Gabe McKay
Terisa Le
Reeve Masilamani

ECS 122A
October 22, 2025

Homework 2

Problem 2

$dp[i] \doteq$ The minimum penalty of the words from $1..i$ with words[i] being the last word.
The words array represents our input.
$dp[0] = 0$
Recurrence:

```
$j \leftarrow 0$
$minPenalty \leftarrow \infty$
len ← 0
while $len < 32$ and $i - j > 0$:
  len += words[i - j].length
  j = j + 1
  penalty = calculatePenalty(len)
  minPenalty ← min(dp[i - j] + penalty, minPenalty)
  len += 1
```

Algorithm:

```
for i = 1 to n:
  $j \leftarrow 0$
  $minPenalty \leftarrow \infty$
  len ← 0
  while $len < 32$ and $i - j > 0$:
    len += words[i - j].length
    j = j + 1
    penalty ← $(16 - len)^2$
    if i = n:
      if $len > 16$ break out
      minPenalty ← min(dp[i - j], minPenalty)
    else:
      minPenalty ← min(dp[i - j] + penalty, minPenalty)
    len += 1
  dp[i] = minPenalty
```

Runtime: We do n * O(1) work meaning O(n) work overall.

We check when i = n to see if we are on the last line in order to both ignore the penalty and also throw out any invalid solutions that use more than 16 characters on the last line.

By the time we get to dp[i], we have filled $dp[1..i-1]$, through our bottom up loop.

In each recurrence call, we consider filling a line with a number of words. We look at many cases where a line could be filled with one word up to 32 characters + extra characters from the last word. This covers all potential cases since any line with more than 32 characters, a potential word would incur a bigger penalty at the end of such a line rather than just being put on the next line.

Overall, this means that the algorithm looks at every possible case and returns dp[n] as the final answer minimizing the penalty.