Algorithm Analysis and Design (CS1.301)

Monsoon 2021, IIIT Hyderabad
22 September, Wednesday (Lecture 10)

Dynamic Programming (contd.)

Edit Distance

The edit distance is the minimum number of insert/delete/overwrite operations to convert one word to another.

A natural way to do this is to consider how two strings can be aligned. For example, consider the edit distance between `SNOWY` and `SUNNY`.

```
S - N O W Y
S U N N - Y
```

The cost of an alignment is the number of columns in which the letters differ. The edit distance between two strings is the cost of their best possible alignment.

We will consider the subproblem of finding the edit distance $E(i,j)$ between some prefix of the first string, `x[1..i]`, and some prefix of the second `y[1..j]`.

Now, for $E(i,j)$, the rightmost column can only be one of three things: `x[i]` aligned with `-`, `-` aligned with `y[j]`, or `x[i]` aligned with `y[j]`.
This gives us $E(i,j) = \min\{1+E(i-1,j), 1+E(i,j-1), \text{diff}(i,j)+E(i-1,j-1)\}$, where $\text{diff}(i,j) = 0$ if `x[i] = y[j]` and 1 otherwise.

We maintain a grid of the $E(i,j)$ values. Since we need to compute $E(i-1,j)$, $E(i,j-1)$ and $E(i-1,j-1)$ before $E(i,j)$, we can fill the table row by row, from left to right.

```
for i = [0..m]:
    E(i,0) = i
for j = [0..n]:
    E(0,j) = j
for i = [1..m]:
    for j = [1..n]:
        E(i,j) = min(1+E(i-1,j), 1+E(i,j-1), diff(i,j) + E(i-1,j-1))
```

Thus this has a running time of $O(mn)$.

If we consider each cell in the table a node, and edges from up to down, left to right, and diagonally down and to the right between all pairs of adjacent nodes, we have the DAG for the problem.