Assembly Line Problem

Cars produced in factory with 2 assembly lines.

Each line has n stations (that add car parts).

- Stations $S_{1,j}$, $S_{2,j}$ for $j = 1, \ldots, n$
- Stations $S_{1,j}$, $S_{2,j}$ perform same task but possibly at different speeds.

- $a_{1,j}$ = time required at $S_{1,j}$
- $a_{2,j}$ = "" $S_{2,j}$

Cars usually stay on one line (time from $S_{1,j}$ to $S_{1,j+1}$ + $S_{2,j}$ to $S_{2,j+1}$ negligible)

"Special" Rush order cars can switch from line 1 to line 2 to speed up.

- $t_{1,j}$ = time to switch from $S_{1,j}$ to $S_{2,j+1}$
- $t_{2,j}$ = "" $S_{2,j}$ to $S_{1,j+1}$
Given values for $a_{ij}$, $a_{ij}$, $t_{ij}$, $t_{ij}$, find set of stations to visit that will minimize overall time.

Optimization problem - goal is to minimize or maximizes a specified value.

Other optimization problems? Coin changing, shortest path, not searching, sorting.

**Example:**

```
 7 → 9 → 3 → 4 → 8
```

```
8 → 5 → 6 → 4 → 5
```

*Note: Greedy doesn't work. $7 + (2) + 5 + (1) + 3 + 4 + 8 = 30$ (Optimal is 27)*

**Brute Force:** Consider all possible combinations.

For each of $n$ pairs of stations, 2 possibilities $\Rightarrow O(2^n)$

**Better:** Consider smaller versions of the problem, use solutions to those to solve larger + larger versions.

"Suppose we knew best time for both lines up to some station."

"Solution up to that station? Minimum of the 2 times."
"How to use this value to find best time to next pair (for lines 1, 2) of stations?"

Let:

\[ c(S_{1,j}) : \text{fastest time through } S_{1,j} \]

\[ c(S_{2,j}) : \quad \quad \quad \quad \quad \quad \quad S_{2,j} \]

How to express \( c(S_{1,j}) \), \( c(S_{2,j}) \) in terms of the previous sub-problem: \( c(S_{1,j-1}) \) \( c(S_{2,j-1}) \)?

\[
C(S_{1,j}) = \begin{cases} 
\min \{ & \begin{align*} 
(1) & \text{if prev station on line 1} \quad C(S_{1,j-1}) + 0 + a_{1,j} \quad \text{if } j > 1 \\
(2) & \text{"" "" "" ""} \quad 2 \cdot C(S_{2,j-1}) + t_{2j-1} + a_{1,j} 
\end{align*} \} 
\end{cases}
\]

Can use this for \( j = n, n-1, n-2 \ldots \) When to stop? When can the problem be solved directly?

\[ \Rightarrow \text{For } j=1: \]

\[ C(S_{1,1}) = a_{1,1} \quad \text{if } j=1 \]
Similarly:

\[ c(S_{2,j}) = \begin{cases} 
    a_{2,1} & \text{if } j = 1 \\
    \min \left\{ \begin{array}{l}
        c(S_{2,j-1}) + a_{2,j} \\
        c(S_{1,j-1}) + t_{1,j-1} + a_{2,j}
    \end{array} \right\} & \text{if } j > 1
\end{cases} \]

Dynamic Programming - solving a problem using solutions to sub-problems

When can we use dynamic programming?

If the problem exhibits:

- optimal sub-structure property - optimal solution to the main problem depends on optimal solutions to sub-problems

Main problem: \( C(S_{1,j}) \), \( C(S_{2,j}) \)
Sub problems: \( C(S_{1,j-1}) \), \( C(S_{1,j-2}) \), \( C(S_{1,j-3}) \), \( \ldots \), \( C(S_{2,j-1}) \), \( \ldots \)

Another problem that has this property? Coin changing!

Optimal number of coins for \( n = 15 \) depends on:

- optimal number of coins for \( 15 - 12 \)
- \( 15 - 5 \)
- \( 15 - 1 \)

Assembly Line Coin Change
(Tabulation vs. Memoization D.P.)