align.py

```python
#############################################################################
# Implementation of the Needleman-Wunsch Global Pairwise Alignment algorithm
#############################################################################

def buildTable(X, Y, score, gap):

    # create table of zeros with dimension (1+|X|)-by-(1+|Y|)
    opt = [ [0] * (1+len(Y)) for _ in range(1+len(X))]

    for k in range(1+len(Y)):
        opt[0][k] = k*gap       # initialize top row
    for j in range(1+len(X)):
        opt[j][0] = j*gap       # initialize left column

    for j in range(1,1+len(X)):
        for k in range(1, 1+len(Y)):
            option1 = opt[j-1][k-1] + score(X[j-1],Y[k-1])  # align last chars
            option2 = opt[j-1][k] + gap                     # last of X with gap
            option3 = opt[j][k-1] + gap                     # last of Y with gap
            opt[j][k] = max(option1, option2, option3)
    return opt

def optScore(X, Y, table):
    return table[len(X)][len(Y)]   # bottom-right corner

def reconstructSolution(X, Y, table, score, gap):
    first = ''         # alignment for X
    second = ''        # alignment for Y
    glue = ''          # line showing matches/mismatches

    # start reconstruction at bottom-right of table
    j = len(X)
    k = len(Y)

    while j>0 or k>0:

        if j>0 and k>0 and table[j][k] == table[j-1][k-1] + score(X[j-1],Y[k-1]):
            # option1 above; match X[j-1] with Y[k-1]
            first =  X[j-1] + first
            second = Y[k-1] + second
            if X[j-1] == Y[k-1]:
                glue = '|' + glue   # designate match
            else:
                glue = '.' + glue   # designate mismatch
            j = j-1
            k = k-1
        elif j > 0 and table[j][k] == table[j-1][k] + gap:
            # option2 above; match X[j-1] with a gap in Y
            first  = X[j-1] + first
            second = '-' + second
            glue   = ' ' + glue
            j = j-1
        else:
            # option3 above; match gap in X with Y[k-1]
            first  = '-'  + first
            second = Y[k-1] + second
            glue   = ' '  + glue
            k = k-1

    return first,glue,second

```