

**Scratch**

S = itwasthebestofqnltimes

**Problem Formulation**

T[i] = Whether the sequence T[1….i] is a valid string

**Recurrence**

T[i] = We will loop back from i-1 to 1 using iterator j and check whether at any j, T[j] is true and dict(T[j+1…i]) is also true. Or if dict(T[1…i]) is true. If yes, we will set T[i] to true, else false.  
  
T[i] = any ((for j in 1 to i-1, if T[j] is true and dict(T[j+1…i]) is true) or dict(T[1…i]) is true

**Base case**

For all i in 1 to n, T[i] = false

**Sample Run (rhymes with Temple Run)**

S = i t w a s t h e b e s t o f t i m e s

T = T T F F T F F T F F F T F T F F F T T

S = i t w a s t h e b e s t o f q n l t i m e s

T = T T F F T F F T F F F T F T F F F F F F F F

**Return Value**

T[n] indicates whether the whole string is valid or not.  
For resolving the valid sequence if the whole sequence is known to be valid, we need to iterate over T from 1 to n, detect the word ending indices by detecting transition from T to F, insert a space after the word ending index and output the resultant string.

**PseudoCode**

for i = 1-> n ----------------- (1)

T[i] = False

for i = 1-> n ----------------- (2)

if (dict(S[1….i]))

T[i] = True

Else

for j = i-1-> 1 ----------------- (3)

if (T[j] == True) and (dict(T[j+1….i]) == True)

T[i] = True

If T[n] == True

k = 1

for i = 1->n-1 ----------------- (4)

Sentence[k] = S[k]

if (T[i] == True) and (T[i+1] == False)

k = k+1

Sentence[k] = ‘ ‘

k = k+1

Sentence[k] = S[k]

Return T[n], Sentence

Else

Return T[n], “Invalid String”

**Runtime Complexity**

Loop 1 takes O(n) time.   
The loops 2 and 3 are nested, hence will take O(n^2) time in worst case.  
The Loop 4 will also take O(n).   
All other statements take O(1) time.   
Hence, Total time complexity is O(n^2).