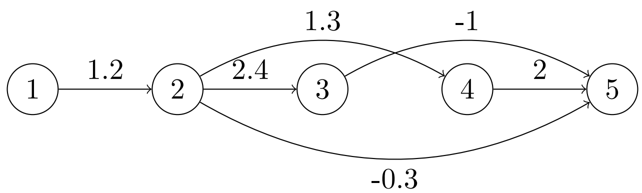
CSCI 3104-Spring 2016: Assignment #8.

Assigned date: Wednesday, 4/6/2016,
Due date: Tuesday, 4/12/2016, before class
Maximum Points: 40 points (includes 5 points for legibility).

Note: This assignment *must be turned in on paper, before class.* Please do not email: it is very hard for us to keep track of email submissions. Further instructions are on the moodle page.

P1 (20 points) Let G be a directed acyclic graph with n vertices and m edges. We will find how to compute single source shortest paths for G running in time $\Theta(m + n\log(n))$. Let us take the example below.



(A) Find a topological ordering of the graph G. How much time does it take for n nodes and m edges. Write down a topological ordering for the graph G above?

(B) Design an algorithm that uses the topological ordering of G to find shortest paths from a single source. Illustrate the working of your algorithm to compute shortest path distances and the resulting tree starting from node 2.

(C) What is the running time of the overall algorithm for finding shortest paths in DAGs.

Solution. (A) The topological sorting takes $\Theta(m + n\log(n))$ time, the cost of running a DFS and sorting the edges in topological order. The topological sort of G yields the order $1, 2, 3, 4, 5$.

(B) The algorithm works as follows:

1. Initialize all distances to ∞ for non source nodes and 0 for the source node.
2. For each vertex v in the topological sorted order,
3. For each outgoing edge e from vertex v,
4. Relax according to e.

The algorithm works as follows on the graph above.

1. We consider vertex 1 and edge $1 \to 2$. This has no effect since $d[2] = 0$.
2. We now consider vertex 2 and edges $(2, 3), (2, 4)$ and $(2, 5)$. This updates $d[3] = 2.4$, $d[4] = 1.3$ and $d[5] = -0.3$.
3. Next we consider vertex 3 and edge $(3, 5)$. But this has no effect.

4. Next we consider vertex 4 and edge $(4, 5)$ but this has no effect.

Final distances are therefore

$$d[1] = \infty, d[2] = 0, d[3] = 2.4, d[4] = 1.3, d[5] = -0.3$$

(C) Running time is dominated by topological sorting and is $\Theta(m + n\log(n))$.

P2 (20 points) A dictionary cipher substitutes each word in the dictionary by a string according to a codebook. Example codebook can be like below.

Word	Code
hello	juuyakjel
world	kjaue
how	ajsuei
are	lloppyy
you	jkjauieu

We have a part of the codebook that allows us to translate some words but are missing the other part. A long cipher-text with n letters is provided.

"jkjkuieuijuuyakjelkjkiekjaueajuseilloppyyaskjirrjkjauieukjkaiejjuyyajjuuyakjel"

Our goal is to split the cipher text to highlight known words and parts that cannot be decoded. For instance, the messge above may be split as follows:

"jkjkuieui + juuyakjel + kjkie + kjaue + ajusei + lloppyy+askjirr+jkjauieu+kjkaiejjuyyaj+juuyakjel"

This split yields 6 known codewords (underlined) from the codebook and 34 characters that are *ciphered*, i.e., not part of a known codeword. The cost of such a split is taken to be $6 + 2 * 34 = 74$.

Formally, we are given a codebook with codewords and a long cipher-text string with n characters. Our goal is to find a *minimum cost split* that splits the given cipher text into a set of known codewords and ciphered characters so that the cost (defined as number of code words + 2 * number of ciphered characters) is minimized.

(A) Show how a given string can be converted in to a graph whose vertices are the positions in the string and edges encode cost. Show how you can add edges to the graph corresponding to codewords and ciphered characters.

(B) Show how the shortest cost path from the first position to the last yields an optimal way to split the string.

Solution. (A) Let w be a word with n letters. The vertices are labelled 1 to $n+1$. From vertex i to $i+1$, we will have an edge of weight 2 that denotes that letter i is taken undeciphered. For each code word, C, if the substring $w[i] \cdots w[j]$ equals C, we add an edge of weight 1 from position i to $j+1$.

(B) The shortest path from 1 to $n+1$ in the graph G above yields the optimum way of splitting the string. If the shortest path goes from i to $j > i+1$, we know that $w[i] \cdots w[j]$ must be a codeword. On the other hand, if it goes from k to $k+1$, the character $w[k]$ is take as ciphered.