Tutorial 5
Graph Algorithms

Graph Traversals

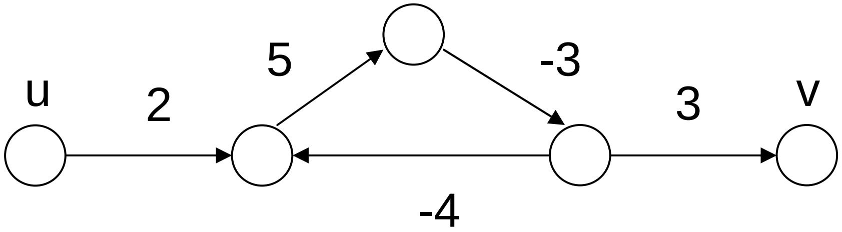
- DFS: Stack
- BFS: FIFO Queue
- Best First Search: Priority Queue (P296:6.7.1,6.7.2; P420 Ex-8.25, P417 Ex-8.9)

DFS

- DAG
  - Topological Order
  - Critical Path
- Directed Graph
  - Strongly Connected Component (P380 Ex-7.26)
- Undirected Graph
  - Biconnected Components (P382 Ex-7.35, 7.37, 7.38, 7.40)
- MST
  - Prim's Algorithm $\Theta(n^2)$ (Depends on the implementation of minPQ, see P417 Ex-8.9 for reference)
  - Kruskal's Algorithm $\Theta(m\lg m)$ (worst case)
- Single-Source Shortest Paths
  - Dijkstra Algorithm
  - Bellman-Ford Algorithm: negative weight cycle problem
- All-Pairs Shortest Paths
  - Floyd Algorithm
- Determine if a digraph has a cycle (P379 Ex-7.17)

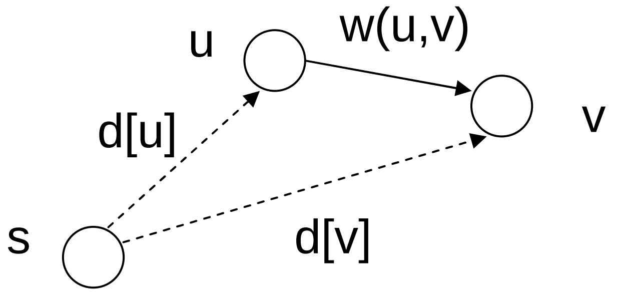
Bellman-Ford Algorithm

- CLRS P588 24.1
- Negative-weight cycles:
  - If a graph G=(V,E) contains a negative-weight cycle, then some shortest paths may not exist.



- Bellman-Ford algorithm: Finds all shortest-path lengths from a source s∈V to all v ∈V or determines that a negative-weight cycle exists

- RELAX(u,v,w)
  If d[v]>d[u]+w(u,v)
  then d[v]=d[u]+ w(u,v);
  pre[v]=u;
- Bellman-Ford(G,w,s)
  For each vertex v $\in$V do
  d[v]=∞;
  Pre=nil;
  d[s]=0;
  For i=1 to |V|-1 do
  for each edge (u,v) $\in$E do
  RELAX(u,v,w);
  for each edge (u,v) $\in$E do
  If d[v]>d[u]+w(u,v) return false;
  Return true;
- Time=O(mn)



Appendix: Solution to P379 Ex-7.17

- Determine if a digraph has a cycle
  - (a) Use DFS scheme:

    If a node see a "gray" node, then there is a cycle.
  - (b) Use BFS scheme, there are two conditions:

    If a node see a "black" node, and the two nodes have an ancestor-descendant relationship, then there is a cycle
  - Revise the DFS and BFS scheme accordingly.