Dynamic Programming II

By Harry Wiggins

Summary

- DP is a confusing name for a programming technique that dramatically reduces the runtime from exponential to polynomial time.
- The trick is to find the subproblem within the problem and to come up with the recursive relationship.
- Then to figure out the right order to fill the table(especially 2D DP problems).

Example 1 : Longest increasing subsequence

- Subproblem : best[i] is the answer for the sequence s_i, … s_n.
- Recursive formula : best[i]=max{best[j] : j>i and s_j>s_i}.
- Order : best[n]=1. Then calculate best[n-1], …. , best[1]. The answer is best[1].

Example 2 : Hidden DP

Given a word what is the least number of letters you need to insert anywhere to make it a palindromic word?

Q : BANANA

A : 1 (add B at end)

Did you know : aibohphobia is the fear of palindromes?

Solution

- Looking for longest palindromic subset.
- Note it's a match between string and its reverse.
- Need to find longest common substring (did last time also)
- (IOI 2000 Day 1 Question 1)

Example 3 : Integer knapsack

You are designing a contest which isn't allowed to be longer than a certain predetermined length. You are also given a set of problems. Each problem has a point value and a certain length. Find the contest which has the maximum number of points but within the length constraint.

Need to be careful!

- Can't re-use a problem.
- Subproblem is most points for length l contest after using m problems.
- best[l][m]=max(best[l][m-1],best[l-length(m)][m-1])
- However we can be space efficient.
- If we update in the right order we only need ??? one dimensional arrays.

Example 4 : Breaking strings

The cost of breaking a string is its length. You are given a string and posititions you want to break it, you need to calculate the least cost (ie the order) of doing that.

Ie. n = 20, break at 3,8,10.

If left to right it is 49. Best?

Solution

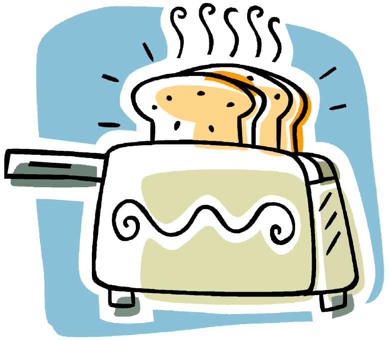
- Look it at differently.
- for each segment (i,j) let cost(i,j) be the least
- cost(i,j) = least{(cost(i,p) + cost(p,j) + length(i,j) : all p}
- This is O(n^3).
- It turns out we can do better.

Improvement

- Let $P(i,j)$ be the position where $cost(i,j)$ is minimized.
- It can be shown (icky maths) that

$$cost(i,j) = \text{least}\{cost(i,k) + cost(k,j) : P(i,j-1) \le k \le P(i+1,j)\}$$

- Complexity?
- $O(n^2)$



The End.