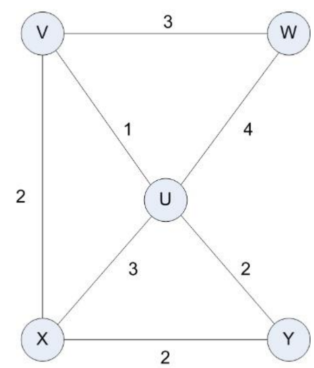
Lab 11 Solutions

Due Monday 2 pm

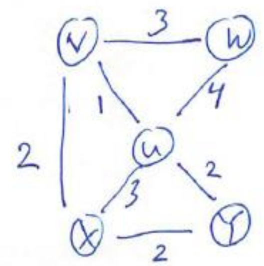
1. Carry out the steps of Dijkstra's algorithm to compute the length of the shortest path between vertex V and vertex Y in the graph below. Show all steps and indicate clearly your final answer (which should be a list A[] which shows shortest distances between V and every other vertex).



Solution

Lab 13

Prob 2



~~Start~~ (V) $A[v] = 0$

pool = $\{ vu, vw, vx \}$

$A[v] + wt(v,u) = 1 \leftarrow$

$A[v] + wt(v,w) = 3$

$A[v] + wt(x,v) = 2$

(u) $A[u] = 1$

pool = $\{ v,w, vx, ux, uw, uy \}$

$A[v] + wt(v,w) = 3$ $\quad A[u] + wt(u,w) = 5$

$A[v] + wt[v,x] = 2 \leftarrow$ $\quad A[u] + wt(u,y) = 3$

$A[u] + wt(u,x) = 4$

(x) $A[x] = 2$

pool : $\{ vw, uw, uy, xy \}$

$A[v] + wt(vw) = 3 \leftarrow$

$A[u] + wt(u,w) = 5$

$A[u] + wt(u,y) = 3$

$A[x] + wt(x,y) = 2+2 = 4$

(w) $A[w] = 3$

(y) pool : $\{uy, xy\}$

$A[u] + wt(u,y) = 3 \leftarrow$

$A[x] + wt(x,y) = 4$

$A[y] = 3$

2. Prove that the "slow" Dijkstra algorithm is also correct. The algorithm is reproduced here:

SLOW DIJSKSTRA

Input: A simple connected undirected weighted graph *G* with nonnegative edge weights, determined by a weight function *wt* (*x*,*y*), and a starting vertex *s* of *G*.

Output: Table A of shortest distances d(s,v) from s to v, for each v in V, so A[v] = d(s,v) for each v
Aux Output: Table B with property that B[v] is a shortest path from s to v.

The Algorithm:

$A[s] \leftarrow 0.$
X ← {s} //Basis step

while X ≠ V **do**
{ POOL ← {(v,w) ∈ E | v ∈ X and w ∉ X} }
//here, we have no control over how much of E is searched, leading to slow running time
(v′,w′) ← search POOL for edge (v,w) for which greedy length A[v] + wt(v,w) is minimal
add w′ to X
A[w′] ← A[v′] + wt(v′,w′)

Recall the Lemma shown in the slides. This will be helpful to use in your proof.

Main Lemma. Suppose G is a weighted graph. Suppose $q : s, \ldots, y, z$ is a true shortest path from s to z in G. Then the path $s, \ldots, y$ is a true shortest path from s to y; the path y, z is a true shortest path from y to z; and $d(s, z) = d(s, y) + \mathrm{wt}(y, z)$.

Proof. Let r and t be the following subpaths of q:

$$r : s, \ldots, y \qquad t : y, z.$$

We claim that $d(s, y)$ is equal to the length of r: If not, then there must be a shorter path r' than r that goes from s to y. But then $r' \cup t$ is a shorter path from s to z than q, which contradicts the fact that q is a true shortest path from s to z. For the same reason, $d(y, z)$ is equal to length of t. In particular, $d(y, z) = \mathrm{wt}(y, z)$.

We have:

$$d(s, z) = \mathrm{length}(q) = \mathrm{length}(r) + \mathrm{length}(t) = d(s, y) + \mathrm{wt}(y, z),$$

as required. □

Reason in the following way: Assume that through the first i stages of the iteration in Slow Dijkstra, the values A[u] for each u in X are correct --- that is, for each u in X, A[u] = d(s,u). We consider stage i+1. Let (v',w') be an edge in G with v' in X and w' not in X with the property that for all edges (v,w) in G with v in X and w not in X, we have A[v'] + wt(v',w') ≤ A[v] + wt(v,w). (This is how the algorithm behaves.) We wish to show A[w'] is correct – that is, we wish to show A[w'] = d(s,w'). Assume that this is not the case. Since A[v'] is correct, it follows that A[w'] ≠ d(s,w') implies A[w'] > d(s,w').

Let q : s, …, y, z, …, w' be a truly shortest path from s to w'. Let L be the length of q. Assume that z is first vertex in V - X encountered on q, and that y is the predecessor of z on q and must therefore belong to X. The subpath s,…, y must be a shortest path from s to y by the Main Lemma and has length A[y] (we are assuming A[u] is correct for all u in X so far).

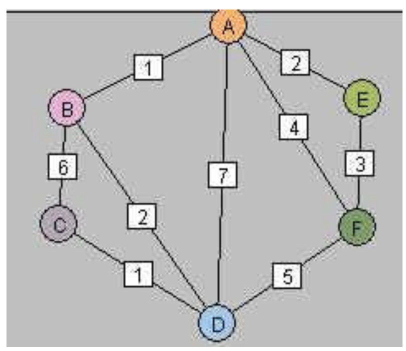
Let q_0 be the path s, …, y, z; we denote its length L_0. Clearly, $A[y] \leq L_0 \leq L$. It suffices to show

$$A[w'] = A[v'] + wt(v',w') \leq L_0$$

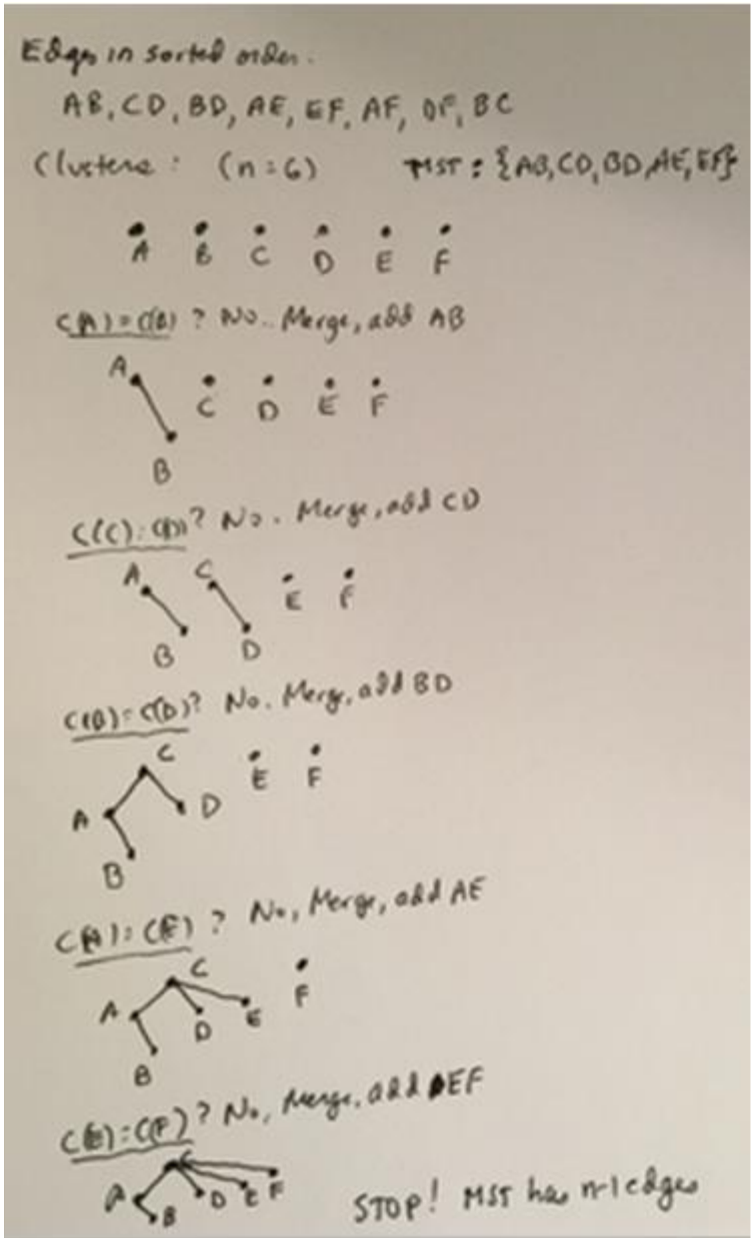
Explain the following:

a. Why must it be true that $L_0 = A[y] + wt(y,z)$? ***Solution:*** By the Lemma, s,…,y is a true shortest path from s to y and y, z is a true shortest path from y to z. But the length of the path y, z is wt(y,z) and the length of the path s,…,y is A[y]. Therefore, $L_0 = A[y] + wt(y,z)$.
b. Why must it be true that $A[v'] + wt(v',w') \leq A[y] + wt(y,z)$? ***Solution.*** The algorithm chose (v',w') so that $A[v'] + wt(v',w') \leq A[u] + wt(u,t)$ whenever (u,t) is an edge with u in X, and t not in X. But y in X and z is not in X. The result follows.
c. Why do parts a and b allow us to conclude that we have reached a contradiction and therefore that A[w'] = d(s,w') after all? ***Solution.*** Parts a and b show that $A[w'] = A[v'] + wt(v',w') \leq L_0$. Since $L_0 \leq L = d(s,w')$, this contradicts the assumption that A[w'] > d(s,w'). Therefore, the selection at stage i + 1 of the edge (v',w') results in the a correct value for A[w'] after all.

3. Carry out the steps of Kruskal's algorithm for the following weighted graph. Manage the evolution of clusters using a tree-based disjoint sets data structure (as shown at the end of the slides). Also, show how the set T of edges evolves during execution (as shown in the slides).
Edges should be sorted initially; to resolve ties, use the alphabetical ordering of vertex names. (You do not need to re-draw the input graph each time to show the evolving MST structure.)



Solution.



4. Create and implement an algorithm IsPrime(n) that accepts positive integer inputs and outputs true if n is prime, false otherwise. What is the asymptotic running time of your algorithm?

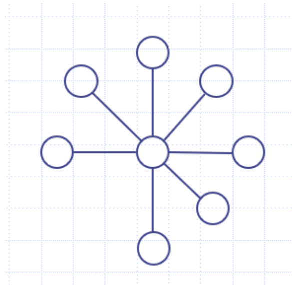
```java
public class IsPrime {
    //Precondition: n is a positive integer
    public static boolean isPrime(int n) {
        if(n== 1) return false;
        for(int i = 2; i * i <= n ; ++i) {
            if(n % i == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        for(int i = 2; i <= 500; ++i) {
            if(isPrime(i)) System.out.println(i + " is prime");
            else System.out.println(i + " is composite");
        }
    }
}
```

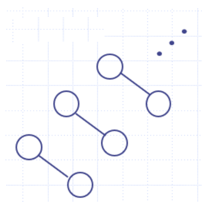
5. Suppose G = (V, E) is an undirected (unweighted) simple graph. A subset U of V is called a *base* for G is every edge in G has at least one endpoint in U. Do the following:
   a. Given G = (V,E) is it true that V itself is a base for G? Explain.
      Solution. Yes – by definition of E, every e in E has an endpoint in V.

   b. Is there a graph G having a base that is the empty set? If so, give an example.
      Solution. Yes – any graph with one or more vertices and no edges is an example.

   c. Give an example of a graph having 8 vertices and having a base of size 1.
      Solution.



   d. Give an example of a graph G having 2n vertices with the property that every base for G has size at least n.



6. Devise an algorithm to solve the Smallest Base Decision Problem: Given G = (V,E) and a nonnegative integer k, is there a base U for G having size ≤ k? What is the running time of your algorithm?

```
Input: A simple undirected graph G = (V,E)
Output: True if a  smallest base for G has size ≤ k, else false
P ← all subsets of V
minU ← V
min ← |V|
for U in P do
   good  ← true
   while good and (x,y) in E
        if  x not in U and y not in U then
           good ← false
   if good then
     if |U| < min then
        minU ←  U
        min  ←  |U|
return minU ≤ k
```