CSCI-163 HW#5 Jesse Mayer

9.2-1:b.

Tree edges	Sorted List of edges
	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
de 1	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
cd 2	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
ef 2	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
ab 3	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
be 3	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
gh 3	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
ij 3	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
cg 4	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
ei 4	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
il 5	de cd ef ab be gh ij ad cg ei ac dh fj il bf gk hi hk kl jl 1 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 8 9
gk 6	

9.2-3: The number of edges for a minimum spanning forest with |V| vertices and |C| connected components is |V|-|C|. Kruskal will never get to the |V|-1 tree edge unless the graph is connected, so in the while loop we replace encounter <|V|-1 with while k<|E| so it stops after exhausting the sorted list of edges.

9.2-10:	V=5, E=7	V=9, E=14
Prim	6.2×10^{-2} ms	7.7×10^{-2} ms
Kruskal	4.7×10^{-2} ms	7.9×10^{-2} ms

9.3-1: a. In order to solve the single-source shortest-paths problem for directed weighted graphs, we just need to take into account edge directions in processing adjacent vertices.

b. In order to solve the single-pair shortest path problem we simply start the algorithm at one of the given vertices and stop the program once the other vertex has been added to the tree.

c. In order to solve the single-destination shortest-paths problem, for an undirected graph we just make the destination the source and then reverse all paths obtained in the solution. However, if the graph is directed, then we must start by reversing all its edges, ~~and~~ then solve for the new digraph with the destination as the source and at the end reverse the direction of all the paths in the solution.

d. In order to solve the single-source shortest-paths problem in a graph with nonnegative numbers assigned to its vertices, we start by creating a new graph where every vertex v is replaced by v' and v" that are connected by an edge equal to the weight of vertex v. Next we make it so all the edges that entered and left v are now entering v' and leaving v". Then we assign a weight of 0 to each of the edges from the original graph.

9.3-2: b.

Tree Vertices	Remaining Vertices
a(-,0)	b(a,3) c(a,5) d(a,4)
b(a,3)	c(a,5) d(a,4) e(b,3+3) f(b,3+6)
d(a,4)	c(a,5) e(d,4+1) f(a,9) h(d,4+5)
c(a,5)	e(d,5) f(a,9) h(d,9) g(c,5+4)
e(d,5)	f(e,5+2) h(d,9) g(c,9) i(e,5+4)
f(e,7)	h(d,9) g(c,9) i(e,9) j(f,7+5)
h(d,9)	g(c,9) i(e,9) j(f,12) k(h,9+7)
g(c,9)	i(e,9) j(f,12) k(g,9+6)
i(e,9)	j(f,12) k(g,15) l(i,9+5)
j(f,12)	k(g,15) l(i,14)
l(i,14)	k(g,15)
k(g,15)	

Shortest Paths

a to b: a-b of length 3
a to d: a-d of length 4
a to c: a-c of length 5
a to e: a-d-e of length 5
a to f: a-d-e-f of length 7
a to h: a-d-h of length 9
a to g: a-c-g of length 9
a to i: a-d-e-i of length 9
a to j: a-d-e-f-j of length 12
a to l: a-d-e-i-l of length 14
a to k: a-c-g-k of length 15

9.3-7: Algorithm ShortestPathsDag(G,s)

For every vertex v do
    $d_v \leftarrow \infty$, $p_v \leftarrow \emptyset$, $d_s \leftarrow 0$
For every vertex u taken in topological order do
For every vertex u adjacent to v do
    if $d_v + w(v,u) < d_u$
        $d_u \leftarrow d_v + w(v,u)$, $P_u \leftarrow v$

9.3-8: The minimum-sum descent problem can be solved by Dijkstra's algorithm by treating the integers as vertices ~~and~~ with the adjacent integers as vertices that share an edge, the value of each integer becomes the corresponding edge's weight, and integers that are not adjacent have an edge weight of ∞.

9.3-10: Algorithm Subway(G,s)

```
dist[s]=0
for each vertex v in G
    if v≠s
        dist[v]=∞
        add v to Q
while Q is not empty
    v=vertex in Q with minimum dist[v]
    remove v from Q
    for each vertex u that shares edge with v
        alt=dist[v]+w(v,u)
        if alt<dist[u]
            dist[u]=alt
return dist
```

11.1-2: $M(n)=2M(n-1)+1 \quad M(1)=1$

$M(n)=2[2M(n-2)+1]+1=2^2M(n-2)+2+1$

$M(n)=2^2[2M(n-3)+1]+2+1=2^3M(n-3)+2^2+2+1$

$M(n)=2^iM(n-i)+2^{i-1}+2^{i-2}+\cdots+2+1=2^iM(n-i)+2^i-1$

$M(n)=2^{n-1}M(n-(n-1))+2^{n-1}-1$

$M(n)=2^{n-1}M(1)+2^{n-1}-1$

$M(n)=2^{n-1}+2^{n-1}-1$

$M(n)=2^{n}-1$

The classic algorithm makes $2^{n}-1$ moves so this is correct.

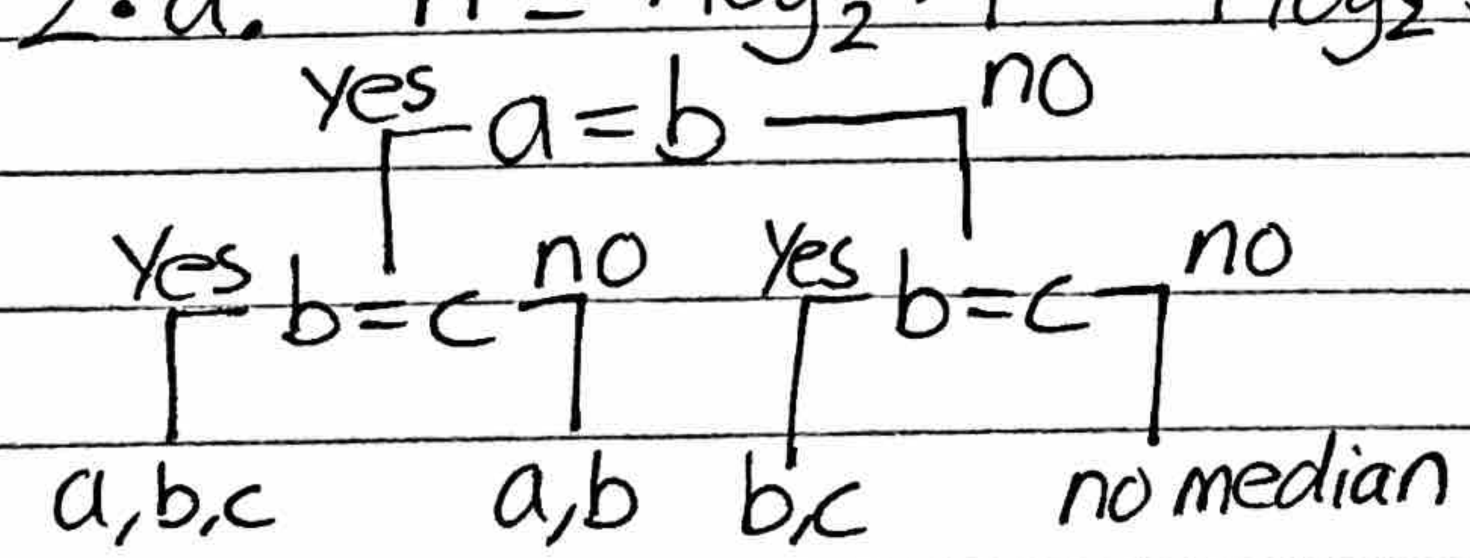
11.1-3: a. To find the largest element in an array we need to check all n elements, so the lower bound is $\Omega(n)$. Because no algorithm can do better this lower bound is tight.

b. For a graph with n vertices there are $n(n-1)/2$ possible edges that must be checked. Therefore the lower bound is $\Omega(n^2)$ and is tight since the algorithm must check all the elements of the upper triangular matrix until either a 0 is found or all elements have been checked.

11.1-4: Yes, because with each weighing the set of coins the fake coin resides in is halved until we get to the fake coin. Variable reduction algorithms could be used by not weighing all the potential fake coins at once but this will also take $\lceil \log_2 n \rceil$ weighings in the worst case.

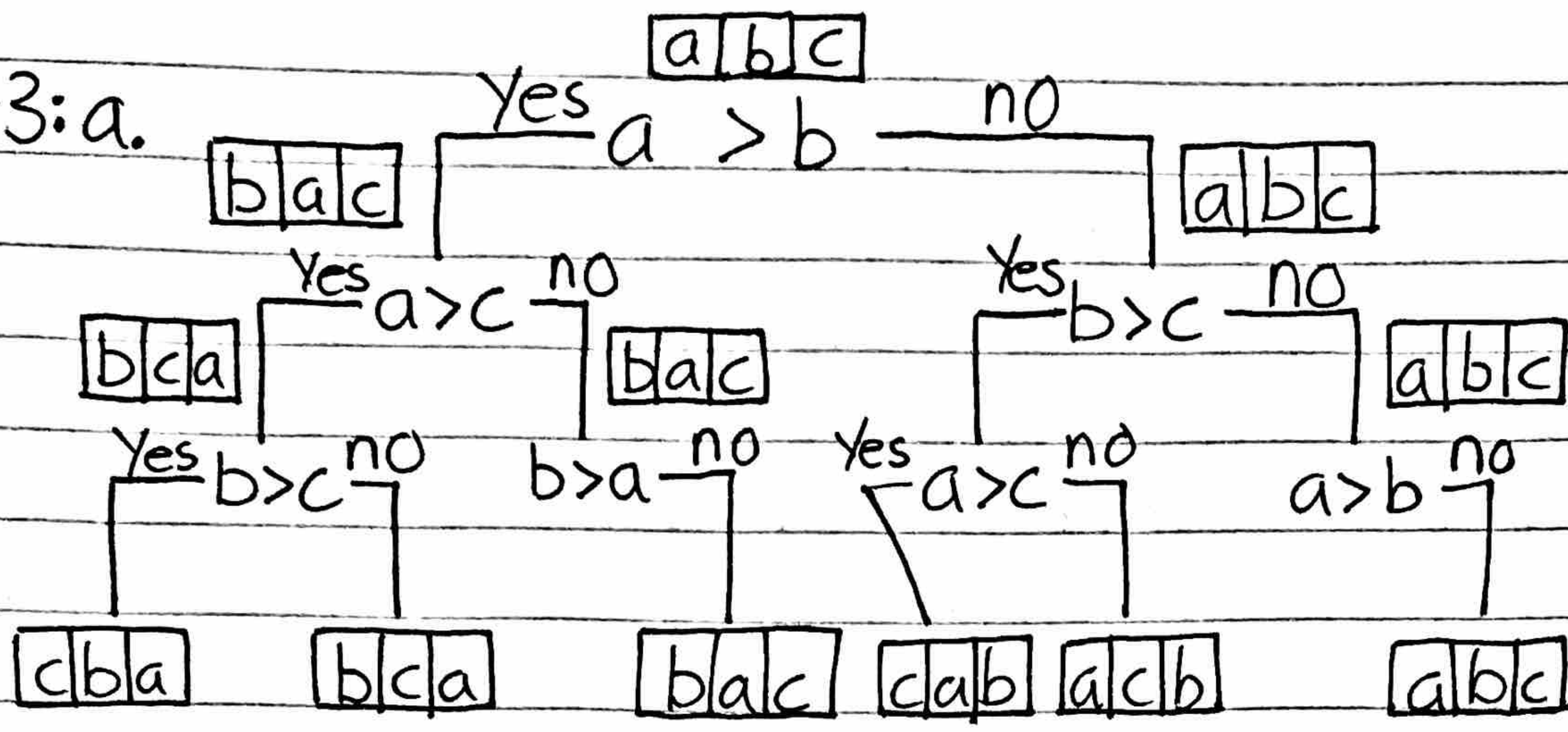
11.2-2: a. $h \geq \lceil \log_2 l \rceil = \lceil \log_2 5 \rceil = \lceil 2.32 \rceil = 3$

b.



c. This algorithm makes 3 comparisons $a=b$, $a=c$, and $b=c$ which agrees with the information-theoretic lower bound, so there does not exist a better algorithm.

11.2-3: a.



$C_{worst} = C_{avg} = \lceil \log_2 6 \rceil = \lceil 2.58 \rceil = 3$ comparisons