Recitation 17

Bottle Breaker

The hit new videogame *Green Zombie Atonement II* features a new minigame, where you can throw balls at a line of bottles, each labeled with a number (positive, negative, or zero). You can throw as many balls at the bottles as you like, and if a ball hits a bottle, it will fall and shatter on the ground. Each ball will either hit no bottle, exactly one bottle, or two bottles (but only when the two bottles were **adjacent in the original lineup**). If a ball hits two adjacent bottles, you receive a number of points equal to the product of the numbers on the bottles. Otherwise, if a ball hits zero or one bottle, you do not receive any points. For example, if the line of bottle labels is $(5, -3, -5, 1, 2, 9, -4)$, the maximum possible score is 33, by throwing two balls at bottle pairs $(-3, -5)$ and $(2, 9)$. Given a line of bottle labels, describe a efficient dynamic-programming algorithm to maximize your score.

Solution:

1. **Subproblems**
   - Points are independent of the order we hit the bottles.
   - All that matters is which bottles are paired.
   - Let $l(i)$ denote the label of bottle $1 \leq i \leq n$, where n is number of bottles
   - $\boxed{x(i)\text{: maximum score possible by hitting only bottles } 1 \text{ to } i}$
2. **Relate**
   - Either the last bottle is part of a pair, gaining points and leaving $i-2$ bottles remaining,
   - or the last bottle is not part of a pair, and we can evaluate the remaining $i-1$ bottles
   - $x(i) = \max\{x(i-2) + l(i) \cdot l(i-1), x(i-1)\}$
   - Subproblems $x(i)$ only depend on strictly smaller i, so acyclic
3. **Base**
   - No points possible if only zero or one bottles
   - $x(0) = x(1) = 0$
4. **Solution**
   - Solution is $x(n)$, the maximum considering all the bottles
   - Store parent pointers to reconstruct hit pairs

5. **Time**
    - # subproblems: $n + 1$
    - work per subproblem $O(1)$
    - $O(n)$ running time

Bottle Breaker Remix

After witnessing the popularity of *Green Zombie Atonement II* (GZA2), a competitor has released a knockoff videogame called *Blue Demon Confessions III* (BDC3). BDC3 features the exact same bottle-breaking minigame as in GZA2 (from PS8) with one key difference: whenever one or more bottles are hit and shatter on the ground, the remaining bottles **get pushed together**, so that the bottles on either side of the bottles that just broke **become adjacent**. As in GZA2, you may throw a ball to hit any single bottle or any pair of adjacent bottles; but in BDC3, you may hit adjacent bottle pairs even if they were not adjacent in the original lineup, to score the product of their labels. For example, if the lineup of bottle labels is $(5, -3, -5, 1, 2, 9, -4)$, then one sub-optimal[1] score attainable is 38: you could throw balls at pairs $(-3, -5)$, $(2, 9)$, and $(5, 1)$ (which become adjacent after $(-3, -5)$ are hit); or, you could throw a ball at pair $(2, 9)$, then throw a ball at 1 to get rid of that bottle, and then throw a ball at pair $(-5, -4)$, to achieve the same score. Given a line of n bottle labels, describe an efficient dynamic-programming algorithm to determine the maximum possible score.

Solution:

1. **Subproblems**
    - Points now depend on the order we hit the bottles
    - Let $l(i)$ denote the label of bottle $1 \leq i \leq n$, where n is number of bottles
    - $\boxed{x(i, j)\text{: maximum score possible by hitting only bottles } i \text{ to } j}$
2. **Relate**
    - Either bottle i contributes to score or not
    - If not scoring, pairs with no bottle, recurse on $x(i + 1, j)$
    - If scoring, pairs with some bottle k for $1 < k \leq j$, recurse on two disjoint subsets
    - $x(i, j) = \max \left\{ \begin{array}{l} x(i + 1, j), \\ \max\{l(i) \cdot l(k) + x(i + 1, k - 1) + x(k + 1, j) \mid i < k \leq j\} \end{array} \right\}$
    - Subproblems $x(i, j)$ only depend on strictly smaller $j - i$, so acyclic

[1] The optimal score for this lineup is 62, achievable by hitting pairs $(-3, -5)$, $(1, 2)$, and $(5, 9)$.

3. **Base**
    - Maximum score on 1 or fewer bottles is zero
    - $x(i, j) = 0$ for all $j - i < 1$
4. **Solution**
    - Solution is $x(1, n)$, the maximum considering all the bottles
5. **Time**
    - # subproblems: n^2
    - work per subproblem $O(n)$
    - $O(n^3)$ running time