Dynamic Programming

Name: Ankit Kumar Singh **GaTech ID**: asingh821

---

1 General Notes

Should be able to break into sub problem and sub problems combines to form an optimal solution. Trick is to try prefix problem, prefix problem with some tweak, consider multiple subproblem to solve the bigger one, substring problem. Every DP problem has an underlying DAG structure.

1.1 Longest increasing subsequence

Subproblem: L(i) in longest increasing subsequence ending at a[i]
Recursion: L(i) = 1 + max(L(j))
Can we seen from POV of DAG of elements by connecing i,j if i ¡ j and $a_i < a_j$.

1.2 Edit distance

Subproblem: dp(i, j) denotes edit distance of prefix string $s1_i$ and $s2_j$.
Recursion: dp(i, j) = min(1 + dp(i-1, j), 1 + dp(i, j-1), diff(i, j) + dp(i-1, j-1))

1.3 Kanpsack with repetition

dp(w) - maximum value achievable from knapsack of capacity w.

1.4 Kanpsack without repetition

dp(w) - maximum value achievable from knapsack of capacity w.

1.5 Chain matrix multiplication

dp(w) - maximum value achievable from knapsack of capacity w.

1.6 Independent sets in Tree

dp(w) - maximum value achievable from knapsack of capacity w.

1.7 Rod cutting

dp(w) - maximum value achievable from knapsack of capacity w.

1.8 Optimal BST

dp(w) - maximum value achievable from knapsack of capacity w.

2 Common Problems

3 Textbook solutions