CS 170 HW 6

Due **2021-03-02, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, write “none”.

2 2-SAT

In the 2SAT problem, you are given a set of clauses, where each clause is the disjunction (OR) of two literals (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value true or false to each of the variables so that all clauses are satisfied that is, there is at least one true literal in each clause. For example, heres an instance of 2SAT:

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$$

This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true, false, false, and true`, respectively.

The purpose of this problem is to lead you to a way of solving 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from from the negation of α to β, and one from the negation of β to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to either of the implications $\overline{\alpha} \implies \beta$ or $\overline{\beta} \implies \alpha$. In this sense, G_I records all implications in I.

(a) Show that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.

(b) Now show the converse of (a): namely, that if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable. (*Hint*: Assign values to the variables as follows: repeatedly pick a sink strongly connected component of G_I . Assign value `true` to all literals in the sink, assign `false` to their negations, and delete all of these. Show that this ends up discovering a satisfying assignment.)

(c) Conclude that there is a linear-time algorithm for solving 2SAT.

3 Perfect Matching on Trees

A *perfect matching* in an undirected graph $G = (V, E)$ is a set of edges $E' \subseteq E$ such that for every vertex $v \in V$, there is exactly one edge in E' which is incident to v.

Give an algorithm which finds a perfect matching *in a tree*, or reports that no such matching exists. Describe your algorithm, prove that it is correct and analyse its running time.

4 Huffman Coding

In this question we will consider how much Huffman coding can compress a file F of m characters taken from an alphabet of $n = 2^k$ characters x_0, x_1, ... , x_{n-1} (each character appears at least once).

(a) Let $S(F)$ represent the number of bits it takes to store F without using Huffman coding (i.e., using the same number of bits for each character). Represent $S(F)$ in terms of m and n.

(b) Let $H(F)$ represent the number of bits used in the optimal Huffman coding of F. We define the *efficiency* $E(F)$ of a Huffman coding on F as $E(F) := S(F)/H(F)$. For each m and n describe a file F for which $E(F)$ is as small as possible.

(c) For each m and n describe a file F for which $E(F)$ is as large as possible. How does the largest possible efficiency increase as a function of n? Give you answer in big-O notation.

5 Minimum Spanning k-Forest

Given a graph $G(V, E)$ with nonnegative weights, a spanning k-forest is a cycle-free collection of edges $F \subseteq E$ such that the graph with the same vertices as G but only the edges in F has k connected components. For example, consider the graph $G(V, E)$ with vertices $V = \{A, B, C, D, E\}$ and all possible edges. One spanning 2-forest of this graph is $F = \{(A, C), (B, D), (D, E)\}$, because the graph with vertices V and edges F has components $\{A, C\}, \{B, D, E\}$.

The minimum spanning k-forest is defined as the spanning k-forest with the minimum total edge weight. (Note that when $k = 1$, this is equivalent to the minimum spanning tree). In this problem, you will design an algorithm to find the minimum spanning k-forest. For simplicity, you may assume that all edges in G have distinct weights.

(a) Define a j-partition of a graph G to be a partition of the vertices V into j (non-empty) sets. That is, a j-partition is a list of j sets of vertices $\Pi = \{S_1, S_2 \ldots S_j\}$ such that every S_i includes at least one vertex, and every vertex in G appears in exactly one S_i. For example, if the vertices of the graph are $\{A, B, C, D, E\}$, one 3-partition is to split the vertices into the sets $\Pi = \{\{A, B\}, \{C\}, \{D, E\}\}$.

Define an edge (u, v) to be crossing a j-partition $\Pi = \{S_1, S_2 \ldots S_j\}$ if the set in Π containing u and the set in Π containing v are different sets. For example, for the 3-partition $\Pi = \{\{A, B\}, \{C\}, \{D, E\}\}$, an edge from A to C would cross Π.

Show that for any j-partition Π of a graph G, if $j > k$ then the lightest edge crossing Π must be in the minimum spanning k-forest of G.

(b) Give an efficient algorithm for finding the minimum spanning k-forest.

Please give a 3-part solution.