Name: Daniel Kim
ID: 102353420

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 6 – Due Fri Feb 28 11:55pm **Spring 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solutions:

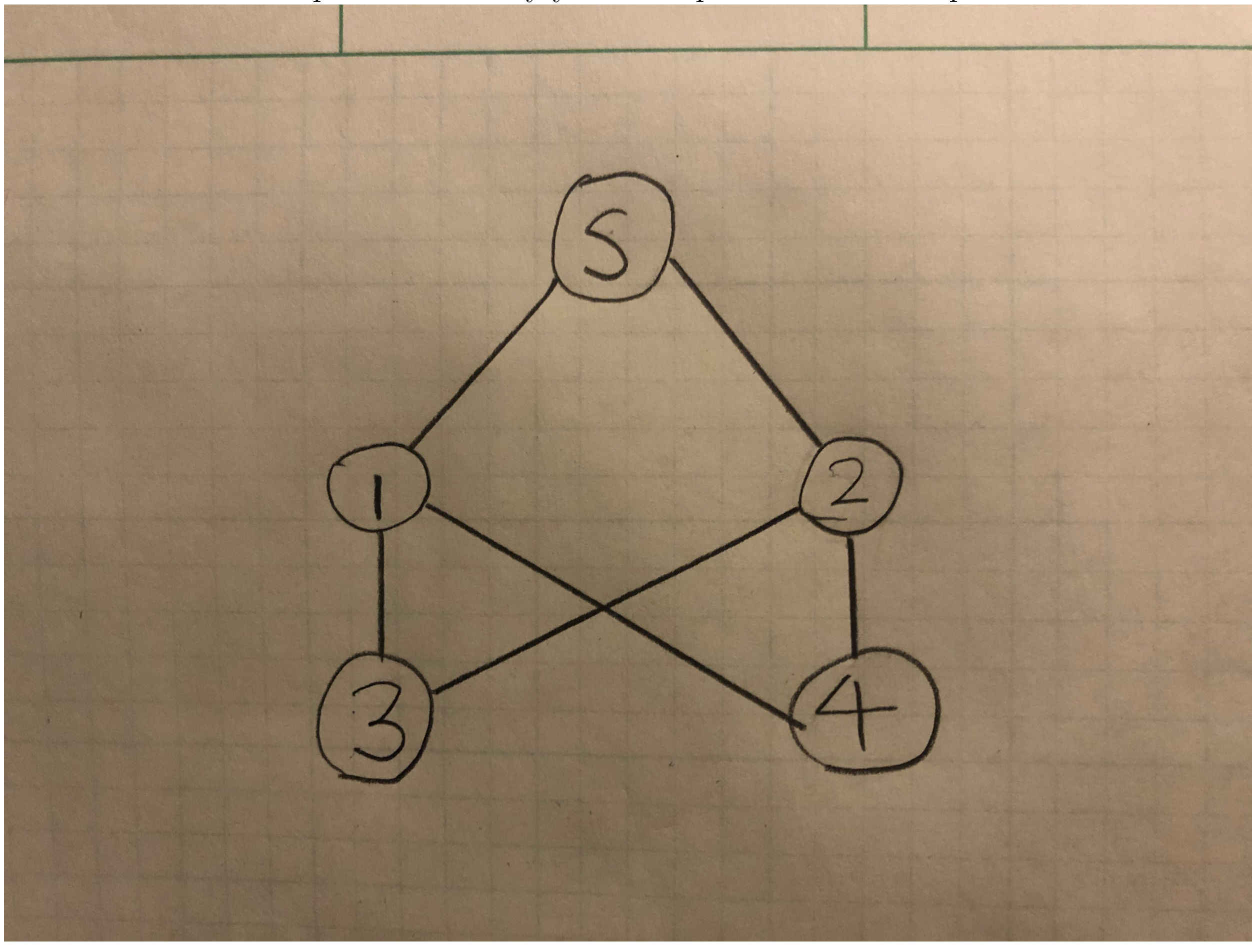
- All submissions must be easily legible.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please allot at least as many pages per problem (or subproblem) as are allotted in this template.
- For drawing graphs, you may include scans of hand-drawn graphs into your PDF file. **However, the rest of your solution (including the explanation of the graph) must be typed. If your words are not typed, you will get a 0 for that part of the question.**

Quicklinks: 1 2 3 4a 4b 4c

Name: Daniel Kim
ID: 102353420

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 6 – Due Fri Feb 28 11:55pm **Spring 2020, CU-Boulder**

1. Give an example of a (simple, undirected) graph $G = (V, E)$, a start vertex $s \in V$ and a set of tree edges $E_T \subseteq E$ such that for each vertex $v \in V$, the unique path in the graph (V, E_T) from s to v is a shortest path in G, yet the set of edges E_T cannot be produced by running a breadth-first search on G, no matter how the vertices are ordered. Include an explanation of why your example satisfies the requirements.



$E_T = (S1, 13, S2, 24)$

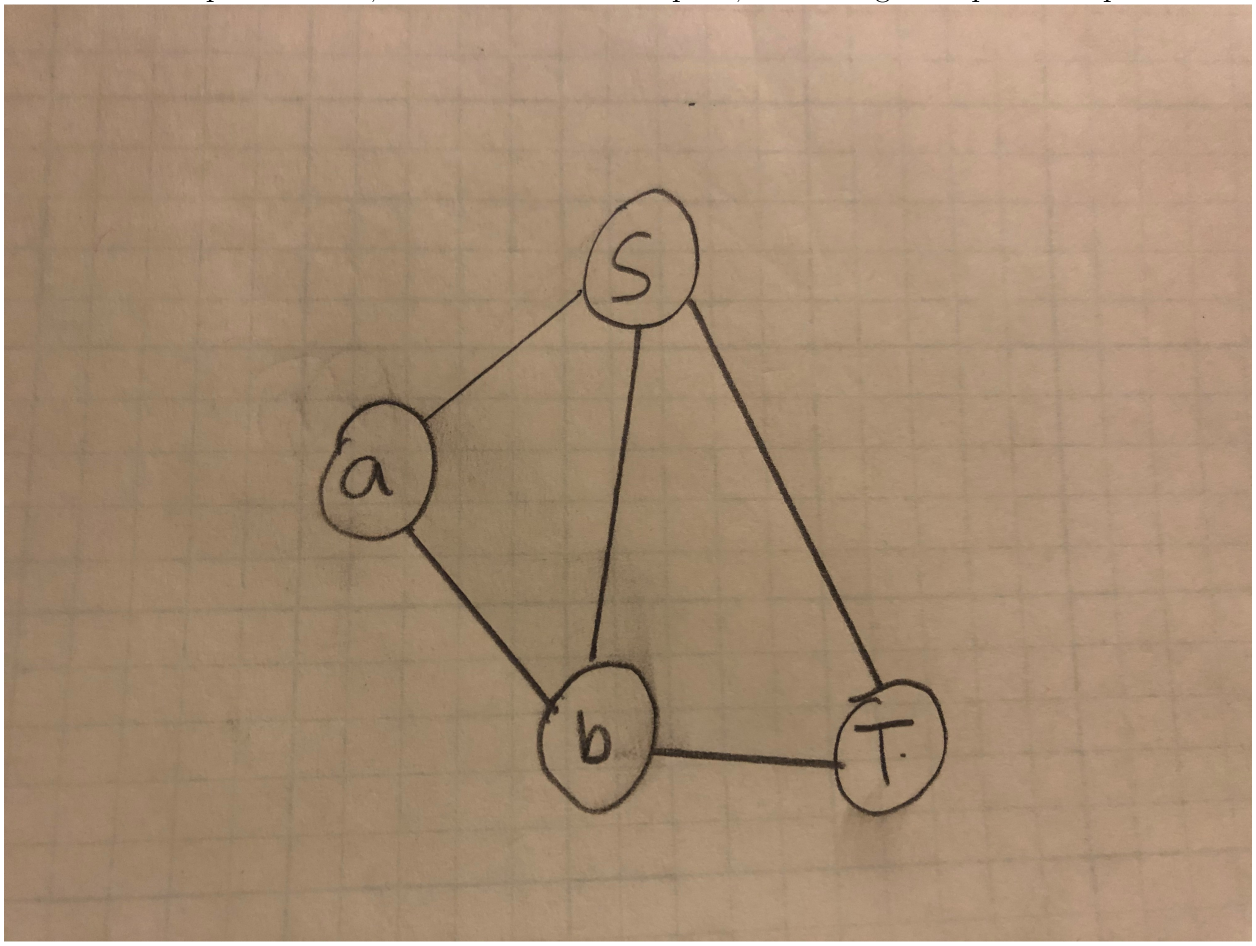
We need to know how BFS tree edges works to construct such a set of tree edges $E_T \subseteq E$. We also need multiple paths to give nodes because once a node is discovered, we will not have another edge continuing to that node.

Let's start at S and let's say that BFS add edges (S,1) and (S,2) to the BFS tree. Node 1 is en-queued before node 2, then BFS adds edges node 1 to node 3 and node 1 to node 4. To move on, if node 2 is en-queued before node 1, then BFS adds edges node 2 to node 3 and node 2 to node 4 to the tree. The set of edges E_T satisfies the requirements since the tree will not put (1,3) and (2,4) in the same tree.

Name: Daniel Kim
ID: 102353420

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 6 – Due Fri Feb 28 11:55pm **Spring 2020, CU-Boulder**

2. A *simple* $s \to t$ *path* in a graph G is a path in G starting at s, ending at t, and never visiting the same vertex twice. Give an example graph (simple, undirected, unweighted) $G = (V, E)$ and vertices $s, t \in V$ such that DFS finds a path from s to t which is neither a shortest path nor a longest simple path. Detail the execution of DFS (list the contents of the queue at each step, and which vertex it pops off the queue), show the final $s \to t$ path it finds, show a shorter $s \to t$ path, and a longer simple $s \to t$ path.



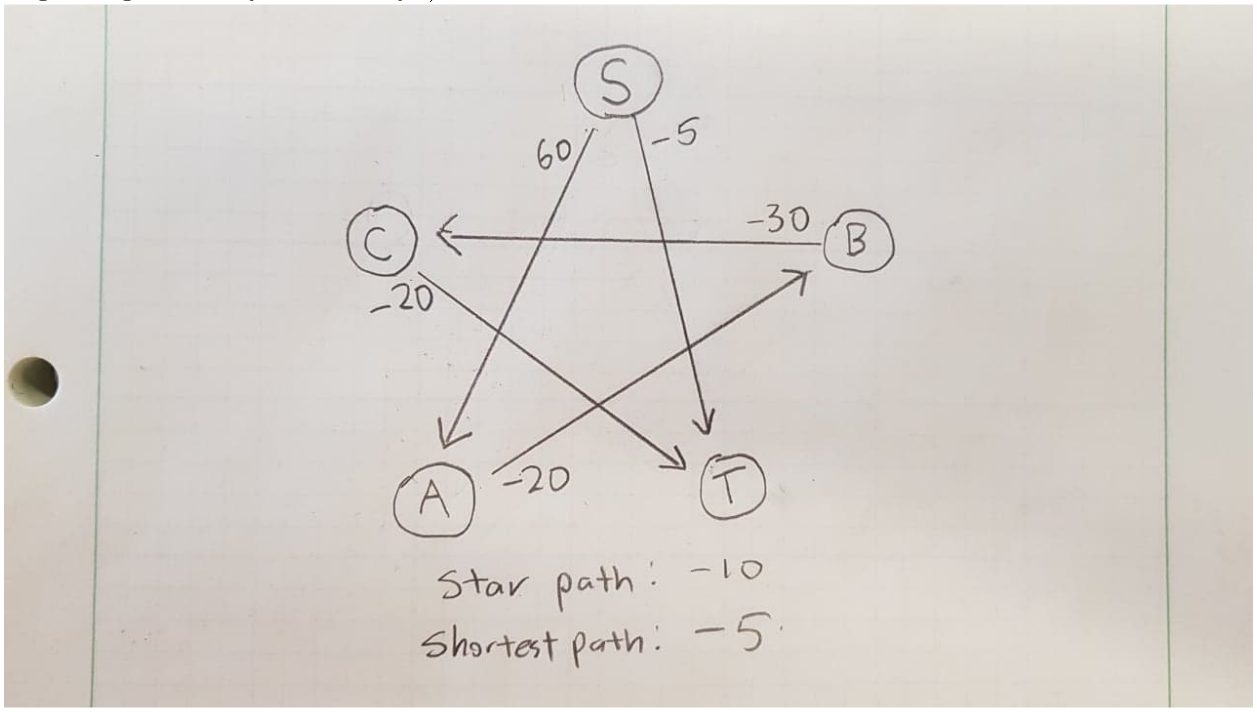
DFS adds S first, which sets queue to S also. Next, DFS will pop off S because of Last In First Out, and adds b which also sets queue to b. After that, DFS will pop off b then adds t.

Shortest $s \to t$ path is $s \to t$
Simple but longer path is $s \to a \to b \to t$
Final path is $s \to b \to t$

Name: $\boxed{\text{Daniel Kim}}$
ID: $\boxed{102353420}$

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 6 – Due Fri Feb 28 11:55pm **Spring 2020, CU-Boulder**

3. Give an example of a simple *directed, weighted* graph $G = (V, E, w\colon E \to \mathbb{R})$ and vertices $s, t \in V$ such that Dijkstra's algortihm started at s does *not* find the shortest $s \to t$ path. *Hint:* You will need to use negative edge weights. (Note: this shows that for finding shortest paths, the greedy choice property *fails* in the presence of negative edge weights. Do you see why?)



By Dijkstra's algorithm, the shortest path would be $s \to t$ (to the bottom right) path which the weight is -5 because it is smaller than other path with 60. However, the weight of that path is not the smallest. Let's say that if the algorithm takes the path to A with weight 60 first as an optimal solution, then path to B with weight -20, path to C with weight -30, and lastly path to t which is -20. Adding all that, the weight of the path is -10 which is lower than $s \to t$ path which is -5. The weight through Dijkstra's algorithm (-5) is higher than the weight in the optimal solution (-10). Therefore, this Dijkstra's algorithm does find the shortest path, but fails in the presence of negative edge weights since the optimal solution has smaller weight than Dijkstra's algorithm.

4. You have three batteries, with 4200, 2700, and 1600 mAh (milli-Amp-hours), respectively. The 2700 and 1600-mAh batteries are fully charged (containing 2700 mAh and 1600 mAh, respectively), while the 4200-mAh battery is empty, with 0 mAh. You have a battery transfer device which has a "source" battery position and a "target" battery position. When you place two batteries in the device, it instantaneously transfers as many mAh from the source battery to the target battery as possible. Thus, this device stops the transfer either when the source battery has no mAh remaining or when the destination battery is fully charged (whichever comes first).

   But battery transfers aren't free! The battery device is also hooked up to your phone by bluetooth, and automatically charges you a number of cents equal to however many mAh it just transfered.

   The goal in this problem is to determine whether there exists a sequence of transfers that leaves exactly 1200 mAh either in the 2700-mAh battery or the 1600-mAh battery, and if so, how little money you can spend to get this result.

(a) Rephrase this is as a graph problem. Give a precise definition of how to model this problem as a graph, and state the specific question about this graph that must be answered.

We need to have a directed and weighted graph G = (V, E) that the state of all three batteries are included in each V in the graph. Also, each E will represent the cost of a transfer. The question will be to find the shortest path from started node to the state node we are looking at.

(b) What algorithm should you apply to solve this problem?

Dijkstra's algorithm.

(c) Apply that algorithm to the question. Report and justify your answer—both the sequence of steps and the total cost. To justify your answer here likely means you will have to detail the steps the algorithm takes.

dist (K) is the distance from start node to node K. This is the calculation when we apply Dijkstra's algorithm to this graph:

Pick the start node A, dist (A) = 0.
dist (B) = dist (A) + w(A, B) = 1600, dist (C) = dist(A) + w(A, C)=2700
dist (D) = dist (B) + 1600 = 3200, dist(E) = dist (B) + 2600 = 4200
dist (F) = dist (C) + 1500 = 4200, dist (G) = dist (C) + 1600 = 4300
dist (H) = dist (D) + 1600 = 4800, dist (I) = dist (E) + 1600 = 5800
dist (J) = dist (F) + 2700 = 6900, dist(K) = dist (G) + 1600 = 5900
dist (O) = dist (H) + 1100 = 5900, dist (Q) = dist (I) 11600 = 7400
dist (L) = dist (K) + 1100 = 7000, dist (P) = dist (O) + 2700 = 8600
dist (M) = dist(J)+ 1500 = 8400, *This is the target node. We have 1200 - mAh in the 2700 - mAh battery.
dist (N)=dist (L) + 2700 = 9700, dist (R) = dist (Q) + 1600 = 9000

Every steps that the algorithm goes through, it won't update the dist of the nodes because the paths that been visited nodes already will always have a larger distance. The paths will keep visit the new nodes now all have a larger dist than dist (M), and node M is one of the expected node that we want to reach.

Dijkstra's algorithm did find the shortest path from A to M, which also means that it is a sequence of transfers that leaves exactly 1200 mAh in the 2700 - mAh battery.

List: (A, C), (C, F), (F,J), (J, M)
Total Cost is 8400

COLLABORATED WITH:
RUIJIANG MA
JINGQI LIU