UBINET, Master 2 IFI Algorithms for telecommunications

Exam, November 2016

1 hour

Course and manuscript notes are allowed. No computers, cellphones, books.

Instruction and comments: the points awarded for your answer will be based on the correctness of your answer as well as the clarity of the main steps in your reasoning. All proposed solutions must be proved. **All the exercises are independent.** *The points are indicated so you may adapt your effort.*

1 Algorithmics on graphs

Exercise 1 (Flow: 4 points, 15 minutes) Consider the elementary network flow depicted in Figure 1 (left) and the initial flow f from s to t in Figure 1 (right).

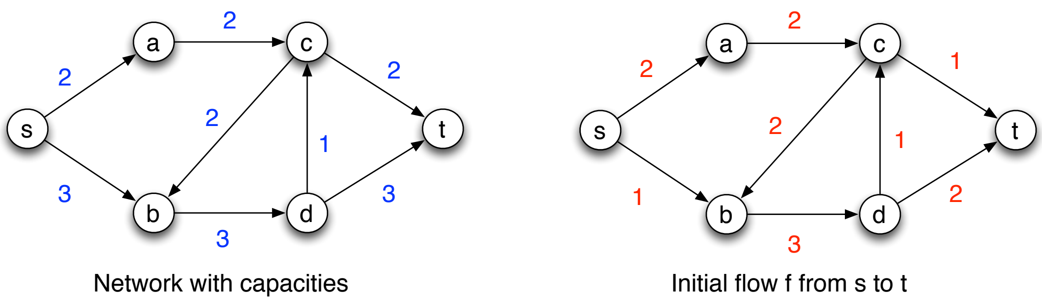

Figure 1: Flow.

- What is the value of the inital flow?
- Apply the Ford-Fulkerson Algorithm to N starting from the flow f. The auxiliary digraphs built during the execution of the algorithm must be given.
- Give the flow and the cut obtained. Conclusion (recall the min cut-max flow theorem)?

Exercise 2 (Traveling Salesman Problem(TSP): 6 points, 45 minutes)

In this part, we are given a connected graph $G = (V, E)$ with a non-negative weight function $w : E \to \mathbb{R}^+$ on its edges. You may see it as a map where vertices are cities, edges are roads and weights are the lengths of the roads. The weight of a subgraph H of G is the sum of the weights of its edges, i.e., $w(H) = \sum_{e \in E(H)} w(e)$.

With no repetition of vertices. In the TSP, a Salesman starts from his home-town $v_0 \in V$, aims at traveling through each city exactly once, coming back to its home-town and minimizing the total length of his journey. More formally, the goal of the TSP is to find an Hamiltonian cycle in G with minimum weight, that is, a sequence $(v_0, \cdots, v_n)$ of the vertices such that:

- for any $0 \leq i < j \leq n$, $v_i \neq v_j$ (no vertex is repeated)
- every vertex belongs to the sequence
- for every $0 \leq i < n$, $e_i = \{v_i, v_{i+1}\} \in E$ (two consecutive vertices are adjacent)
- $e_n = \{v_n, v_0\} \in E$ (we come back to the starting vertex)
- $\sum_{0 \leq i \leq n} w(e_i)$ is minimum

Question 1. Give a example of a connected graph where no solution exists. That is, a graph with no Hamiltonian cycle.

Question 2. Give an (exponential) algorithm that, given a graph, finds a minimum Hamiltonian cycle if it exists.

The problem of deciding if such cycle exists and to find a minimum one is NP-complete. Moreover, it cannot be well approximated (unless $P = NP$). Therefore, in what follows, we remove the constraint that every vertex (and edge) must be visited exactly once.

Repetitions of vertices and edges are allowed. Given a connected weighted graph $G = (V, E)$, $w : E \to \mathbb{R}^+$, we aim at computing a sequence $(v_0, \cdots, v_r)$ of vertices such that, every vertex is met at least once, for every $0 \leq i < r$, $e_i = \{v_i, v_{i+1}\} \in E$ (two consecutive vertices are adjacent), $e_r = \{v_r, v_0\} \in E$ (we come back to the starting vertex), and $\sum_{0 \leq i \leq r} w(e_i)$ is minimum. Such a sequence is called a *tour with minimum weight in G*.

Question 3. Give a tour (starting from v_0) with minimum weight on the example depicted in Figure 2. What is the weight of your solution?

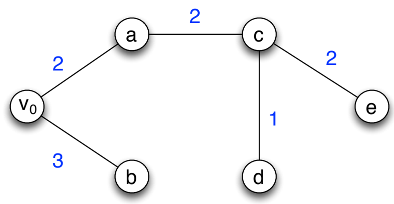

Figure 2: Example.

More generally, when the input graph is a tree (connected graph without cycles) $T = (V, E)$, give the weight of an optimal solution in function of the sum of the weights of all edges: $\sum_{e \in E} w(e)$.

Recall that, given a connected graph $G = (V, E)$, a spanning tree $T = (V', E')$ of G is a subgraph of G such that: T is a tree and T spans all vertices of G (i.e., $V' = V$). We also recall that, computing a spanning tree with minimum weight can be done in polynomial-time (e.g., using Kruskal algorithm).

Question 4. Let G be a connected weighted graph. Show that the minimum weight of a spanning tree in G is at most the minimum weight of a tour in G.

Hint: start from a tour C in G and show that there exists a spanning tree T with $w(T) \leq w(C)$.

Question 5. Give a 2-approximation algorithm for the problem of computing a tour with minimum weight in graphs. Prove that all properties of an approximation algorithm are well satisfied by your solution.