Algorithms and Data Structures II.
Test1 (example exercises)

22 Elementary Graph Algorithms

22.2-1 Present breadth-first search on the directed graphs below[1], using the given vertex as source. Illustrate the run of the algorithm as you have seen it in the classroom. For each vertex, show the d and π values; and for all the times, show the transformations of the queue. Draw the breadth-first tree represented by the final π values.

22.2-1a Source vertex: 3

1 → 2; 4. 2 → 5. 3 → 5; 6.
4 → 2. 5 → 4. 6.

22.2-1b Source vertex: 5

1 → 4. 2 → 1; 3; 5. 3.
4 → 2; 5. 5 → 3; 4. 6 → 3; 5.

22.2-2 Illustrate the run of the breadth-first search on the undirected graph below[2], using vertex 4 as the source.

1 – 2; 5. 2 – 1; 6. 3 – 4; 6; 7. 4 – 3; 7; 8.
5 – 1. 6 – 2; 3; 7. 7 – 3; 4; 6; 8. 8 – 4; 7.

22.3-2 Show how depth-first search works on the graph below. Assume that in the indeterministic cases the DFS procedure considers the vertexes in alphabetical order. Show the discovery and finishing times for each vertex, and show the classification of each edge.

q → s; t; w. r → u; y. s → v. t → x; y.
u → y. v → w. w → s. x → z.
y → q. z → x.

22.4-1 Show the ordering of vertexes produced by TOPOLOGICAL-SORT when it is run on the DAGs below, under the assumption of Exercise 22.3-2.

q → s; t; w. r → u; y. s → v. t → x; y.
u → y. v → w. w. x → y; z.
y. z → v.

p → q; r. q → s; u. r → s; t. s → v.
t → v. u. v → u.

[1] $u \to v_1; \ldots v_n.$ means that the graph has the directed edges $(u, v_1), \ldots (u, v_n)$.
[2] $u - v_1; \ldots v_n.$ means that the graph has the undirected edges $(u, v_1), \ldots (u, v_n)$.

22.x-1. Let us suppose that we represent graph $G = (V, E)$ as an adjacency matrix $A/1 : \mathbb{B}[n, n]$ where $n = |V|$. Give a simple implementation

$$\textbf{BFS}(A, s, d, P)$$

of the Breadth-first search algorithm for this case that runs in $O(n^2)$ time ($d/1 : \mathbb{N}[n]$ and $P/1 : \mathbb{Z}[n]$). As a result of performing BFS, for each vertex $u \in 1..n$ of the graph:
- if $P[u] \geq 1$ then $P[u]$ is the parent of u in the breadth-first tree,
- if $P[u] = 0$ then u is the source vertex of the BFS.
- if $P[u] = -1$ then u is not reachable from the source vertex in the graph.

22.x-2. Let us suppose that we represent graph $G = (V, E)$ with adjacency list representation using pointer array $A/1$:E*$[n]$ where $n = |V|$ and objects of class E{ $+v$:$\mathbb{N}$; $+next$:E* } represent the edges. Give a simple implementation

$$\textbf{BFS}(A, s, d, P)$$

of the Breadth-first search algorithm for this case that runs in $O(n+m)$ time ($m = |G.E|$, $d/1 : \mathbb{N}[n]$ and $P/1 : \mathbb{Z}[n]$). As a result of performing BFS, for each vertex $u \in 1..n$ of the graph:
- if $P[u] \geq 1$ then $P[u]$ is the parent of u in the breadth-first tree,
- if $P[u] = 0$ then u is the source vertex of the BFS.
- if $P[u] = -1$ then u is not reachable from the source vertex in the graph.

22.x-3. Let us suppose that array $P/1 : \mathbb{Z}[n]$ contains the breadth-first tree of a BFS on a directed graph. For each vertex $u \in 1..n$ of the graph:
- if $P[u] \geq 1$ then $P[u]$ is the parent of u in the breadth-first tree,
- if $P[u] = 0$ then u is the source vertex of the BFS.
- if $P[u] = -1$ then u is not reachable from the source vertex in the graph.

Write procedure **printPath**(P, v) which prints the optimal path from the source vertex to vertex $v \in 1..n$, or the text "NO PATH", if v is not reachable from the source vertex.

22.x-4. Let us suppose that we represent graph $G = (V, E)$ as an adjacency matrix $A/1 : \mathbb{B}[n, n]$ where $n = |V|$. Write procedure

$$\textbf{adjMtx2adjList}(A, G)$$

which runs in $\Theta(n^2)$ time and makes a copy of this graph into G where G is going to be an adjacency list representation of the same graph i.e. $G/1$:E*$[n]$ where objects of class E{ $+v$:$\mathbb{N}$; $+next$:E* } represent the edges.

22.x-5. Let us suppose that we represent DAG $G = (V, E)$ with adjacency list representation using pointer array $A/1$:E*$[n]$ where $n = |V|$ and objects of class E{ $+v$:$\mathbb{N}$; $+next$:E* } represent the edges. Give a simple implementation of topological sort based on DFS:

topologicalSort(A, TO)

where a topological order of the vertices of G is to be put into array $TO/1 : \mathbb{N}[n]$. This implementation should run in $O(n + m)$ time where $m = |G.E|$.

22.x-6. Let us suppose that we represent graph $G = (V, E)$ as an adjacency matrix $A/1 : \mathbb{B}[n, n]$ where $n = |V|$. Give a simple implementation of topological sort based on DFS:

topologicalSort(A, TO)

where a topological order of the vertices of G is to be put into array $TO/1 : \mathbb{N}[n]$. This implementation should run in $O(n^2)$ time.