CPSC 320 2016W2: Quiz 6 Pre-Release Information

March 26, 2017

This week's quizzes will follow up on the problem domains described below. It's worth spending a few minutes reading and understanding each domain before your tutorial! In general the problems revolve around reductions, NP-completeness and DP.

Note that if you collaborate on understanding this information, you should follow the academic conduct guidelines. Among other rules, take down the GradeScope student #s or account names of collaborators for acknowledgement, but take **no other notes** away from those collaborators!

NOTE: The names of problems (on the quiz) subproblems are often intended to be fun and are completely irrelevant. Feel free to ignore them here and on the quiz!

Please, remember your GradeScope # when coming to the tutorial quiz.

1 Greedy banks resequencing debits again :(

Predatory banks take the debits to an account that occur over the day and reorder them to maximize the fees they can charge. For each debit that results in taking an account into overdraft (having negative balance in the account) or where the account is already in overdraft, the bank charges the customer an overdraft fee: 10% of the debited amount. For example if the sequence of debits D is \$3, \$4, \$5 and the initial account balance $B = \$8$, the optimal order (for the bank) is: \$4, \$5, \$3 with overdraft fees: $0.1 * (\$5 + \$3) = \$0.80$. Assume that the debit amounts $D = [d_1, \ldots, d_n]$ and initial balance B are in whole dollars (so \$4 is ok, but \$4.1 is not).

SUBSET SUM problem: Suppose we have an array A of length n containing positive integers. For some value k, we want to know if A contains a subset of elements that sums to exactly k.

An example of an instance for SUBSET SUM: $A = [3, 7, 13, 19, 29, 37]$ and $k = 55$. This is a YES-instance, since $55 = 7 + 19 + 29$. Another example: the same A with $k = 54$, which is a NO-instance (feel free to try all 64 combinations).

Note that SUBSET SUM is NP-complete.

2 3-SAT Variations

Recall the 3-SAT problem. Given a collection of 3-literal clauses, we want to decide whether there is a satisfiable assignment (of $true/false$ values to variables) that satisfies each clause. This problem is NP-complete even if we assume that clauses cannot contain the same variable multiple times. For instance, clauses $(x_1 \vee x_1 \vee x_1)$ or $(x_1 \vee \overline{x_1} \vee x_2)$ are not allowed.

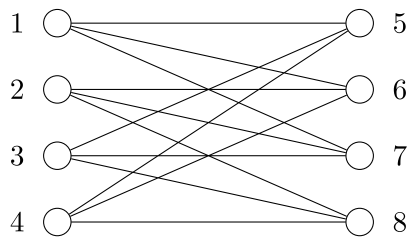
The number of occurrence of a variable x_i in the instance of 3-SAT is the number of times literals x_i or $\overline{x_i}$ appears in the instance. For example, in instance

$$(x_1 \vee \overline{x_2} \vee x_6) \wedge (\overline{x_6} \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge (\overline{x_2} \vee x_3 \vee x_5)$$

there is 1 occurrence of x_1, 3 occurrences of x_2, 3 occurrences of x_3, 1 occurrence of x_4, 2 occurrences of x_5 and 2 occurrences of x_6.

A *bipartite graph* is a undirected graph $G = (V, E)$ in which vertices can be partitioned into two sets V_1, V_2 (so $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \{\}$) such that every edge $(u, v) \in E$ has one end point in V_1 and the other end point in V_2. A bipartite graph is often denoted as $G = (V_1, V_2, E)$.

A graph is called *regular*, if every vertex has the same degree. For example, the following graph is a regular bipartite graph:



Note that the sizes of the two partitions (the left and right set of vertices) in a regular bipartite graph are the same.

BIPARTITE MATCHING problem: Given a bipartite graph $G = (V_1, V_2, E)$. Find a maximal matching $E' \subset E$. (Recall: no two edges in a matching share a vertex.)

The BIPARTITE MATCHING problem can be solved in time $O(|V||E|)$ by Ford-Fulkerson algorithm.

In addition, we have the following theorem:

Theorem 1 (Hall's Theorem)**.** *A regular bipartite graph has a matching with exactly $|V_1| = |V_2|$ edges (so it involves all vertices).*

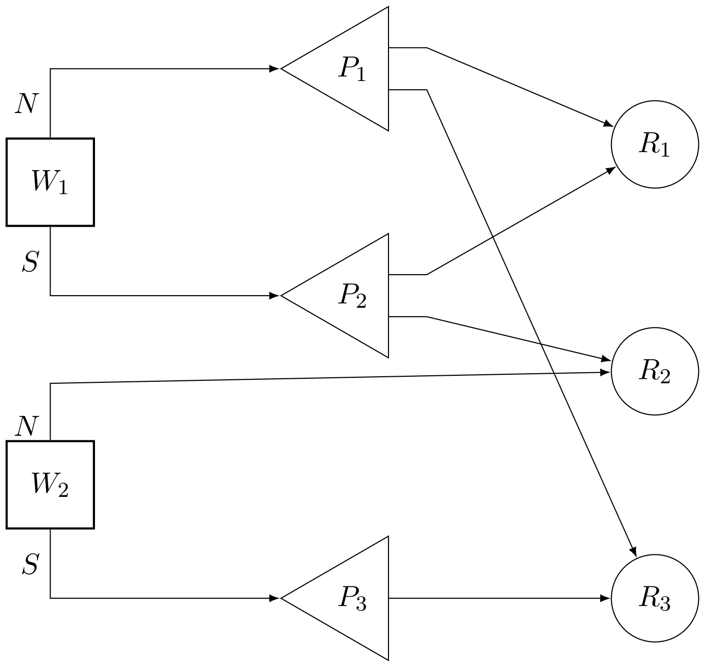
3 Pipelines

In the PIPELINE problem, you're given a network of pipelines which can be represented as a directed, acyclic graph (DAG) with three types of nodes:

- "Oil wells" nodes that produce oil. They have **no** pipelines coming in and two pipelines going out labeled "N(orth)" and "S(outh)". They also have a switch. If the switch is in the "N" position, then the oil flows into the northern pipeline. If the switch is in the "S" position, then oil flows into the southern pipeline.
- "Pump station" nodes can have one pipeline coming in (which may or may not carry oil depending on the configuration of oil well switches) and any number of pipelines going out. If the pipeline coming in carries oil, then all pipelines going out also carry oil. Otherwise, none of the pipelines carries oil.
- "Refinery" nodes require oil supply to produce other products. They have one or more pipelines coming in and none going out. If any pipeline coming in carries oil, the refinery is operational. Otherwise, it is not.

The solution to an PIPELINE instance is YES if some configuration of the oil well switches supplies oil to all refineries; otherwise, it's NO.

Here is an example of a PIPELINE instance:



Oil well nodes are labeled W, pump station nodes P, and refinery nodes R.