CS 6230: Mid Term Examination

Spring 2017

For all problems, at least state the basic idea for your approach. This should be possible in 1-2 sentences for all problems. **Then** follow up with pseudocode. You can assume rank r, size p are known.

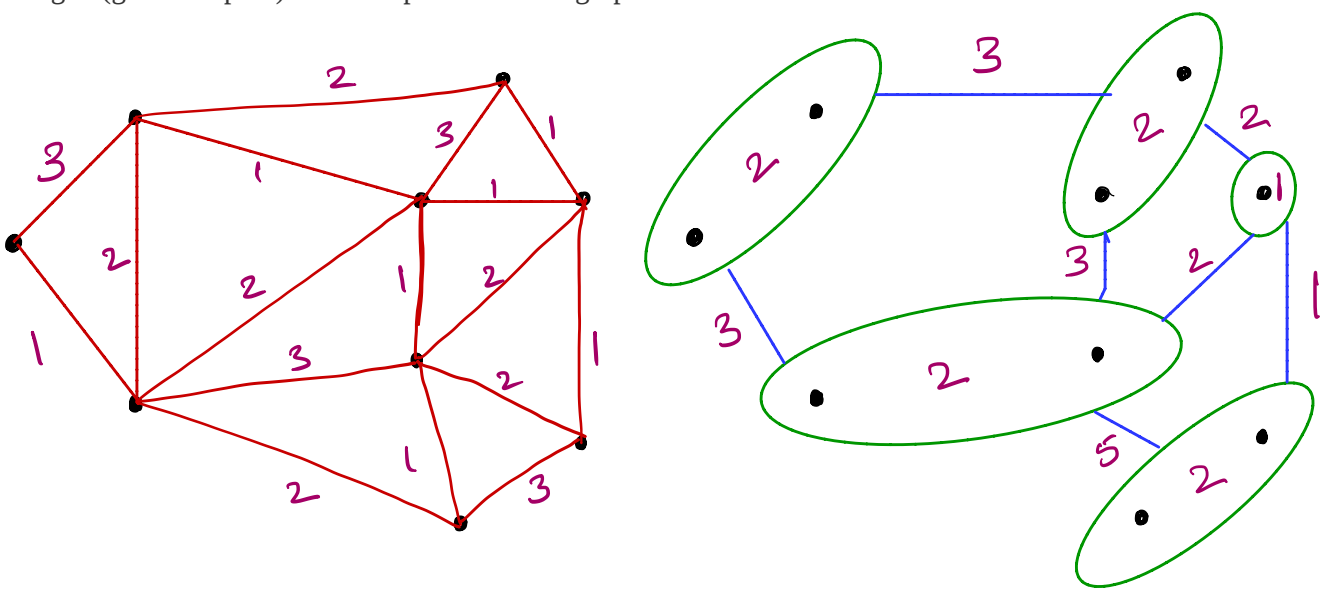
Q1

Let X be an arbitrary sequence of elements from a linearly ordered set S, and let $y \in S$. Suppose that we want to determine all the occurrences, if any, of y in X. Develop an $\mathcal{O}(\log n)$ time EREW PRAM algorithm to solve this problem, using a linear number of operations.

Q2

The Graph Partitioning approaches we discussed in class can be accelerated by using a multi-level approach. We will replace the problem of partitioning the original graph $G_0 = (N_0, E_0)$ by the simpler problem of partitioning a *coarse approximation* $G_1 = (N_1, E_1)$ to G_0. Given a partitioning of G_1, we will use that to get a starting guess for a partitioning of G_0, and refine it by an iterative process, like Kernighan-Lin. The problem of partitioning G_1 (or more generally G_i) is solved recursively, by approximating it by a yet coarser graph G_2 (or G_{i+1}).

The key question is on how to produce the coarse graph. This is done by collapsing edges to "merge vertices" to produce the coarse vertex. Edges exist between coarse vertices if an edge existed between any of its constituent vertices. When multiple edges exist, the edge weight of the coarse edge is the sum of all fine edges. A simple example is shown below: on the left is the fine graph, we identify vertices that can be merged (green ellipses) and the updated coarse graph.



How would you figure out which vertices to merge? Note that we do not want to merge 3 vertices together. Give a parallel algorithm to compute the set of all vertices to be merged. Give the expected complexity of your algorithm (without proof) if known.

Q3

Suppose we are given a set of n elements stored in an array A together with an array L such that $L(i) \in 1, 2, \ldots, k$ represents the label of element $A(i)$, where k is a constant. Develop an optimal $\mathcal{O}(\log n)$ time EREW PRAM algorithm that stores all the elements of A with label 1 into the upper part of A while preserving their initial ordering, followed by the elements labeled 2 with the same initial ordering and so on. For example,

$$\begin{aligned} A &= [6, \quad 5, \quad 3, \quad 9, \quad 11, \quad 12, \quad 8, \quad 17, \quad 21, \quad 2] \\ L &= [1, \quad 1, \quad 2, \quad 3, \quad 2, \quad 1, \quad 1, \quad 2, \quad 3, \quad 3] \end{aligned}$$

produces $A = [6, 5, 12, 8, 3, 11, 17, 9, 21, 2]$.

Q4

Let $p(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n$ be a given polynomial. *Horner's algorithm* to compute $p(x)$ at a point x_0 is based on rewriting the expression for $p(x_0)$ as follows:

$$p(x0) = (\cdots((a0x0 + a1)x0 + a2)x0 + \cdots + an - 1)x_0 + a_n.$$

An obvious sequential algorithm for this problem has $\mathcal{O}(n)$ complexity. Is it possible to develop a **work-optimal** parallel algorithm whose complexity is $\mathcal{O}(n/p + \log n)$ for p processors. Give pseudocode for the best possible parallel algorithm.