```
def LCS-Soln(X,Y,L):
  solution = []
  j,k= len(X), len(Y)
  while L[j][k] > 0:                         // common chars remain
    if X[j-1] == Y[k-1]:
      solution.append(X[j-1])
      j-=1
      k-=1
   elif L[j-1][k] >= L[j][k-1]:
      j-=1
  else
      k-=1
 return ''.join(reversed(solution))          // return left-to-right version
```

Text Compression/Greedy Method

→ Huffman Algo

- begins with each of d distinct chars of string X to encode being the root node of single-node BT
- in each round algo takes the 2 BT's with smallest freq. and merges into single BT
- repeat until there is only one BT

→ each iteration of while loop can be implemented in O(log d) using prio queue/heap

- also takes 2 nodes out of Q and adds one, repeated n-1 times until one node is left in Q

Psuedocode:

```
Compute frequency f(c) of each char in X
Init priority queue Q
for each char c in X do
    Create single node binary tree T storing c
    insert T into Q with key f(c)
while len(Q) > 1 do
    (f1,T1) = Q.remove_min()
    (f2,T2) = Q.remove_min()
    Create new bt T with left subtree T1 and right T2
    Insert T into Q with key f1+f2
(f,T) = Q.remove_min()
return T
```