Joseph Baca
Homework 2

CS 4364/5364
Spring 2022

Due: February 14, 2022

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 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.

   My Solution to the following problem

   Traversing the RNA text T and collecting all substrings of size 3, each codon being 3 nucleotides long (S_i for each codon i in the string T). This is done a fixed number of times that gets absorbed by big-O notation $O(3m) => O(m)$. After converting each codon to amino acid (S_{ci} where c is the amino acid conversion of i) we can do a maximum prefix overlap and calculate M_i values per each converted codon in pattern P ($S_{ci}\$P$) in $O(3n) => O(n)$ time. We can find if $S_{ci} \in P$ if $|S_{ci}| = M_i$. The final complexity analysis would result in $O(3m + 3n) => O(m+n)$ time.

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
   - Set of *in* edges to a vertex v can be written as $E_{(*v)} = \left\{e_{(a,b)} | b = v, e_{(ab)} \in E\right\}$
   - 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 in flow to a node v is $\sum_{e \in E_{(v*)}} f_e$

   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).

   Conservation of flow across nodes

   $$\sum_{e \in E_{(*v)}} f_e \leq M_e \geq \sum_{e \in E_{(v*)}} f_e$$

   This equation describes the conservation of flow. Conservation of flow begins with the inflow from previous edge to the weight of a verticie M_e. Of course, the weight of the vertice must be greater or equal to the summation of the inflow edges described by $\sum_{e \in E_{(*v)}} f_e \leq M_e$. Now the outflow of a vertice can only realistically be less than or equal to the weight of the corresponding verticie, described by $M_e \geq \sum_{e \in E_{(v*)}} f_e$ because such a bottleneck may exist the flow is conserved through the verticies

Maximum flow across an edge

$$Max(f_{e(a,b)}) = M_{e(a)}$$

This equation describes the total maximum flow accross an edge. The max flow (f) in an edge $e(a, b)$ must be equal to the previous verticies weight a denoted by: $M_{e(a)}$. This is because $e(a, b)$ can not aquire more feed than what verticie a has aquired.

Outflow at source is equal to the inflow at sink

$$\sum_{e \in E_{(s*)}} f_e = \sum_{e \in E_{(*t)}} f_e$$

This equation describes the outflow from the source must be equal to the inflow at sink. Once every path from $S- > T$ has been discoverd (by DFS algorithm or something else) each path can be augmented (Reference to Ford-Fulkersons solution) to meet the conservation of flow equation between the nodes. Once every path has been augmented the output from the source will be equal to the input of the sink.