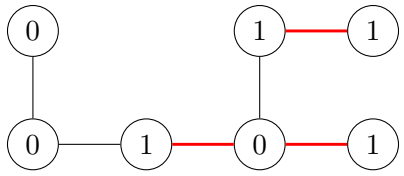
CSE421: Design and Analysis of Algorithms April 3th, 2024

Homework 2

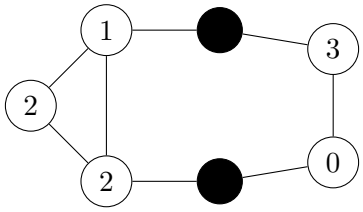
Shayan Oveis Gharan Due: April 10, 2024 at 11:59 PM

P1) Given a tree T (with $n \geq 2$ vertices). Suppose we have written 0 or 1 on each vertex of T such that the sum of all numbers is an even number. Prove that it is possible to choose a subset F of the edges of T such that every vertex with label 0 is adjacent to an even number of edges of F and every vertex with label 1 is adjacent to an odd number of edges of F. For example, given the tree below you can let F be the set of edges colored red.

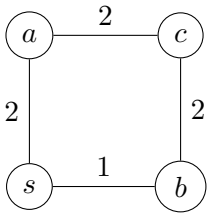


P2) Given a graph $G = (V, E)$, design a polynomial time algorithm to partition edges of G into edge disjoint cycles and output the cycles. If no such partitioning is possible, output "Impossible". For example, give the graph on the left you may output the two cycles in red and blue and given the graph on the right you should output "Impossible".

P3) You are given a connected undirected graph G with n vertices and m edges where some of the vertices are "dead". That means that you cannot travel over them. Every "alive" vertex v has a value, $c_v \geq 0$. You want to start from an alive vertex of G and travel around (while **not** going over any dead vertex) and collect the largest possible value. Note that you can visit a vertex multiple times (but you get the reward only once). If you travel to $v_1, v_2, \ldots, v_k$ your score will be $c_{v_1} + \cdots + c_{v_k}$ Design an $O(m+n)$-time algorithm for this task. Your algorithm should output the largest collectible value. For example in the following graph the dead vertices are colored in black and the score of every alive vertex is written on it. The largest collectible value is 5.



P4) Given a weighted connected graph $G = (V, E)$ with $m = |E|$ edges and $n = |V|$ vertices where every edge $e \in E$ has a weight $w_e \in \{1, 2\}$. Furthermore you are given a pair of vertices s, t. For a path $s = v_0, v_1, \ldots, v_k = t$ from s to a vertex t define its weight to be $w_{(v_0,v_1)} \cdot w_{(v_1,v_2)} \ldots w_{(v_{k-1},v_k)}$, i.e., instead of taking the sum of weights we take their product. Design an $O(m+n)$-time algorithm to output the length of the smallest weight path from s to t. You will receive 18 points (out of 20) if your algorithm runs in polynomial time (as opposed to $O(m + n)$). For example, in the above picture the length of the shortest path from s to b is 1 and the length of the shortest paths from s to each of a and c is 2.



P5) **Extra Credit:** Prove that we can color the edges of every graph G with two colors (red and blue) such that, for every vertex v, the number of red edges touching v and the number of blue edges touching v differ by at most 2.