Lab 21 Notes

CS 61BL Summer 2019 — Lab 21

August 12, 2019

Name:	SID:

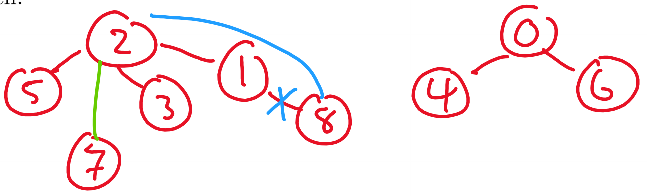
Please complete this worksheet during your lab, and turn it in to your TA by the end of your section. You are encouraged to work with your neighbors collaboratively.

Section Number: 01 02 03 04 05 06 07 08 09 10 11 12

1 Disjoint Sets

1.1 Run the following `union` and `find` calls on a weighted quick union structure with path compression, and write down the final array representation and the output of each `find` operation next to where it was called. If we are `union`-ing two sets of the same size, break ties by choosing the first set to be the root. You may find it helpful to draw out the tree structure as well!

```
union(2, 3); union(5, 7);
union(3, 5); find(3); __2___
union(1, 8); union(7, 1);
union(0, 6); union(6, 4);
find(8); __2__
find(6); __0__
```



wqupc =

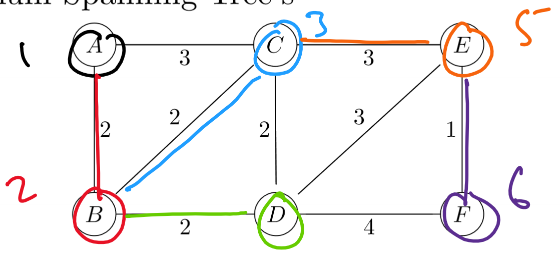
0	1	2	3	4	5	6	7	8
-3	2	-6	2	0	2	0	2	2

1.2 For each of the following implementations of the disjoint sets data structure, write down the worst case runtimes for `union` and `find` in Θ() notation. ⇒Θ

Note: WQU stands for weighted quick union. Weighted quick union with path compression is not included in this table because its runtime is usually analyzed in terms of M total `union` and `find` operations since the two methods are highly dependent on each other.

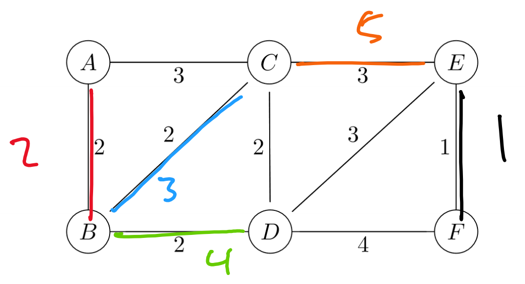
	union	find	Notes (not graded)
Quick Find	Θ(N)	Θ(1)	
Quick Union	Θ(~~1~~ N)	Θ(N)	
WQU	Θ(logN)	Θ(logN)	

2 Minimum Spanning Tree's



2.1 Perform Prim's algorithm on the graph above to find the MST and fill in the boxes corresponding to the edges in the MST. Pick A as the starting node. Whenever there is more than one node with the same cost, process them in alphabetical order.

[✓ $\overline{AB}$] ☐ $\overline{AC}$ [✓ $\overline{BC}$] [✓ $\overline{BD}$] ☐ $\overline{CD}$ [✓ $\overline{CE}$] ☐ $\overline{DE}$ ☐ $\overline{DF}$ [✓ $\overline{EF}$]



2.2 Perform Kruskal's algorithm on the graph above to find the MST and fill in the boxes corresponding to the edges in the MST. When deciding between equiweighted edges, alphabetically sort the edge, and then pick in lexicographic order.For instance, edges are always written as $\overline{AB}$ or $\overline{AC}$, never $\overline{BA}$ or $\overline{CA}$. If deciding between $\overline{AB}$ and $\overline{AC}$, pick $\overline{AB}$ first.

[✓ $\overline{AB}$] ☐ $\overline{AC}$ [✓ $\overline{BC}$] [✓ $\overline{BD}$] ☐ $\overline{CD}$ [✓ $\overline{CE}$] ☐ $\overline{DE}$ ☐ $\overline{DF}$ [✓ $\overline{EF}$]

vertices in heap

↓

2.3 For each of MST finding algorithms that we discussed during lab, write down the worst case runtime in Θ() notation.

process E times, heap size V

	Runtime	Assumptions
Prim's	$\Theta(E \log V)$	Use a binary min heap as the priority queue.
Kruskal's	$\Theta(EV)$	If we want to check if u and v are connected, use DFS/BFS from u and see if we reach v.
Kruskal's	$\Theta(E \log E)$	If we want to check if u and v are connected, use disjoint sets to see if u and v are connected.

DFS/BFS runtime

E logV for WQU

E logE to sort

E logE + E logV

↳ assume connected → $V \in O(E) \Rightarrow$

$= \Theta(E \log E)$