Dynamic Programming

Reduce the problem size by a small fraction typically, there will be several subproblems

Combine the results

operation: \( \min \) \( \max \)

Shortest paths in a DAG

\[
\begin{align*}
    d(B) &= d(N) + w(A, B) \\
    d(N) &= ?
\end{align*}
\]

What is the shortest path to B from a source in the DAG

\[
d(B) = \min \left\{ d(x) + w(x, B) \right\}
\]
\[ d(B) = \min \left\{ \begin{array}{l} d(x) + \omega(x, B) \\ \text{consider all incoming edges (x, B)} \end{array} \right\} \]

\[ d(x) = \min \left\{ \begin{array}{l} d(s) + \omega(s, x), d(c) + \omega(c, x) \end{array} \right\} \]

\[ d(v) = \min \left\{ \begin{array}{l} d(x) + \omega(x, v) \end{array} \text{all incoming edges into } v \right\} \]

\[ \text{prev}(v) = \arg\min_{(x, v) \in E} d(x) \]

\[ \text{base case: } d(x) = 0 \text{ if } x \text{ is a source vertex in the DAG} \]

1) recursive formula is defined "backwards" in terms of smaller problem

2) computation proceeds in the "forward" direction from smaller to larger subproblems

\[ O(|V| + |E|) \]

All Pairs Shortest Paths \(|V| \text{ vertices} \)
All Pairs Shortest Paths

**APSP**

| \(|V|\) vertices
| --- |
| find \(d(x, y) \neq x, y \in V\)

- \(\Omega(|V|^2)\) lower bound on the cost

- Given some graph
- allow negative weights
- negative cycles are not allowed
- sum of the weights on two edges in the cycle is \(< 0\)

```
O \(-5\)
```

```
O \(-10\)
```

```
O \(-b\)
```

Bellman-Ford

- \(|V| - 1\) steps
- start at some \(d(s, s) = 0, d(s, x) = \infty\)
- update all distances considering each edge in each step

- \(O(|V| \cdot |E|)\)

**Alg 1 APSP:** run Bellman-Ford \(|V|\) times

- with each vertex as the starting vertex
\[ |V| \times O(|V| \cdot |E|) \]
\[ O(|V|^2 \cdot |E|) \rightarrow \text{Sparse: } |E| = O(|V|) \]
\[ O(|V| \cdot |E|) \rightarrow O(|V|^3) \]
\[ \text{dense: } |E| = O(|V|^2) \]
\[ O(|V|^4) \] X

\[ \text{Alg 2: PESP (Dynamic Programming)} \]

\[ \ell_{ij}^{(m)} = \text{shortest path from } i \text{ to } j \text{ that uses at most } m \text{ edges} \]

\[ M = n \]

\[ \leq m \text{ edges} \]
Algo 3: APsp Dynamic Programming (Attempt 2)
\[ l_{ij}^{(m)} = \min_{1 \leq k \leq |V|} \left\{ l_{ik}^{(m-1)} + l_{kj}^{(m-1)} \right\} \]

**Base case:**
\[ l_{ij}^{(1)} = \begin{cases} 
\omega(ijj) & (i,j) \in E \\
\infty & \text{otherwise}
\end{cases} \]

\[
\text{while } m < |V| - 1 \\
\text{initialize } l_{ij}^{(m)} = l_{ij}^{(1)} + \omega(ijj)
\]

\[
\text{for } i = 1, \ldots, |V| \\
\text{for } j = 1, \ldots, |V| \\
\text{let } \ell^{(2m)} = \min_{1 \leq k \leq |V|} \left\{ l_{ik}^{(m-1)} + l_{kj}^{(m-1)} \right\}
\]

\[
m = 2 \cdot m
\]

\[
\Omega(\log \frac{1}{\epsilon})
\]

\[
\Omega(|V|)
\]

\[
\Omega(|V|^2)
\]
\[
\begin{array}{c}
\text{Floyd-Warshall method} \\
O(1V^3) \text{ time also}
\end{array}
\]

Attempt 3 for DP

1) Prev approach, \( Q^{(m)}_{ij} \)

Restrict the set of vertices that can be on the path

Set \( \phi \) allowed intermediate vertices

\[ \phi : \{ i \} \rightarrow \{ j \} \]

\( i \rightarrow \phi \rightarrow j \)

\( \text{short edge} \)

Go directly from \( i \) to \( j \) using \( \phi \) set

\[ \{ 1 \} : \{ i \} \rightarrow \{ 1 \} \rightarrow \{ j \} \]
\[ d(i, j, k) = \min \{ d(i, j, k-1), d(i, k, k-1) + d(k, j, k-1) \} \]

**Pseudo code**

Initilize

\[ d(i, j, 0) = \begin{cases} \infty & \text{if } (i,j) \notin E \\ w(i,j) & \text{otherwise} \end{cases} \]

\[ \text{for } l = 1 \text{ to } n \]

\[ d(i, j, l) = \min \{ d(i, j, l-1), d(i, k, l-1) + d(k, j, l-1) \} \]
\( O(n^3) \) 
\[
\begin{aligned}
&\text{for } k = 1, \ldots, |V| \\
&\text{for } i = 1, \ldots, |V| \\
&\text{for } j = 1, \ldots, |V| \\
\end{aligned}
\]
\( d(i,j,k) = \min \left\{ d(i,j,k-1), \right. \\
\left. d(i,k, k-1) + d(k,j,k-1) \right\} \)

Floyd-Warshall: \( O(n^3) \) time.

\[ \text{Even II} \rightarrow \text{DFS} \]

Chapter 3: pre-visit \rightarrow cycle, topo-sorv

\( \text{SCC} \)

for any directed graph

1) SCC's
2) DAG over the SCC
Chapter 4: BFS

Negative - weights

Negative cycle ×

Chapter 5:

Spanning trees

Cut property

Greedy strategy

Prim's (PQ)

Kruskal's → Union-Find

Huffman Codes: PQ

Set cover:

Greedy strategy

Choose the lowest degree
Choose the lowest degree among the unCONNECTED vertices.

PQ

DP: shortest path

\[ \rightarrow APSP \]