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 $S_{1,j}$ and $S_{2,j}$ perform same task but possibly at different speeds.
- $a_{1,j}$ = time required at $S_{1,j}$
- $a_{2,j}$ = "" at $S_{2,j}$

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

"Special" Rush order cars can switch from line1 to line2 to speed up.
- $t_{1,j}$ = time to switch from $S_{1,j}$ to $S_{2,j+1}$
- $t_{2,j}$ = "" to $S_{1,j+1}$
Given values for $a_{ij}$, $a_{2j}$, $t_{ij}$, $t_{2j}$:

find set of stations to visit that will minimize overall time.

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

Other optimization problems: coin changing, shortest path, not searching, sorting.

**Example:**

```
7 ——> 9 ——> 3 ——> 4 ——> 8

3

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 and larger versions.

Know times for station 1: 7, 8

How to use these for station 2?
"Consider just Line 1 for station (2). Two ways to get there: from Line 1 or from Line 2."

Station 2 Line 1 (2 choices):
From Line 1?: $7 + 9 = 16 \quad \Rightarrow \operatorname{Best\ option\ for\ Line}1,$
" 2?: $8 + 2 + 9 = 19 \quad \Rightarrow \operatorname{Station\ 2\ is\ to\ come\ from\ line\ 1.}\"

$C(S_{i,j})$: fastest time through $S_{i,j}$
$C(S_{2,j})$: " " " $S_{2,j}$

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


\[
C(S_{i,j}) = \min \left\{ \begin{array}{ll}
\text{(1) prev station on line 1} & C(S_{1,j-1}) + 0 + a_{i,j} \quad \text{if } j \geq 1 \\
\text{prev station on line 2} & C(S_{2,j-1}) + t_{2,j-1} + a_{i,j} \quad \text{if } j < 1
\end{array} \right. \\
\text{maximal time through prev station } + \text{ transfer time through station}
\]

Can use this for $j = 2, 3, \ldots, n$.

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

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

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

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.)
Implement \( w \) \( 2 \times n \) array \( c \):
\[
C[1:j, n]: \text{best time through } S_{1, n} \\
C[2:j, n]: \quad \text{"""" } S_{2, n}
\]

Make \( Car(a_1, a_2, t_1, t_2, n) \). \( \// a_1, a_2: \) lists of station times
\( \// t_1, t_2: \) """" transfer

// Base Cases
\( \// n: \) number of stations.
\[
C[\text{odd}, 0] = a_1, \quad \text{"""" time through } S_1 \\
C[\text{even}, 0] = a_2, \quad \text{"""" } S_2
\]

for (\( j = 2 \) to \( n \))
\( \// \) times through \( S_{1, j} \)
\[
\text{from1} = 0 + a_{1,j} \quad \text{\"\" coming from \ line 1} \\
\text{from2} = 0 + a_{2,j} \quad \text{\"\" transferring} \\
\quad \text{\"\" from line 2}
\]

if (\( \text{from1} < \text{from2} \))
\( C[\text{odd}, j] = \text{from1} \)
prevline[\( j \)] = 1 \( \// \) best prev station on \ line 1

else
\( C[\text{even}, j] = \text{from2} \)
prevline[\( j \)] = 2 \( \// \) best prev station on \ line 2

\( \// \) times through \( S_{2, j} \)
\[
\text{from2} = 0 + a_{2,j} \quad \text{\"\" coming from \ line 2} \\
\text{from1} = 0 + a_{1,j} + t_{1,j-1} \quad \text{\"\" transferring} \\
\quad \text{\"\" from line 1}
if(from2<from1)
    \text{prevline}[2][j]=2 \quad /\text{best prev station on line 2}

else
    \text{prevline}[2][j]=1 \quad /\text{best prev station on line 1}

//end for

//Final Min cost ?
if(c[1][n]<c[2][n])
    c^*=c[1][n]
    \text{lastline}=1 \quad /\text{last station on line 1}

else
    c^*=c[2][n]
    \text{lastline}=2 \quad /\text{last station on line 2}
Try for example. (W later)

(from line 1, from line 2)

Look at previous entries of table to fill next entry

\[
C^* = \min (29, 27) = 27
\]

Now, how to find the sequence of lines to visit?

* To Do: Update code to do this.

First, last line to visit?

= line that yields \( C^* \)

(add to code)

Now, sequence? Can see this in \( C \) array.
How to keep track of this while filling C?

Another array: prevline

prevline \[J][j] \quad (j = 2, n) \quad \text{previous best line before } S_{nj}

\[ \text{prevline } [2][j] \quad S_{nj} \]

\[ \langle \text{add to code} \rangle \]

Now trace prevline starting from lastline to -1.

lastline = 2 \Rightarrow 2 \leftarrow 2 \leftarrow 1 \leftarrow 2 \leftarrow 2 \leftarrow -1

(Station 5)

\[ \text{prevline } [2][5] = 2 \]

\therefore \text{best line at Station 4 was 2.}
Run Time: n Stations

to fill [[]][[]]: for loop from 1 to n: $O(n)$
"preLine[][]" : $O(n)$

Total time: $O(n)$