CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Nagesh

Problem Set 2

Due Date ..... September 12, 2022 8pm MT
Name ..... **Aidan Reese**
Student ID **108418975**
Collaborators ..... **Miles Sanders, Tyler Carr, Heather Monteson, DJ Richardson, Vikki Wong**

Contents

Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- 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 submit this document with no fewer pages than the blank template (or Gradescope has issues with it).

- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign)

Problem HC.
- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

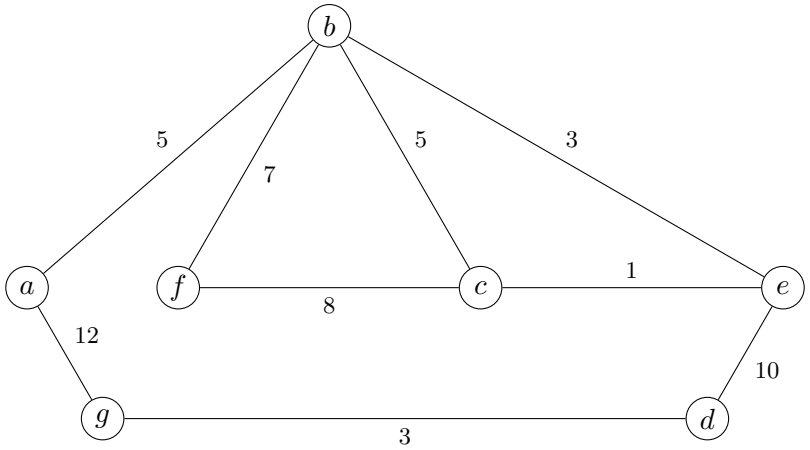
Agreed (Aidan Reese). □

3 Standard 3 – Dijkstra's Algorithm

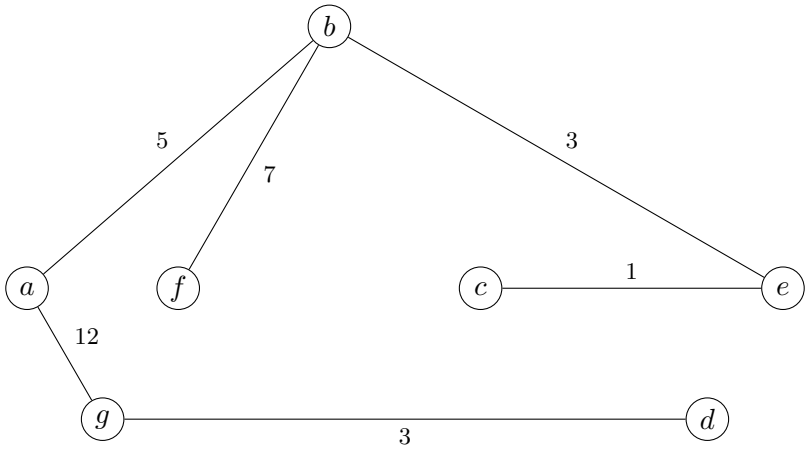
3.1 Problem 1

Problem 1. Consider the undirected weighted graph $G(V, E, w)$ pictured below. Work through Dijkstra's algorithm on the following graph, using the source vertex a. **Note:** In order to get full credits consider the following:

- Clearly include the contents of the priority queue, the distance from a and the parent of each vertex at each iteration.
- If you use a table to store the distances, clearly label the keys according to the vertex names rather than numeric indices (i.e., `dist['B']` is more descriptive than `dist['1']`).
- You do **not** need to draw the graph at each iteration, though you are welcome to do so. [This may be helpful scratch work, which you do not need to include.]
- Finally represent the shortest path graph.



Answer. Priority Queue is placed on the Next page. □



Priority Queue :

	A	B	C	D	E	F	G
Dist[V]	0	∞	∞	∞	∞	∞	∞
Pred[V]	A	NULL	NULL	NULL	NULL	NULL	NULL

	A	B	C	D	E	F	G
Dist[V]	0	5	∞	∞	∞	∞	∞
Pred[V]	A	A	NULL	NULL	NULL	NULL	NULL

	A	B	C	D	E	F	G
Dist[V]	0	5	∞	∞	8	∞	∞
Pred[V]	A	A	NULL	NULL	B	NULL	NULL

	A	B	C	D	E	F	G
Dist[V]	0	5	9	∞	8	∞	∞
Pred[V]	A	A	E	NULL	B	NULL	NULL

Removing (B,C) edge because its a more expensive path to C

	A	B	C	D	E	F	G
Dist[V]	0	5	9	∞	8	12	∞
Pred[V]	A	A	E	NULL	B	B	NULL

	A	B	C	D	E	F	G
Dist[V]	0	5	9	∞	8	12	12
Pred[V]	A	A	E	NULL	B	B	G

	A	B	C	D	E	F	G
Dist[V]	0	5	9	15	8	12	12
Pred[V]	A	A	E	G	B	B	G

	A	B	C	D	E	F	G
Dist[V]	0	5	9	15	8	12	12
Pred[V]	A	A	E	G	B	B	A

Removing (F,C) edge because its a more expensive path to C

Removing (D,E) edge because its a more expensive path to E

Final :

	A	B	C	D	E	F	G
Dist[V]	0	5	9	15	8	12	12
Pred[V]	A	A	E	G	B	B	A

3.2 Problem 2

Problem 2. You have three batteries, with capacities of 40, 25, and 16 Ah (Amp-hours), respectively. The 25 and 16-Ah batteries are fully charged (containing 25 Ah and 16 Ah, respectively), while the 40-Ah battery is empty, with 0 Ah. 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 Ah from the source battery to the target battery as possible. Thus, this device stops the transfer either when the source battery has no Ah 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 dollars equal to however many Ah it just transfered.

The goal in this problem is to determine whether there exists a sequence of transfers that leaves exactly 10 Ah either in the 25-Ah battery or the 16-Ah battery, and if so, how little money you can spend to get this result. Do the following.

3.2.a Problem 2(a)

(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. [**Note:** While you are welcome to draw the graph, it is enough to provide 1-2 sentences clearly describing what the vertices are and when two vertices are adjacent. If the graph is weighted, clearly specify what the edge weights are.]

Answer.
We can set a graph to where each of the nodes have 3 values, representing each of the batteries. The Source Node would be Initialized as (0,25,16). From here each edge would represent the possible moves that the values inside of the node could take. The weights of these edges would be defined by how many amps are being transfered between each battery. Given enough space there will be at least one node that has one of its 3 values set to 10 and that would be a solution, else if there are no new nodes being formed there is not a solution. □

3.2.b Problem 2(b)

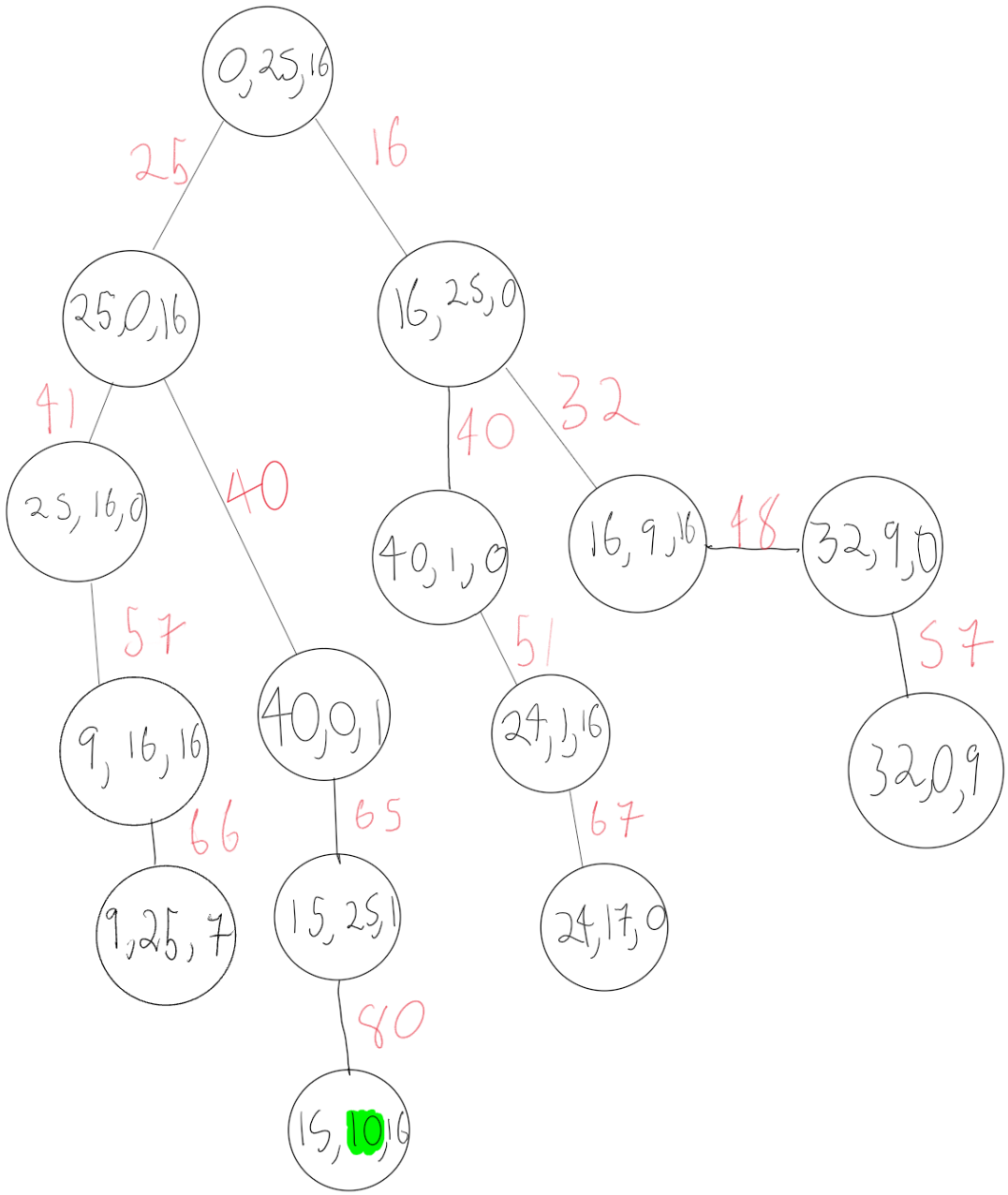
(b) Clearly describe an algorithm to solve this problem. If you use an algorithm covered in class, it is enough to state that. If you modify an algorithm from class, clearly outline any modifications. Make sure to explicitly specify any parameters that need to be passed to the initial function call. You need not write the algorithm.

Answer.
Because we are attempting to minimize the amount of amps being used in the problem we can use Dijkstra that only explores where there is a low number to total amps moved. Exploring farther and farther into the graph until it finds a node with a value of 10 inside of it. This will minimize the number of amps as once a solution is found, all other cheaper routs to nodes will of been attempted and any other solution will not of been discovered as it will cost more amps to find. □

3.2.c Problem 2(c)

(c) Apply that algorithm to the question. Report and justify your answer. Here, justification includes the sequences of vertices visited and the total cost. **Note:** For full credits make sure to draw the graph with all the battery states. Then apply the algorithm that you have described until the desired final states are reached.

Answer. Notably the Algorithm will not include any nodes that were previously visited as inherent to a greedy algorithm. Such here is the shortest path tree, finding the answer at the very bottom at a cost of 80amps. In addition to this, the edge weights that I included is the sum of the path to get to the given node and not necicarily the edge weight itself. □



4 Standard 4 – Examples Where Greedy Algorithms Fail

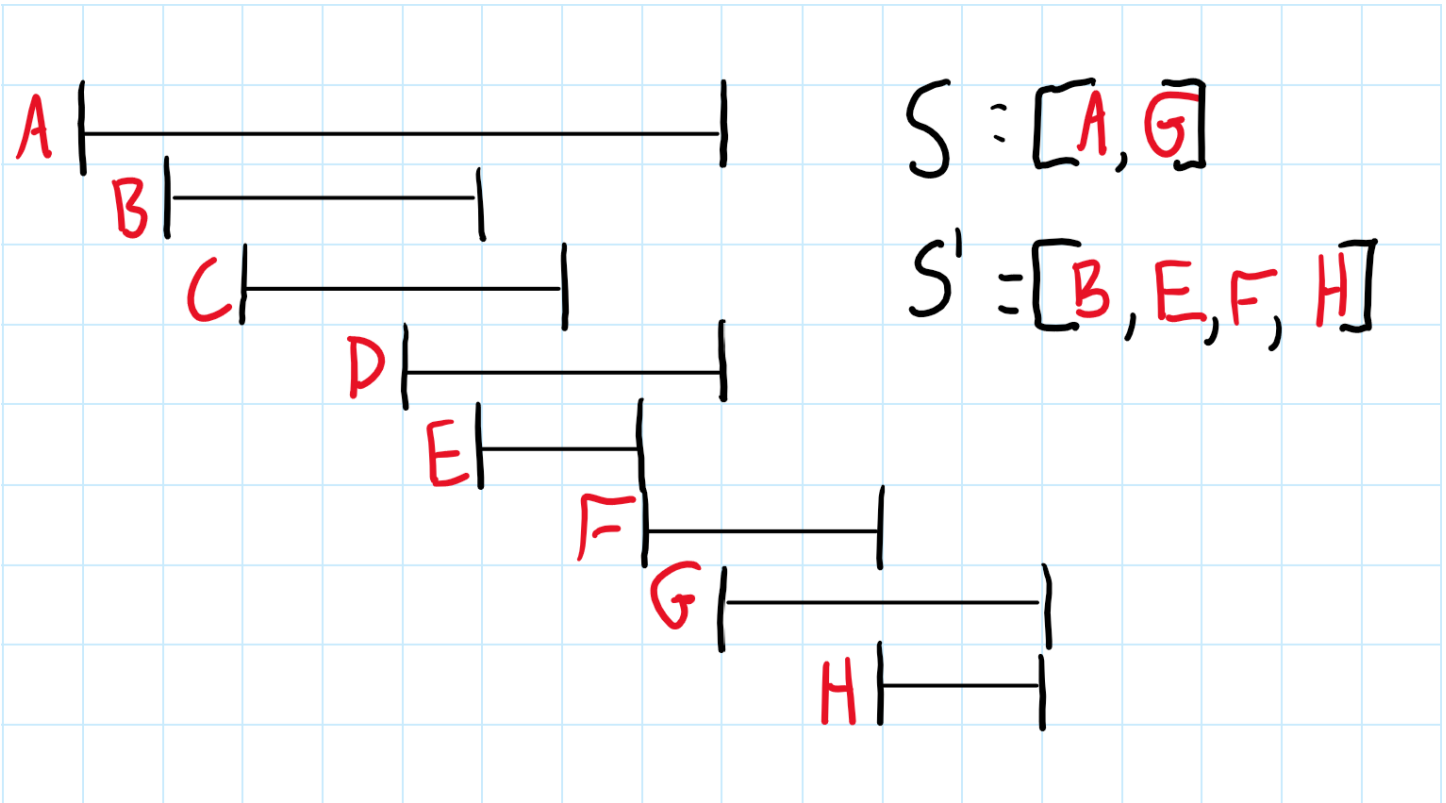
4.3 Problem 3

Problem 3. Recall the Interval Scheduling problem, where we take as input a set of intervals $\mathcal{I}$. The goal is to find a maximum-sized set $S \subseteq \mathcal{I}$, where no two intervals in S intersect. Consider the greedy algorithm where we place all of the intervals of $\mathcal{I}$ into a priority queue, ordered earliest start time to latest start time. We then construct a set S by adding intervals to S as we poll them from the priority queue, provided the element we polled does not intersect with any interval already in S.

Provide an example with at least 5 intervals where this algorithm fails to yield a maximum-sized set of pairwise non-overlapping intervals. Clearly specify both the set S that the algorithm constructs, as well a larger set of pairwise non-overlapping intervals.

You may explicitly specify the intervals by their start and end times (such as in the examples from class) or by drawing them. **If you draw them, please make it very clear whether two intervals overlap.** You are welcome to hand-draw and embed an image, provided it is legible and we do not have to rotate our screens to grade your work. Your justification should still be typed. If you would prefer to draw the intervals using LaTeX, we have provided sample code below.

Answer.
Here we can see that if the greedy algorithm chooses to do A first then it will only be able to do one more task. If it does B first and is optimal to how many it can fit S' can fit 4 intervals in the set. □



4.4 Problem 4

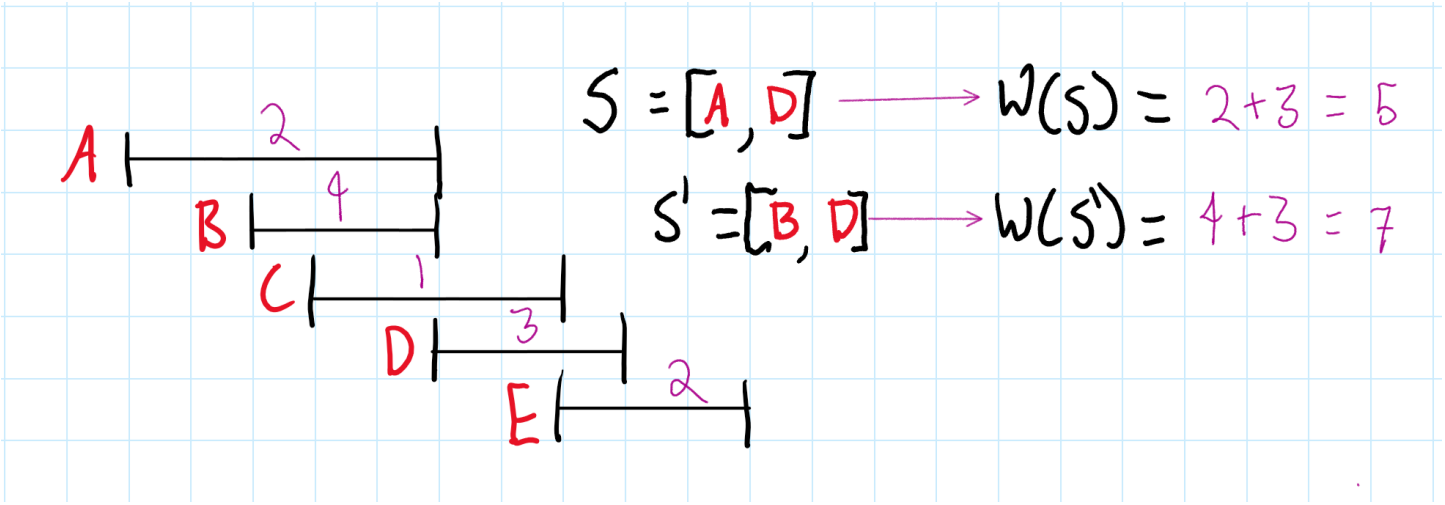
Problem 4. Consider now the Weighted Interval Scheduling problem, where each interval i is specified by

$$([\text{start}_i, \text{end}_i], \text{weight}_i).$$

Here, the weight is an assigned value that is independent of the length $\text{end}_i - \text{start}_i$. Here, you may assume $\text{weight}_i > 0$. We seek a set S of pairwise non-overlapping intervals that maximizes $\sum_{i \in S} \text{weight}_i$. That is, rather than maximizing the number of intervals, we are seeking to maximize the sum of the weights.

Consider a greedy algorithm which works identically as in Problem 3. Draw an example with at least 5 appointments where this algorithm fails. Show the order in which the algorithm selects the intervals, and also show a subset with larger weight of non-overlapping intervals than the subset output by the greedy algorithm. The same comments apply here as for Problem 3 in terms of level of explanation.

Answer.
Because the weights are independent of the length the algorithm from 3 will produce S, which has less of a weight but longer run times. S' shows a solution that optimizes the weights and because it starts with B it is able to acheive a higher score. □



5 Standard 5 – Exchange Arguments

5.5 Problem 5

Problem 5. Recall the Making Change problem, where we have an infinite supply of pennies (worth 1 cent), nickels (worth 5 cents), dimes (worth 10 cents), and quarters (worth 25 cents). We take as input an integer $n \geq 0$. The goal is to make change for n using the fewest number of coins possible.

Prove that in an optimal solution, we use at most 2 dimes.

Proof. Given the statement in the problem states that at optimal solution will only have 2 dimes. So in every case that there are 3 or more dimes the solution can be reduced.

In a case of (3) dimes, the more optimal solution will be (1) Quarter and (1) nickel (2 Coins). For (4) Dimes, it can be broken into (1) Quarter and (1) dime and (1) Nickel (3 Coins). 5 or more dimes can be represented a combination of (2) Dimes and (3) Dimes, and we know that (3) Dimes can be broken down. For all $d \geq 5$, it can be represented as a combination of either (2) Dimes, 3 (Dimes), (4) Dimes.

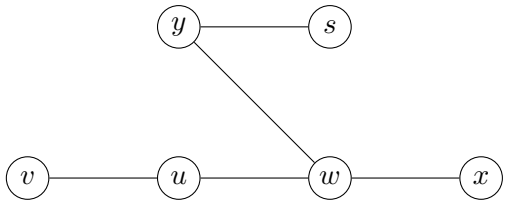
2 Dimes is optimal.
3 Dimes will produce a remainder of 0 dimes.
4 Dimes will produce a remainder fo 1 dime.

For any given set of dimes ≥ 3 , it can be represented by a combination of the stated subsets that can be reduced. Else if there are leftover dimes after reduction, this can form a new set of dimes and either be reduced further or the number of dimes is ≤ 2. □

5.6 Problem 6

Problem 6. Let $G = (V, E)$ be a graph. A *vertex cover* of G is a set of vertices C such that for every edge $uv \in E$, either $u \in C$ or $v \in C$. That is, every edge of G has at least one endpoint in C.

(a) Let T be the graph



Find a vertex cover C of T that includes the vertex v. Now, explain why the set $C' = (C \setminus \{v\}) \cup \{u\}$ obtained from your C by removing v and adding (if it is not already present) u is also a vertex cover of T.

(b) Now prove this property in general. That is, let T be an arbitrary tree, suppose that C is a vertex cover of T, and suppose that C contains a leaf vertex v. Let u be the unique neighbor of v in T, and let $C' = (C\setminus\{v\})\cup\{u\}$ be the set of vertices obtained from C by obtained from C by removing v and adding (if it is not already present) u. Carefully explain why C' is a vertex cover of T.

Answer.
a.) Because the vertices of u and v are adjacent to eachother and on the end of a branch, if u were to be added and v was removed they would both either be in the cover or adjacent to the cover without needing to adjust the rest of the graph.

b.) The two cases are C and C', where in C, the parent node of v is u, u is apart of the cover and because u is v's unique neighbor and share an edge, the cover is valid.

For the case of C' where the cover is now on v, it would still be a valid cover as the edge between v and u would still have an endpoint in the cover without changing the rest of the tree.

□