Assembly Line Problem

- Cars produced in factory with 2 assembly lines
- Each line has n stations (that odd 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:
    - $a_{1,j} = \text{time required at } S_{1,j}$
    - $a_{2,j} = \text{time required at } S_{2,j}$

- Cars can stay on the same line from one station to next (time from $S_{1,j}$ to $S_{1,j+1}$ and time from $S_{2,j}$ to $S_{2,j+1}$ is negligible)
- Or, cars can transfer from line 1 to line 2 and line 2 to line 1
  
  $t_{1,j} = \text{time to transfer from } S_{1,j} \text{ to } S_{2,j+1}$
  $t_{2,j} = \text{""""""""""""" S}_{2,j} \text{ """"""""""""" S}_{1,j+1}$
Goal: Given values for $a_{ij}, a_{ji}, t_{ij}, t_{ji}$, find sequence of lines to visit that minimizes total time.

Modified Goal: Start with finding minimum total time, then go back and find sequence.

Optimization problem - problem for which there are many correct solutions, but we want the "best" one, i.e., one that minimizes or maximizes a specified value.


Example:

Brute Force? Consider all possible combinations. For each of $n$ stations, 2 possibilities (Line 1 or 2) ⇒ Total # combinations = $2^n$.

Greedy? Always move to the line for which transfer time + station time is minimum.

$7 + (2) + 5 + (1) + 3 + 4 + 8 = 30 \leftarrow$ not optimal

Optimal = 27
Other ideas?

Consider smaller versions of the problem, use solutions to these to solve larger versions.

Find minimum time through Station 1.
Use this to find min time through Station 2.
And so on....

Min time through Station 1? \( \min (7, 8) = 7 \)
Group Work:
- Use these times to find min. time through station 2
- Find a general formula to find min. time through Station 2

"Consider line 1 for station 2."
Two ways to get there: from Line 1 or from Line 2.

Station 2, Line 1, (2 ways:)
From line 1? 7 + 9 = 16 \text{ min time for} \quad 2? 8 + 2 + 9 = 19 \quad \text{station 2, line 1}

Station 2, Line 2, (2 ways:)
From line 1? 7 + 2 + 5 = 14 \quad 2? 8 + 5 = 13 \quad \text{min time for station 2 line 2}

Note: Not always best to come from the same line.

100 \rightarrow 0 \quad \text{best option is to transfer from line 2.}
0 \rightarrow 101 \quad 101 \text{ vs } 2

Min time through station 2? = 13.
"Let's generalize this for any station $j$:"

\[
C(S_{1j}) = \text{fastest time through } S_{1j} \\
C(S_{2j}) = \text{ fastest time through } S_{2j}
\]

Need to express $C(S_{1j})$ and $C(S_{2j})$ in terms of previous problem: $C(S_{1j-1})$, $C(S_{2j-1})$.

1. **Dynamic Programming Formulation**

\[
C(S_{1j}) = \begin{cases} 
\min & \text{if prev station on line 1}: & C(S_{1j-1}) + 0 + a_{1j} & \text{if } j > 1 \\
2 & C(S_{2j-1}) + t_{2j-1} + a_{1j} & \text{if } j = 1.
\end{cases}
\]

Which one?

For which values of $j$? $j = 2, 3, \ldots, n$

For $j = 1$? $C(S_{11}) = a_{11}$ (2) Base Case
Similarly:

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

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: \( C(S_1, n) \), \( C(S_2, n) \)

Sub-problems: \( C(S_1, n-1) \), \( C(S_1, n-2) \), \( C(S_2, n-1) \), \( C(S_2, n-2) \).

Another problem with this property? Change-Making

Optimal number of coins for \( n = 15 \) depends on \( 15 - 12, 15 - 5, 15 - 1 \).

Assembly Line
D.P. with tabulation
Bottom-up

Change-Making
D.P. with memoization
Top-Down
For D.P. solution, need:

1. **Dynamic Programming Formulation** (already did this)
2. **Base cases** (can be solved w/o d.p. formulation)
3. **Size of array/matrix**? $2 \times n$
4. **How to fill array/matrix**.
5. **How to find final answer (value)** once array/matrix is filled.
6. **How to find sequence that corresponds to final value**.
7. **Running Time**.
Implement with 2xn array C:

\[ C[I][1..n] \text{ best time through } S_I, 1..n \]
\[ C[2][1..n] \text{ " " } S_{2,1..n} \]

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
\( C[I][0] = a_{1,i} \) // time through \(S_I\)
\( C[2][1] = a_{2,1} \) // " " \(S_2\),

4) How to fill matrix
\[
\text{for } (j = 2 \text{ to } n) \\
\text{// times through } S_I, j
\text{from 1 to } j = C[I][j-1] + a_{1,j} \quad \text{// coming from line 1 }
\text{from 2 to 1} = C[2][j-1] + t_{2,j-1} + a_{2,j} \quad \text{// transferring }
\quad \text{// from line 2 }
\]

\[
\text{if } (\text{from 1 to } j < \text{from 2 to } 1) \\
\quad \text{from 1 to } j = \text{from 1 to } j \\
\quad \text{prevline}[I][j] = 1 \quad \text{// best prev station on line 1 }
\] else

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

\text{// times through } S_{2, j}
\text{from 2 to 2} = C[2][j-1] + a_{2,j} \quad \text{// coming from line 2}
\text{from 1 to 2} = C[I][j-1] + t_{1,j-1} + a_{2,j} \quad \text{// transferring }
\quad \text{// from line 1.}
if (from2to2 < from1to2 )
  \[ \mathcal{C}_{2j} = \text{from2to2} \]
  \[ \text{prevline}[2][j] = 2 \] // best prev station on line 2

else
  \[ \mathcal{C}_{2j} = \text{from1to2} \]
  \[ \text{prevline}[2][j] = 1 \] // best prev station on line 1

else

(5) How to find final answer:

// Final min cost?
if (\[ \mathcal{C}_{1n} < \mathcal{C}_{2n} \])
  \[ \mathcal{C}^* = \mathcal{C}_{1n} \]
  \[ \text{lastline} = 1 \] // last station on line 1

else
  \[ \mathcal{C}^* = \mathcal{C}_{2n} \]
  \[ \text{lastline} = 2 \] // last station on line 2
Try for example:

(from line 1, from line 2)

\[
C = \begin{bmatrix}
1 & 7 & (16, 19) & (19, 17) & (21, 25) & (29, 32) \\
2 & 8 & (14, 13) & (26, 19) & (22, 23) & (29, 27)
\end{bmatrix}
\]

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

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 \(C^*\)
- \(\text{add to code}\)

Now, (6) find sequence of lines? Can see in \(C\) array.