PSO 2 — Week 9: Solutions

CS381, Fall 2026

Problem 1

(a) Numbers in a row

Let $B[i]$ be the largest total we can get by choosing from $w_1, \ldots, w_i$ only. Then $B[0] = 0$, $B[1] = w_1$, and for $i \geq 2$

$$B[i] \;=\; \max\big(B[i-1],\;\; B[i-2] + w_i\big).$$

```
B[0] := 0;  B[1] := w_1
for i = 2 to n: B[i] := max(B[i-1], B[i-2] + w_i)
return B[n]
```

Correctness. Take a best choice from $w_1, \ldots, w_i$. Either it does not use w_i, and then it is a choice from $w_1, \ldots, w_{i-1}$, worth at most $B[i-1]$; or it uses w_i, and then it cannot use w_{i-1}, so the rest of it is a choice from $w_1, \ldots, w_{i-2}$, worth at most $B[i-2]$, and the whole is worth at most $B[i-2]+w_i$. Both options can actually be achieved: adding w_i to a choice from $w_1, \ldots, w_{i-2}$ never puts two adjacent numbers together. So $B[i]$ is the larger of the two. $\square$

Recovering the choice. Walk back from $i = n$: if $i \leq 0$, stop; if $B[i] = B[i-1]$, w_i is not needed, so move to $i-1$; otherwise choose w_i and move to $i-2$.

Time. $O(n)$ to fill the table and $O(n)$ to walk back. If only the total is wanted, keep just the last two entries, for $O(1)$ extra space.

Taking the largest number first does not work: on $3, 4, 3$ it takes the 4, which blocks both 3s, while $3 + 3 = 6$ is better.

(b) Numbers in a circle

The only new constraint is that w_1 and w_n are adjacent, so a best choice skips at least one of them. If it skips w_n, it is a valid choice for the row $w_1, \ldots, w_{n-1}$. If it skips w_1, it is a valid choice for the row $w_2, \ldots, w_n$. Conversely, a valid choice for either row is valid on the circle, because neither row contains both w_1 and w_n. So the answer is

$$\max\big(\text{(a) on } w_1, \ldots, w_{n-1},\;\; \text{(a) on } w_2, \ldots, w_n\big),$$

which is two runs of (a), $O(n)$ in total. To recover the choice, walk back through whichever run gave the larger answer.

Problem 2

Put the ends of the stick in with the cuts: let $x_0 = 0$, let $x_1, \ldots, x_m$ be the cut positions $c_1, \ldots, c_m$, and let $x_{m+1} = L$. For $0 \leq i < j \leq m+1$, let $C[i][j]$ be the minimum cost of making all the cuts strictly between x_i and x_j on the piece from x_i to x_j. We want $C[0][m+1]$. A piece with no cuts inside costs nothing, so $C[i][i+1] = 0$, and for $j \geq i+2$

$$C[i][j] \;=\; (x_j - x_i) \;+\; \min_{i<k<j}\big(C[i][k] + C[k][j]\big).$$

```
x[0] := 0;  x[1..m] := c[1..m];  x[m+1] := L
for i = 0 to m: C[i][i+1] := 0
for gap = 2 to m+1:                          // shorter pieces first
    for i = 0 to m+1-gap:
        j := i + gap
        C[i][j] := (x[j] - x[i]) + min over i < k < j of (C[i][k] + C[k][j])
return C[0][m+1]
```

Correctness. Look at the piece from x_i to x_j and suppose its first cut is at x_k. That cut costs $x_j - x_i$, the length of the piece, and leaves two pieces, from x_i to x_k and from x_k to x_j. Every later cut lies in one of the two, and its cost is the length of whatever piece it is made in, which depends only on cuts made earlier *in that same piece*. So the two pieces never affect each other: the cheapest way to finish is to handle each one as cheaply as possible, at cost $C[i][k] + C[k][j]$. Trying every possible first cut x_k and taking the best gives the recurrence. The loop fills the table by increasing $j - i$, so the two smaller pieces are always ready when they are needed. □

Why intervals. The cost of a cut depends on both ends of the piece it falls in. So a subproblem has to remember both ends, which is why the table is indexed by pairs (i, j) rather than by a single prefix, as in the earlier dynamic programs.

Time. There are $O(m^2)$ pairs (i, j), and each takes a minimum over $O(m)$ choices of k, for $O(m^3)$ in total.

The example. With $x = 0, 4, 5, 6, 10$, the table gives $C[0][4] = 18$, from first cutting at 4 or at 6; cutting at 5 first costs 20. The cheap orders cut the outer stretches away first, so that the closely spaced cuts are made on short pieces. To output an optimal order rather than just its cost, record the minimizing k for each pair (i, j): the first cut is at x_k for $(0, m + 1)$, followed by the orders found in the same way for the two pieces.