1 Introduction and Objective

This is a short tutorial which will be a prelude to the third part of CS2010: Dynamic Programming, abbreviated as DP.

DP is usually viewed as a problem solving paradigm but it can also be viewed as an algorithm on a special class of graph: Directed Acyclic Graph (DAG). The problems discussed in this tutorial involves DAGs. And since we will work with DAGs, we will also re-discuss topological sort on DAG which is an essential concept behind DP technique. We will see how topological ordering of the vertices of our DAG allows us to perform efficient (DP) algorithms on it.

We will heavily use http://visualgo.net/recursion.html and also the topological sort feature in http://visualgo.net/dfsbfs.html in this tutorial.

The tutor will try to end the tutorial earlier so that you can ask about past Written Quiz 2 papers, if any.
2 Tutorial 09 Questions

Review Topological Sort

Q1. Given the DAG in Figure 1, show how to use One-Pass Bellman Ford’s (a.k.a. DP) to find both the Shortest Paths and the Longest Paths from source vertex 2 to every other (reachable) vertices. The tutor is empowered to actually change the DAG in the picture during the actual tutorial. Steven has not actually implement this feature in http://visualgo.net/sssp.html so this has to be semi-manual for now.

Q1.a). Find the topological order of the DAG in Figure 1 using the modified DFS routine as taught in Lecture 06!

Q1.b). Now perform One-Pass Bellman Ford’s to get the Shortest Paths from source vertex 2 to every other (reachable) vertices.

Q1.c). Now perform One-Pass Bellman Ford’s to get the Longest Paths from source vertex 2 to every other (reachable) vertices.

Q2. The modified DFS topological sort algorithm given in the Lecture 06 only gives only one valid topological ordering. How can we find all possible valid topological orderings for a given DAG? For example, there are 1008 possible valid topological orderings of the DAG in Figure 1 on top of one that you find in Q1.a).

Dynamic Programming - Basic Ideas

Q3. What are the two important properties of a problem that allows for a DP solution?
Q4. You are given the following implementation for finding the \( n \)-th Fibonacci number. Is it a DP solution? If yes, explain! If no, how to write a DP solution (either Top-Down or Bottom-Up) for finding the \( n \)-th Fibonacci number?

\begin{algorithm}
\textbf{Algorithm 1 fib(}n\textbf{)}
\begin{algorithmic}
\If {\( n == 0 \) or \( n == 1 \)}
\State return \( n \)
\EndIf
\State return fib(\( n-1 \)) + fib(\( n-2 \))
\end{algorithmic}
\end{algorithm}

Q5. Given a target value \( n \) in cents, and a list \( C \) of \( k \) different coin denominations (each coin denomination is also given in terms of cents), what is the minimum number of coins that we must use in order to obtain the amount \( n \)? You can assume we have an unlimited supply of coins of any type, \( 1 \leq n, k \leq 1000 \), and the smallest coin denomination is a 1 cent coin.

Example: Given \( n = 10 \), \( k = 2 \) and \( C = \{1,5\} \), we can use:

1. Ten 1 cent coins = 10x1 = 10. Total coins used = 10
2. One 5 cents coin + five 1 cent coins = 1x5 + 5x1 = 10. Total coins used = 6
3. Two 5 cents coins = 2x5 = 10. Total coins used = 2 → Optimal

Now your tasks:

1. Model the coin change problem above as a DAG (the key point).
2. Once you have that DAG, solve the problem with a standard graph algorithm that you have actually learned in Lecture 06, 08, and 09. Alternatively, write either a Top-Down or Bottom-Up DP solution for this problem.

**Discuss Past Written Quiz 2**

Before dismissing the class, the tutor will give opportunity for student to ask any one random question from past WQ2 papers and it will be quickly discussed in class. For subsequent questions, students can meet the tutor outside class.