CPSC 320 2016W2: Assignment #6

March 31, 2017

Please submit this assignment via GradeScope at `https://gradescope.com`. Detailed instructions about how to do that are pinned to the top of our Piazza board: `https://piazza.com/ubc.ca/winterterm22016/cpsc320/`. Briefly, your GradeScope account **must** use the "GradeScope Student #" we distributed in your Connect gradebook so that we can link your account with you!

Submit by the deadline **Thursday April 6 at 10PM**. **For this assignment, submit solutions to two questions only! Choose one question from 1.2, 1.4, 1.5 and 2.1, and one question from 1.3, 2.2 and 3.1. (Question 1.1 is intentionally excluded and its solution is provided below.) If you submit more than the two solutions requested, we will make an arbitrary choice as to what to mark, such as choosing randomly or choosing the submitted question with the lowest mark. So, SUBMIT ONLY TWO SOLUTIONS as requested (one from each group!).**

For credit, your group must make a **single** submission via one group member's account, marking all other group members in that submission **using GradeScope's interface**. Your group's submission **must**:

- Be on time.
- Consist of a single, clearly legible file uploadable to GradeScope with clearly indicated solutions to the problems. (PDFs produced via LaTeX, Word, Google Docs, or other editing software work well. Scanned documents will likely work well. **High-quality** photographs are OK if we agree they're legible.)
- **Include a page that says "DID NOT CHOOSE" and point the 5 questions that you didn't choose to answer to this page.**
- Include prominent numbering that corresponds to the numbering used in this assignment handout (not the individual quiz postings). Put these **in order** starting each problem on a new page, ideally. If not, **very clearly** and prominently indicate which problem is answered where!
- Include at the start of the document the **GradeScope Student #s** of each member of your team. (No names are necessary.)
- Include at the start of the document the statement: "All group members have read and followed the guidelines for academic conduct in CPSC 320. As part of those rules, when collaborating with anyone outside my group, (1) I and my collaborators took no record but names (and GradeScope information) away, and (2) after a suitable break, my group created the assignment I am submitting without help from anyone other than the course staff." (Go read those guidelines!)
- Include at the start of the document your outside-group collaborators' GradeScope student #s or account names. (Be sure to get those when you collaborate!)

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 $A = [a_1, \ldots, a_n]$ 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.

1.1 Imaginary max profit

Knowing only that the largest debit in D is d_{max} and the sum of all debits is $d_{sum} > B$, give a (non-asymptotic) upper-bound on the overdraft fees the bank could collect. Answer this question in terms of d_{max}, d_{sum} and B without knowing anything else about the debit amounts.

Sample Solution

The bank will collect fees for the debits that already in the overdraft, so in order to maximize profit they would want to have the largest debit amount putting the account in overdraft in such a way that the most of this debit is paid by the available balance and only a tiny bit is covered by borrowed money, as this way the bank could charge 10% on the largest amount that is technically still covered by the balance. The smallest tiny bit is \$1 (since we assume debits and the initial balance are integers), so let's assume that the sum of debits processed up to d_{max} (including d_{max}) is $B + 1$. Hence, the overdraft fee 10% is collected from d_{max} and the subsequent debits, which sum up to $0.1 * (d_{max} + d_{sum} - (B + 1))$.

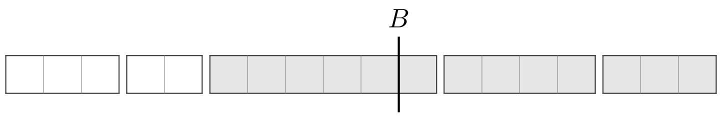
1.2 Greedy CEO Part 1

After hearing about the possibility of collecting the amount of fees described in the previous part, the CEO of Greedy Banks Consortium wants to find out for which collections of debits D and initial balances B (both containing only integer values) it's possible to charge this maximal fee

$$0.1 * (d_{max} + d_{sum} - (B + 1)),$$

where d_{max} is the largest and d_{sum} is the sum of all debits in D. Let's call this problem the **GREEDY CEO problem**. So the answer to an instance of this problem is "YES" if it is possible to achieve this upper-bound on fees, and the answer is "NO", otherwise.

An example of a YES-instance (with $D = 2, 3, 3, 4, 6$ and $B = 10$) is illustrated in the following diagram in which rectangles represents different debits (the number of boxes in the rectangle shows the amount of debit), shaded rectangles are debits in overdraft and the vertical line show the initial account balance:



Note that the largest debit (\$6) is putting the account into overdraft and most of it (except exactly \$1) is covered by the account balance.

An example of a NO-instance is the following: $D = 1, 3, 4$ and $B = 5$. Here, if we "place" $d_{max} = 4$ to right position (just \$1 over the balance B), it creates two gaps, one before and one after d_{max}, of size 2, which we cannot "fill" with debits \$1 and \$3 (as we are not allowed to break debits into smaller pieces).

You were tasked to write an efficient (polynomial) algorithm for the GREEDY CEO problem. You suspect that such an algorithm might not exist, so you want to prove to your boss that the problem cannot be solved by any efficient algorithm (assuming $P \neq NP$). You came up with the following reduction from the SUBSET SUM problem to the GREEDY CEO problem:

Reduction: Let $a_1, \ldots, a_n, k$ be an instance of the SUBSET SUM problem. Let $a_{max} = \max\{a_1, \ldots, a_n\}$. In order to achieve this maximum overdraft fee, we have to "pack" a subset of debits just before d_{max}. That means they would have to sum up to $B + 1 - d_{max}$. Let's construct an instance of the GREEDY CEO problem from the instance of the SUBSET SUM problem by using numbers in A as the debit amounts (so $D = A$) and setting $B = k + a_{max} - 1$.

Explain why this reduction is **incorrect**! Your explanation **must** include a simple counterexample to the correctness of the reduction.

1.3 Greedy CEO Part 2

Fix the reduction and **prove** it's correct!

1.4 Greedy CEO Part 3

You were tasked to write an efficient (polynomial) algorithm the GREEDY CEO problem. You found out the problem is NP-hard, but there is a hope. The SUBSET SUM problem is solvable in polynomial time in terms of n and k, which is fine as long as k is only modestly large (bounded by some polynomial in n).

Design a DP algorithm that solves SUBSET SUM problem in time $\Theta(nk)$.
Hint. Base your recurrence for SUBSETSUM($[a_1, a_2, \ldots, a_n], k$) on whether the last element (a_n) is included in the subset or not. Then use the recurrence to write a pseudocode for the DP algorithm.

1.5 Greedy CEO Part 4

Solve the GREEDY CEO problem in time $\Theta(nB)$ by reducing it to the SUBSET SUM problem (and using a $\Theta(nk)$-time SUBSET SUM solver as a black box).

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.

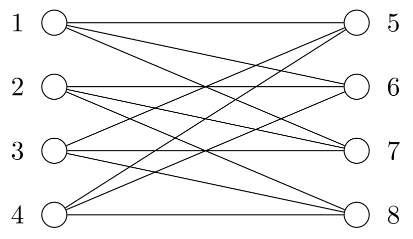
The number of occurrence of a variable x_i in the instance of 3-SAT is the number of times literals x_i or $\overline{x_i}$ appears in the instance. For example, in instance

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

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).*

2.1 Solvable 3-SAT!

Consider a special version of 3-SAT that requires that each variable has exactly 3 occurrences and no clause can contain multiple occurrences of the same variable. Let's call this problem 3-SAT3.

Show that 3-SAT3 can be solved in polynomial time by reducing it to the BIPARTITE MATCHING problem. Determine the running time of your algorithm!

Hint. Built a bipartite graph from a 3-SAT3 instance. Note that every instance of 3-SAT3 must contain the same number of variables and clauses.

2.2 Darn 2-clauses!

Consider a special version of 3-SAT that requires that each variable has exactly 3 occurrences, but allows each clause to have either 2 or 3 literals. Let's call this problem 2,3-SAT3. Here is an example of an instance of 2,3-SAT3 problem:

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

Note that each variable has exactly 3 occurrences.

We want to show that 2,3-SAT3 is NP-hard[1]. Follow the following steps to reduce 3-SAT to 2,3-SAT3. For simplicity, you may assume that in each instance of 3-SAT, every variable has at least 2 occurrences.

(a) Given variables $y_1, y_2, \ldots, y_k$, $k \geq 2$, design a set of 2-literal clauses such that there are exactly two assignments for these variables that satisfy the set of your designed clauses: (i) $y_1 = y_2 = \cdots = y_k = false$, and (ii) $y_1 = y_2 = \cdots = y_k = true$. Each variable should have exactly 2 occurrences in these clauses.

(b) Now consider an instance S of 3-SAT in which variable x_i has $k \geq 2$ occurrences. Using part (a) construct a new instance S' by replacing all occurrences of x_i with new variables $y_1, \ldots, y_k$. Add necessary clauses to make the original instance S and the new instance S' equivalent (one is satisfiable if and only if the other is satisfiable). Argue why each of these new variables has exactly 3 occurrences in S'.

(c) Describe a reduction from 3-SAT to 2,3-SAT3 as an algorithm.

(d) Prove correctness of your reduction!

Bonus. Describe a reduction from 3-SAT to 2,3-SAT3 without assuming the simplifying assumption (that each variable in 3-SAT instance has at least 2 occurrences).

[1]This is somewhat surprising, since both 2-SAT and 3-SAT3 are in P.

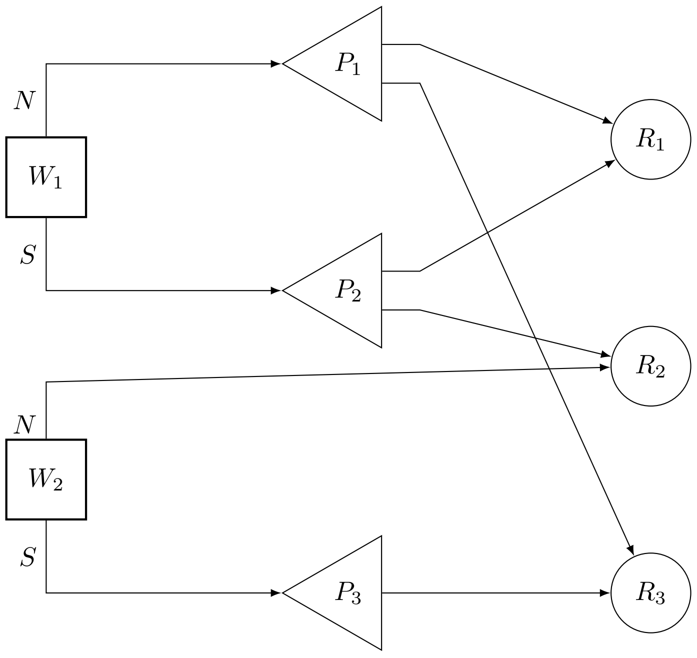
3 Pipelines

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 well" 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.

3.1 Reduction from SAT to PIPELINE

(a) List all configurations of the oil well switches in the network on the previous page that supply oil to all refineries.

(b) Give a reduction from SAT to PIPELINE.

Hint: Consider that a variable can be positive or negated, the positive (or negated) literal can appear in many clauses, and each clause needs at least one true literal in it.

(c) Prove correctness of your reduction!