Lecture 4

# Misc

1. String Tokenize: <http://www.cplusplus.com/reference/cstring/strtok/>
2. Big Integer: <https://github.com/anudeep2011/programming/blob/master/bigint.cpp>

# Graph

1. Disjoint Set Union  
   ~<https://www.topcoder.com/community/data-science/data-science-tutorials/disjoint-set-data-structures/>   
   ~<https://www.hackerearth.com/notes/disjoint-set-union-union-find/>
2. Minimum Spanning Tree
   1. Kruskal: <http://www.geeksforgeeks.org/greedy-algorithms-set-2-kruskals-minimum-spanning-tree-mst/>
   2. Maxi Mini Problem: <https://en.wikipedia.org/wiki/Widest_path_problem>
3. Warshall <http://www.geeksforgeeks.org/dynamic-programming-set-16-floyd-warshall-algorithm/>
   1. DP <https://www.youtube.com/watch?v=Qdt5WJVkPbY>
   2. Cycle Detection   
      Initiate the diagonal values as infinity. After applying Warshall, if any of the diagonal value is less than infinity, then there is a cycle.
   3. Negative Cycle Detection   
      Initiate diagonal value with 0. If any diagonal value is negative after applying warshall, then there is negative cycle.
   4. Print Path   
      <https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm> Look into Path reconstruction.
   5. Complexity: O(N^3)
4. Dijkstra  
   ~<https://www.youtube.com/watch?v=gdmfOwyQlcI>   
   ~[http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm](http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/)
   1. Weighted SSSP
   2. Conditions to fulfill  
      All edge must have cost >= 0.
   3. Complexity: O(E+VlogV)
   4. Minimum Length Cycle  
      Select an edge between two nodes A, B with cost C. Remove the edge and then run dijkstra from A and find shortest from A to B. If it is possible to reach B from A, then there is a cycle of length dist(A,B)+C.
5. Shortest Path Graph
   1. Detect whether edge in on a shortest path or not  
      How to determine if an edge between node U-V is part of shortest path or not?   
      <http://www.forthright48.com/#/cpps101/notes/shortestPathGraph.md> Check “Build of Shortest Path Graph” section.
6. DFS  
   <http://courses.csail.mit.edu/6.006/fall10/handouts/quiz2review.pdf> Read page 2 and 3.
   1. Color: Black, White, Gray
   2. Edge Name
7. Topological Sort  
   ~<https://class.coursera.org/algo-003/lecture/52>
   1. Using DFS: <http://www.geeksforgeeks.org/topological-sorting/>
   2. Using Indegree: <https://www.cs.usfca.edu/~galles/visualization/TopoSortIndegree.html>
8. SCC
   1. Tarjan  
      ~<http://www.geeksforgeeks.org/tarjan-algorithm-find-strongly-connected-components/>   
      ~[https://en.wikipedia.org/wiki/Tarjan's\_strongly\_connected\_components\_algorithm](https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm)

Code written during class:

<https://drive.google.com/file/d/0B9qzXJ_Svdc8SEp1R1FlXzZ4aWc/view>