Algorithmic Discrete Mathematics
3. Exercise Sheet

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics
Andreas Paffenholz
Silke Horn

SS 2013
15/16 May 2013

Groupwork

Exercise G1

A *spanning forest* for a graph $G = (V, E)$ with $c(G)$ connected components is a forest $T = (V, F)$ with $F \subseteq E$ and $|F| = |V| - c(G)$. In particular, a spanning forest of a connected graph is a spanning tree.

Generalize the breadth-first-search algorithm so that it computes a spanning forest of a not necessarily connected graph. Also determine the running time.

Exercise G2

Reconsider the depth-first-search algorithm presented in the lecture:

Algorithm 2: Depth-First-Search (DFS)

Input: graph $G = (V, E)$, $V = \{1, \dots, n\}$ given as adjacency list
Output: predecessor function $\text{pred} : V \to V \cup \{0\}$

foreach $v \in V$ **do**
  $\text{pred}(v) \leftarrow 0$
  $\text{seen}(v) \leftarrow 0$
foreach $v \in V$ **do**
  if $\text{seen}(v) = 0$ **then**
    DFSvisit(G, v)

Function DFSvisit(G,r)

Input: graph $G = (V, E)$ given as adjacency list, root node $r \in V$

$\text{seen}(r) \leftarrow 1$
foreach $v \in \text{Adj}(r)$ **do**
  if $\text{seen}(v) = 0$ **then**
    $\text{pred}(v) = r$
    DFSvisit(G, v)

Show that the algorithm correctly computes a spanning forest and determine its running time.

Exercise G3

Recall Kruskal's algorithm:

Algorithm 3: Kruskal's Algorithm

Input: graph $G = (V, E)$, weight function $w : E \to \mathbb{R}$
Output: Minimal spanning tree $T = (V, F)$ of G

1 $F \leftarrow \emptyset$
2 $L \leftarrow E$
3 Sort the edges in L increasingly by weight
4 **while** $L \neq \emptyset$ **do**
5   $e \leftarrow \text{pop\_front}(L)$
6   **if** $(V, F \cup \{e\})$ *is acyclic* **then**
7     $F \leftarrow F \cup \{e\}$

The goal of this exercise is to show that the loop in lines 4–7 can be implemented so that it runs in time $\mathcal{O}(m \log n)$.

To this end, we have to verify whether inserting the edge e in step 6 encloses a cycle. We will at each step keep track of the connected components of the forest. We define a function $\mathsf{find}: V \to V$ that maps a vertex to some unique representative of its connected component. It then suffices to check whether the two endpoints of $e = \{u, v\}$ are in the same component, *i.e.,* e encloses a cycle if and only if $\mathsf{find}(u) = \mathsf{find}(v)$.

If we add e to the forest, we have to form the union of the two connected components containing u and v. To this end, we need a function union that forms the union.

(a) Describe an easy $\mathcal{O}(n)$ implementation of find and union.

(b) We can do faster if we arrange the elements of each connected component in a rooted tree with the representative in the root. Describe the details of such an implementation and show that $\mathsf{find}(v)$ and $\mathsf{union}(u, v)$ run in $\mathcal{O}(\log n)$ time.

(c) Conclude that the loop in lines 4–7 runs in time $\mathcal{O}(m \log n)$.

(d) Can you think of even more improvements?

Exercise G4

Let $G = (V, E)$ be a d-*regular* graph on n vertices, *i.e.,* each vertex $v \in V$ has degree d. Show that the total number of triangles in G and $\overline{G}$ equals $\binom{n}{3} - \frac{n}{2}d(n - d - 1)$. (Recall that the complementary graph $\overline{G}$ of G is defined as $\overline{G} = (V, \binom{V}{2} \setminus E)$.)

Homework

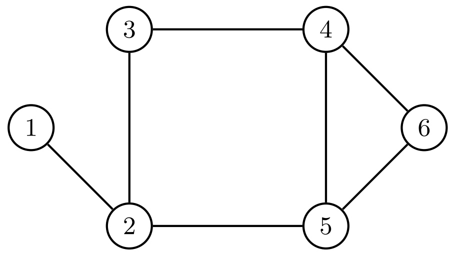
Exercise H1 (5 points)

Perform

(a) the BFS algorithm and

(b) the DFS algorithm

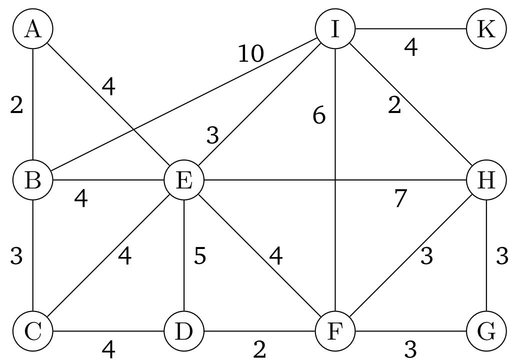
on the following graph with root node $s = 1$:



Always go through the vertices in the adjacency list in increasing order. Determine the values of pred, seen and L (only for BFS) in each step. Moreover, give the spanning tree that is constructed.

Exercise H2 (5 points)

(a) Perform Kruskal's algorithm on the following graph:



You can use the template on the website for the drawings of the graph.

(b) Prove: If the weights of the edges are pairwise distinct then the minimal spanning tree is unique.

Exercise H3 (5 points)
Let T be a minimal spanning tree in a graph $G = (V, E)$.

(a) Let $\{i, j\} \in E$. Describe an algorithm that finds a minimal spanning tree in the graph $G_1 = (V, E \setminus \{\{i, j\}\})$ obtained by deleting the edge $\{i, j\}$.

(b) Let $\{k, \ell\} \notin E$. Describe an algorithm that finds a minimal spanning tree in the new graph $G_2 = (V, E \cup \{\{k, \ell\}\})$ obtained by adding the edge $\{k, \ell\}$.

In both cases show that your algorithm is correct and determine its running time.

Exercise H4 (5 points)
A *tournament* $T = (V, A)$ is a directed graph in which there is exactly one edge between any two vertices. Show that in every tournament there is a vertex v such that there is a path of length ≤ 2 from v to any other vertex.