How many key comparisons does this algorithm do for finding the min and the max of n=2k data items:

1. for (i=0; i < n; i+=2)

    if (A[i] > A[i+1]) swap(A[i], A[i+1])

Then use a linear scan (like in MaxSort) to

2. Find the min of

    A[0], A[2], A[4], … , A[n-2]

3. Find the max of

    A[1], A[3], A[5], … A[n-1]

Old final Exam Question

Answer true or false and justify your answer:

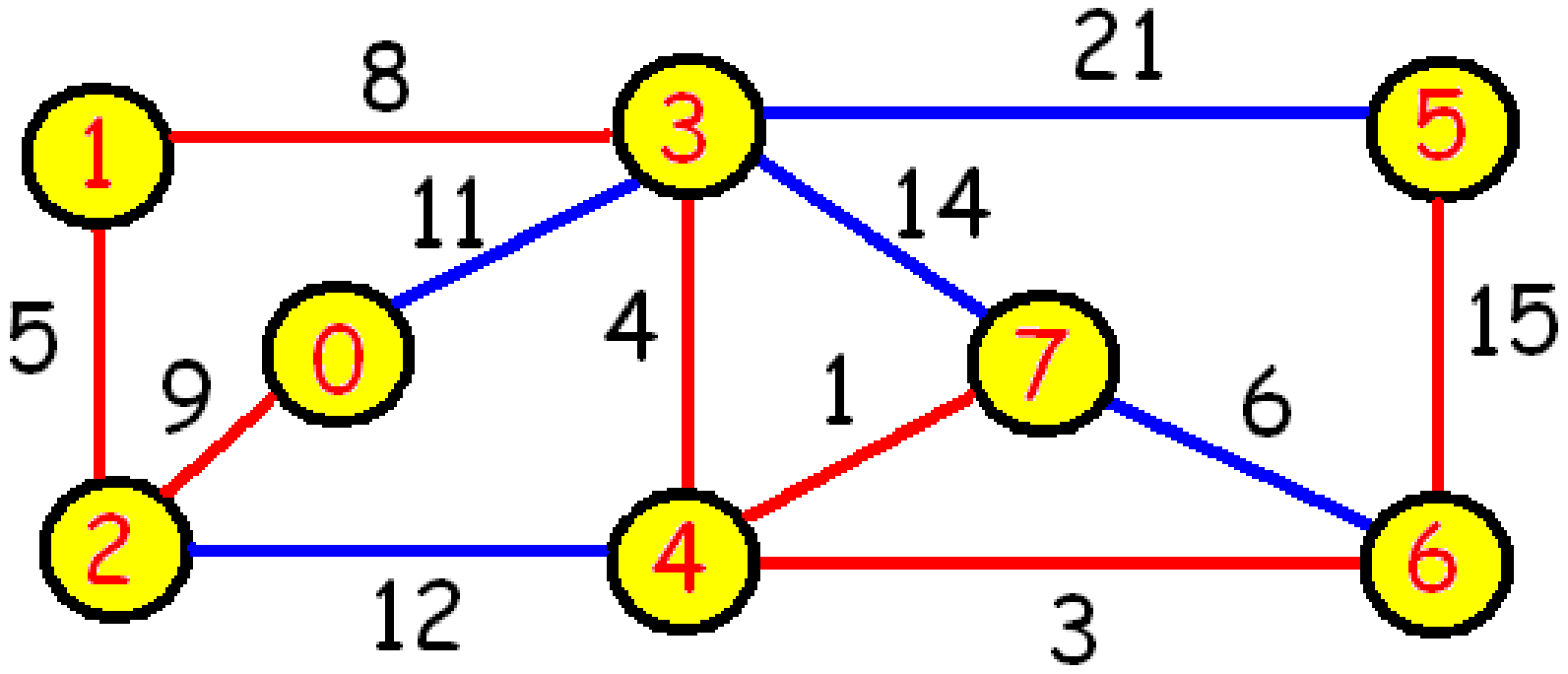
Since it takes at least n-1 key comparisons to find the min of n data items and it takes at least n-1 key comparisons to find the max of n data items, it takes at least 2n-2 key comparisons to find both the min and the max.

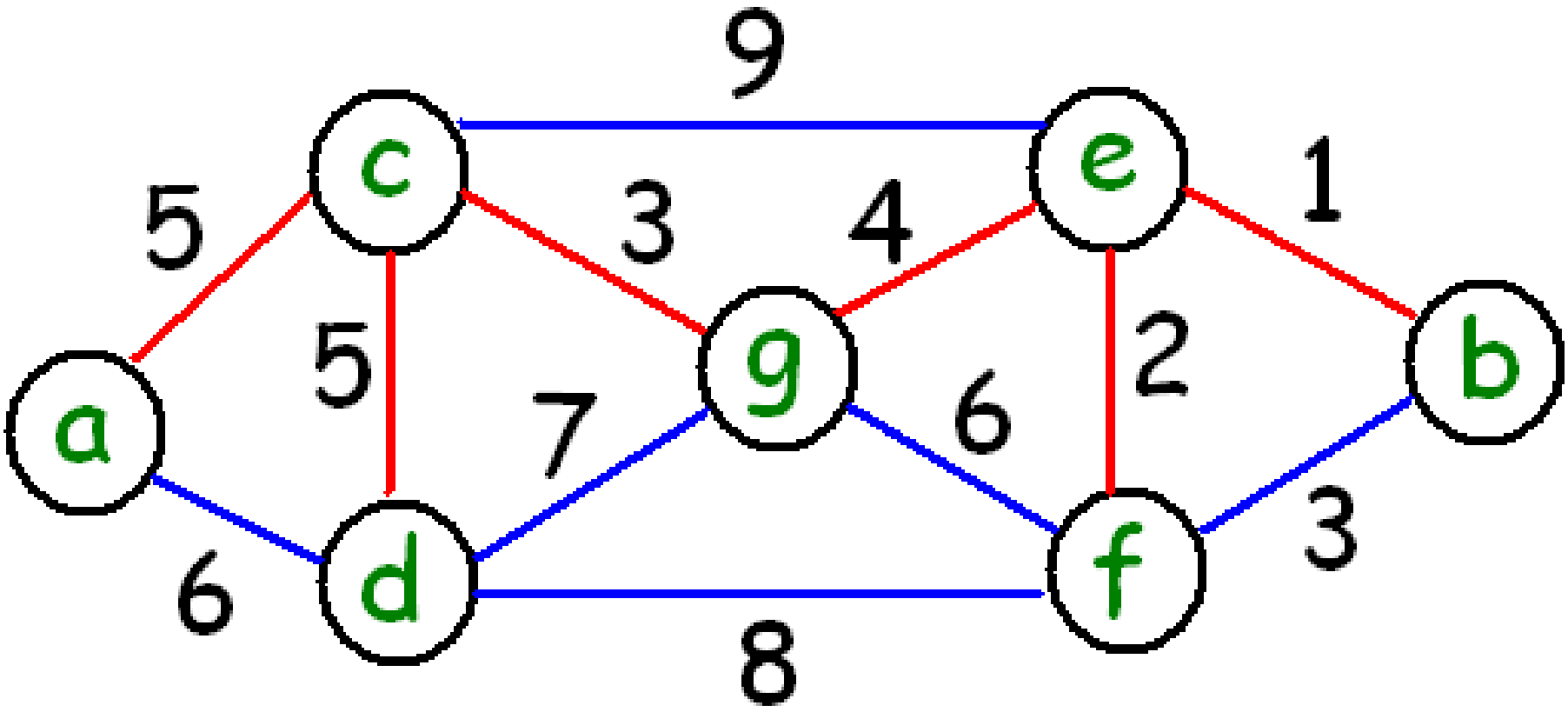
Final exam tutorial time:
Thursday Dec. 5: from 11am until all questions are answered. ECS 116.

.

Written Assignment #5: due Wed. Dec. 4, beginning of class.

For those of you worried about failing the assignments: you should write the final exam.


8
21
1
3
5
11
14
5
4
15
0
7
9
1
6
2
4
6
12
3

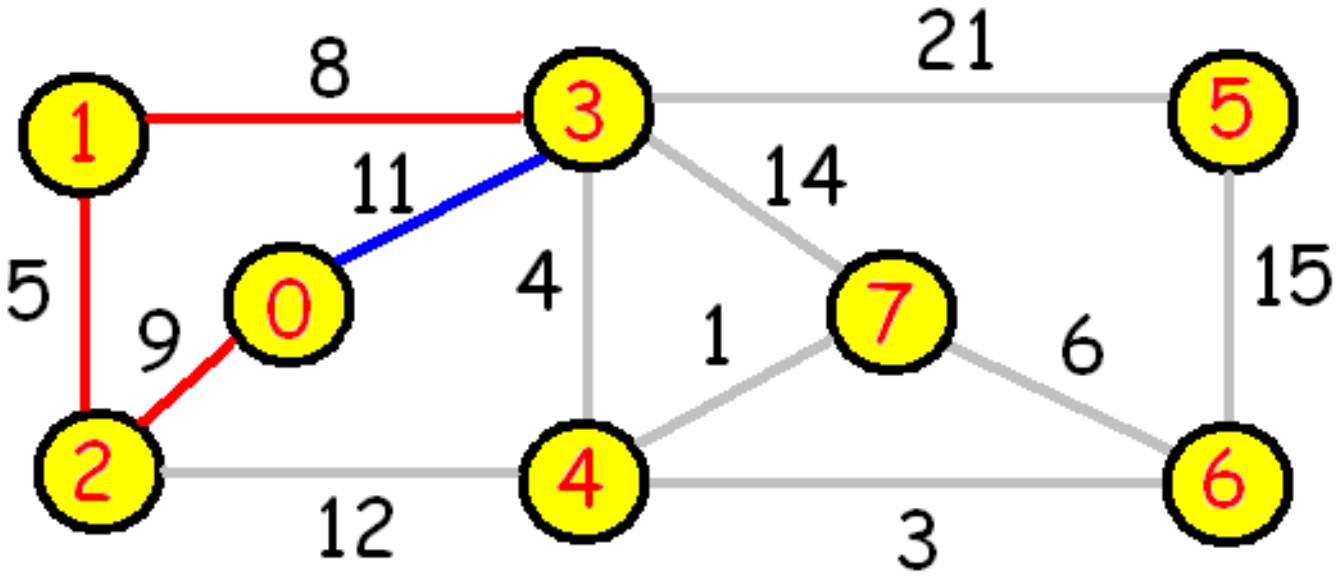

9
c
e
5
3
4
1
5
2
a
g
b
7
6
6
d
f
3
8

Proof of Correctness for the Kruskal and Dijkstra/Prim MST algorithms.

Why do these algorithms correctly compute a minimum weight spanning tree?

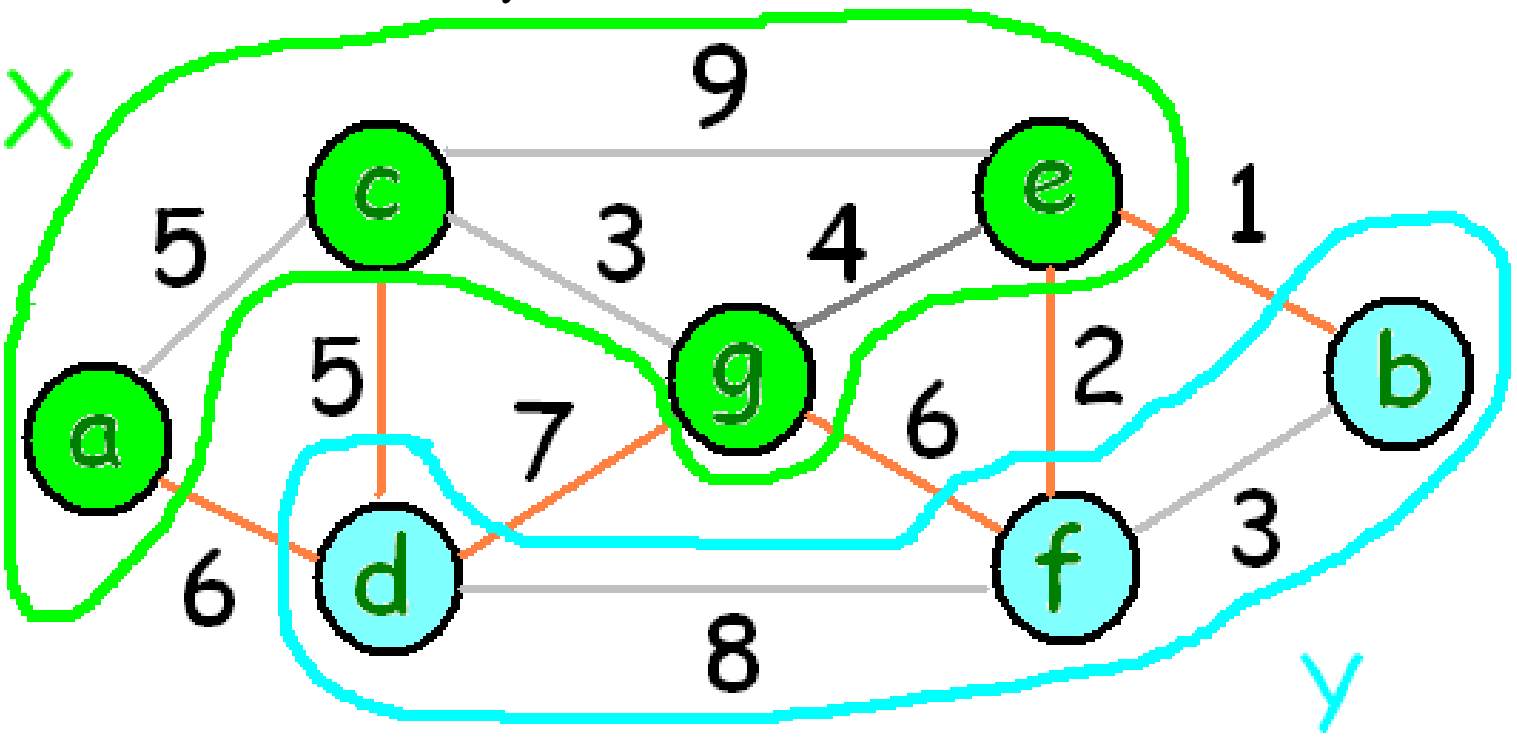
What's a Cycle?

A cycle of a graph is an alternating sequence of vertices and edges of the form $v_0, (v_0, v_1), v_1, (v_1, v_2), v_2, (v_2, v_3), \ldots, v_{k-1}, (v_{k-1}, v_k), v_k$ where except for $v_0 = v_k$ the vertices are distinct



What's a Cut?

Let (X,Y) be a partition of the nodes of a graph. The *cut* induced by that partition is the set of all edges $\{x,y\}$ with x in X and y in Y.



The Red/Blue Rules for MST

Author- Frank Ruskey.

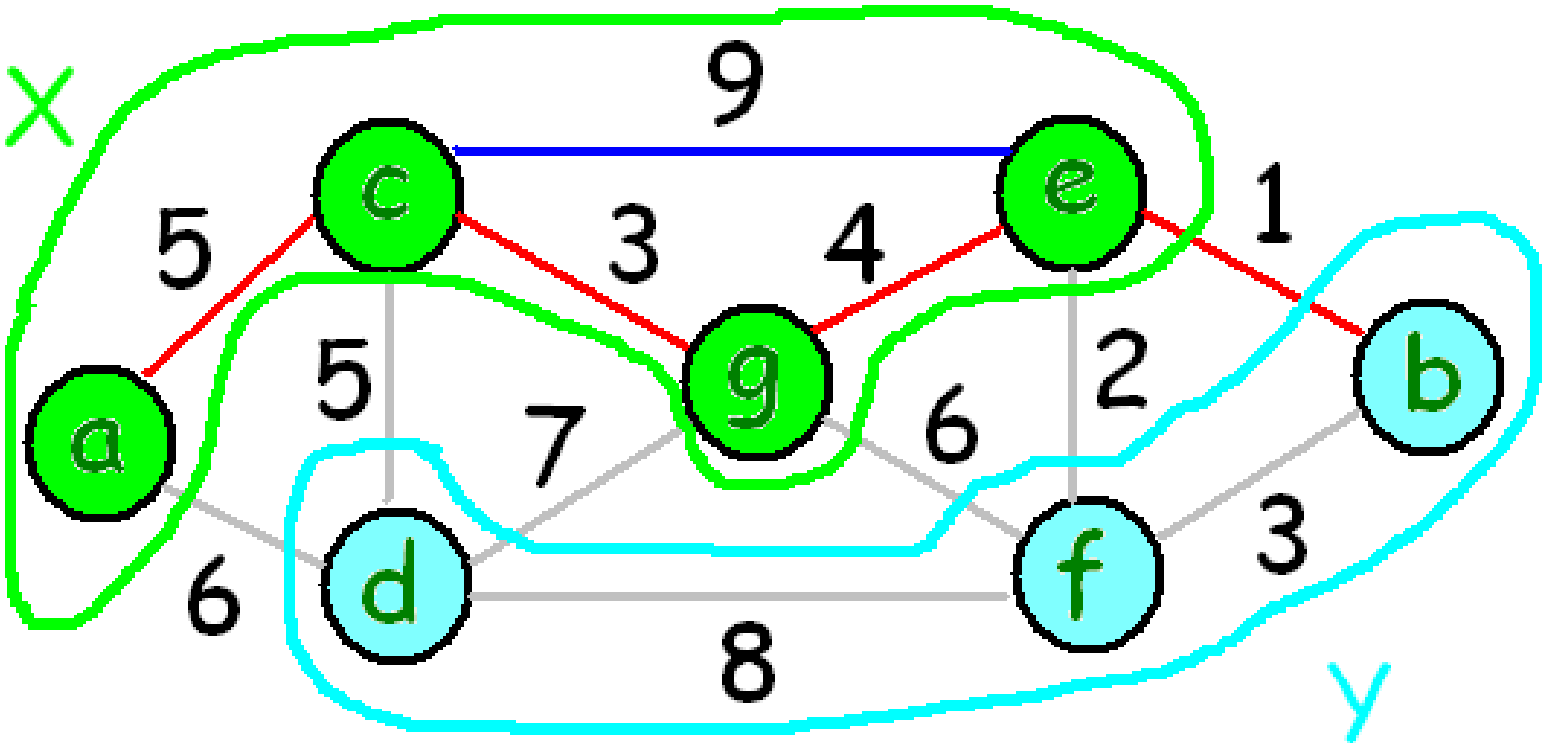
Color Invariant:
There is a MST containing all the red edges and none of the blue edges.

Red Rule:
Let C be a cut without red edges. Color red the smallest uncolored edge of C.

Blue Rule:
Let C be a cycle without blue edges. Color blue the largest uncolored edge of C.

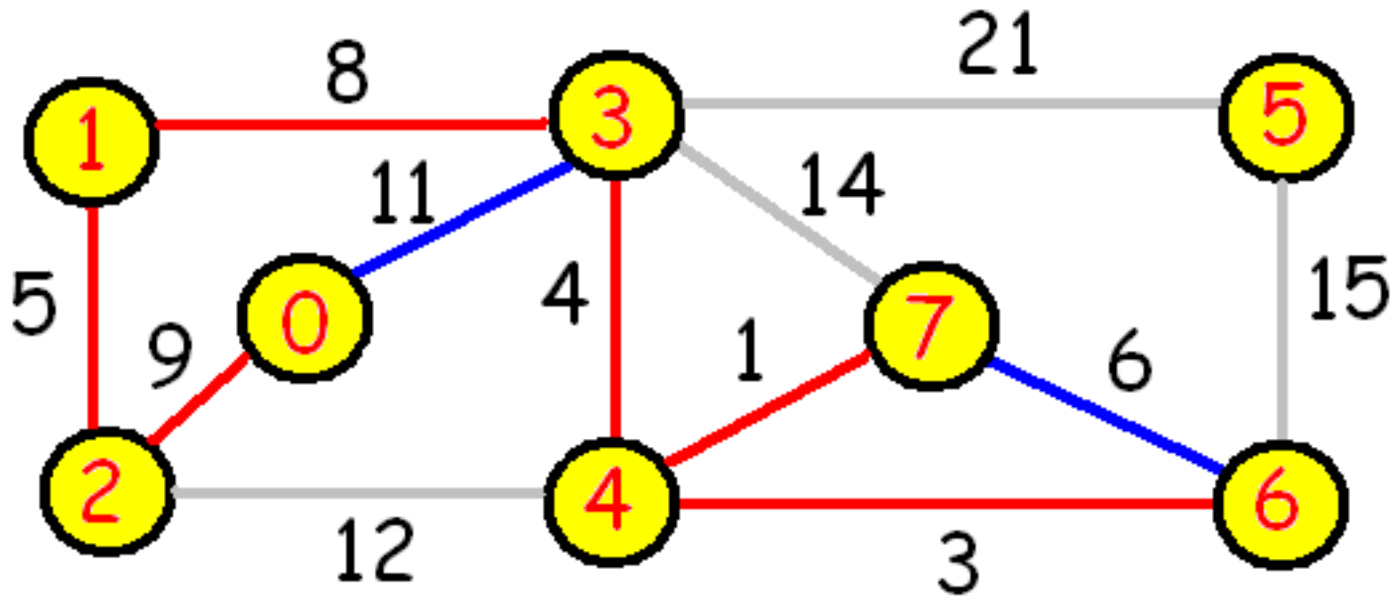
Red Rule: Dijkstra/Prim

Let C be a cut without red edges. Color red the smallest uncolored edge of C.



Blue Rule: Kruskal's

Let C be a cycle without blue edges. Color blue the largest uncolored edge of C.



Generic Greedy Algorithm:

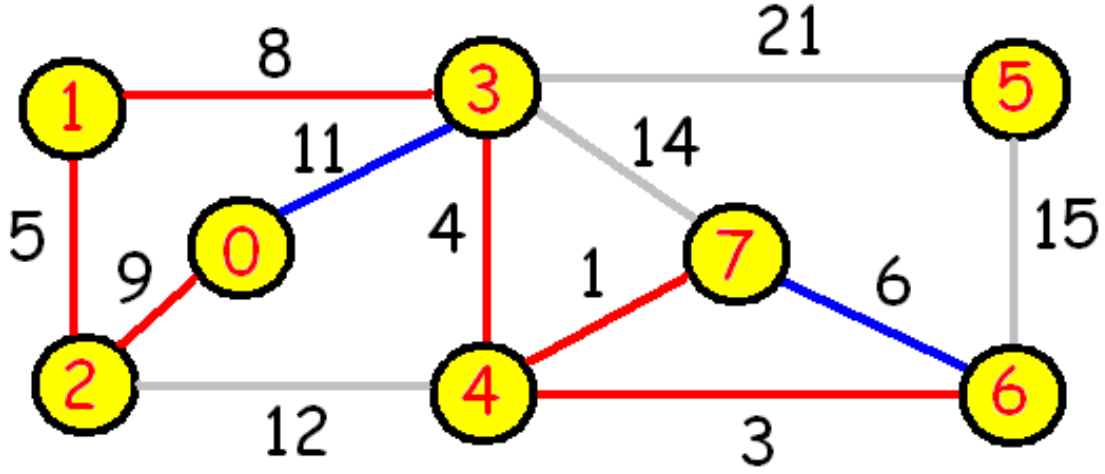
Nondeterministically apply the Red and Blue rules until all edges are colored. The red edges form a MST. (Can stop when there is n-1 red edges)

Kruskal's Algorithm:

T initially has no edges.

Consider edges in order according to their weights.

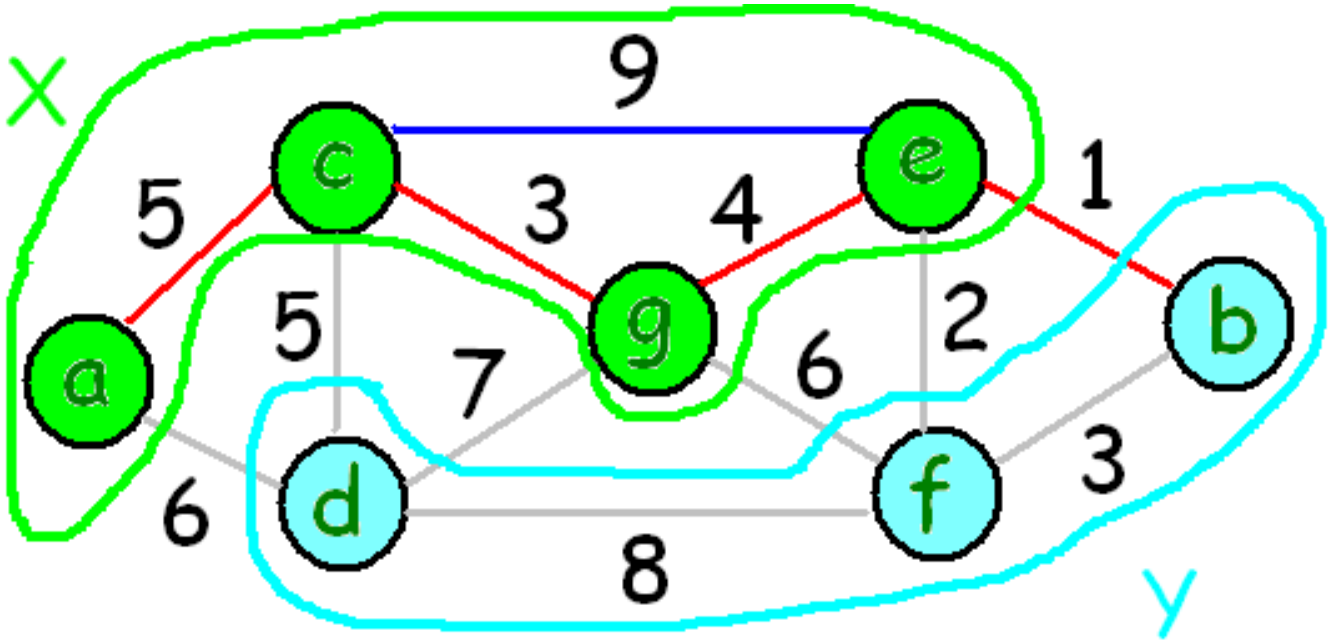
For edge (u, v), apply the Blue rule if $T+(u,v)$ creates a cycle, otherwise apply the Red rule.



Dijkstra/Prim Algorithm:

T initially contains one vertex.

Successively apply the Red rule to all edges of the form (u, v), with u in T and v not in T.



The Red/Blue Rules for MST

Author- Frank Ruskey.

Color Invariant:
There is a MST containing all the red edges and none of the blue edges.

Red Rule:
Let C be a cut without red edges. Color red the smallest uncolored edge of C.

Blue Rule:
Let C be a cycle without blue edges. Color blue the largest uncolored edge of C.

Theorem: The colour invariant is maintained if we apply the blue rule or the red rule.

Proof (by contradiction).

Suppose not. Consider the first application of a blue or red rule which violates the colour invariant. Let R and B denote the red and blue edges already coloured so far without violating the colour invariant. Let T denote a MST of G which contains all of the edges in R but none of the edges in B.

Case 1: the red rule violates the colour invariant.

In this case, we have an edge e=(u, v) which is the minimum weight edge in a cutset C having no edges from R . Violating the colour invariant means that there is no MST of the graph containing all of the edges in the set R ∪ {e} but none of the edges in B.

Let (X, Y) be the partition of the vertices representing the cut C, and without loss of generality, u is in X and v is in Y. Consider the unique cycle in T + e. Because u is in X and v is in Y, the path in T which connects u and v must contain at least one edge e' in the cut (X, Y). This edge e' cannot be in R because by the red rule, C has no red edges. Also, e' is not in B because it is an edge of T.

Red Rule:
Let C be a cut without red edges.
Color red the smallest uncolored edge of C.

Color Invariant:
There is a MST containing all the red edges and none of the blue edges.

The tree T' = T + e - e' has wt(T') = wt(T) + wt(e) - wt(e'). This is less than or equal to wt(T) because wt(e) is less than or equal to wt(e)' (both edges are in C and e is a minimum weight edge of C). If wt(e) < wt(e'), this contradicts the initial assumption that T was a MST. Otherwise, T' is a MST which contains all of R union {e} and none of B, so the red rule does not violate the colour invariant.

Blue Rule:
Let C be a cycle without blue edges. Color blue the largest uncolored edge of C.

Color Invariant:
There is a MST containing all the red edges and none of the blue edges.

Case 2: the blue rule violates the colour invariant.

In this case, we have an edge e=(u, v) which is the maximum weight edge in a cycle C with no edges from B. Violating the colour invariant means that there is no MST of the graph containing all of the edges in the set R but none of the edges in B ∪ {e}. This implies that T must contain e.

Delete e from T to create a 2-component forest. Let X be the component containing u and Y be the component containing v. Because the cycle C has a path from u to v, it must contain some edge e'=(x, y) which has x in X and y in Y. This e' is not in R because it is not in T. Further, it is not in B because by the blue rule, C has no edges in B.

The tree T' = T + e' - e has wt(T') = wt(T) + wt(e') - wt(e). This is less than or equal to wt(T) because wt(e) is greater than or equal to wt(e)' (both edges are on cycle C and e is a maximum weight edge of this cycle). If wt(e) > wt(e'), this contradicts the initial assumption that T was a MST. Otherwise, T' is a MST which contains all of R and none of B ∪ {e}, so the blue rule does not violate the colour invariant.

Boruvka's algorithm:

Repeat

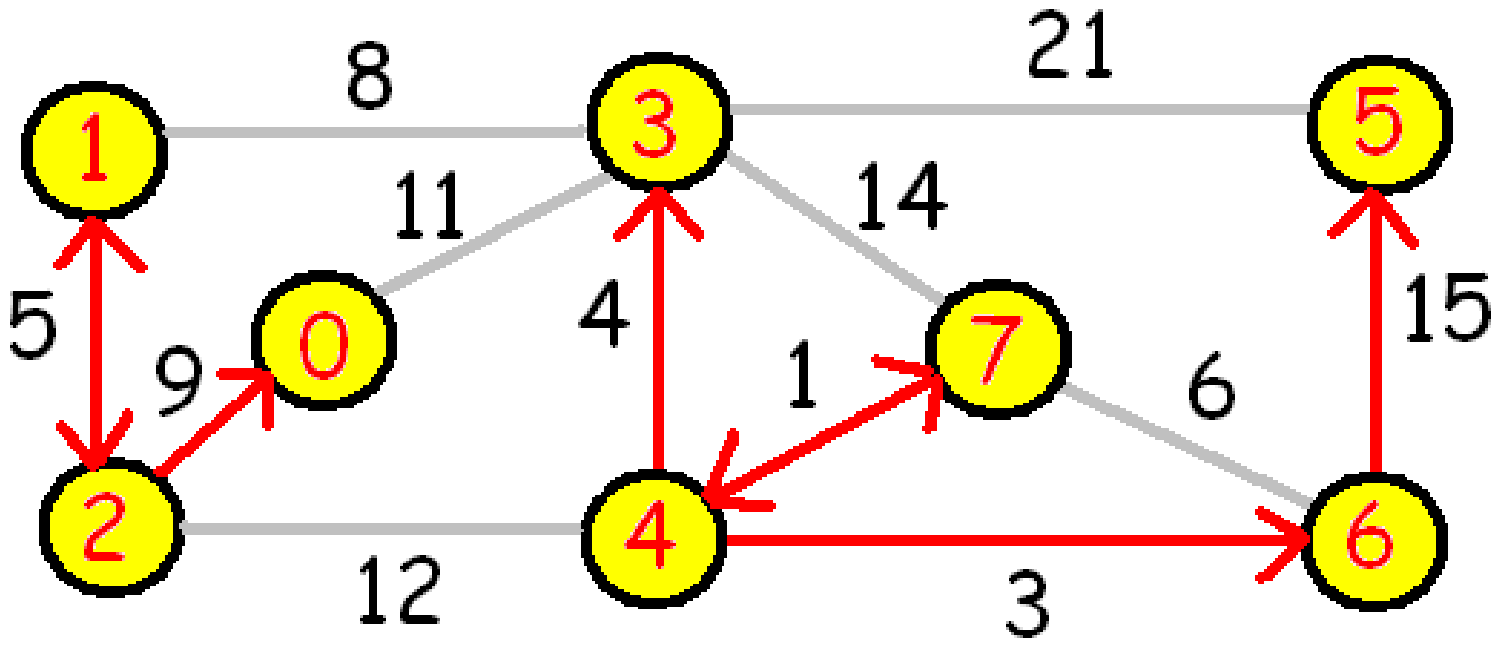
For each vertex v do:

Choose the min weight edge incident to the v and colour it red.

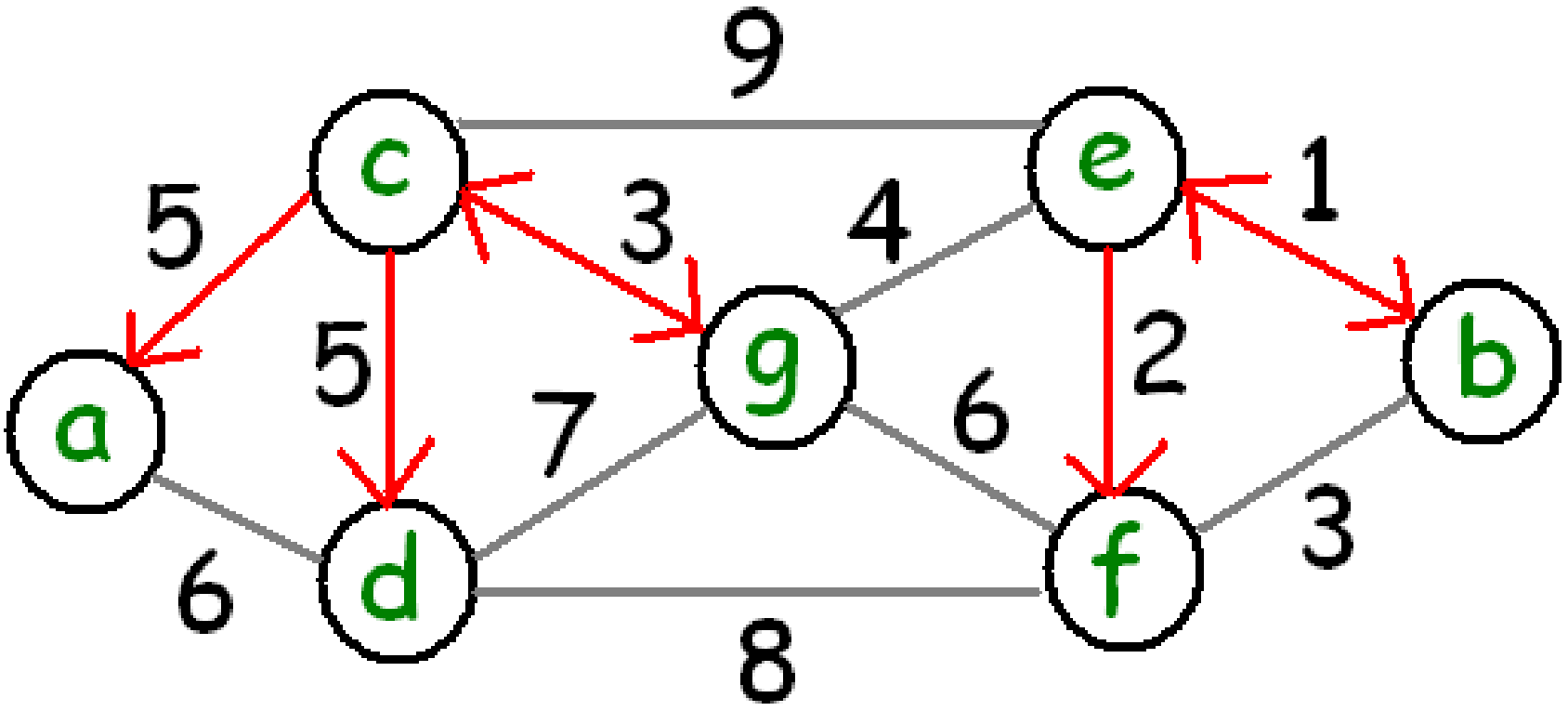
Contract edges in the MST- if you get multiple edges you can keep the one with min weight.

Until one vertex remains

First iteration: edge for each vertex is oriented so that it enters the vertex.



First iteration: edge for each vertex is oriented so that it enters the vertex.



If edges have the same weight, break ties by always selecting the one which has minumum label (u, v).

For example:

Both (0,4) and (0, 7) have weight 3:

$(0, 4) < (0, 7)$

Correctness:

At one phase, the edges selected never form a cycle.

The number of edges chosen at one phase is at least n/2.

Therefore, the number of components is at least halved at every iteration.

So the number of iterations is $O(\log_2(n))$.

Time: $O(m \log_2 n)$ [adjacency lists]