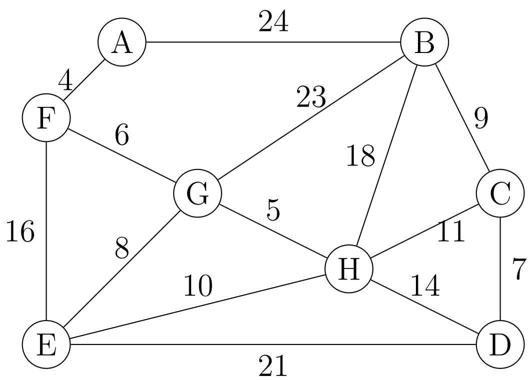
**cmput 204** **seminar 7** **fall 2015**

1. Continue the trace of Prim's MST from node D. At each step, show the parent and cost arrays, and give the edge added to the MST.




| parent | cost | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ABCDEFGH | A | B | C | D | E | F | G | H | pick |
| ---D---- | - | - | - | 0 | - | - | - | | D |
| --DDD--D | - | - | 7 | 0 | 21 | - | - | 14 | C |

2. Does Kruskal's MST algorithm work correctly if edges can have negative weights? Explain briefly.

3. For each edge in the above graph, multiply its weight by $-1$. Now trace Kruskal's algorithm. Give the weight of the final MST.

4. (i) Let $G$ be a connected acyclic graph with $n \geq 2$ nodes. Let $P = (v_1, \ldots, v_t)$ be a longest path of $G$. Prove that $v_2$ is the only node in $G$ adjacent to $v_1$, that $G^- = G - v_1$ is connected, and that $G^-$ is acyclic.

   (ii) Using (i), prove by induction on $n$ that $G$ has n-1 edges.

   (iii) Using (ii), prove that an acyclic graph with $n$ nodes and $c$ components has $n - c$ edges.

5. ```
   def find(v,P):
     while P[v] != v:
       v = P[v]
     return v

   def u(v,w,P): # union
     rv,rw = find(v),find(w)
     P[rv] = rw

   def uBR(v,w,P,R): # union by rank
     rv,rw = find(v),find(w)
     if   R[rv] < R[rw]:
       P[rv] = rw
     elif R[rv] > R[rw]:
       P[rw] = rv
     else:
       P[rv] = rw
       R[rw] += 1
   ```

   An algorithm uses the above union/find algorithms. Here are the current parent values:

| node | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| P | A | B | C | D | E | F | G |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

   (i) Give more meaningful names for `rv,rw,P,R`.

   (ii) Draw the current union/find forest.

   (iii) Show P, R, and the forest after the following operations: `u(A,B) u(B,C) u(A,D) u(E,F) u(B,F)`.

   (iv) Repeat (iii) using `uBR( )` instead of `u( )`.
   ```

1.

| ABCDEFGH | A | B | C | D | E | F | G | H | pick |
|---|---|---|---|---|---|---|---|---|---|
| ---D---- | - | - | - | 0 | - | - | - | - | D |
| --DDD--D | - | - | 7 | 0 | 21 | - | - | 14 | C |
| -CDDD--C | - | 9 | 7 | 0 | 21 | - | - | 11 | B |
| BCDDD-BC | 24 | 9 | 7 | 0 | 21 | - | 23 | 11 | H |
| BCDDH-HC | 24 | 9 | 7 | 0 | 10 | - | 5 | 11 | G |
| BCDDGGHC | 24 | 9 | 7 | 0 | 8 | 6 | 5 | 11 | F |
| FCDDGGHC | 4 | 9 | 7 | 0 | 8 | 6 | 5 | 11 | A |
| FCDDGGHC | 4 | 9 | 7 | 0 | 8 | 6 | 5 | 11 | E |

2. yes. we just want a tree whose sum of edges is minimum. at each point, we pick the available edge with minimum weight. there is no problem if edge weights are negative. if you take a problem with negative weight edges, and subtract the most negative weight from all edges, you will get a graph with non-negative edge weights, and the set of edges picked, and the order in which they are picked, will be the same.

3. AB -24 BG -23 ED -21 BH -18
FE -16 HD -14 HC -11

4. (i) $v_1$ is not adjacent to any other vertex in $P$, else there is a cycle. $v_1$ is not adjacent to any vertex not in $P$, else $P$ can be extended into a longer path.

If there is a path in $G$ between two nodes, not including $v_1$, that uses $v_1$, then the path must at some point go from $v_2$ to $v_1$ and immediately back to $v_2$. So removing $v_1$ from the path leaves a path between the two nodes. So removing $v_1$ from $G$ leaves a connected graph.

(ii) Assume that this holds for all $n$ up to a fixed integer $t$. Consider a graph $G$ with $t+1$ nodes. Find the end of a longest path, call this node $v_1$. Removing $v_1$ leaves a graph with number of nodes and number of edges both decreased by 1. So the number of edges in $G$ is the number of edges in $G' = G - v_1$ plus 1. By our inductive hypothesis, the number of edges in $G'$ is one less than the the number of nodes in $G'$, so $(t+1-1)-1$. So, the number of edges in $G$ is one more than this, so $(t+1-1)-1+1 = (t+1-1)$. So $G$ has $t+1-1$ edges, i.e. one less than the number of nodes. So we are done.

(iii) Let $n_1 \ldots n_c$ be the number of nodes in the $c$ components of $G$. Then the number of edges in $G$ is $(n_1 - 1) + \ldots n_c - 1 = (n_1 + \ldots n_c) - (1 + \ldots + 1) = n - c$.

5. (i) rootv, rootw, parent, rank

(ii) each node is in a tree with one node

(iii)

| node | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| P | B | C | D | F | F | F | G |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(iv)

| node | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| P | B | F | B | B | F | F | G |
| R | 0 | 1 | 0 | 0 | 0 | 2 | 0 |