Shortest path algorithm  
 - DAG - use topological sort to find shortest path  
 - negative cycles - no solution to shortest path problem  
 - negative edges -  
      - use Dijkstra's implementation from the book - works in E logV and worst case is EV  
      - use Bellman-ford to detect negative cycles as well. EV in all cases.  
  
Maxflow-mincut  
 - use Ford-Fulkerson   
     - find an augmented path (no full forward edges, no empty backward edges)  
     - keep looking for paths until no augmented paths exists  
     - find a mincut by traversing the graph (starting with s)  
         - follow a forward edge that is not full or backward edge that is not empty  
         - mark the vertex as in s-cut. Once no more marking is possible, then s-cut contains a set of nodes that minimizes the cut (fewest number of nodes) and maximizes the flow (maximum outflow possible)  
         - t-cut is the rest of the node  
         - one property to check is that all forward edges from s-cut to t-cut are full   
           and all backward edges from t-cut to s-cut are empty  
           (note that if the forward edges were not full, they would have been included in the s-cut and if the backward  
            edges are no empty, they would have been included in the s-cut)  
  
Seam Carving assignment  
  - the idea is to find a min-cost seam (vertical or horizontal)  
  - Example:  
      2   5    3  
      6   1    2  
      2   1    0  
   So the min vertical seam here would be as marked.  
      \*2   5    3  
      6   \*1    2  
      2   1    \*0  
   That also give the minimum sum (2+1+0) in a cumulative approach. All others have higher cumulative sums. rest is up to you to figure out. good luck.