $\delta(A, B)$ is [circle the correct answer]: (1) 0; (2) -1; (3) ∞; (4) undefined.

   (b) In order to use a *single-source* shortest paths algorithm to solve a *single-destination* shortest paths problem, we must: (1) negate edge weights (only); (2) reverse edges (only); (3) both negate edge weights and reverse edges; (4) use a completely different algorithm.

   (c) True or false: If $\langle A, B, C, D, E\rangle$ is a shortest path from A to E then running Dijkstra's shortest path algorithm with source vertex B must produce $\langle B, C, D\rangle$ as a shortest path from B to D, due to the *optimal substructure* of the problem.

   (d) Annotate each vertex in the following graph with the *shortest-path weight* from A (or indicate that it is undefined).

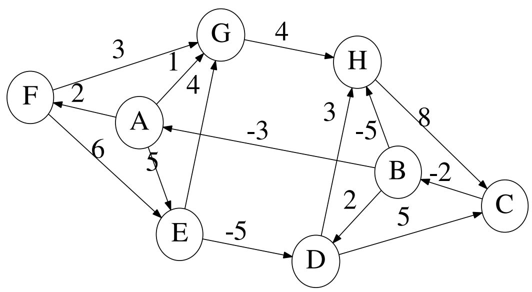


   (e) A shortest-paths algorithm can limit attention to *cycle-free* paths only if [circle all valid cases]: (1) there are no negative-weight edges; (2) there are no negative-edge cycles; (3) there are no cycles; (4) there are no unreachable vertices.

   (f) If a a graph has negative-weight edges but no negative-weight cycles, shortest paths may be computed using [circle all correct options]: (1) Dijkstra's algorithm; (2) Bellman-Ford algorithm; (3) Depth-first search; (4) none of the above.

   (g) The *predecessor subgraph* induced by the $v.\pi$ values produced by algorithms in this chapter (always): [circle all correct options]: (1) is connected; (2) is a tree; (3) is a DAG; (4) none of the above.

3. Trace the execution of the Bellman-Ford single-source shortest paths (SSSP) algorithm on the following directed graph, with vertex A as the source. Use the textbook's Fig. 24.4 (p. 652) as a model. Relax edges in lexicographic order. Annotate predecessor edges with check marks.



[additional space for answering the earlier question]

4. Repeat Question 3 using Dijkstra's SSSP algorithm, treating all negative edge weights as positive, and using the textbook's Fig. 24.6 (p.659) as a guide.

[additional space for answering the earlier question]

5. Informal homework: Augment the answer to Question 4 with the states of the primary data structures at each step (pairing heap, union-find).