4th tutorial

Strings

1. Most Frequent Occurence of Length k. Find which substring of length k appears the most times as a substring of σ.

Flows

Refresher: The *reserve* of an edge uv is $r(uv) = (c(uv) - f(uv)) + f(vu)$. An edge e is *unsaturated* if $r(e) > 0$. A path P is *unsaturated* or *augmenting* if $\forall e \in P : r(e) > 0$.
The Ford-Fulkerson algorithm starts with $f \equiv 0$ (the flow over each edge is 0), and then, while an augmenting $s - t$ path P exists, it augments the current flow f as follows. Let $\epsilon = \min_{e \in P} r(e)$. For each edge $uv \in P$, let $\delta = \min\{\epsilon, f(vu)\}$, and set $f(vu) = f(vu) - \delta$ and $f(uv) = f(uv) + \epsilon - \delta$. This preserves the Kirchhoff's law, and increases $|f|$ by ϵ.

In all the tasks below, assume that if Ford-Fulkerson terminates, it returns the maximum flow (we will prove this in the next lecture).

2. Multiple Sources and Terminals. How could you use it to find the maximum flow when there are multiple sources and terminals?

3. Bad Net. Give an example of a small network in which the F-F algorithm may perform more than a million iterations. ("May perform" means there is a sequence of choices of unsaturated paths; you may adversarily choose these.)

4. Ford-Fulkerson with Unit Capacities. How many iterations does Ford-Fulkerson make if all capacities are 1?

5. Edge Disjoint Paths. Find an algorithm which finds the maximum number of edge disjoint paths between given two vertices $u, v \in V(G)$.

6. Vertex Disjoint Paths. Find and algorithm which computes the maximum number of vertex disjoint paths between given two vertices $u, v \in V(G)$.

7. Maximum Bipartite Matching. Design an algorithm which computes the maximum size matching in a bipartite graph $G = (V, E)$. A matching is a subset of edges $M \subseteq E$ such that no two edges overlap, i.e. $\forall e_1, e_2 \in M : e_1 \cap e_2 = \emptyset$.