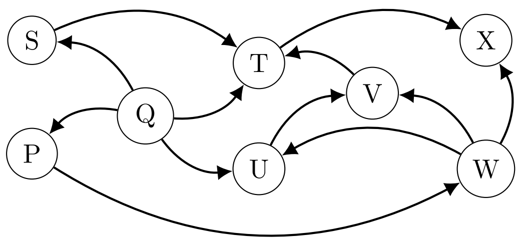
Week 10. Problem set (solutions by Danil Elgin)

1. Consider the following directed graph $\mathcal{G}_1$ and answer questions:



   (a) Write down **all** possible topological sortings for the vertices of $\mathcal{G}_1$.

   Answer: *1) QSPWUVTX*
   2) QPSWUVTX
   3) QPWSUVTX
   4) QPWUSVTX
   5) QPWUVSTX

   (b) Write down **all** edges that can be safely removed from $\mathcal{G}_1$ without affecting the set of possible topological sortings.

   Answer: *1)* $Q->T$
   2) $Q->U$
   3) $W->V$
   4) $W->X$

   (c) What is the maximum possible number of distinct topological sortings for the vertices of a graph with 8 vertices? Briefly justify your answer (1–2 sentences).

   Answer: $8! = 40320$
   The largest number of possible topological sorts will be if we have the fewest dependencies (edges) between vertices, because in this case we will have no difference in which order the vertices should be placed when sorting. The largest number will be when we have no edges at all.

   (d) What is the maximum possible number of distinct topological sortings for the vertices of a graph with 8 vertices and 12 edges? Briefly justify your answer (1–2 sentences).

   Answer: *1440*
   For the largest number of topological sorts, we need the maximum number of vertices with the same dependencies and the minimum dependencies between each other. We know that the maximum number of topologically sorted items is $8!$*, we will start from this and reduce it. Let's try* $7! * 1!$ *- however, this is impossible without a cycle, then comes* $6! * 2!$ *- it is possible to make such a graph, it appears if 2 vertices have 0 occurrences, and the remaining 6 vertices have 2 occurrences each.*

2. Consider a large undirected graph $\mathcal{G}_2 = (V, E)$ where vertex labels are integers. We represent this graph with a variation of the adjacency-list graph representation [GTG, §14.2.2], where instead of lists we use maps, represented by B-trees [CLRS, §18] with minimum degree t and vertex labels serving as keys in the search tree. What are the worst-case time complexities of the following Graph ADT operations over this modified representation, in terms of $|V|$, $|E|$, and t?

   (a) `areAdjacent(v, u)`
   Answer: $O(t * log_t(|V|))$

   (b) `insertVertex(v)`
   Answer: $O(1)$

   (c) `insertEdge(from, to, e)`
   Answer: $O(t * log_t(|V|))$

(d) `removeEdge(from, to)`

Answer: $O(t * log_t(|V|))$

(e) `removeVertex(v)`

Answer: $O(|V| * t * log_t(|V|))$

Assume that v, `from`, and `to` arguments above are *references* to vertex objects, not just labels. Briefly justify your answer (1–2 sentences).

Justification: *I assumed that only the adj-list is based on a B-tree, and the lists for vertices and edges are linked lists. In the worst case,* $O(t * log_t(|V|))$ *will be needed to search through the tree, to remove from the tree, and to add to the tree. 1) We go through the tree of one of the vertices is* $t * log_t(|V|)$ *2) Adding it to the linked list is a constant, 3) Adding the corresponding vertex to two trees is* $2 * t * log_t(|V|)$ *and adding the vertex to the linked list is a constant, 4) removing it from two trees is* $2 * t * log_t(|V|)$ *and we remove a vertex from the list that we already have a reference to, so the constant is 5) In the worst case, the vertex will have an edge with all other vertices, so we need to remove each co-corresponding edge from* $|E|$ *and remove an edge from each corresponding vertex, this is* $|V| * t * log_t(|V|)$.

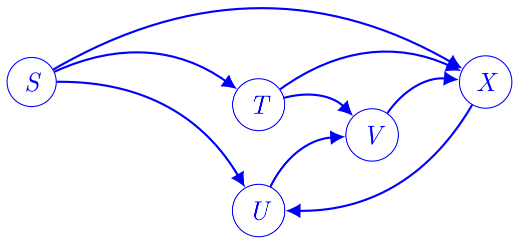
3. Provide a directed graph $\mathcal{G}_3 = (V, E)$, vertex $s \in V$, and a subset of edges $T \subseteq E$ such that

- both (V, E) and (V, T) are connected graphs,
- $|V| \leq 5$,
- T forms a tree with root s,
- the set of edges T cannot be produced by running BFS [CLRS, §20.2] on $\mathcal{G}_3$, no matter how the vertices are ordered in the adjacency lists,
- the set of edges T cannot be produced by running DFS [CLRS, §20.3] on $\mathcal{G}_3$, no matter how the vertices are ordered in the adjacency lists.

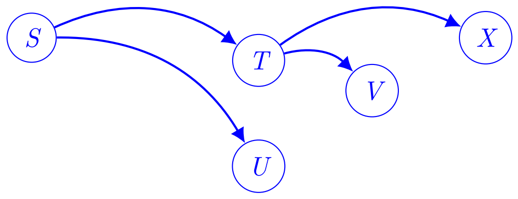
Briefly justify your answer (1–2 sentences).

Answer: $s = S \in V,\ T = \{S->T,\ S->U,\ T->V,\ T->X\}$

(a) $\mathcal{G}_3 = (V, E)$



(a) $\mathcal{G} = (V, T)$



Justification: *DFS:*
1) $S->U->V...$ *- is not the same as in (V, T)*
2) $S->T->V->X->U...$ *- is not the same as in (V, T)*
3) $S->T->X->U...$ *- is not the same as in (V, T)*
4) $S->X...$ *- is no the same as in (V, T)*

BFS:
$S->U, S->T, ->S->X...$ *- is not the same as in (V, T)*
If we do not start from the vertex S, then we will not be able to reach the edges coming from S.
We have considered all possible options, so it is impossible to get a set of edges T using BFS and DFS, also graphs (V, E) and (V, T) are connected and (V, T) forms a tree with the root S

References

[CLRS] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., 2022. *Introduction to algorithms, Fourth Edition*. MIT press.

[GTG] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser. *Data Structures and Algorithms in Java.* WILEY 2014.