CS 161 (Stanford, Winter 2026) Section 8

1 LCS Warm-up

1. Given sequences $X = \texttt{ABCB}$ and $Y = \texttt{BDCAB}$, fill in the LCS dynamic programming table $C[i, j]$ below, where $C[i, j]$ is the length of the LCS of $X[1 : i]$ and $Y[1 : j]$. Then use the completed table to recover an actual longest common subsequence.

	$\emptyset$	B	D	C	A	B
$\emptyset$	0	0	0	0	0	0
A	0					
B	0					
C	0					
B	0					

Solution

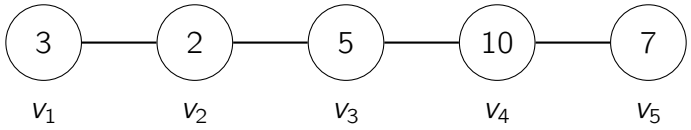
The completed table is:

	$\emptyset$	B	D	C	A	B
$\emptyset$	0	0	0	0	0	0
A	0	0	0	0	1	1
B	0	1	1	1	1	2
C	0	1	1	2	2	2
B	0	1	1	2	2	3

The LCS has length $\boxed{3}$. To recover it, trace back from $C[4, 5]$: $X[4] = Y[5] = \texttt{B}$ (match, go to $C[3, 4]$), then $C[3, 4] = C[3, 3]$ so decrement j to $(3, 3)$, then $X[3] = Y[3] = \texttt{C}$ (match, go to $C[2, 2]$), then $C[2, 2] = C[2, 1]$ so decrement j to $(2, 1)$, then $X[2] = Y[1] = \texttt{B}$ (match, go to $C[1, 0]$), done. Reading matches in order gives LCS = $\boxed{\texttt{BCB}}$.

2 DP Exercises

1. Consider a path graph with 5 vertices, where each vertex v_i has weight w_i as shown. An **independent set** is a subset of vertices such that no two are adjacent. Find a maximum weight independent set.



(a) Define $A(i)$ to be the weight of the maximum weight independent set considering only vertices $v_1, \ldots, v_i$. Write a recurrence for $A(i)$.

Solution

Either v_i is in the optimal set or it is not. If v_i is included, then v_{i-1} cannot be, so we get $w_i + A(i-2)$. If v_i is not included, we get $A(i-1)$. The recurrence is:

$$A(i) = \max\{A(i-1),\ w_i + A(i-2)\}, \quad A(0) = 0,\ A(1) = w_1.$$

(b) Compute $A(i)$ for $i = 0, 1, \ldots, 5$. Which vertices are in the optimal set?

Solution

$$A(0) = 0, \quad A(1) = 3, \quad A(2) = \max(3, 0+2) = 3,$$
$$A(3) = \max(3, 3+5) = 8, \quad A(4) = \max(8, 3+10) = 13,$$
$$A(5) = \max(13, 8+7) = \boxed{15}.$$

Backtracking: $A(5) = A(3) + w_5$, so v_5 is included. $A(3) = A(1) + w_3$, so v_3 is included. $A(1) = w_1$, so v_1 is included. The optimal set is $\boxed{\{v_1, v_3, v_5\}}$ with weight $3 + 5 + 7 = 15$.

(c) What is the runtime of the resulting algorithm?

Solution

Each entry $A(i)$ depends on at most two previous entries and takes $O(1)$ to compute. There are n entries, so the runtime is $\boxed{O(n)}$.

2. In how many ways can you tile a $2 \times n$ board using 1×2 dominoes (placed either horizontally or vertically)?

(a) Let $T(n)$ be the number of tilings of a $2 \times n$ board. Write a recurrence for $T(n)$ and justify it.

[**Hint:** *Consider how the rightmost column can be filled.*]

Solution

The rightmost column can be filled in two ways:

- One vertical domino covering the entire last column. The remaining board is $2 \times (n-1)$, giving $T(n-1)$ tilings.
- Two horizontal dominoes covering the last two columns. The remaining board is $2 \times (n-2)$, giving $T(n-2)$ tilings.

These cases are disjoint and exhaustive, so $T(n) = T(n-1) + T(n-2)$ with base cases $T(1) = 1$ and $T(2) = 2$.

(b) Compute $T(n)$ for $n = 1, 2, \ldots, 7$.

Solution

$T(1) = 1,\ T(2) = 2,\ T(3) = 3,\ T(4) = 5,\ T(5) = 8,\ T(6) = 13,\ T(7) = \boxed{21}$.

(c) Does this recurrence look familiar?

Solution

Yes. $T(n) = T(n-1)+T(n-2)$ is the **Fibonacci recurrence**. In particular, $T(n) = F_{n+1}$ where $F_1 = 1, F_2 = 1, F_3 = 2, \ldots$ is the standard Fibonacci sequence.

3 Activity Selection: Greedy Strategies

The activity selection problem asks: given n activities with start and finish times, find a maximum-size set of non-conflicting activities. In lecture, we saw that always selecting the activity with the earliest **finish time** (that doesn't conflict) gives an optimal solution.

For each of the following alternative greedy strategies, give a counterexample showing it does not always produce an optimal solution.

1. Always select the activity with the earliest **start time** that doesn't conflict with already-selected activities.

   Solution

   Consider activities $(0, 10)$, $(1, 3)$, $(3, 5)$, $(5, 7)$. The earliest-start-time greedy selects $(0, 10)$ first, which conflicts with all other activities, giving a set of size 1. However, $\{(1, 3), (3, 5), (5, 7)\}$ is a non-conflicting set of size $\boxed{3}$.

2. Always select the activity with the **shortest duration** that doesn't conflict with already-selected activities.

   Solution

   Consider activities $(0, 4)$, $(3, 5)$, $(4, 8)$ with durations 4, 2, 4. The shortest-duration greedy selects $(3, 5)$ first (duration 2). This conflicts with $(0, 4)$ since $3 < 4$, and with $(4, 8)$ since $4 < 5$, giving a set of size 1. However, $\{(0, 4), (4, 8)\}$ is a non-conflicting set of size $\boxed{2}$.

4 Minimum Refueling Stops

You need to drive from city A to city B, a total distance of L miles. Your car can travel at most D miles on a full tank. Gas stations are located at distances $g_1 < g_2 < \cdots < g_n$ from city A. You start with a full tank. You are guaranteed that the distance between any two consecutive stops is at most D (where city A and city B are also considered stops), so it is always possible to reach city B.

Design an algorithm that finds the minimum number of refueling stops needed to reach city B.

(a) Describe a greedy strategy for this problem.

Solution

Go as far as possible before refueling: at each point, drive to the farthest gas station you can reach on your current tank. Only stop to refuel when you cannot reach the next station (or the destination) without refueling.

(b) Apply your strategy with $L = 20$, $D = 7$, and stations at positions $3, 5, 8, 12, 14, 17$. List the positions where you stop to refuel.

Solution

- Start at position 0, range $[0, 7]$. Reachable stations: 3, 5. Farthest: 5. **Stop at 5.**
- At 5, range $[5, 12]$. Reachable stations: 8, 12. Farthest: 12. **Stop at 12.**
- At 12, range $[12, 19]$. Reachable stations: 14, 17. Cannot reach destination ($20 > 19$). Farthest: 17. **Stop at 17.**
- At 17, range $[17, 24]$. Destination at $20 \leq 24$. **Arrive.**

Total stops: $\boxed{3}$ (at positions 5, 12, 17).

(c) Prove your greedy strategy is optimal.

[**Hint:** *Show by induction that after k stops, the greedy car is always at least as far along as in any other solution.*]

Solution

Let $s_1, s_2, \ldots, s_m$ be the greedy stop positions and $s_1^*, s_2^*, \ldots, s_k^*$ be the stop positions in any optimal solution. We show $m \leq k$.

Claim: For all i, $s_i \geq s_i^*$.

Base case ($i = 1$): Both cars start at 0 with a full tank. The greedy car stops at the farthest reachable station, so $s_1 \geq s_1^*$.

Inductive step: Assume $s_i \geq s_i^*$. After refueling, both cars have a full tank of range D. Since $s_i \geq s_i^*$, the greedy car can reach every station that is reachable from s_i^* (and possibly more). The greedy car picks the farthest reachable station,

so $s_{i+1} \geq s^*_{i+1}$.
By the claim, $s_k \geq s^*_k$. The optimal solution reaches city B from s^*_k, meaning $s^*_k + D \geq L$. Since $s_k \geq s^*_k$, we also have $s_k + D \geq L$, so the greedy car can reach city B from its k-th stop as well. Therefore the greedy solution uses at most k stops, i.e. $m \leq k$. $\square$