Name: Kevina Wong

ID: 109179049

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.**

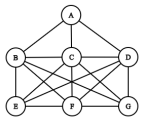
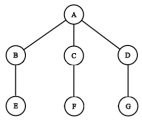
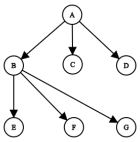
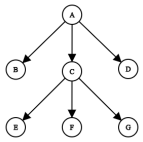
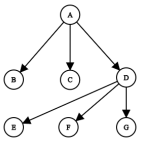
Quicklinks: 1 2 3 4a 4b 4c

Name: Kevina Wong
ID: 109179049

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.

   Here is my graph *G=(V,E)* with all V and E displayed:
   **Note that my start vertex, s, is A.*

F.1: Graph (V, E) *F.2: Graph(V, E_T)* *F.3: BFS (L)* *F.4: BFS (C)* *F.5: BFS (R)*

   Requirements to meet:

   (a) $E_T \in E$, **E_T is the set of tree edges.**
   This requirement is met because looking at Figures 1 and 2, we can see that all of the edged in the set of edges E_T can be found in graph $G(V, E)$. We can further confirm that $G(V, E_T)$ is, in fact, a tree because paths are unique and do not have any cycles, which is true for this example as shown in Figure 2.

   (b) **The unique path in $G(V, E_T)$ from s to v is the shortest path in G.**
   This requirement is met because in every path from s to v in graph $G(V, E_T)$, it will take at most the traversal of two edges at most to reach from any s to v, which yields the shortest path. Furthermore, because the paths are unique, we can conclude the path from s to v is the shortest possible path.

   (c) **The set of edges cannot be produced by running a BFS.**
   This requirement is met because F.3, F.4, and F.5 show the possible set of edges when running a BFS. Since there does not exist a BFS path that matches $G(V, E_T)$ (shown in F.2), we can conclude that the set of edges cannot be produced by running a BFS.

Name: Kevina Wong
ID: 109179049

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.

**Note: For this particular problem, I am assuming that the length of each edge between two vertexes are all equal. **Note: For this particular problem, I am assuming that the nodes are added to the queue from right to left.*

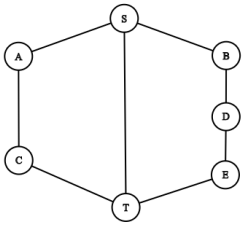


Figure I: Graph G

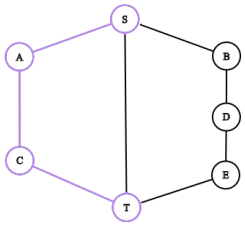


Figure II: DFS Path

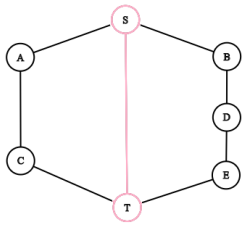


Figure III: Shortest Path

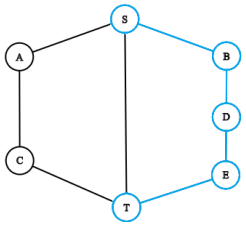


Figure IV: Longest Path

Queue	Pop	Add Child(ren)	Visited
$[S]$	S	B,A	$[S]$
$[B, A]$	A	C	$[A, C]$
$[B, C]$	C	T	$[S, A, C, T]$
$[B, T]$	T	None	$[S, A, C, T]$
$[B]$	B	D	$[S, A, C, T, B]$
$[D]$	D	E	$[S, A, C, T, B, D]$
$[E]$	E	None	$[S, A, C, T, B, D, E]$

**Note that after you pop T in the queue, the DFS search for the path from S to T has already been found, and the last three steps in the queue table doesn't change what path is outputted.*

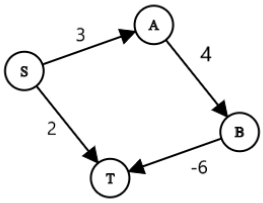
DFS Path Found from S to T: $S \to A \to C \to T$

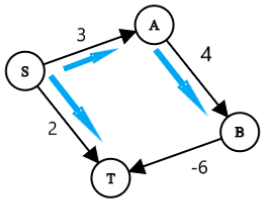
When finding a path from $S \to T$ using DFS, we can see using Figure II and the table that the path found is neither the shortest nor the longest path. Figure III shows that a shorter path exists, and Figure IV shows that a longer path exists.

Name: Kevina Wong
ID: 109179049

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 algorithm 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?)





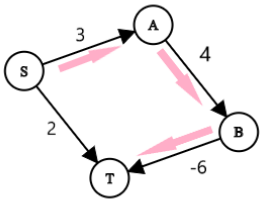


Figure A: graph G *Figure B: Dihkstra's Path(s)* *Figure C: Optimal Path*

In this example (Figure A), Dijkstra's algorithm fails to yield the lowest cost path. This is because when implementing Dijkstra's algorithm on this particular weighted graph, the total cost of the path that it yields is 2. However, the total cost of the lowest cost path of graph G is 1. When using Dijkstra's algorithm, these are the steps that it would pick along with the cost:

	Dijkstra's		**Lowest Cost**	
Step	Vertex	Cost of Path	Vertex	Cost of Path
1	S	0	S	0
2	**T**	**2**	A	3
3	A	3	B	7
4	B	7	**T**	**1**
Lowest cost	**2**		**1**	

From this table, it is clear that the shortest path from s to t that Dijkstra's algorithm yields is not the shortest path possible as Dijkstra's path yields a cost of 2, while the most optimal path that was not found by Dijkstra's algorithm has a cost of 1. You can also refer to Figure B and Figure C above to visually see the paths.

**Note that when I say "shortest", this implies lowest cost for this particular question.*

Name: Kevina Wong
ID: 109179049

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 transferred.

   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.

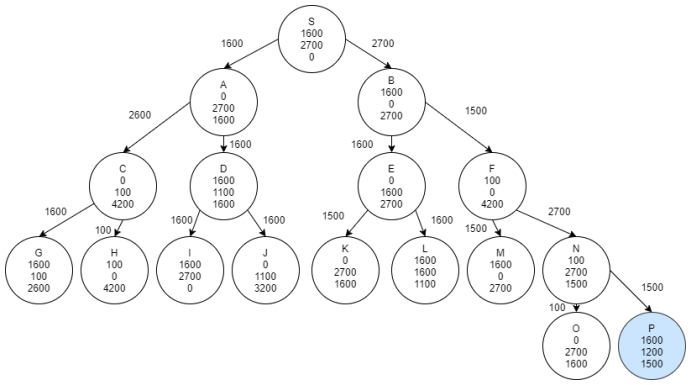
We have a directed, weighted graph $G(N, E)$, such that each N has properties V and M, representing value and max value, respectively. Values of V are initially $V = \{1600, 2700, 0\}$, and is not constant as the algorithm progresses. The weights, W, are also not constant. For an edge, E, connecting two vertices, N_1 and N_2, N_1 transfers as much value to N_2. There is also a cost associated with this transfer. The cost is equal to the amount of value transferred between vertexes. The max possible vertex values are as listed $M = \{1600, 2700, 4200\}$, and these are constant values. Also note that when a vertex hits its max, the value of the vertex of the max will be its max value and the remainder will remain in the source vertex. Provide an example of weights such that when Dijkstra's algorithm is applied, we have a vertex with exactly 1200 as its value; however, the vertex with exactly 1200 as its value cannot be the vertex with a max value of 4200. Provide the weights that were used to produce this outcome and sum them together to show the total cost. Lastly, the goal is to have the smallest total cost.

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

We would use Dijkstra's algorithm to solve this problem. This is because the goal is to find the most cost-efficient path of weighted edges, and Dijkstra's algorithm works to find the shortest path between vertexes.

(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.

**Assume that the starting node is S.* Here is a graph that describes each step of Dijkstra's algorithm for this problem and a table that shows the cost of each path:



Path	Cost
$S \to A$	1600
$S \to B$	2700
$S \to A \to D$	3200
$S \to B \to E$	3200
$S \to B \to F$	3200
$S \to A \to C \to H$	3300
$S \to A \to C$	4200
$S \to A \to D \to I$	4200
$S \to A \to D \to J$	4800
$S \to B \to F \to M$	5700
$S \to A \to C \to G$	5800
$S \to B \to E \to K$	5800
$S \to B \to E \to L$	5900
$S \to B \to F \to N$	6900
$S \to B \to F \to N \to O$	7000
$S \to B \to F \to N \to P$	8400

The path $S \to B \to F \to N \to P$ produces a node where one of the nodes has exactly a value of 1200. The cost to get to this node is 8400 cents.