Other examples of dynamic programming:

Assembly line scheduling:

- two assembly lines, each with n stations.
- time for station i is not necessarily the same in the two assembly lines.
- there is a time cost to switch between assembly lines.

Ex: Station 1  Station 2  Station 3

![Diagram of assembly line scheduling]

Question: What is the fastest way through the assembly line?

Brute force approach of trying all possibilities would have to consider $2^n$ solutions.

Try dynamic programming:

1) Recursive description of solution:
    Key idea: fastest way to get from station i through station j in line 1 is:
    fastest way through station j-1, then either stay in line 1 or switch to line 2,
whichever is faster.

Let $F[i, j]$ be the fastest time to get from the entry through station $j$ on line $i$, $j = 1, \ldots, n$ and $i = 1, 2$.

Desired final answer is

$$\min (F[1, n] + x_1, F[2, n] + x_2).$$

**Formula for $F$:**

**Basic:** $F[i, 1] = e_i + a_{i, 1}$, $i = 1, 2$.

**Induction:**

For $j > 1$:

$$F[i, j] = \min (F[i, j-1] + a_{i,j}, F[i', j-1] + t_{i', j-1} + a_{i,j}).$$

where $i'$ is the "other" line,

i.e., if $i = 1$, then $i' = 2$ and if $i = 2$, then $i' = 1$.

Compute dependencies:

$F[i, j]$ depends on $F[i, j-1]$ and $F[i', j-1]$. 
Good order: go from left to right, doing both rows.

So there are $O(n)$ entries to compute and each takes $O(1)$ time.
So total time is $O(n)$.

See text for pseudocode & how to keep track of the schedule in addition to the time.