Dynamic Programming

Repeated use of a small (polynomial) number of subproblems to solve a given problem optimally.

We will see:

- shortest paths (Dijkstra)
- shortest paths using at most k edges
- all pairs shortest paths

- longest increasing subsequence
- edit distance / longest common substring

- knapsack
- matrix chains
- polygon triangulation

- max indep. set in a tree
$A = (V, E) \quad \lambda: E \rightarrow \mathbb{R}^+$

$\lambda(e) \geq 0$

Find shortest path from $s$ to $t$ in $A$.

Shortest $s$-$t$ path = shortest $s$-$u$ path + $\lambda(u, t)$

for some $u \in V \setminus \{s, t\}$.

Maintain cumulative distances from $s$ to all nodes.

$T(s) = 0 \quad T(u) = \lambda(s, u) \quad$ if $(s, u) \in E$

$T(u) = \infty \quad (s, u) \notin E$

$S = \{s\}$ - Find node $u$ with minimum $T(u)$

and add to $S$.

Update $T(v) = \min \{T(v), T(u) + \lambda(u, v)\}$

Nodes are included in $S$ in order of increasing distance from $s$.

Complexity? \( O(|V|^2) \) - $|V|$ steps

- Each is a min

Can be made \( O(|E| + |V| \log |V|) \) with efficient data structures
shortest s-t path with at most k edges.

\[ P(a, b, i) : \text{length of shortest a, b path with at most } i \text{ edges.} \]

\[ P(a, b, i) = \min_{c} \{ P(a, c, i-1) + l(c, b), P(a, b, i-1) \} \]

Fix a. \hspace{1cm} u_1 \hspace{1cm} u_2 \hspace{1cm} \ldots \hspace{1cm} u_i \\
1 \hspace{1cm} l(a, u_1) \hspace{1cm} \ldots \hspace{1cm} l(a, u_i) \\
2 \\
\vdots \\
K \\
\hdots \\
\text{Fill this table.}

Two: Size of Table \times \text{two per entry} \leq K \times n \times n
All pairs shortest paths.

Using Dijkstra: $|V| |E|$

Negative edges allowed (but no negative cycles)
Dijkstra does not work.

Floyd-Warshall: Order nodes $1, 2, \ldots, n$

$$d(i, j, k) = \text{shortest ij path using intermediate nodes numbered } k \text{ or less}.$$  

$$d(i, j, k) = \min \left\{ d(i, j, k-1), d(i, k, k-1) + d(k, j, k-1) \right\}$$

Table is $n \times n \times n$

Time $= O(n^3) = O(|V|^3)$. 
longest increasing subsequence

A_1, A_2, \ldots, A_n

S(j) = longest inc. subse. ending with A_j.

S(j) = \max \left\{ 1 + \max_{i < j} S(i) \right\}

\quad \text{subject to } A_i < A_j

Return \max_j S(j).

To find sequence also store prev(j).

III to shortest paths.
**Edit distance**

\[ \begin{align*}
\text{RAIN} & \quad \text{RAIN} \\
- \text{WIND} & \quad \text{WIND} \\
\end{align*} \]

\[ D(i, j) = \text{Edit distance} \left(a_1 \ldots a_i, b_1 \ldots b_j\right) \]

Possible alignments:

\[
\begin{array}{cccc}
\text{a}_i & \text{a}_i \\
\text{b}_j & \text{b}_j \\
\end{array}
\]

\[ \text{cost} = \begin{cases} 
1 & \text{if } a_i \neq b_j, \\
0 & \text{if } a_i = b_j.
\end{cases} \]

\[ D(i, j) = \min \left\{ 1 + D(i, j-1), 1 + D(i-1, j), 1 + \left( a_i \neq b_j \right) + D(i-1, j-1) \right\} \]
Matrix chain

\[ A_1 \times A_2 \cdots \times A_n \]

\[ m_1 \times m_2 \quad m_2 \times m_3 \quad m_3 \times m_4 \]

Can do

\[ m_1 m_2 m_3 + m_1 m_3 m_4 = m_1 m_3 (m_2 + m_4) \]

or

\[ m_2 m_3 m_4 + m_1 m_2 m_4 = m_2 m_4 (m_1 + m_3) \]

\[ 2 \times 6 \quad 6 \times 10 \quad 10 \times 6 \quad 20 \cdot 12 \]

\[ 1 \times 2 \quad 2 \times 1 \quad 1 \times 3 \quad 3 \times 1 \quad 1 \times 3 \]

What is the best sequence?

\[ C(i, j) = \text{best sequence of multi. for } A_i \cdot A_{i+1} \cdots A_j \]

\[ C(i, j) = \min_{i \leq k \leq j} \left\{ C(i, k) + C(k, j) + m_i m_k m_j \right\} \]
Max Independent in a tree.

\[ \text{root } v. \]

Subtree at node \( u \), \( T(u) \)

if \( u \notin I \) then children of \( u \) \( \notin I \).

Let \( I(u) = \max \) wid set \( w \) in \( T(u) \)

\[ I(u) = \max \left\{ \delta_I + \sum_{v \text{ : grandchild of } u} I(v), \sum_{u \text{ : children of } u} I(u) \right\} \]