

1 Schedule

Part 1 of the homework is **due Oct 13**.
Part 2 of the homework is **due Oct 20**.
The **QUIZ** will be on **Thursday, Oct. 22**.

2 List of algorithms covered in the class

(B-basic, I-intermediate, A-advanced):

- B: DFS, BFS (p.85, DSV; p.105, DSV).
- B: Topological sort (p.90, DSV).
- A: Strongly connected components (p.91, DSV).
- B: Dijkstra's algorithm (p.108, DSV).
- I: Bellman-Ford algorithm (p.117, DSV).
- A: Floyd-Warshall algorithm (p.173, DSV).
- B: Prim's algorithm (p.137, DSV) (a.k.a., Borůvka's algorithm).
- I: Kruskal's algorithm (p.131, DSV).
- I: Union-find (p.132, DSV).
- A: Maximum matching in bipartite graphs.
- A: Maximum-weight matching in bipartite graphs.

3 Basic material

Important concepts, problems, theorems, and algorithms:

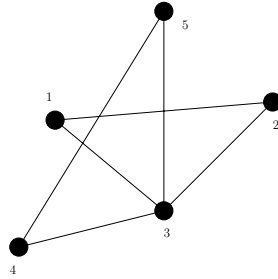
- Graph, directed graph, representing graphs (adjacency list representation, adjacency matrix).
- Connected component, strongly connected component.
- Topological sorting.
- Shortest path.
- Minimum spanning tree.

Testing method:

- Trace BFS and DFS algorithms.
- Given a drawing of graph give its adjacency matrix and/or adjacency lists representation.
- Given an adjacency matrix of a graph, draw the graph.
- Given a directed graph, find its strongly connected components.
- Given a directed acyclic graph (DAG), find a topological sorting of the graph.
- Given a weighted graph and two vertices u, v , find a shortest path from u to v . The question could (but doesn't have to) ask you to trace Dijkstra's algorithm.
- Given a weighted graph, find the minimum spanning tree of the graph. The question could (but doesn't have to) ask you to trace Kruskal's algorithm.

Example test problems (homework):

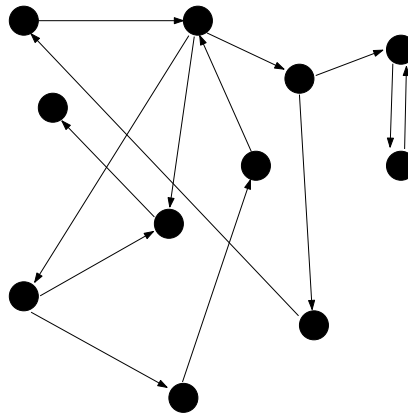
3.1 (due Oct 13) Give the adjacency matrix representation and the adjacency lists representation of the following graph:



3.2 (due Oct 13) Draw the graph with the following adjacency matrix:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

3.3 (due Oct 13) Find the strongly connected components of the following graph:



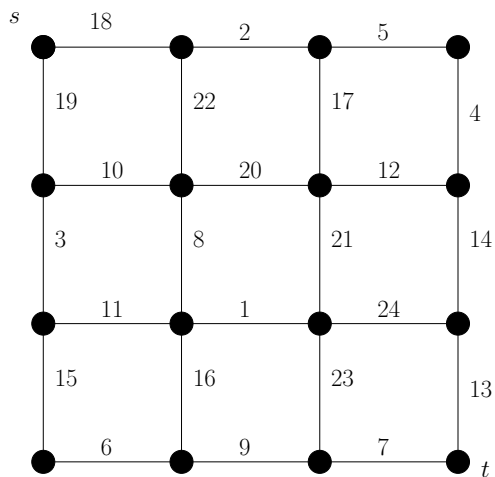
3.4 (due Oct 13) Find a topological sorting of the following directed acyclic graph $G = (V, E)$ where

$$V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\},$$

and

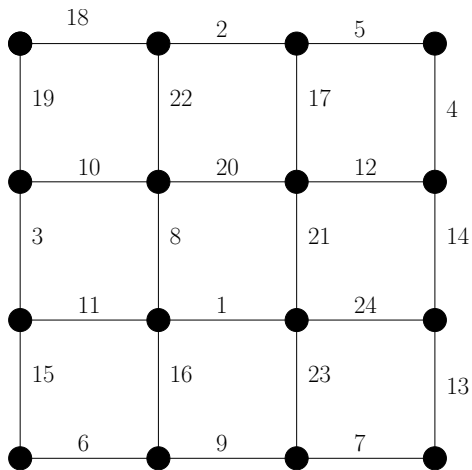
$$E = \{(7, 9), (9, 8), (6, 8), (10, 6), (2, 3), (3, 6), (1, 3), (4, 10), (5, 2), (7, 5)\}.$$

3.5 (due Oct 13) Find the shortest $s - t$ path in the following graph:



Use Dijkstra's algorithm. Show the content of the `dist` array after each step.

3.6 (due Oct 13) Find the minimum spanning tree of the following graph:



4 Additional homework

3.7 (due Oct 13) Let $G = (V, E)$ be a digraph given in the adjacency-list representation (i. e., for each vertex $v \in V$ we have a (linked) list of out-neighbors of v).

1. Write pseudocode for a procedure which outputs an adjacency-list representation of the reverse digraph (i. e., G with each edge reversed). The procedure should run in time $O(|V| + |E|)$.
2. Write pseudocode for a procedure which outputs the adjacency-list representation of G in which the out-neighbors of each vertex are listed in the increasing order. The procedure should run in time $O(|V| + |E|)$.
3. Write pseudocode for a procedure which checks if G is undirected (i. e., the reverse of every $e \in E$ is also in E). The procedure should run in time $O(|V| + |E|)$.

Definitions needed for problems 3.10 and 3.14 below:

Let $G = (V, E)$ be an undirected graph. A *matching* of G is a set of edges $E' \subseteq E$ such that every vertex of G is in *at most one* edge of E' . A *perfect matching* of G is a set of edges $E' \subseteq E$ such that every vertex of G is in *exactly one* edge of E' . A matching E' of $G = (V, E)$ is *maximal* if there does not exist $e \in E \setminus E'$ such that $E' \cup \{e\}$ is a matching. A matching E' of $G = (V, E)$ is *maximum* if it has the largest cardinality (that is, the number of elements) among the matchings of G . To understand the definitions, try to answer the following two questions (do not turn in):

- Is every maximum matching maximal matching?
- Is every maximal matching maximum matching?

Let $G = (V, E, w)$ be an undirected graph. Let $E' \subseteq E$ be a matching of G . The weight of E' is

$$\sum_{e \in E'} w(e).$$

We will consider the following two problems:

MAX-WEIGHT MATCHING:

INPUT: weighted graph $G = (V, E, w)$

OUTPUT: a matching $E' \subseteq E$ of G with maximum weight.

MAX-WEIGHT PERFECT MATCHING:

INPUT: weighted graph $G = (V, E, w)$

OUTPUT: a perfect matching $E' \subseteq E$ of G with maximum weight.

3.8 (due Oct 13) Consider the following algorithm to find a maximal matching of a given input graph $G = (V, E)$:

- $E' \leftarrow \{\}; V' \leftarrow \{\};$
- For each edge $\{u, v\} \in E$ do
 - if $u \notin V'$ and $v \notin V'$ then $E' \leftarrow E' \cup \{\{u, v\}\}; V' \leftarrow V' \cup \{u, v\}$

Give an example of a graph for which the algorithm does not find a maximum matching. (The algorithm is a little underspecified — it does not specify the order in which the edges are considered. You can pick any order.)

3.9 (due Oct 13) Prove that the algorithm from the previous problem always returns a matching whose size is at least half of the maximum.

3.10 (due Oct 13) Suppose that you have an algorithm (let's call it A) which solves the MAX-WEIGHT MATCHING problem on any weighed graph $G = (V, E, w)$ in time $O(E + V)$. Show how you can use algorithm A to solve the MAX-WEIGHT PERFECT MATCHING problem in time $O(E + V)$. (HINT: consider the weighted graph $G' = (V, E, w')$ where $w(e) = w'(e) + C$ where C is some large number.)

3.11 (due Oct 20) Let $G = (V, E)$ be an undirected graph. A 2-coloring of G is an assignment $\phi : V \rightarrow \{1, 2\}$ (that is, each vertex gets one of two colors) such that for all $\{u, v\} \in E$ we have $\phi(u) \neq \phi(v)$ (that is, endpoints of each edge get different colors). Give an $O(m + n)$ algorithm which finds either a 2-coloring of a graph or an odd cycle.

3.12 (due Oct 20) Let $G = (V, E)$ be an undirected graph. Give an $O(m + n)$ algorithm that decides whether G contains a cycle and if it does then it outputs one (the output should be a sequence of vertices in the order they occur on the cycle).

3.13 (due Oct 20) Show that if we exchange the lines “for $k = 1$ to n :” and “for $i = 1$ to n :” of Floyd-Warshall algorithm (DPV (draft), p. 188) the resulting algorithm does not correctly compute all-pairs-shortest paths. Give an example graph for which the modified algorithm fails.

3.14 (due Oct 20)[BONUS PROBLEM] Suppose that you have an algorithm (let's call it B) which solves the MAX-WEIGHT PERFECT MATCHING problem on any weighed graph $G = (V, E, w)$ in time $O(E + V)$. Show how you can use the algorithm B to solve the MAX-WEIGHT MATCHING problem in time $O(E + V)$.

3.15 (due Oct 20)[BONUS PROBLEM] The *max-weight* of a spanning tree T is the maximum weight of an edge of T . A *min-max-weight spanning tree* is a spanning tree with the minimum max-weight. Give $O(E + V)$ algorithm which finds min-max-weight spanning tree of a given input graph G .

5 Additional problems from the book

Try to solve the following problems. A few of them will be on the quiz. We will go over the ones that you choose in the problem sessions.

- 3.1, 3.2, 3.3, 3.4, 3.6, 3.7, 3.8, 3.9, 3.11, 3.12, 3.13, 3.15, 3.18, 3.19, 3.21, 3.22, 3.23, 3.24, 3.25, 3.26, 3.27,
- 4.1, 4.2, 4.3, 4.4, 4.5, 4.7, 4.8, 4.10, 4.11, 4.13, 4.15, 4.17, 4.18, 4.19, 4.20, 4.21,
- 5.1, 5.2, 5.3, 5.7.