Step 0: Recursive structure.

Consider an arbitrary optimum solution $s_1 \cdots s_{p_\ell} / s_{p_\ell+1} \cdots s_m$ (for some $1 \leqslant \ell \leqslant k$). Then breaks $p_1, \ldots, p_{\ell-1}$ must be performed in some optimum order on $s_1 \cdots s_{p_\ell}$ and breaks $p_{\ell+1}, \ldots, p_k$ must be performed in some optimum order on $s_{p_\ell+1} \cdots s_m$—otherwise we would get a better overall solution.

Step 1: Array definition.

Let $T[i,j]$ be the minimum time to perform all the breaks on substring $s_{p_i+1} \cdots s_{p_j}$, for $0 \leqslant i < j \leqslant k+1$ (where we set $p_0 = 0$ and $p_{k+1} = m$).

Step 2: Recurrence relation.

- $T[i,i+1] = 0$ for $0 \leqslant i \leqslant k$ (no break in between positions p_i and p_{i+1}).
- $T[i,j] = \min_{\ell=i+1}^{j-1} \{(p_j - p_i + 1) + T[i,\ell] + T[\ell,j]\}$ for $0 \leqslant i < j-1 \leqslant k$ (by the reasoning in Step 0, where $p_j - p_i + 1$ is the cost for the break at p_ℓ).

Step 3: Iterative algorithm.

Compute values for decreasing i to ensure smaller subproblems solved first.

```
for i ← k, k−1, ..., 0:
    T[i, i+1] ← 0
    for j ← i+2, i+3, ..., k+1:
        T[i, j] ← ∞
        for ℓ ← i+1, i+2, ..., j−1:
            t ← T[i, ℓ] + T[ℓ, j] + p_j − p_i + 1
            if t < T[i, j]:
                T[i, j] ← t
```

Step 4: Solution reconstruction.

Use second array $U[i,j]$ to record value of ℓ that makes $T[i,j] = T[i,\ell] + T[\ell,j] + p_j - p_i + 1$, then recursively build break order using values of U.

```
MinCost(p_0 = 0, p_1, ..., p_k, p_{k+1} = m):
    for i ← k, k−1, ..., 0:
        T[i, i+1] ← 0;   U[i, i+1] ← 0
        for j ← i+2, i+3, ..., k+1:
            T[i, j] ← ∞;   U[i, j] ← 0
            for ℓ ← i+1, i+2, ..., j−1:
                t ← T[i, ℓ] + T[ℓ, j] + p_j − p_i + 1
                if t < T[i, j]:
                    T[i, j] ← t;   U[i, j] ← ℓ
    return BreakOrder(U, 0, k+1)

BreakOrder(U, i, j):
    if U[i, j] = 0:
        return [ ]
    else:
        return [U[i, j]] + BreakOrder(U, i, U[i, j]) + BreakOrder(U, U[i, j], j)
```

Total running time is $\Theta(k^3 + k) = \Theta(k^3)$.