ASSIGNMENT 5

DUE: Monday, July 29, 6 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read http://www.student.cs.uwaterloo.ca/~cs341 for general instructions and policies. Also read Assignment section of course outline for clarification of what “justify” means.

Note: All logarithms are base 2 (i.e., log x is defined as \( \log_2 x \)).

Note: For all algorithm design questions, you must give the algorithm, argue the correctness, and analyze time complexity.

1 Warmup... Decision vs Optimization...

1.1. [5 marks] Fair split. Suppose you have a polynomial time algorithm for the FairSplit decision problem: given a list of \( n \) integers, \( a_1, a_2, \ldots, a_n \), indexed by \( S = \{1, \ldots, n\} \), is there a partition \( S = A \cup B \) with \( A \cap B = \emptyset \) such that \( \sum_{i \in A} a_i = \sum_{i \in B} a_i \).

Show that you can use this algorithm to find such a partition \( A, B \) (if it exists) in polynomial time. If the FairSplit algorithm runs in time \( O(n^p) \), give a bound on the run time of your algorithm of finding a fair partition.

1.2. [7 marks] Satisfiability. Recall that a literal is a variable \( x_i \) or the negation of a variable \( \neg x_i \). Consider the following variant of satisfiability problem: Max2-SAT.

Input: a number \( k > 0 \), a set of \( n \) Boolean variables, \( x_1, x_2, \ldots, x_n \) and a set \( C \) of \( m \) clauses, where each clause has the form \( (l_i \lor l_j) \) where \( l_i \) and \( l_j \) are literals.

Question: is there an assignment of truth-values to the variables that makes at least \( k \) of the clauses true?

Suppose you have a polynomial time algorithm for the above Max2-SAT decision problem. Show that you can use this algorithm to find the maximum number of clauses that can be made true, and to find a truth-value assignment that satisfies that number of clauses, both in polynomial time.

2 P, NP ...

2.1. [3 marks] Fair split is in NP. Show that FairSplit \( \in \text{NP} \). Be clear about your certificate and about the details of your verification algorithm and its run-time.

2.2 [3 marks] Max2-SAT \( \in \text{NP} \). Show that Max2-SAT \( \in \text{NP} \). Be clear about your certificate and about the details of your verification algorithm and its run-time.

2.3. [4 marks] In the Clique4 problem, we are given a graph \( G = (V, E) \) with maximum degree 4 and a positive integer \( k \); we must determine if \( G \) has a clique of size at least \( k \) or not. (A graph \( G \) has maximum degree \( d \) if every vertex in \( G \) is incident to at most \( d \) edges.)

Prove that Clique4 \( \in \text{P} \).
3 NPC ...

3.1. [7 marks] Prove that **FairSplit** ∈ NPC.

3.2. [7 marks] Prove that the following problem is NP-complete. Given two graphs, \( H = (V_H, E_H) \), and \( G = (V_G, E_G) \), is \( H \) a subgraph of \( G \), i.e. is there a mapping \( \pi \) of the vertices of \( H \) to the vertices of \( G \) such that \( \pi \) is one-to-one (it never maps two vertices of \( H \) to the same vertex of \( G \)) and such that for every pair of vertices \( u, v \in V_H \), we have \((u, v) \in E_H \) iff \((\pi(u), \pi(v)) \in E_G \).

4 More NPC.

4.1. [7 marks] Consider the following modification of the **FairSplit** problem: **FairSplit100**

Input: a list of \( n \) integers, \( a_1, a_2, \ldots, a_n \), indexed by \( S = \{1, \ldots, n\} \).

Question: is there a partition \( S = A \cup B \) with \( A \cap B = \emptyset \) such that \( \sum_{i \in A} a_i - \sum_{i \in B} a_i < 100 \)?

Prove that **FairSplit100** ∈ NPC.

4.2. [7 marks] Show that the following decision problem is NP-complete: given a graph \( G \) in which every vertex has even degree, and an integer \( k \), does \( G \) have a vertex cover with at most \( k \) vertices?

(The degree of a vertex is the number of edges incident to it.)

Hint: given an arbitrary graph \( G \), find a way to modify it by adding some vertices and edges so that all the vertices of the new graph have even degree. You can use the following fact without proof: in any undirected graph \( G \), the total number of vertices of odd degree is always an even number (possibly zero).