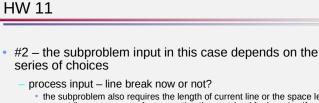


HW 11

CPSC 327: Data Structures and Algorithms . Spring 2025

 avoid unnecessary work – even though the result may still be polynomial, it could be a better polynomial
 e.g. a computeLineLength(i,j) function to compute the cost of the remaining spaces for a line containing words i through j is a nice idea for organization, but consider its context
<pre>for j = i to n-l do length ← computeLineLength(i,j)</pre>
 this adds up the lengths of words ii, then ii+1, ii+2, ii+3, – a lot of

- this adds up the lengths of words *i..i*, then *i..i*+1, *i..i*+2, *i..i*+3, ... a lot of repeated computation
- instead incrementally add the length of word *j* to the running total in each iteration



- the subproblem also requires the length of current line or the space left on current line – necessary for computing the cost (and for knowing if not breaking now is an option)
- base case cost >= 0 because it is necessary to consider the cost of spaces on the current line if there are words on it
- produce output where is the next line break?
 - the subproblem only needs the remaining words because this decision is about the entire contents of the line – the cost of the space left can be computed from the length of the words between the current point and the next line break
 - base case cost is 0 because there are no words left to go on a line (the current line is empty)

CPSC 327: Data Structures and Algorithms • Spring 2025