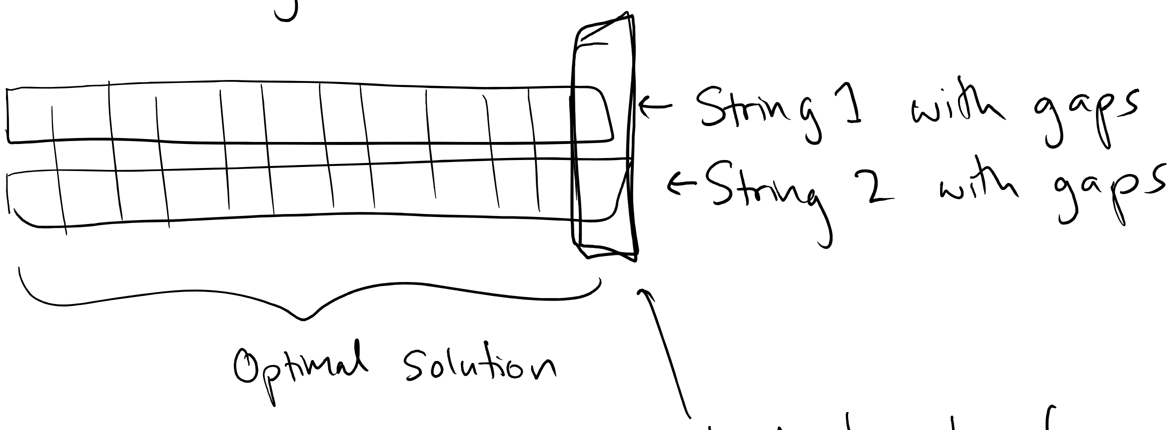
S.KIMMEL

Create D.P. Algorithm

$\boxed{\underline{1}}$



Form of optimal solution:

(i) String 1 has gap, String 2 does not

(ii) String 2 has gap, String 1 does not

String 1 & String 2 have letters at final position and

(iii) match

(iv) mismatch

S.KIMMEL

$\boxed{2}$ Let x' be the first $n-1$ letters of x
Let y' " $n-1$ " of y

Show how optimal solution is formed from optimal solution to subproblems in

(i) If A is optimal alignment for (x, y), A is first optimal alignment for (x, y'), then $\boxed{\begin{matrix} - \\ y_m \end{matrix}}$

(Usual proof by contradiction, we'll skip)

(ii) If A is optimal alignment for (x, y), A is first optimal alignment for (x', y), then $\boxed{\begin{matrix} x_n \\ - \end{matrix}}$

(iii) and (iv) If A is optimal alignment for (x, y), A is first optimal alignment for (x', y'), then $\boxed{\begin{matrix} x_n \\ y_m \end{matrix}}$

match or mismatch

S.KIMMEL

3 Create recurrence / Pseudo code to fill out array

$P(i,j)$ = penalty of optimal alignment of $(x_1, \dots x_i)$ and $(y_1, \dots y_j)$

for $(i=1$ to $n)$
  $P(i,0) = i \cdot P_{gap}$
for $(j=1$ to $m)$
  $P(0,j) = j \cdot P_{gap}$

for $(i=2$ to $n)$
  for $(j=2$ to $m)$
    $P(i,j) = \min:$

$$\{ P(i,j-1) + P_{gap},\ P(i-1,j-1) + P_{gap}$$
$$+ P(i-1,j-1) + \Delta_{ij} \cdot P_{mus} \}$$

$$\Delta_{ij} = \begin{cases} 0 & \text{if } x_i = y_j \\ 1 & \text{if } x_i \neq y_j \end{cases}$$

Running Time: $O(nm)$