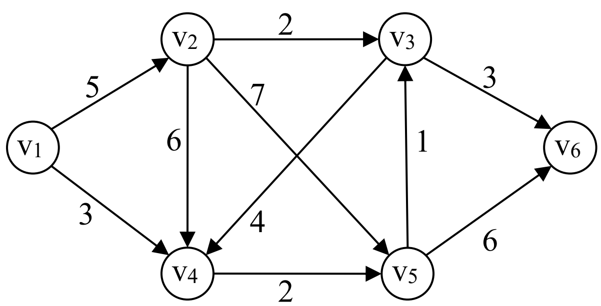
Washington State University
School of Electrical Engineering and Computer Science
Spring 2022

CptS 223 Advanced Data Structures in C++
Homework 12
Due: April 27, 2022 (11:59pm pacific time)

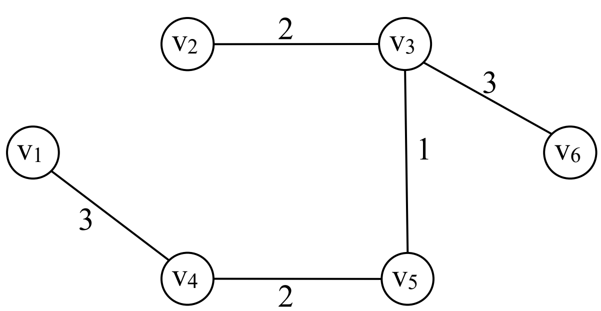
General Instructions: Put your answers to the following problems into a PDF document and upload the document as your submission for Homework 12 for the course CptS 223 Pullman on the Canvas system by the above deadline.

Questions 1-5 refer to the following graph G.



1. Show the minimum spanning tree of G assuming the edges are undirected.

   Solution:

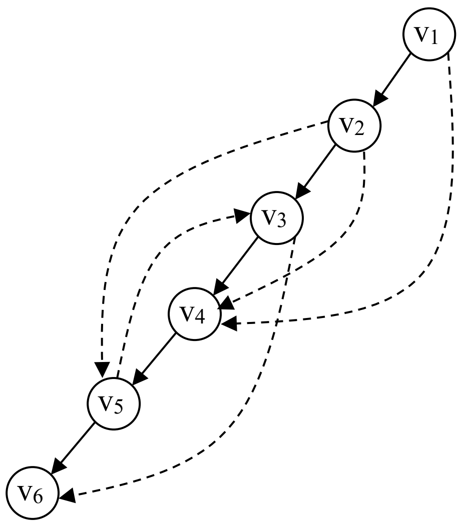


2. Give the articulation points, if any, in graph G after removing edge (v_5,v_6) and assuming the edges are undirected and unweighted.

   Solution: There is one articulation point: v_3.

3. Show the depth-first spanning forest (similar to that in Figure 9.77) that results from running depth-first search on graph G assuming the edges are unweighted. Be sure to show tree edges as solid arrows and forward/back/cross edges as dashed arrows. When there is a choice as to which vertex to visit next, always prefer the lower-numbered vertex.

   Solution:



4. Give the strongly-connected components of graph G assuming the edges are unweighted.

   Solution: {v1}, {v2}, {v3, v4, v5}, {v6}

5. Give the Euler circuit of graph G starting from v_1 assuming the edges are undirected and unweighted. When there is a choice as to which vertex to visit next, always prefer the lower-numbered vertex.

   Solution:

   - There are several Euler circuits, but there are only two consistent with the preference for lower-numbered vertices.
   - If you follow the efficient strategy in the book, then you always choose the lowest-numbered vertex on the most recently added cycle that still has untraversed edges. This yields the circuit: v1, v2, v4, v5, v3, v6, v5, v2, v3, v4, v1.
   - If you always choose the lowest vertex in the entire sequence so far, then you will start from v3, rather than v5, when adding the third cycle. This yields the circuit: v1, v2, v4, v5, v2, v3, v5, v6, v3, v4, v1.

6. Consider the following two decision problems.
   - Partition: Given a set A of integers, can you partition the set into two subsets A1 and A2 such that the sum of the elements of A1 equals the sum of the elements of A2, i.e., $A1 \cup A2 = A$, $A1 \cap A2 = \varnothing$, and $\sum_{i \in A1} i = \sum_{i \in A2} i$?
   - Subset-Sum: Given a set A of integers and an integer K, can you find a subset A1 of A whose elements sum to K, i.e., $\sum_{i \in A1} i = K$?

   Assuming Partition is NP-Complete, prove that Subset-Sum is NP-Complete.

   Solution: To show Subset-Sum is NP-Complete, we must show that it is in the set NP and that it is as hard as any problem in NP (i.e., it is in the set NP-Hard).

   To show a problem is in the set NP, we must show that a potential solution to the problem can be verified in polynomial time. A solution A1 to the Subset-Sum problem can be verified by checking that each element of A1 is also an element of A, and that the sum of the elements in A1 is equal to K. Both these checks can be done in time linear in the size of the set, which is within polynomial time. Thus Subset-Sum $\in$ NP.

   To show a problem is NP-Hard, we must show that some NP-Complete problem can be reduced, in polynomial time, to the problem. We choose the Partition problem to reduce to the Subset-Sum problem. The reduction is to pass the set A in Partition as the set A in Subset-Sum, and let $K = (\frac{1}{2} * \sum_{i \in A} i)$. If we can find a subset A1 of A whose elements sum to K, then this same A1, and $A2 = A - A1$, will solve the Partition problem. Computing A and K for Subset-Sum from the A of Partition, and computing the A1 and A2 for Partition from the A1 found for Subset-Sum, can all be done in time linear in the size of A, and thus the reduction is polynomial time. Therefore, Subset-Sum is NP-Hard.

   Since Subset-Sum is both in NP and NP-Hard, it is thus in NP-Complete.