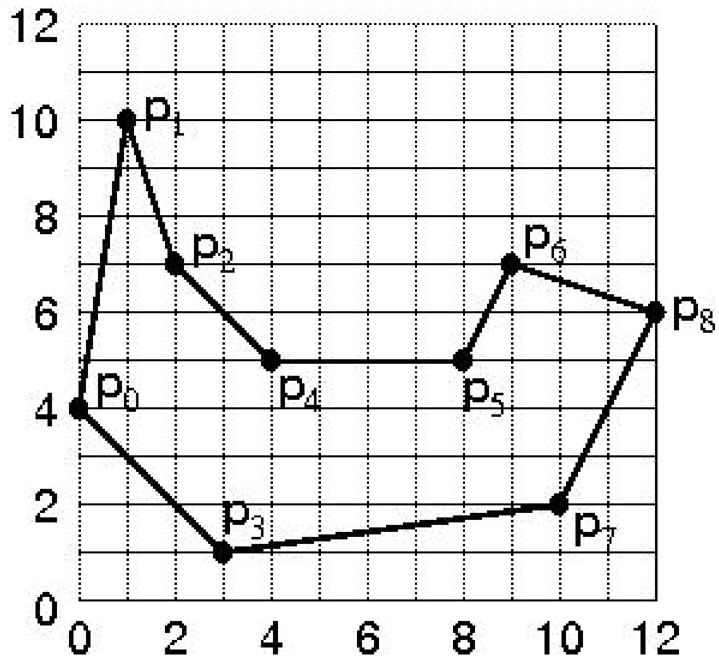
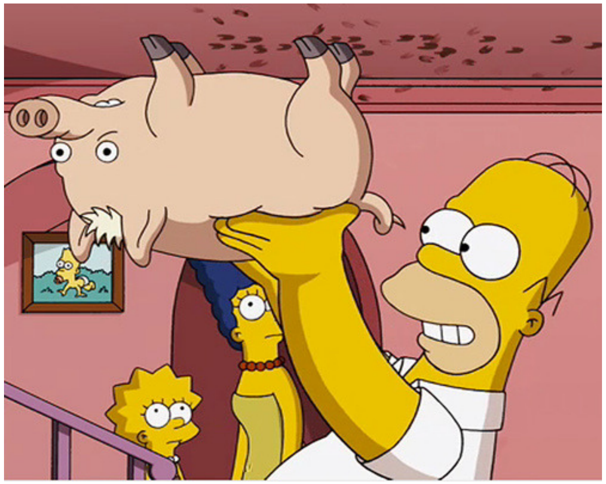
Spider Pig's Doughnut-Eating Spree: Solution

Problem: Given a set of coordinates. Find the shortest path that moves monotonically to the right and then monotonically to the left.

Example: (0,4) (1,10) (2,7) (3,1) (4,5) (8,5) (9,7) (10,2) (12,6)





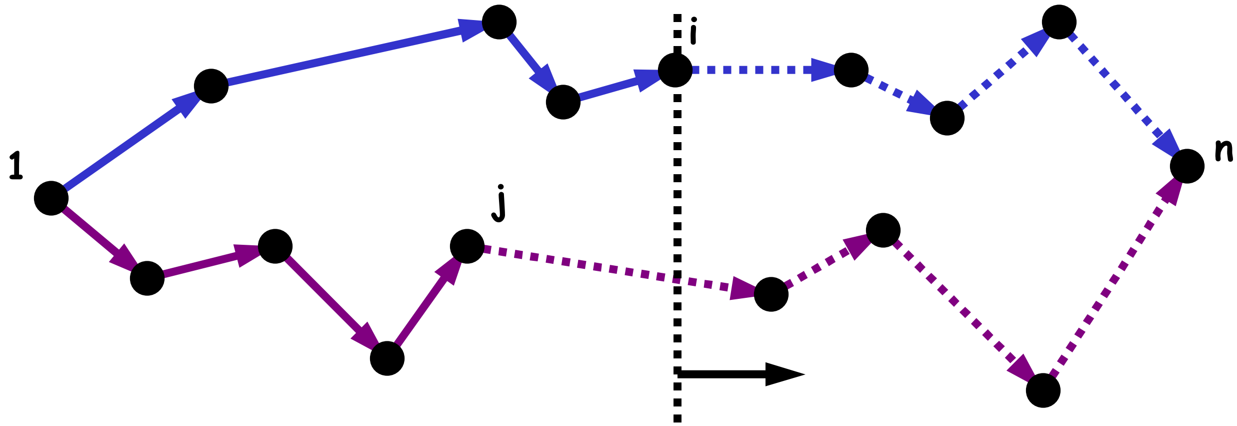
Dynamic Programming Solution

Dynamic Programming:

- Let n be the number of doughnuts.
- We construct an array C[n,n], where C[i,j] holds the cost of a **partial solution**.
- C[i,j] is sum of lengths of the shortest pair of paths that traverse the doughnuts 0..max(i,j), where the outbound path **ends at doughnut i** and the inbound path **ends at doughnut j**.

Basis Case C[1,1] = 0.

Final Optimal Path Length: C[n,n].

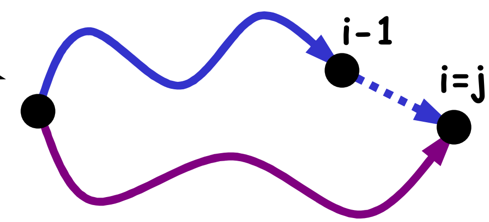


Dynamic Programming Solution

Computing C[i,j]: (Assume that $j \le i$. Other case symmetrical.)

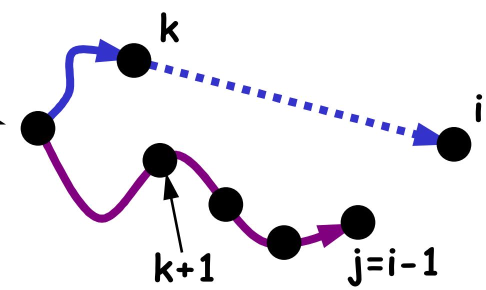
j = i: Edge (i-1,i) must be on one path or the other. Take the shorter of the two:

$$C[i,i] = \begin{cases} C[i-1,i] + \text{distance}(i-1,i) \\ C[i,i-1] + \text{distance}(i-1,i) \end{cases}$$



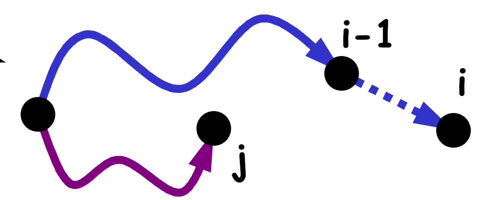
j = i-1: Doughnut i is preceded by one of i-2, i-3, ..., 0. Don't know which, so try them all:

$$C[i,j] = \min_{0 \le k \le i-2} (C[k,j] + \text{distance}(k,i)).$$



j < i-1: Doughnut i must be preceded by i-1:

$$C[i,j] = C[i-1,j] + \text{distance}(i-1,i).$$



Recovering the Path

Observe: Each value $C[i,j]$ arises by adding a single segment onto some prior entry:

- If $i > j$: Prior entry is of the form $C[k, j]$.
- If $i < j$: Prior entry is of the form $C[i, k]$.
- Path Array: $P[i,j]$ holds this value k of the prior entry.

Computing P:

Basis case: $P[1,1] = 0$.

General cases:

$j = i$: $C[i, i] = C[i-1, i] + \text{distance}(i-1, i)$.
$P[i, i] = i - 1$.

$j = i-1$: $C[i, j] = \min_{0 \le k \le i-2} (C[k, j] + \text{distance}(k, i))$.
$P[i, j] =$ the value k giving the minimum

$j < i-1$: $C[i, j] = C[i-1, j] + \text{distance}(i-1, i)$.
$P[i, j] = i - 1$.

Computing the Path:

- Start with $P[n,n]$. By tracing the values of $P[i, j]$ back, we can determine the last edge added to the path. Repeat, working our way back to start.