Breadth first search

\[ G = (V, E) \]

\[ w(x,y) = \text{weight on the edge (x,y)} \]

"length" = weight = cost of the entire path

Task: finding shortest paths

\[ \rightarrow \text{Dijkstra's method (weighted, +ve)} \]

\[ \rightarrow \text{BFS (o/1 e/300)} \]

\[ G, \text{source x, find shortest path to all other vertices} \]

BFS:
Dijkstra's Algorithm

Replace # of hops with weighted hops.

Always pick the current smallest cost vertex and expand it.

Total cost: \(O((|V| + |E|) \log |V|)\)

**PQ:** Array
- delete min: \(O(|V|)\)
- insert/decrease: \(O(1)\)

Dijkstra's: \(O(|V|)\) delete mins

\[O(|V|^2) + O(|V| + |E|) = O(|V|^2)\]
Dijkstra's:
- $O(|V|)$ delete mins
- $O(|V| + |E|)$ decrease keys

Binary PQ

$|V| \log |V| + |E| \log |V| \quad \text{vs} \quad O(|V|^2)$

Good for sparse graphs

$|E| = O(|V|)$

$\approx O(|V| \log |V|)$

dense: $|E| = O(|V|^2)$

$\frac{|V|^2 \log |V|}{|E|} \quad \text{vs} \quad O(|V|^2)$

Array PQ

$|V| \log |V| + |E| \log |V| \quad \text{vs} \quad O(|V|^2)$

Away is good!

Negative weights

Counter-example to Dijkstra's
Bellman–Ford method

Update the weights of all vertices in each iteration

∀ vertex \( u \in V \)

for all neighbors \( y \) of \( u \)

update weight:

\[
Cost(u) + w(u, y) \leq Cost(y)
\]

\[
Cost(y) = Cost(u) + w(u, y)
\]

\[
\text{max \# of edges in any path from } x \leq |V| - 1
\]

exact \( |V| - 1 \) iterations
Exactly \(|V| - 1\) iterations

In each iteration we update all cells

\[
\text{total cost: } (|V| - 1) \times \left( |V| \cdot |E| \right)
\]

\[
= \frac{|V|^2 + |V| \cdot |E|}{2}
\]

\[
= \mathcal{O}(|V| \cdot |E|)
\]

sparse: \(\mathcal{O}(|V|)
\)

dense: \(\mathcal{O}(|V|^3)
\)
If we restrict to "paths"
Bellman Ford will not work

1) Find all possible paths between \( \bigtimes \) any pair \( \bigcirc \)
   - Choose the min cost path

\[
O(V \cdot 15)
\]

Exponential time method

Greedy Algorithms

- "Local" information
- \( \Rightarrow \) pick the best available option
**Minimum Spanning Tree**

**Output:** \( G = (V, E) \), with weights

**Output:**

1) **Tree**

2) Span all the vertices

\[ T = (V', E') \]

**Spanning tree:** set of vertices in \( T \) includes all vertices in \( G \)

\[ E' \subseteq E \]

3) \( \text{Cost} \; \& \; T = (V, E') \)

\[ \text{Cost}(T) = \sum_{e \in E'} w(e) \]

Find a tree \( T \) with smallest possible \( \text{Cost}(T) \)

\[ T \text{ is a spanning tree} \]

\[ \text{Cost}(T) = 23 \]
Prim's Algorithm (similar to Dijkstra's)

1. Sort the edges in increasing order

2. $T = \{ \text{edge in sorted order} \}$

For each edge in sorted order $\leq E$

Add if and only if one end point is in $T$ and we do not create a cycle

$O(\log E)$

$G$ is undirected

1. $ac, cd, ab, bc, bd, ef, ef, df$

2. $\{a, c\}$

$Cov(T) = 16$
1) sort the edges
2) $T = \{ \text{smallest edge} \}$

Iteratively add the least weight edge with one end point not in $T$

Greedy: add the least cost edge