Intro to Algorithms: Homework #8

Due on April 1, 2021

Prof. Zaki

Jared Gridley

Lab Results:

```
Results:

Without Path Compression:

Graph100
Total Weight: 263492
Max rank: 14
Max Height: 15
Time: 11.464974880218506

Graph1000
Total Weight: 786722
Max rank: 14
Max Height: 15
Time: 3155.329099178314

With Path Compression:

Graph100
Total Weight: 263492
Max rank: 14
Max Height: 4
Time: 0.15575194358825684

Graph1000
Total Weight: 786722
Max rank: 14
Max Height: 5
Time: 5.8285439014434814

Graph10000
Total Weight: 7448724
Max rank: 17
Max Height: 6
Time: 67.20316171646118
```

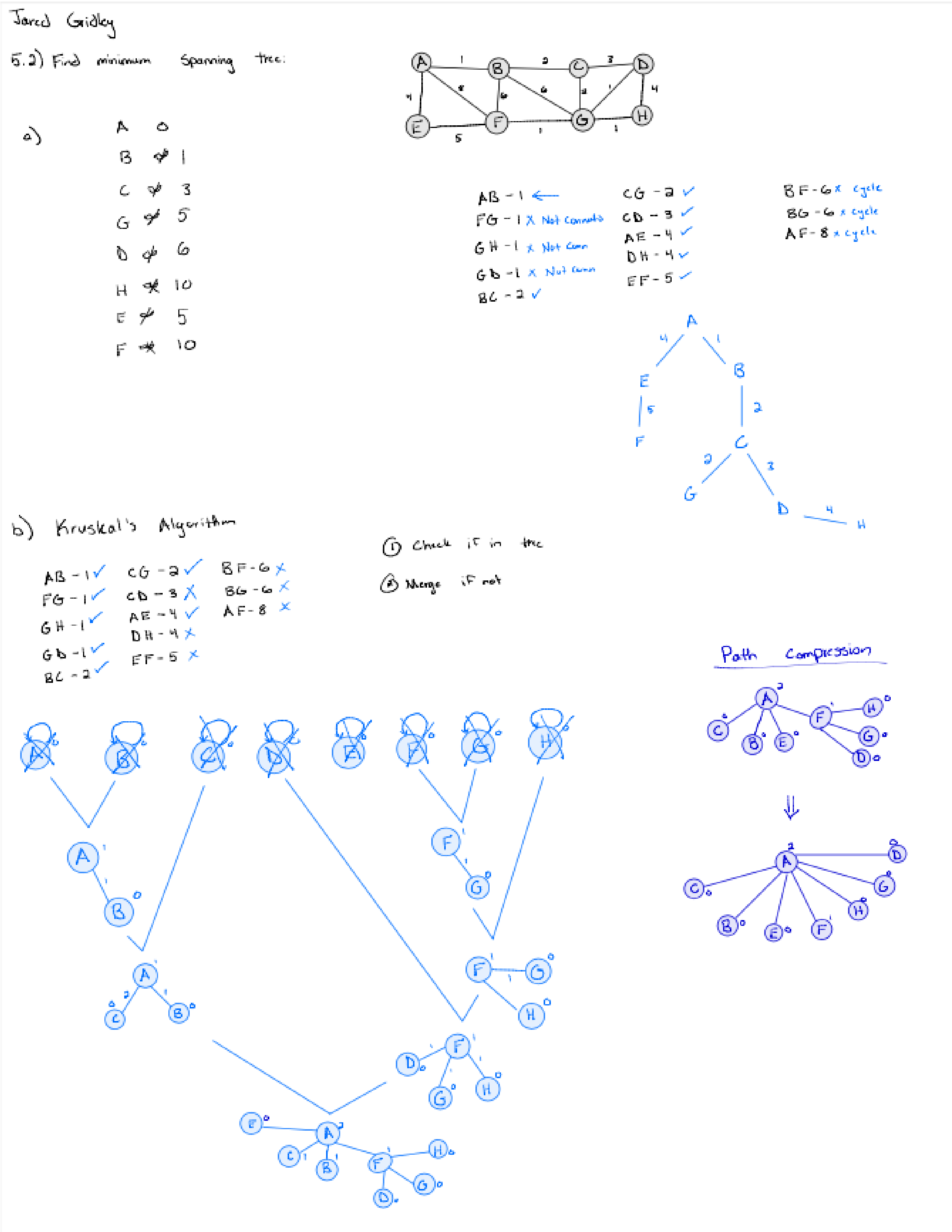
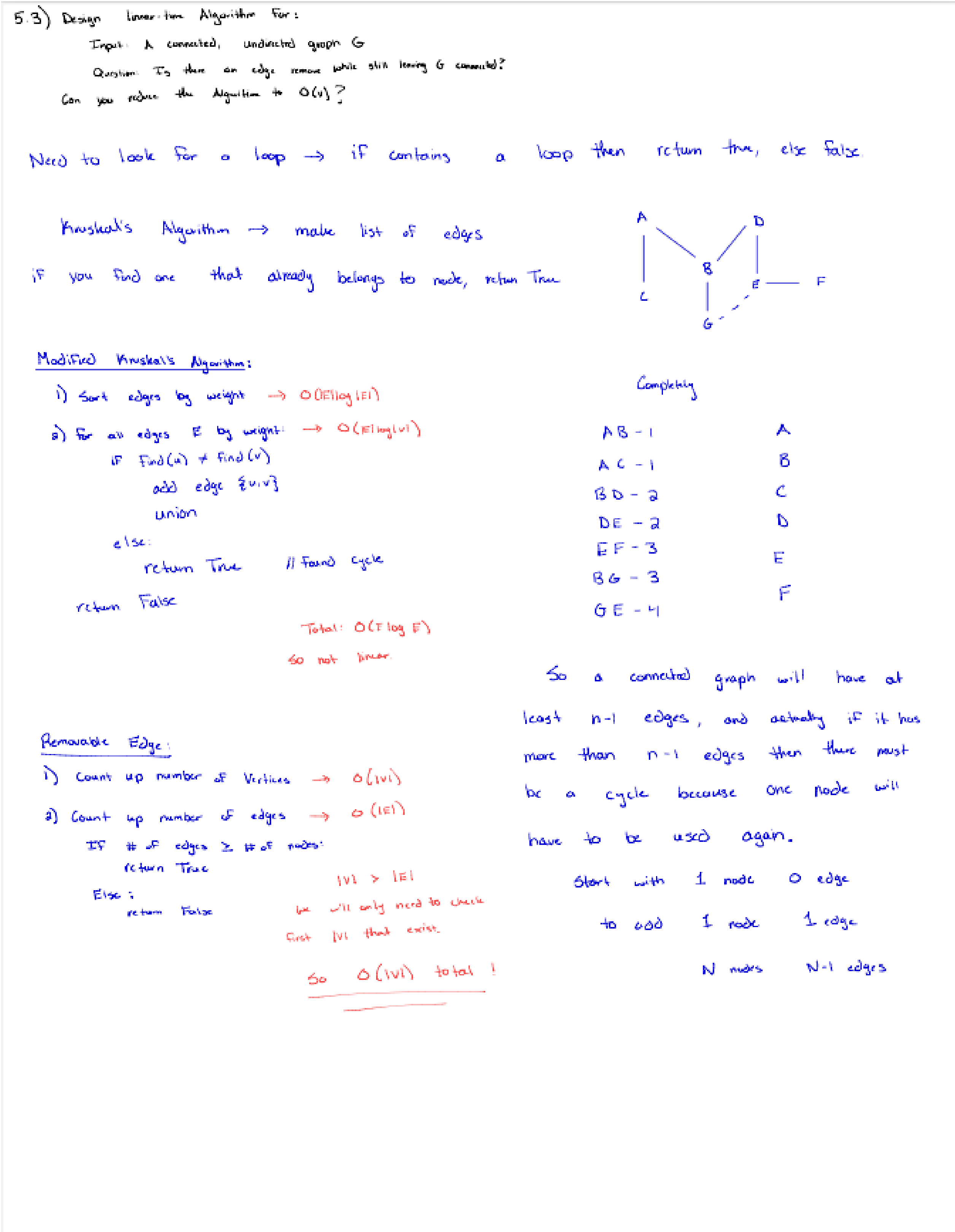


Figure 1: Page 1



5.3) Design linear-time Algorithm for:

Input: A connected, undirected graph G

Question: Is there an edge remove while still leaving G connected?

Can you reduce the Algorithm to O(v)?

Need to look for a loop → if contains a loop then return true, else false

Kruskal's Algorithm → make list of edges

if you find one that already belongs to node, return True

A D B E F C G

Modified Kruskal's Algorithm:

1) Sort edges by weight → $O(|E|\log|E|)$
2) for all edges E by weight: → $O(|E|\log|V|)$
   - if Find(u) ≠ Find(v)
     - add edge {u,v}
     - union
   - else:
     - return True // found cycle
   - return False

Total: $O(E \log E)$

so not linear.

Complexity

AB - 1	A
AC - 1	B
BD - 2	C
DE - 2	D
EF - 3	E
BG - 3	F
GE - 4	

So a connected graph will have at least n-1 edges, and actually if it has more than n-1 edges then there must be a cycle because one node will have to be used again.

Start with 1 node 0 edge

to add 1 node 1 edge

N nodes N-1 edges

Removable Edge:

1) Count up number of Vertices → $O(|V|)$
2) Count up number of edges → $O(|E|)$
   - If # of edges ≥ # of nodes:
     - return True
   - Else:
     - return False

$|V| > |E|$

we will only need to check first |V| that exist.

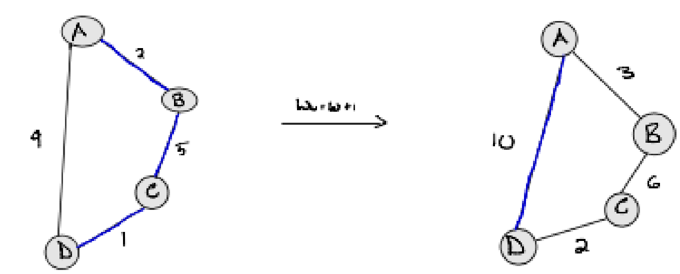
So $O(|V|)$ total!

Figure 2: Page 2

5.5) When weights of MST all increase by 1

a) The MST will not change because all of the weights will be increasing by the same amount. So
when we go through Kruskal's it will be the same for adding all the vertices to makeset
because their values did not change. Then, when we go to sort, since they were all increased by the
same amount, the sorted list of edges based on weight will be the same because
they will all be the same in terms of relationship to each other. So since all the
nodes have the same head/tail values as before and the weights are increased by the
same amount, then all the operations will proceed in the same order and will create the same MST ∎

b) Consider the example where the shortest path is made of multiple components:



Minimum A → D:

A → B → C → D

Minimum A → D

A → D

So yes the shortest paths can change

Figure 3: Page 3

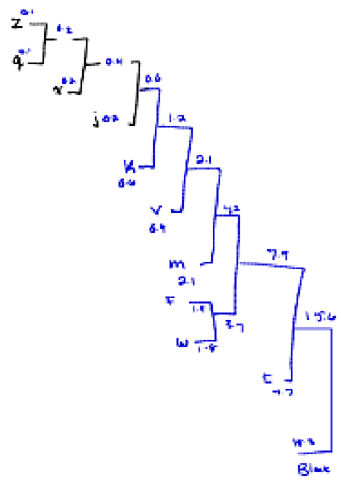
5.18)

blank : 1
e : 01
t : 001
a : 0001
o : 00001
i : 000001
n : 0000001
⋮

So expected number of bits/letter is 13.5 → way too big

Not optimal

Construct Tree based on letter Frequencies :



b) Expected Bits/Letter

$$= \frac{\sum f_a l_a}{\text{Sum of freq}} = \frac{\quad}{101}$$

d) Yes the data that we compress should be more limited, then we would be able to compress it by a smaller set of possibilities, like if it only contained a limited number of words with only certain letters.

Figure 4: Page 3