Homework 2

CS 4364/5364
Spring 2022

Due: 9 February 2022

Because of the reliance of the particular assignments in this class on mathematical notation, and the fact that all assignments will be submitted electronically, students are encouraged to use LaTeX to formalize their responses. **For those enrolled in the graduate section the use of latex is *required.*** This assignment (like all others) will be posted on the course `github`[1] as source code as well as in PDF form on the course website. Please submit your assignment to the professor via email, either as a link to your assignment online (i.e. overleaf or github) or as an attachment. Graduate students will need to include the `.tex` files as well as a PDF, this is optional but encouraged for undergraduates.

1. **(25 points)** The first method we saw to find a pattern P in a text T was using the maximum prefix overlap values on a special string (calculate M_i for each i in the string $P\$T$). We know that we can compute these M_i values in $O(m+n)$ time (assuming $|P| = n$ and $|T| = m$). The solution described in class assumes both strings are over the same alphabet: what if they were not?

   Consider the following: Given a protein sequence pattern $P \in \Sigma_{AA}^*$ over the amino acid alphabet, and a RNA text $T \in \Sigma_{RNA}^*$ (see footnote [2]). Develop an algorithm that uses the maximum prefix overlap method to determine where the pattern P is in the text T (if it exists), and runs in $O(m+n)$ time.

   Translating from RNA codons (3 nucleotides) to amino acids is done using the standard codon table (can be found in the slides, though its not actually needed for this exercise). The problem is that to reverse the translation of an amino acid $p_i \in P$ there are multiple choice of codon that could have produced it, thus there is an exponential number of possible un-translations of P. Therefore, we cannot simply enumerate all possible RNA un-translations of the pattern and run the maximum prefix overlap, it would be exponential time which is not feasible.

[1] `github.com/deblasiolab/CS4364-documents`

[2] $\Sigma_{AA} = \{\mathtt{A},\mathtt{R},\mathtt{N},\mathtt{D},\mathtt{B},\mathtt{C},\mathtt{E},\mathtt{Q},\mathtt{Z},\mathtt{G},\mathtt{H},\mathtt{I},\mathtt{L},\mathtt{K},\mathtt{M},\mathtt{F},\mathtt{P},\mathtt{S},\mathtt{T},\mathtt{W},\mathtt{Y},\mathtt{V}\}$ and $\Sigma_{RNA} = \{\mathtt{A},\mathtt{C},\mathtt{U},\mathtt{G}\}$

Hint: you can run the algorithm multiple times, but as long as this number of repetitions is constant it is absorbed by the big-O.

2. **(20 points)** Given a directed graph $G = (V, E)$, source and sink vertexes $s, t \in V$, and a limit on the flow allowed through nodes m_e (defined $\forall e \in E$). Write the max flow problem as a linear program, and explain each (set of) equation in your definition. Remember the max flow problem assigns a flow (weight), f_e, to each *edge* in a graph while maximizing the total flow going from the source to the sink. For each node (other than the source and sink, i.e. $v \in V \setminus \{s, t\}$) the total in flow must equal the out flow. Some helpful notation to use:

   - a directed edge $e_{(ab)} \in E$ goes from a to b
   - the set of *in* edges to a vertex v can be written as $E_{(*v)} = \left\{e_{(a,b)} | b = v, e_{(ab)} \in E\right\}$
   - the set of *out* edges to a vertex v can be written as $E_{(v*)} = \left\{e_{(a,b)} | a = v, e_{(ab)} \in E\right\}$
   - we can say the sum of the *out* flow to a node v is $\sum\limits_{e \in E_{(v*)}} f_e$

   Your variables will be the set of f_e's, some of these (but not all) will end up in the optimization function. The things to keep in mind that must be satisfied are: (1) conservation of flow across nodes, (2) maximum flow across an edge, and (3) the outflow at the source should equal the inflow at the sink (though this may not need to be explicit in your program).

   Writing down an LP

   In this example below we are minimizing our objective function, our variables are $x_1, x_2, \& x_3$, and we have a collection of maximum values $m_1, m_2, \& m_3$. As a reminder the objective function (which is being maximized or minimized) defines what makes a solution *optimal*, and the set of constraints (linear inequalities) define what makes a solution *valid* As an aside, even though the values of m_x are arbitrary that are not technically variables that we're trying to optimize.

$$\begin{array}{lll} \text{minimize} & 7x_1 + 14x_2 - 12x_3 & \\ \text{subject to} & x_i \le m_i & 1 \le i \le 3 \\ & \sum\limits_{1 \le i \le 3} x_i \ge 4, & \end{array}$$

   (This LP has no grounding in a real-world problem, I made it up.)
   Note that the first line of the constraints section actually defines a collection of inequalities, one for each value of i.