CS3100 : Algorithms

Dynamic Pro. (pt 3)

Announcements

Recap: Dynamic programming is just smart recursion.

- Recurse - don't repeat

Often computed values are stored in some table for later lookups

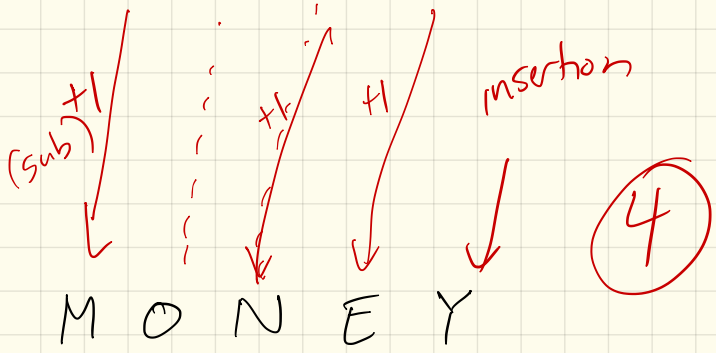
- or -

Can rearrange to fill table from ground up.

Edit Distance

The minimum number of deletions, insertions, or substitutions of letters to transform between two strings.

Ex: F O O D

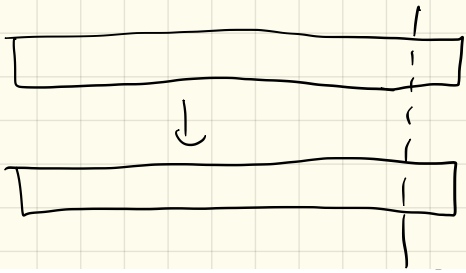


Recursive formulation:

If I align like this, can observe:

If you delete last (aligned) column, the rest will still be optimal for shorter substrings edit distance.

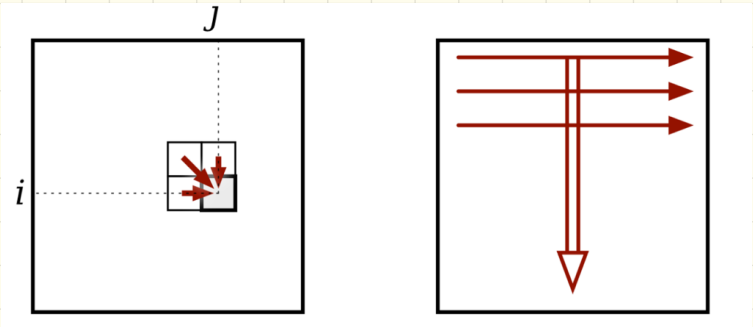
Why?

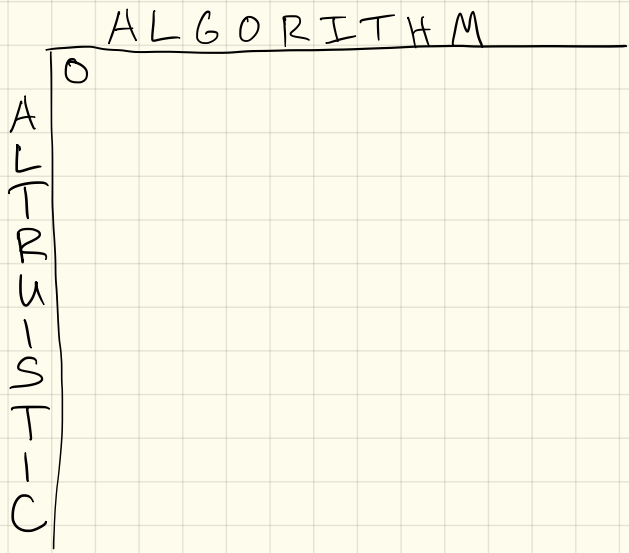


$$Edit(A[1..m], B[1..n]) = \min \left\{ \begin{array}{l} Edit(A[1..m-1], B[1..n]) + 1 \\ Edit(A[1..m], B[1..n-1]) + 1 \\ Edit(A[1..m-1], B[1..n-1]) + [A[m] \neq B[n]] \end{array} \right\}$$

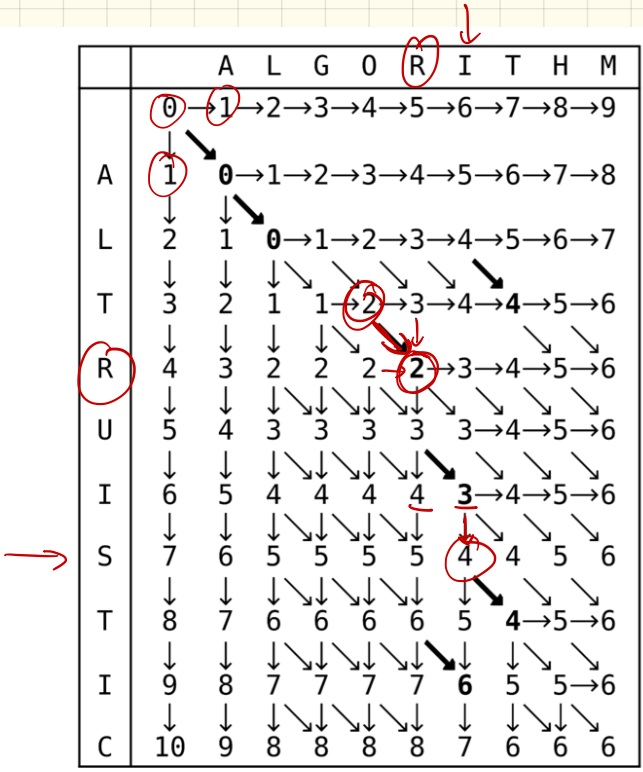
Turn into "nice" recursion:

$$Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \left\{ \begin{array}{l} Edit(i-1,j)+1, \\ Edit(i,j-1)+1, \\ Edit(i-1,j-1)+\left[A[i] \neq B[j]\right] \end{array} \right\} & \text{otherwise} \end{cases}$$





$$Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \left\{ \begin{array}{l} Edit(i-1,j)+1, \\ Edit(i,j-1)+1, \\ Edit(i-1,j-1)+\left[A[i] \neq B[j]\right] \end{array} \right\} & \text{otherwise} \end{cases}$$



		A	L	G	O	R	I	T	H	M
	0	1	2	3	4	5	6	7	8	9
A	1	**0**	1	2	3	4	5	6	7	8
L	2	1	**0**	1	2	3	4	5	6	7
T	3	2	1	1	2	3	4	**4**	5	6
R	4	3	2	2	2	**2**	3	4	5	6
U	5	4	3	3	3	3	3	4	5	6
I	6	5	4	4	4	4	**3**	4	5	6
S	7	6	5	5	5	5	4	4	5	6
T	8	7	6	6	6	6	5	**4**	5	6
I	9	8	7	7	7	7	**6**	5	5	6
C	10	9	8	8	8	8	7	6	6	6

The memoization table for *Edit*(ALGORITHM, ALTRUISTIC)

```
A L G O R   I   T H M
A L T   R U I S T I C
```

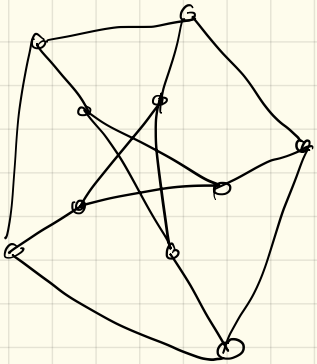
Correctness:

Runtime

```
EditDistance(A[1..m], B[1..n]):
  for j ← 1 to n
      Edit[0, j] ← j

  for i ← 1 to m
      Edit[i, 0] ← i
      for j ← 1 to n
          if A[i] = B[j]
              Edit[i, j] ← min {Edit[i − 1, j] + 1, Edit[i, j − 1] + 1, Edit[i − 1, j − 1]}
          else
              Edit[i, j] ← min {Edit[i − 1, j] + 1, Edit[i, j − 1] + 1, Edit[i − 1, j − 1] + 1}
  return Edit[m, n]
```

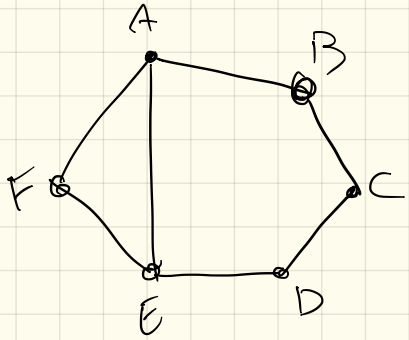
Dynamic Pro. on trees

Dfn: An independent set in a graph is a subset of the vertices that have no edges going between them.

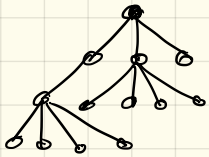


Why should we care?

Can we be greedy?



Even on a tree?



How would we do the recursion?

Consider any node v:

Pseudocode:

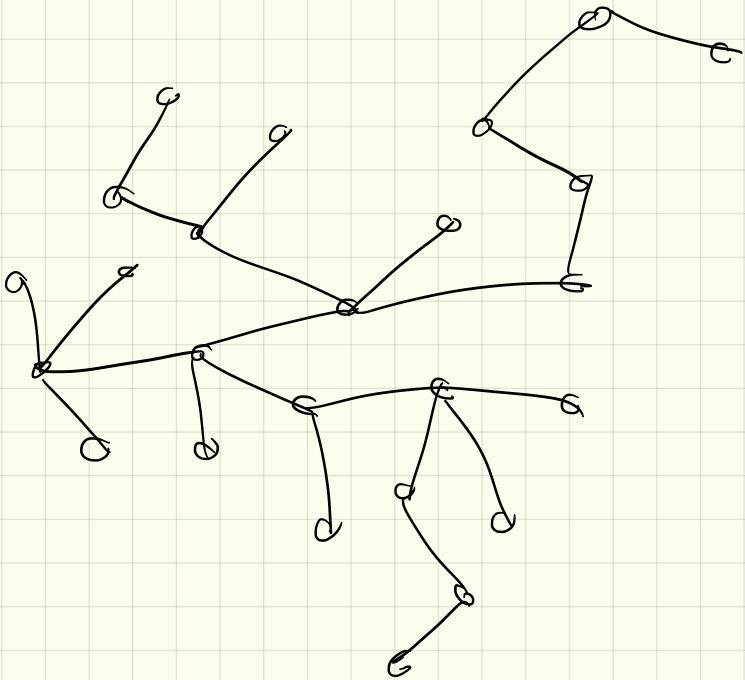
```
MaximumIndSetSize(G):
    if G = ∅
        return 0

    v ← any node in G
    withv ← 1 + MaximumIndSetSize(G \ N(v))
    withoutv ← MaximumIndSetSize(G \ {v})
    return max{withv, withoutv}.
```

Runtime?

In a tree: (or acyclic graph)

- Each component of $T-v$ is also a tree!

Idea:



Pseudocode

```
MaximumIndSetSize(v):
  withoutv ← 0
  for each child w of v
      withoutv ← withoutv + MaximumIndSetSize(w)
  withv ← 1
  for each grandchild x of v
      withv ← withv + x.MIS
  v.MIS ← max{withv, withoutv}
  return v.MIS
```

Data structure?

Nicer way:

```
MaximumIndSetSize(v):
  v.MISno ← 0
  v.MISyes ← 1
  for each child w of v
      v.MISno ← v.MISno + MaximumIndSetSize(w)
      v.MISyes ← v.MISyes + w.MISno
  return max{v.MISyes, v.MISno}
```

Correctness:

Runtime: