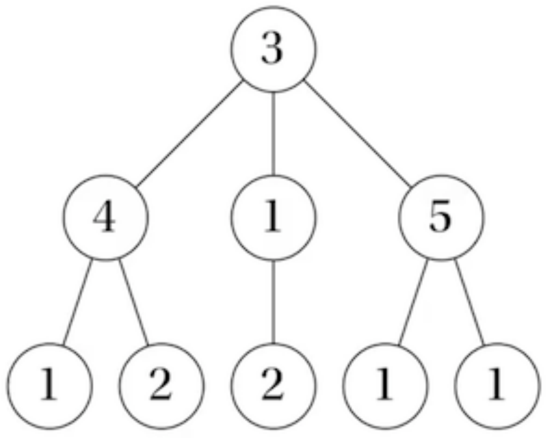
Unit 2 Exam: Algorithms & Landscapes

QUESTION 1: Dynamic Programming

Maximum independent set. Try to maximize the value of nodes you can select. Can take any subset of the nodes as long as no two nodes you take are connected by an edge. (a) What is the maximum value of the independent set? (b) Describe a dynamic programming algorithm that can find the maximum independent set from a tree - a rooted graph with no loops - of any size.



Important! Your answer should include (a) a numerical answer corresponding to the maximum independent set, and (b) a narrative description or fully annotated code of a dynamic programming algorithm that can find the maximum independent set from any tree. This description (b) should include the ordered series of logical steps through which your algorithm proceeds. NOTE: You do not need to provide working code. It is only important that you articulate the logic of your algorithm.

(a) Before discussing a dynamic programming solution, we note a greedy approach to the problem yields:

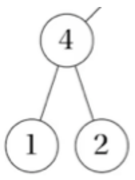
5->4->2 (middle leaf) = 11 which we will see is the maximum value of any independent set in the tree.

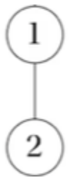
(b) Dynamic programming solution:

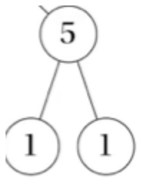
Start at the top level node with value 3. Now there are two cases (i, ii) either this node is a member of the maximum weighted independent set or it is not. So let's consider these in turn

i) **The top level node 3 is a member**. In this case this node's children 4, 1, 5 can not be members of the maximum weighted independent set as they are adjacent. Excluding 4, 1, 5 means we can and must include all the leaf nodes: 1, 2, 2, 1, 1 since none of these are adjacent. We must included all leaf nodes as omitting any one would contradict the maximal independent set assumption. Then in this case the total weight is 3+1+2+2+1+1 = 10

ii) **The top level node 3 is not a member.** In this case, there are three disconnected subtrees to consider :





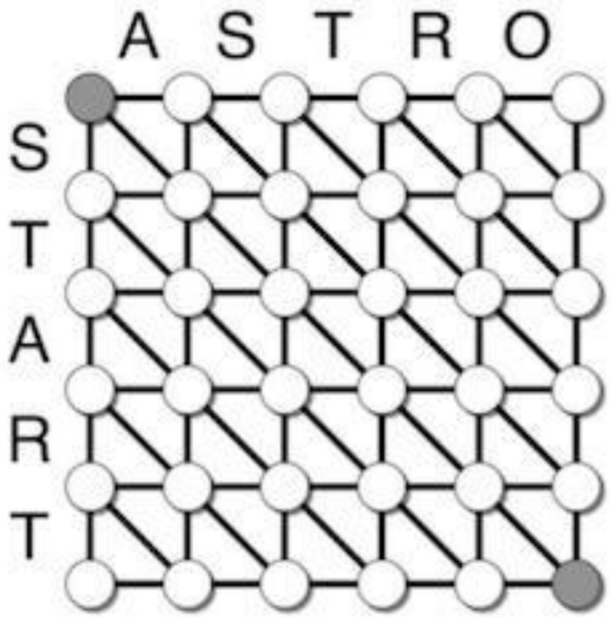


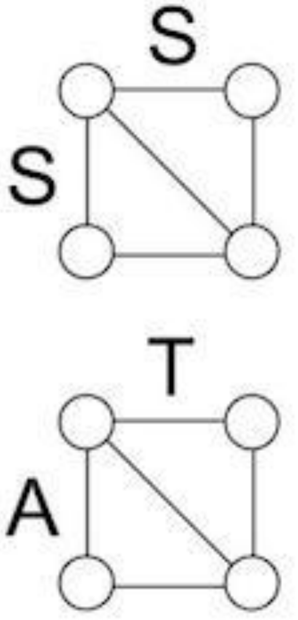
To insure the top level node 3 is covered at least one of the subtrees root nodes must be a member of the maximum weighted independent set. In fact, both 4 and 5 must both be members since they are maximal weighted independent sets for their respective subtrees. Now only the middle subtree remains to be analyzed. In this last case either 1 or 2 must be included, clearly 2 must be included since 2 is the maximum weighted independent set for the middle subtree. In conclusion, if the top **level node 3 is not a member** then the maximal weighted set is 4+2+5 = 11.

So combining cases (i and ii) the maximal weighted set has value = 11.

QUESTION 2: Reductions & Translations

(a) Assign weights (numerical values) to the edit distance rules - insertion, deletion, substitution, copy - and describe how those operations relate to the graphical representation. It may be helpful to use the examples of the two subgraphs included below and label the edges with the rule (insertion, deletion, substitution, copy) represented. (b) Find the shortest path from "ASTRO" to "START" - from the node in the upper left to the node in the bottom right. What is the weight of this path? Include a trace of the path with the weights of all included edges labeled.





(a) Insertion, deletion, and substitution will have weight +1. Copy will have a weight of 0.

	Empty	A	S	T	R	O
Empty	0 C. (0)	1 D (1)	2 D (2)	3 D (3)	4 D (4)	5 D (5)
S	1 I (1)	1 S (1)	1 D, 1 C (1)	2 D, 1 C (2)	3 D, 1 C (3)	4 D, 1 C (4)
T	2 I (2)	1 S, 1 I (2)	1 D, 1 C, 1 I (2)	1 D, 2 C (1)	2 D, 2 C (2)	3 D, 2 C (3)
A	3 I (3)	2 I, 1 C (2)	1 D, 1 C, 2 I (3)	1 D, 2 C, 1 I (2)	1D, 2 C, 1 S (2)	2 D, 2 C, 1 S (3)
R	4 I (4)	3 I, 1 C (3)	2 I, 1 C, 1 S (3)	1 D, 2 C, 2 I (3)	1 D, 3 C, 1 I (2)	2 D, 3 C, 1 I (3)
T	5 I (5)	4 I, 1 C (4)	3 I, 1 C, 1 S (4)	2 I, 2 C, 1 R (3)	1 D, 3 C, 2 I (3)	1 D, 3 C, 1 I, 1 S (3)

(b) The table given above used Dynamic Programming to construct the asked for weighted edge graph. The shortest path transforming ASTRO to START is highlighted in pink. Starting at the upper left hand corner cell (0). The path can be described as: ASTRO (delete A) -> ST (copy S and T) -> ST (insert A) -> STA->(copy R)->STAR (replace O by T) -> START for a **total weight is 3.**

An explanation of the codes used in the table is as follows:

I = Insert, D = Delete, S = Substitute, C = Copy.

Moving down vertically one cell corresponds to an I (insertion), moving horizontally one cell corresponds to a D (deletion), and moving diagonally left to right corresponds to S (substitution).

Now any entry in the table corresponds to the moves and minimum weight needed to convert a given letter subsequence of "ASTRO" to a given letter in subsequence of "START". For example "AST" could be transformed into "ST" by 1 D, 2 C (1) i.e. 1 deletion and 2 copies for a total weight of 1. The last code 1 D, 3 C, 1 I, 1 S (3) in the table means that "ASTRO" can be transformed into START by: 1 deletion, 3 copies, 1 insertion, and I substitution for a total weight of 3. This total weight of 3 is what was noted above.

How the table was generated using DP is as follows:

Case 1 If the letters match use the previous diagonal value with the copy weight C increased by 1. For example the entry 1 D, 2 C (1) corresponding to the

	A	S	T
S			
T			1 D, 2 C (1)

In the table, is obtained from 1 D, 1 C (1) and becomes 1 D, 2 C (1) since the horizontal "T" matches the vertical "T"

Case 2 If the letters do not match then choose a minimum weight cell from the prior horizontal, vertical and diagonal cells. If the selected minimum weight cell was found in the horizontal direction increase the D weight by +1. If the selected minimum weight cell was found in the vertical direction increase the I weight by +1. Lastly, if the selected minimum weight cell was found in the diagonal direction increase the S weight by +1.

For example the entry 1 D, 2 C (1) corresponding to the

	A	S	T
S			
T			
A	1 D, 2 C, 1 I (2)		

In the table, is obtained from 1 D, 2 C (1) vertically by inserting 1 I.