Written Problem Set 5: Reductions

The first thing you should do in ps5.tex is set up your name as the author who is submitting by replacing the line, `\submitter{TODO: your name}`, with your name and UVA email id, e.g., `\submitter{Grace Hopper (gmh1a)}`.

Next, put your response for problem 1 into problem1.tex, problem 2 into problem2.tex, etc.

Before submitting , also remember to:

– List your collaborators and resources, replacing the TODO in `\collaborators{TODO: replace ...}` with your collaborators and resources. (Remember to update this before submitting if you work with more people.)

– Replace the second line in ps5.tex, `\usepackage{algo}` with `\usepackage[response]{algo}`.

Collaborators: TODO: replace this with your collaborators (if you did not have any, replace this with *None*)

Resources: TODO: replace this with your resources (if you did not have any, replace this with *None*)

Problem 1a: Shift Scheduling

You are the shift manager at "Nate's Noms", the hottest new bakery in all of Charlottesville. You must assign shifts to the employees over the final exam period. The problem is that most of the employees are students, who must fit in their shifts between their exams, studying, and parent-mandated "family time". Your task is to write an algorithm to create a work schedule.

Your algorithm will be given the following:

– A number k which represents how many shifts that need to be staffed.

– A list of numbers c, where c_i represents the number of employees required to work during shift i.

– A number n which represents how many bakery employees there are.

– For each employee i, a list of preferred shifts P_i that employee i would like to work.

– A number x which represents the maximum number of shifts each employee may be assigned outside of their preference list (recall that P_i is the preference list for employee i).

Your task is to design an efficient algorithm which assigns exactly c_i employees to each shift i, but does not give any employee more than x shifts outside of their preferences. It should also indicate when it is not possible to satisfy these constraints (e.g., by outputting "No solution."). Analyze your algorithm's running time and correctness. (**Hint:** Use Maxflow)

Algorithm: Describe your algorithm

Correctness: Justify the correctness of your algorithm (doing so informally is fine).

Running time: Present the running time of your algorithm, and describe how you came to that time.

Problem 1b: Minimizing the Extra Shifts

Use the algorithm from Problem 1 to minimize the maximum number of extra shifts assigned to any employee. In other words, find the smallest value of x that would allow for a valid schedule. Your algorithm must run in polynomial time (in the length of the input).

To get you started, here is an exponential time algorithm which would give the correct answer.

Exponential-Time Algorithm. First, invoke the algorithm from Part 1 with $x = 0$. If there is no schedule, then re-try with $x = 1$. Keep retrying until you find a value of x which allows for a schedule, return that value.

To evaluate the running time of this algorithm, first observe that k is an upper bound on x, since no employee can be assigned to more than k extra shifts, so the procedure will run at most k times. The reason this is exponential, though, is that running time is expressed in terms of the *size* of the input. To give the number k as input requires $\log_2(k)$ bits, so if our algorithm runs in linear time relative to k, then it's running exponentially relative to $\log_2(k)$, which is the input size.

Algorithm: Describe your algorithm

Running time: Present the running time of your algorithm, and describe how you came to that time.

Problem 2: Maximum R.O.I.

Typically, stock prices are reported by percentage change from the previous day. For example, Beyond Meats Inc (BYND) closed on July 6, 2020 with a value of \$29.03 per share. It closed on July 7, 2020 with a value of \$31.76 per share. The stock change on July 7 is therefore reported as $+9.04\%$ (because $29.03 \cdot 1.09404 = 31.76$), thus someone who bought shares on May 6 and sold on May 7 would have made a 9% profit. If the market share were to go down by 10% (e.g. if the value was \$100 yesterday and \$90 today) then the change would be reported as -10%.

Suppose you are given a list of daily gains (and losses) for a particular stock, reported as a percentage change from the previous day. Write an algorithm to identify two days, b and s, such that buying stock at the end of day b and selling at the end of day s would yield the maximum percentage return on investment. For example, if the daily gains were reported as follows:

Gain:	$+10\%$	$+30\%$	-40%	$+20\%$	-10%	$+100\%$	-70%
Day:	0	1	2	3	4	5	6

Then the best time to buy would be on day $b = 2$ and the best day to sell would be on day $s = 5$, yielding a return of $1.2 \cdot 0.9 \cdot 2.0 = 2.16$, or a profit of 116%.

Solve this problem by showing how to reduce it to another problem we have seen previously in this class. Your algorithm's overall running time (time for the reduction plus time to solve the problem you reduced to) should be $O(n)$.

Reduction: Describe your reduction. Make it clear which problem you are reducing to, how instances map to instances, and solutions map to solutions.

Running Time: Justify that your algorithm's running time is $O(n)$ (including the time of the reduction as well as the time required to solve the problem you reduce to).

Problem 3: Insert/Delete/Min Data Structure

Consider a data structure with the following operations:

– `INSERT`(x), which inserts a given element x into the data structure

– `DELETE`(x), which removes a given element x from the data structure, should it be present

– `MIN`$()$ which returns but doesn't remove the minimum element from the data structure

Show that it is impossible for any data structure to have all of `INSERT`, `DELETE`, and `MIN` run in worst case $O(1)$ time.

Problem 4: Random Permutations

Consider the following approach for shuffling a list of n elements:

> Apply the following procedure for $i = 1, \ldots, n$:
>
> – Choose a random index of the list.
>
> – Swap the element at index i with the element at the randomly-selected index.

We say that a list is perfectly shuffled if, after applying the shuffling algorithm to any initial configuration of the list, all possible permutations of the list are equally likely. Use a decision tree to show that for every $n > 2$, this shuffling algorithm fails output a perfectly shuffled list. In other words, show that for every choice of $n > 2$, some permutations of the list are more likely than others.

To do this:

– Describe/draw a decision tree for a list of length n, keeping in mind that the nodes in a decision tree are going to be the steps that influence the final output

– Compare the number of leaves in this decision tree with the number of permutations in the list

– Use that comparison to arrive at the conclusion that not all permutations can be equally likely (**Hint:** you may use, without justification, that integers i and $i - 1$ have a greatest common denominator of 1).

