CMPT 706, Spring 2020
Take Home Final Exam

Due Date: April 14, 2020, 23:59.
Submit your solutions to Coursys in pdf format.

Type your solutions using Word or Latex.
Poor quality photos will not be accepted.

Explain all your answers! In each question explain the algorithm and prove the bound on the runtime.

Question 1 (9 points)

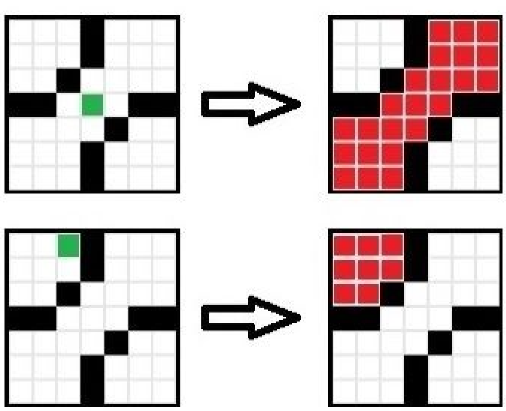
Write a parallel algorithm that gets two integers each N-bit long, and computes their product using at most $O(N^{1.7})$ processors in $O(N)$ parallel time.

Question 2 (9 points)

Write an algorithm that gets an NxN matrix of colors, and a starting point, and colors the WHITE area connected to the starting point with RED.
The algorithm must run in time $O(N^2)$.

See examples below:

The *green point* on the left represents the starting point.
The *red points* are the area containing the starting point.



Question 3 (9 points)

A *feedback edge set* of an undirected graph G = (V,E) is a subset of edges E' $\subseteq$E that intersects every cycle of the graph. That is, removing the edges E' from G makes the graph acyclic.

Design a polynomial time algorithm that gets an undirected graph G = (V,E) with weights on edges w :E $\rightarrow$ $R_{>0}$, and outputs a feedback edge set of G of minimal weight, i.e., a *feedback edge set* that minimizes the sum of the weights of all edges in the set.

What is the runtime of your algorithm? Explain your answer

Question 4 (9 points)

Given an undirected graph G = (V,E), an *independent set* in G is a subset vertices I$\subseteq$V such that no two vertices in I share an edge.
[4 points] Prove that a set of vertices S$\subseteq$V is a vertex cover if and only if V\S is an independent set.
[5 points] Design a polynomial time algorithm that gets a graph G on N vertices and has the guarantee that if G contains an independent set of size 0.6N, then the algorithm returns an independent set in G of size at least 0.2N.
[Hint: use the polytime 2-approximation algorithm for vertex cover we saw in class]

Question 5 (9 points)

Write a linear time algorithm for the following problem.
The input is an array A of n integers, and the goal is to find 3 indices i<j<k such that A[i] - A[j] + A[k] is maximized.
The algorithm must run in time O(n).

For example if the array is A = [2, 8, 2, 6, 4, 1, 9, 3, 10], then the output should be 17, because we can take 8 -1+10 = 17.

[There is a trivial O(n^3) algorithm that tries all triples of i<j<k. However, you need an algorithm that runs in O(n) time. Hint: use dynamic programming]`