CPSC 320: Intermediate Algorithm Design 2016 Summer Term 1

Tutorial 7

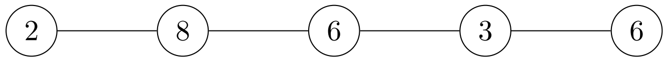
Prof. Nick Harvey *University of British Columbia*

Recurrences describing the value of optimal solutions to subproblems — Dynamic Programming

Note: Tutorial A students are at a slight disadvantage as they have to do this tutorial before attending Lecture 9. Alireza will tailor his presentation appropriately.

1. **(Independent Set on a Path)** Let $G = (V, E)$ be an undirected graph with n nodes. A subset of the nodes is called an *independent set* if no two of them are joined by an edge. Finding large independent sets is difficult in general, but it can be done efficiently if the graph is simple enough.

   Let us call the graph G a *path* if its nodes can be written as $v_1, v_2, \ldots, v_n$ with an edge between v_i and v_j if and only if the numbers i and j are consecutive ($j = i \pm 1$). With each node v_i we associate a positive integer *weight* w_i. For example, in the following path, the weights are the numbers drawn inside the nodes.

   Define recurrence relations for

   - $With[i]$: the maximum sum we can obtain using non-consecutive elements from $v_1, \ldots, v_i$, including the element v_i in the sum.
   - $Without[i]$: the maximum sum we can obtain using non-consecutive elements from $v_1, \ldots, v_i$, without including the element v_i in the sum.

2. **(Dynamic Programming)** As some of you know well, and others of you may be interested to learn, a number of languages (including Chinese and Janapese) are written without spaces between the words. Consequently, software that works with text written in these languages must address the *word segmentation* problem: inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like "meetateight" and deciding that the best segmentation is "meet at eight" (and not "me et at eight", or "meet ate ight", or any of the huge number of even less plausible alternatives). How could we automate this process?

   A simple approach that is at least reasonably effective is to find a segmentation that simply maximizes the cumulative "quality" of its individual constituent words. Thus, suppose you are given a black box that for any string of letters $x = x_1 x_2 \ldots x_k$ will return a number $Q(x)$. This number can be either positive or negative; larger numbers correspond to more plausible English words. So $Q(me)$ would be positive, while $Q(ght)$ would be negative.

   Given a long string of letters $y = y_1 y_2 \ldots y_n$, a segmentation of y is a partition of its letters into contiguous blocks of letters; each block corresponds to a word in the segmentation. The *total quality* of a segmentation is determined by adding up the qualities of each of its blocks. So we would get the right answer for the problem above provided that $Q(meet) + Q(at) + Q(eight)$ was greater than the total quality of any other segmentation of the string.

   Give a recurrence relation for $TQ(i)$: the maximum total quality of any segmentation of the letters $y_0, y_1, ..., y_i$. **Hint:** look for the position of the first letter of the current "word".