UNIVERSITY OF OSLO

INF2220 - Summary

Ivar Haugaløkken Stangeby

November 12, 2014

CONTENTS

1 GRAPHS

1.1 DEFINITIONS

Definition (Connected graph)**.** An undirected graph is called **connected** if there is a path from any vertex to any other vertex in the graph. If the graph is directed, we call it **strongly connected**.

Definition (Biconnected graph)**.** A **biconnected** graph is a connected graph with the property that if any vertex were to be removed, the graph will remain connected.

Definition (Minimum spanning tree)**.** Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. We can also assign a weight to each edge and use it to assign a weight to a spanning tree. A **minimum spanning tree** (MST) is a spanning tree with weight less than or equal to the weight of every other spanning tree.

Definition (Acyclic graph)**.** An **acyclic graph** is a directed graph that contains no cycles

Definition (Topological sort)**.** A **topological sort** is an ordering of vertices in a directed acyclic graph, such that if there is a path from v_i to v_j, then v_j appears *after* v_i in the ordering.

1.2 Representation of graphs

Adjacency matrix There are several ways to represent a graph. The most simple way is called an **adjacency matrix**. The adjacency matrix is a two-dimensional array where for each edge (u, v) we set A$[u][v]$ to **true**. This representations is good for dense graphs. This means that there are a lot of edges in relation to the number of vertices.The space requirement is $\Theta\left(|V|^2\right)$

Adjacency List If the graph on the other hand is not dense, an **adjacency list** representation is a better option. For each vertex, keep a list of the adjacent vertices. The space requirement is then $O(|E|+|V|)$.

1.3 Graph algorithms

1.3.1 Dijkstra's algorithm

Dijkstra's algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs. For a given source vertex in the graph, the algorithm finds the path with the lowest cost between that vertex, and any other vertex. In essence, the algorithm picks the unvisted vertex with the lowest distance, calculates the distance through it to each unvisited neighbor and updates the neighbor's distance if it is smaller. Mark the vertex as visited when all the neighbors have been checked.
It has a worst case performance of $O\left(|E|+|V|\log|V|\right)$

1. Set starting point distance to 0, and ∞ for the rest.
2. Mark all nodes as unvisited, set initial node to current-node. Fill a list with all the unvisited nodes.
3. For current node - consider all of its unvisited neighbors and calculate their tentative-distance = currentnodedistance + cost of edge traversal.
4. When done considering all the neighbors of the current, mark current as visited and remove it from unvisited-list.
5. If the destination node has been marked visited, or if the smallest tentative distance among the nodes in unvisited set is ∞, then stop. The algorithm has finished.
6. Select the unvisited node with the smallest tentative distance, and set it as new current-node and repeat from step 3.

1.3.2 PRIM'S ALGORITHM

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph.
Its complexity relies entirely on the data structure used. Using an adjacency matrix the algorithm has a worst case performance of $O\left(|V|^2\right)$.

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph
2. Find the minimum weight edge from a vertex in the tree to a vertex not in the tree and transfer it to the tree.
3. Repeat step 2 - until all vertices are in the tree.

1.3.3 KRUSKAL'S ALGORITHM

Kruskal's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted graph. If the graph is not connected, it finds a minimum spanning forest.

1. Create a forest F, where each vertex in the graph is a separate tree.
2. Create a set S, containing all the edges in the graph.
3. while S is nonempty and F is not yet spanning.
    a) remove an edge with minimum weight from S.
    b) if that edge connects two different trees, then add it to the forest, combining the two trees into a single tree.