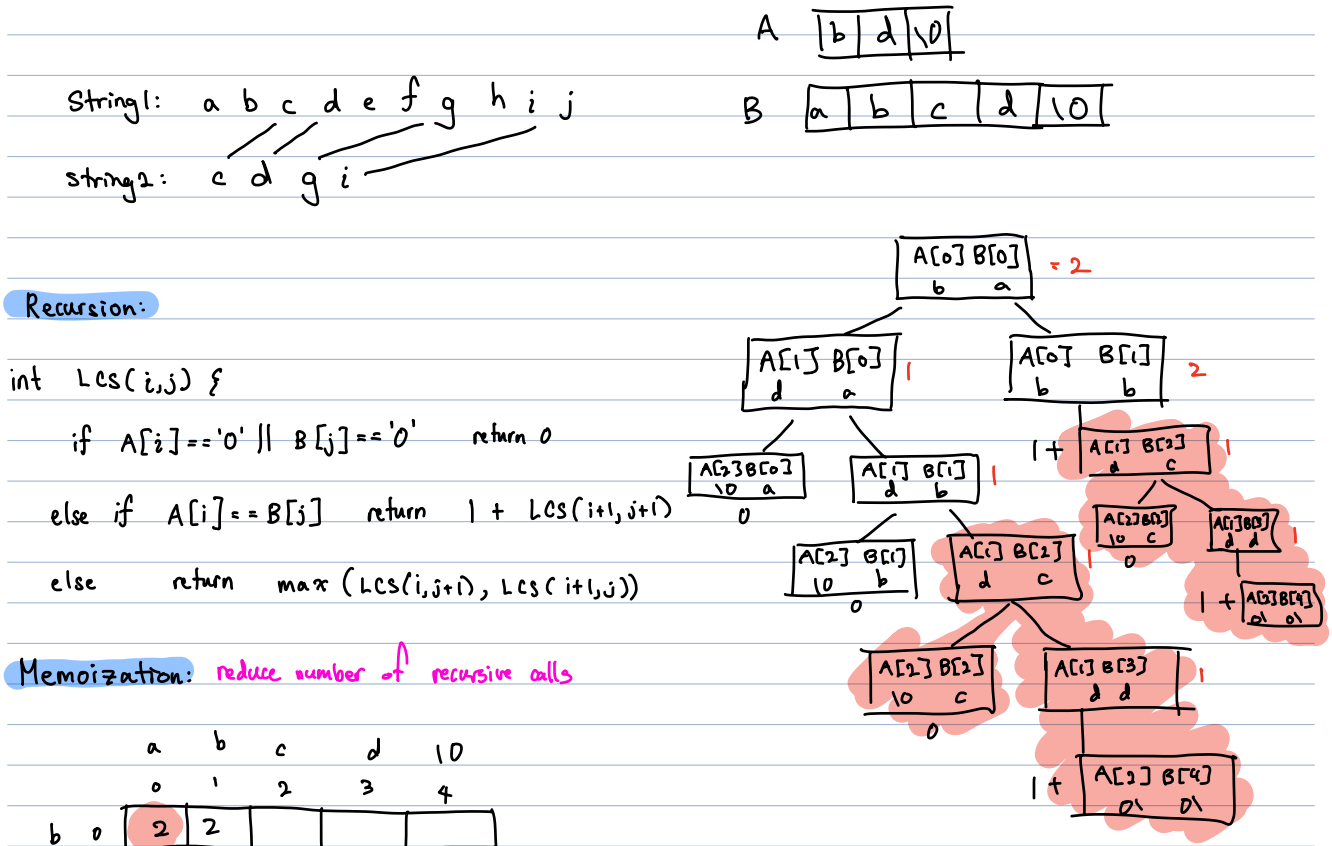
A
b
d
\0
String1: a b c d e f g h i j
string2: c d g i
B
a
b
c
d
\0
Recursion:
int LCS(i,j) {
if A[i] == '0' || B[j] == '0' return 0
else if A[i] == B[j] return 1 + LCS(i+1, j+1)
else return max (LCS(i,j+1), LCS(i+1,j))
A[0] B[0]
b a
= 2
A[1] B[0]
d a
1
A[0] B[1]
b b
2
A[2] B[0]
\0 a
0
A[1] B[1]
d b
1
1 + A[1] B[2]
d c
1
A[2] B[1]
\0 b
0
A[1] B[2]
d c
1
A[2] B[2]
\0 c
0
A[1] B[3]
d d
1
1 + A[2] B[3]
0\ 0\
1 + A[3] B[4]
0\ 0\
Memoization: reduce number of recursive calls
a b c d \0
0 1 2 3 4
b 0 2 2
d 1 1 1 1 1
\0 2 0 0 0 0
top-down
O(m x n) time complexity
a b c d \0
0 1 2 3 4
b 0 0 0 0 0 0
d 1 0 0 1 1 1
\0 2 0 2
if A[i] == B[j]:
LCS[i,j] = 1 + LCS[i-1,j-1]
else:
LCS[i,j] = max(LCS[i-1],j], LCS[i,j-1])