Recitation 12

Dynamic Programming

12.1 Announcements

- *DPLab* has been released and is due *Dec 1*.
- Happy Thanksgiving

12.2 Matrix Chain Product

Definition 12.1. *In the* matrix chain product problem*, we are attempting to find the cheapest way to multiply a chain of n matrices. I.e., determine a parenthesization of the expression*

$$A_1 \times A_2 \times \ldots \times A_n$$

such that cost of evaluating the expression is minimized.

Task 12.2. *Write a top-down solution to the matrix chain product problem. Specifically, assuming you have a cost function*

val `cost : int * int * int → real`

where `cost`(x, y, z) *is the cost of multiplying two matrices with dimension (x, y) and (y, z), write a function*

val `MCP : (int * int) Seq.t → real`

which takes a sequence of pairs (h_i, w_i) (the dimensions of the i^{th} matrix) and returns the cheapest cost of multiplying those matrices.

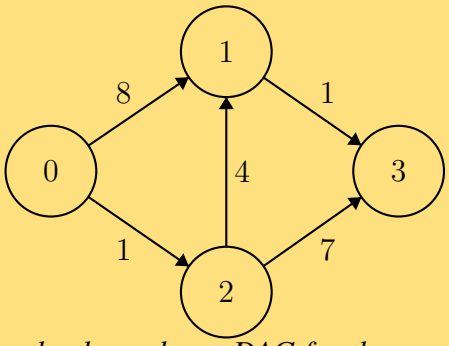
Be sure to clearly specify your subproblems. Determine the work of your algorithm by assuming that each distinct subproblem is computed only once. Determine the span of your algorithm by describing the DAG of subproblem dependencies and identifying a bottom-up ordering of the subproblems.

The *Bellman-Ford* algorithm, which we covered in the shortest path section, is another example of dynamic programming.

Task 12.3. *The code for Bellman-Ford given in the textbook is written in a bottom-up fashion. Rewrite this code in a top-down style. Once again, be sure to identify the subproblems and the DAG of dependencies. Use these observations to re-derive the work and span of Bellman-Ford.*

12.3 Additional Exercises

Exercise 12.4. *Describe the DAG of dependencies between subproblems $\delta(v, k)$ in the Bellman-Ford algorithm; i.e., each vertex is a pair (v, k), and there is an arc from (v', k') to (v, k) if we need to know the value $\delta(v', k')$ in order to calculate $\delta(v, k)$.*



Draw the dependency DAG for the example graph given above.