Course: Telecommunication Network Design
Teacher: Piotr Chołda piotr.cholda@agh.edu.pl
Studies: Electronics and Telecommunications
Speciality: Networks and Services
Semester: 2^{nd} sem. MSc stud., Fall

Draft of the lecture

Piotr Chołda

October 12, 2017

1 Algorithms Defined on Graphs

1.1 Combinatorial optimization for network programming

1. The notions of network programming and combinatorial optimization.

2. Problem of searching for minimum spanning tree (MST):

   - Kruskal's algorithm:

     procedure KRUSKAL($G = (V, E, f)$)
     $\triangleright$ Output edges of MST: T
         $T \leftarrow \emptyset$
         $\mathcal{E} = E$
         while $|T| < |V| - 1$ **do**
             $e \leftarrow \arg\min_{i \in \mathcal{E}} \{f(i)\}$
             $\triangleright$ e: the 'lightest' edge in a set of edges not dealt with before
             $\mathcal{E} \leftarrow \mathcal{E} \setminus \{e\}$
             if $T \cup \{e\}$ does not contain a cycle **then**
                 $T \leftarrow T \cup \{e\}$
             end if
             $\mathcal{E} \leftarrow \mathcal{E} \setminus \{e\}$
         end while
     end procedure

   - Prim's algorithm (also known as Prim-Dijkstra's algorithm):

     procedure PRIM($G = (V, E, f)$, $r \in V$)
     $\triangleright$ Output edges of MST: T
     $\triangleright$ r: the root
         $T \leftarrow \{r\}$
         $\mathcal{V} = V$
         while $\mathcal{V} \neq \emptyset$ **do**
             $\mathcal{L} = \{i \in E : i = \{t, v\}, t \in T, v \in \mathcal{V}\}$
             $e \leftarrow \arg\min_{i \in \mathcal{L}} \{f(i)\}$
             $\triangleright$ e: the 'lightest' edge in a set of edges that can extend the tree
             $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v \in \mathcal{V} : e = \{t, v\}, t \in T\}$
             $T \leftarrow T \cup \{e\}$
         end while

end procedure

3. Greedy algorithms.

4. Examples of ‘difficult’ network programming problems:

   - Steiner’s problem;
   - (vertex) coloring in graphs; the four color theorem, planar graphs.

5. Breadth-First Search (BFS), a tree of the hop-based shortest paths:

procedure BFS($G = (V, A)$, $r \in V$)
▷ r: the root
▷ Initialization:
    $\mathcal{S} \leftarrow \{r\}$
▷ $\mathcal{S}$: set of vertices reachable (via directed path/s) from r
    $\mathcal{L} \leftarrow (r)$
▷ $\mathcal{L}$: an ordered list of already found vertices
    $\mathcal{L}' \leftarrow V \smallsetminus \{r\}$
▷ $\mathcal{L}'$: set of not yet searched vertices
    $\mathit{predecessor}(r) = 0$
▷ $\mathit{predecessor}(j) = k$: means that vertex k is a predecessor of vertex j at a directed path from root r
▷ The root does not have a predecessor
▷ Main loop:
    while $\mathcal{L} \neq \emptyset$ **do**
        for all $k \in \mathcal{L}$ **do**
            for all $j \in \mathcal{L}'$ **do**
                if $(k, j) \in A$ **then**
▷ Only for admissible arcs
                    $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
                    $\mathit{predecessor}(j) = k$
                    $\mathcal{L} \leftarrow (\mathcal{L}, j)$
                    $\mathcal{L}' \leftarrow \mathcal{L}' \smallsetminus \{j\}$
                end if
            end for
            $\mathcal{L} \leftarrow \mathcal{L} \smallsetminus \{k\}$
        end for
    end while
    return $(\mathcal{S}, \mathcal{P}(\mathcal{S}))$
▷ $\mathcal{P}(\mathcal{S})$: list of the predecessor of vertices contained by $\mathcal{S}$
end procedure

6. Depth-First Search (DFS):

procedure DFS($G = (V, A)$, $r \in V$)
▷ r: root
▷ Initialization:
    $\mathcal{S} \leftarrow \{r\}$
▷ $\mathcal{S}$: set of vertices reachable (via directed path/s) from i
    $\mathcal{L}' \leftarrow V \smallsetminus \{r\}$
▷ $\mathcal{L}'$: set of not yet searched vertices
    $\mathit{predecessor}(r) = 0$

$\triangleright$ Main loop:
    SearchDeep(r,G,$\mathcal{S}$,$\mathcal{L}'$)
    return ($\mathcal{S}$,$\mathcal{P}(\mathcal{S})$)
end procedure

A subprocedure performed recurrently:

procedure SearchDeep(v,G,$\mathcal{S}$,$\mathcal{L}'$)
    for all $j \in \mathcal{L}'$ **do**
        if $(v, j) \in A$ **then**
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
            $predecessor(j) = v$
            $\mathcal{L}' \leftarrow \mathcal{L}' \setminus \{j\}$
            SearchDeep(j,G,$\mathcal{S}$,$\mathcal{L}'$)
        end if
    end for
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{v\}$
end procedure

7. Maximum flow problem (`max flow`). Ford-Fulkerson's theorem. Algorithm for solving max flow problem based on usage of a residual graph and augmenting flows.

1.2 Exercises

- Give example of a weighted graph G, that has all the following properties:
  - ⋆ graph G is connected,
  - ⋆ all the weights are natural numbers,
  - ⋆ there are nine vertices in graph G,
  - ⋆ the sum of weights of the minimum spanning tree of graph G is equal to the half of the sum of all the weights in this graph.
- Show that the minimum spanning tree of a weighted full mesh graph K_9 is a bipartite graph.

1.3 Reading

1.3.1 Contents of the lecture

Problems described in this lecture are generally dealt with in the following positions:

- Wayne D. Grover. *Mesh-Based Survivable Networks. Options and Strategies for Optical, MPLS, SONET, and ATM Networks.* Prentice Hall PTR, Upper Saddle River, NJ, 2004: section 4.10.
- Deepankar Medhi and Karthikeyan Ramasamy. *Network Routing. Algorithms, Protocols, and Architectures.* Morgan Kaufmann Publishers—Elsevier, San Francisco, CA, 2007: chapter 2.
- Michał Pióro and Deepankar Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers—Elsevier, San Francisco, CA, 2004: appendix C.1-C.2.

1.3.2 Auxiliary references

- Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Minimum Cost Flow Problem. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2095–2108. Springer Science+Business Media, LLC., New York, NY, 2009: minimum cost flow problem.

- Ramesh Bhandari. *Survivable Networks. Algorithms for Diverse Routing.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999: overview of various algorithms useful in network design (mainly for resilient networks).