Shortest Paths

The problem:

- Let $G = (V, E)$ be a directed graph
- Edge weights $\text{wt} : E \to \mathbb{R}$
- Convention: $\text{wt}(u, v) := \infty$ if $(u, v) \notin E$
- The weight of a path $p = \langle v_0, v_1, \ldots, v_k \rangle$:
  \[ \text{wt}(p) := \sum_{i=1}^{k} \text{wt}(v_{i-1}, v_i) \]
- The shortest path weight from $u$ to $v$:
  \[ \delta(u, v) := \min\{\text{wt}(p) : p \text{ is a path from } u \text{ to } v\} \]
Some extremes:

• If there is no path from \( u \) to \( v \), then \( \delta(u, v) := \infty \)

• If there is a path from \( u \) to \( v \) that contains a negative weight cycle, then \( \delta(u, v) := -\infty \)

Cycles:

• A shortest path cannot contain either:
  ○ a negative weight cycle, or
  ○ a positive weight cycle

  but may contain a zero-weight cycle

• If there is a shortest path:
  ○ there is always a shortest path with no cycles
  ○ there is always a shortest path with \( \leq |V| - 1 \) edges
Currency conversion: an application with negative edge weights

We have $n$ currencies (or other financial instruments), which are represented as vertices in a graph.

Suppose we can convert 1 unit of currency $u$ to $x$ units of currency $v$

we make an edge $u \rightarrow v$ with weight $-\log(x)$

Consider a path of currencies $v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k$ with edge weights $-\log(x_1), \ldots, -\log(x_k)$

The path weight is $-\sum_i \log(x_i) = -\log(x_1 \cdots x_k)$ which is minimized when the exchange rate $x_1 \cdots x_k$ is maximized.

Negative weight cycles can be very interesting!
An arbitrage opportunity

\[0.741 \times 1.366 \times 0.995 = 1.00714497\]

A negative cycle that represents an arbitrage opportunity

\[-ln(0.741) - ln(1.366) - ln(0.995)\]

\[0.2998 - 0.3119 + 0.0050 = -0.0071\]

replace each weight \(w\) with \(-ln(w)\)
Shortest Path Variations:

- Single source
- Single destination
- Single pair
- All pairs
**Single source shortest paths**

**Goal:** compute shortest paths from a given node \( s \) to all other nodes

We will calculate \( d[v] = \delta(s, v) \) for all \( v \in V \)

We will also calculate an implicit “shortest path tree”:

- \( \pi[v] = \text{predecessor of } v \text{ on the tree path from } s \text{ to } v \)

Code to print a shortest path to \( v \), in reverse:

```plaintext
while \( v \neq s \) do: print \( v \), \( v \leftarrow \pi[v] \)
```
A shortest path tree:
Dijkstra’s Algorithm

**Assumption:** No negative edge weights

**Idea:**

Beginning at $s$, we grow a shortest path tree, edge by edge

We will use a “greedy” strategy, choosing the edge that yields a new path of least weight
To see why this strategy works, we need a definition:

**Definition (Cluster)**

A subset \( C \subseteq V \) is called a cluster about \( s \) if

- \( s \in C \), and
- \( \delta(s, u) \leq \delta(s, v) \) for all \( u \in C \) and \( v \in V \setminus C \)

**Intuition:** nothing outside the cluster is closer to \( s \) than anything inside the cluster.
**Definition (Cluster Path / Cluster Distance)**

For cluster $C$ about $s$ and $v \in V \setminus C$, define a $C$-path to $v$ as a path that

- starts at $s$,
- ends at $v$,
- and (except for $v$) contains only nodes in $C$.

Define $D_C(v) := \min \left( \{ \text{wt}(p) : p \text{ is a } C\text{-path to } v \} \cup \{\infty\} \right)$
Lemma (Cluster Properties)

1) \( \{s\} \) is a cluster

2) If \( C \) is a cluster, and \( v^* \in V \setminus C \) with minimal \( D_C(v^*) \), then \( \delta(s, v^*) = D_C(v^*) \) and \( C \cup \{v^*\} \) is also a cluster

(1) is clear

For (2), we will prove (below)

**Claim:** \( \delta(s, v) \geq D_C(v^*) \) for all \( v \in V \setminus C \)

First, apply claim with \( v := v^* \):

\[
\delta(s, v^*) \geq D_C(v^*)
\]

- \( \delta(s, v^*) \leq D_C(v^*) \) \([D_C(v^*) \text{ is the weight of some path}]\)
- \( \therefore \delta(s, v^*) = D_C(v^*) \)

Second, apply claim with arbitrary \( v \in V \setminus C \):

\[
\delta(s, v) \geq D_C(v^*) = \delta(s, v^*)
\]

which implies that \( C \cup \{v^*\} \) is a cluster
**Proof of Claim:** $\delta(s, v) \geq D_C(v^*)$ for all $v \in V \setminus C$

We may assume $\delta(s, v) \neq \infty$

Let $s = v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_\ell = v$ be a shortest path from $s$ to $v$.

Let $k$ be the *smallest* index such that $v_k \notin C$

We have:

$$
\delta(s, v) = \text{wt}((v_0, \ldots, v_\ell)) \quad \text{[it's a shortest path]}
\geq \text{wt}((v_0, \ldots, v_k)) \quad \text{[wt $\geq 0$]}
\geq D_C(v_k) \quad \text{[def'n of $D_C(v_k)$]}
\geq D_C(v^*) \quad \text{[def'n of $v^*$]}
$$
Algorithm:

// Initialization
for each \( v \in V \):
\[
d[v] \leftarrow \infty
\]
\[
\pi[v] \leftarrow Nil
\]
\[
d[s] \leftarrow 0
\]
\[
Q \leftarrow \{s\}  \quad // \quad C \leftarrow \emptyset
\]

while \( Q \) not empty
\[
\text{remove } v^* \in Q \text{ with minimal } d[v^*]  \quad // \quad \text{add } v^* \text{ to } C
\]

for each \( w \in \text{Successor}(v^*) \) do
\[
\text{if } d[v^*] + \text{wt}(v^*, w) < d[w] \text{ then}
\]
\[
\text{if } d[w] = \infty \text{ then add } w \text{ to } Q
\]
\[
d[w] \leftarrow d[v^*] + \text{wt}(v^*, w)
\]
\[
\pi[w] \leftarrow v^*
\]
Correctness follows from Cluster Properties:

- $C$ is the set of all nodes that have been removed from $Q$

**Loop invariants:**

- for each $v \in C$: $d[v] = \delta(s, v)$
  - and $\pi$ traces out a shortest path from $s$ to $v$ through nodes in $C$
- for each $v \in V \setminus C$: $d[v] = D_C(v)$

**Special case:** First loop iteration makes the cluster $C = \{s\}$

**The general case:** when we add $v^*$ to $C$, the only nodes whose cluster distances can change are those in $\text{Successor}(v^*)$

- Let $C^* := C \cup \{v^*\}$, and consider a $C^*$-path $p : s \rightsquigarrow v \rightarrow w$
- If $v \neq v^*$, then $v \in C$ and $\text{wt}(p) \geq D_C(w)$
- So the only “interesting” $C^*$-paths are those of the form $s \rightsquigarrow v^* \rightarrow w$
Implementation: priority queue

- $n := |V|$ ExtractMin’s / Insert’s
- $m := |E|$ Decrease’s

Running Time:

- unsorted list: $O(n^2)$
  - $\text{ExtractMin}: O(n)$, $\text{Decrease} / \text{Insert}: O(1)$
- binary heap: $O((n + m) \log n)$
  - all operations: $O(\log n)$
- Fibonacci heap: $O(n \log n + m)$ (an advanced data structure)
A linear time special case:

- assume all edges weights are integers in the range $[0..B]$ for some small bound $B$
- we can implement Dijskstra in time $O(nB + m)$
- so for constant $B$, this is linear time

Some observations:

- nodes are removed from $Q$ (and added to $C$) in order of increasing distance from $s$ — why?
- the maximum distance from $s$ of any node is $\leq (n - 1)B$ — why?
- the $d$-value of any node only decreases over time — why?
- the maximum $d$-value of any node in $Q$ is $\leq nB$ — why?
An implementation (first attempt):

- Use an array $A[0..nB]$
- Entry $A[i]$ is a “bucket” of nodes in $Q$ whose current $d$-value is $i$
- Initialize $next \leftarrow 0$

**ExtractMin:**

- while $A[next]$ empty do: increment $next$
- remove and return any node from bucket $A[next]$
- **Total cost:** $O(nB)$

**Insert / Decrease:**

- add/move node to appropriate bucket
- **Key fact:** the node will never land in a bucket of index smaller than $next$ — why?
- **Total cost:** $O(n + m)$
More observations:

• Let \( \text{min}(Q) := \) the smallest \( d \)-value for any node in \( Q \)

• when a node is added to \( Q \), it’s \( d \)-value is at most \( \text{min}(Q) + B \) — why?

• at any point in time, the \( d \)-value of any node in \( Q \) lies in \([ \text{min}(Q), \text{min}(Q) + B]\) — why?

• In the above implementation, at any point in time, most buckets are empty:
  ◦ only entries \( \text{min}(Q), \ldots, \text{min}(Q) + B \) of \( A \) are non-empty

• A more space efficient representation:
  ◦ use a “circular array” \( A'[0..B] \) to represent the non-empty part of \( A \)
  ◦ \( A[i] \) is stored at \( A'[i \mod (B + 1)] \)
Problem solving by reduction

You are given a directed graph $G = (V, E)$ along with nodes $s, t \in V$

Edges are colored red and green

A path is called admissible if it contains at most 3 red edges

Determine if there is an admissible path from $s$ to $t$, and if so, find one with the minimal number of green edges

Solve this problem by recasting it as a standard shortest path problem: running time $O(|V| + |E|)$. 
Idea:

Make 4 copies of graph: $G^{(0)}, G^{(1)}, G^{(2)}, G^{(3)}$

Green edge $u \rightarrow v$ in $G$ maps to $u^{(i)} \rightarrow v^{(i)}$ with edge weight 1

Red edge $u \rightarrow v$ in $G$ maps to $u^{(i)} \rightarrow v^{(i+1)}$ with edge weight 0

Add 0-weight edges from each $t^{(i)}$ to new node $t'$

Find shortest path from $s^{(0)}$ to $t'$