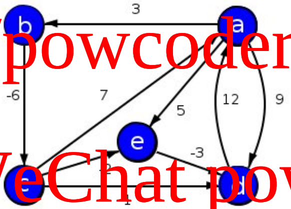
More Terminology

Definition (Directed Graph)

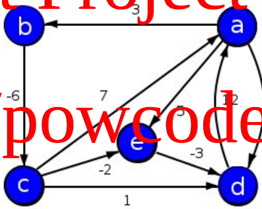
A directed graph is a graph $G = (V, E)$ where V is a set (of objects) and E is a set of ordered pairs of elements of V.



- In a directed graph each edge (u, v) has a direction
- Edges (u, v) and (v, u) can both exist, and have different weights
- An undirected graph can be seen as a special type of directed graph

Shortest Paths

With weighted edges a simple breadth-first search will not find the shortest paths



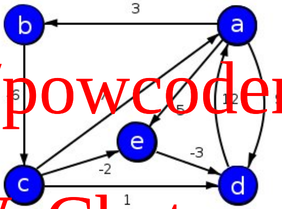
- The "shortest" path from a to e is $\langle a, b, c, e \rangle$

Questions

- What might a "brute force" algorithm do?
- How long would it take?

Bellman-Ford

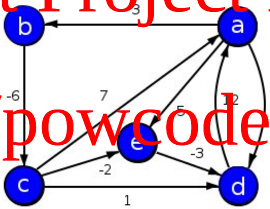
The Bellman-Ford algorithm solves the general problem where edges may have negative weights



- A `distance` array is used again
- `distance[v]` is the current estimate of the shortest path to v
- The algorithm proceeds by gradually reducing these estimates

Bellman-Ford

Relaxing edge (u, v) checks if $s \rightsquigarrow u \rightarrow v$ reduces `distance[v]`



Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then:
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

Bellman-Ford

Bellman-Ford (Input: weighted graph $G = (V, E)$ and vertex s)

- Set *distance*$[v] = \infty$ for all vertices
- Set *distance*$[s] = 0$
- Repeat $|V| - 1$ times:
  - For each edge $e \in E$
    - Relax e
- For each edge $(u, v) \in E$
  - If *distance*$[v]$ is greater than *distance*$[u] + w(u, v)$
    - Return *FALSE*
- Return *TRUE*

Question

Why does the loop run $|V| - 1$ times?

Bellman-Ford

Bellman-Ford (Input: weighted graph G and vertex s)

- Set $distance[v] = \infty$ for all vertices
- Set $distance[s] = 0$
- Repeat $|V| - 1$ times:
  - For each edge $e \in E$
    - Relax e

- For each edge $(u, v) \in E$
  - If $distance[v]$ is greater than $distance[u] + w(u, v)$
    - Return *FALSE*

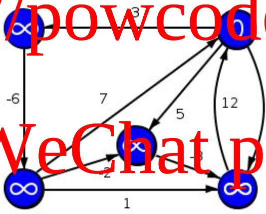
- Return *TRUE*

- All edges are relaxed $|V| - 1$ times so all paths are tried
- The algorithm returns *FALSE* if a negative weight cycle occurs

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

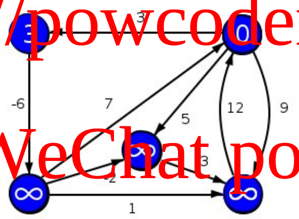


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If *distance*$[v]$ is greater than *distance*$[u] + w(u, v)$ then
  - *distance*$[v]$ is *distance*$[u] + w(u, v)$
  - Parent of v is u

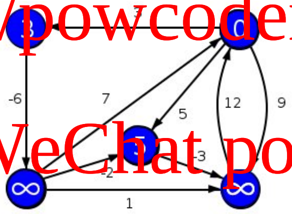


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

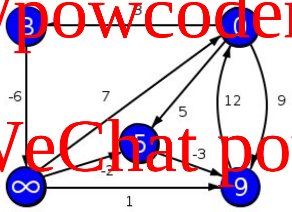


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then:
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

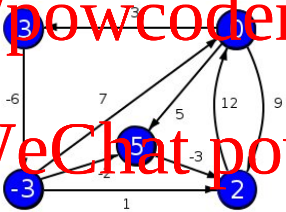


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

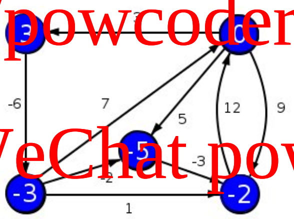


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

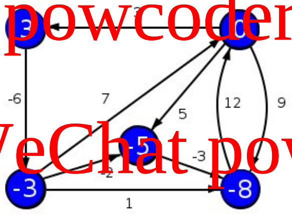


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Relax (Input: weighted edge (u, v))

- If $distance[v]$ is greater than $distance[u] + w(u, v)$ then
  - $distance[v]$ is $distance[u] + w(u, v)$
  - Parent of v is u

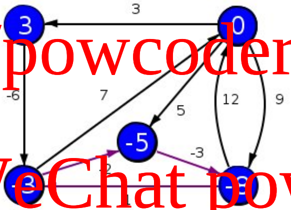


- In iteration i all edges in paths containing i edges have been relaxed
- The most edges in any (simple) path is $|V| - 1$

Bellman-Ford

Definition (Negative Weight Cycle)

A path $C = \langle v_1, v_2, \ldots, v_i \rangle$ in a directed graph is a negative weight cycle iff C is a cycle and $\sum_{i=1}^{n-1} w(v_i, v_{i+1}) < 0$.



If a directed graph G contains a negative weight cycle $\langle v_1, v_2, \ldots, v_n \rangle$ then:

- The shortest paths to all vertices reachable from $v_1, \ldots, v_n$ are undefined
- In this case Bellman-Ford will return *FALSE*

Time

Question

What is the time complexity of Bellman-Ford?

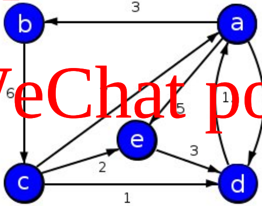
Bellman-Ford (Input: weighted graph G and vertex s)

- Set $distance[v] = \infty$ for all vertices
- Set $distance[s] = 0$
- Repeat $|V| - 1$ times:
  - For each edge $e \in E$
    - Relax e
- For each edge $(u, v) \in E$
  - If $distance[v]$ is greater than $distance[u] + w(u, v)$
    - Return *FALSE*
- Return *TRUE*

Dijkstra's Algorithm

If G has non-negative edges only then we can use Dijkstra's Algorithm

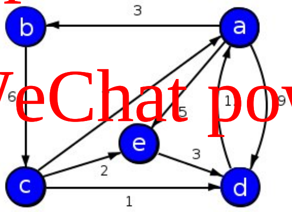
- Bellman-Ford relaxes every edge of every path
- The running time of Bellman-Ford is $O(VE)$
- Dijkstra's algorithm instead uses a greedy strategy



Dijkstra's Algorithm

Basic idea:

- Relax edges from one vertex
- Will have then found shortest path to at least one other vertex
- Repeat



Dijkstra's Algorithm

Dijkstra's algorithm maintains a set of vertices whose *distance*[*v*] is correct

Dijkstra (input: weighted graph $G = (V, E)$, vertex s)

```
distance[v] = infinity for all vertices
distance[s] = 0
S = {}
while V - S != {}
  u is vertex in V - S with least distance[u]
  for v in G.adj[u]
    relax (u,v)
  S = S + {u}
```


- The next vertex added to S is the one with the least *distance*[*u*]
- This value is now assumed to be minimal. Is this correct?

Correctness

In the following, the function p represents the (actual) length of the shortest path from the source to a given vertex

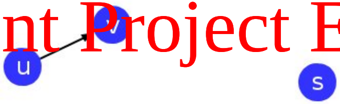
- If there is no path $s \rightsquigarrow v$, then $p(v) = \infty$
- $\infty + x = \infty$, for all $x \in \mathbb{R}$

Theorem (Correctness of Dijkstra)

At the start of the while loop of Dijkstra's algorithm, run on weighted, directed graph $G = (V, E)$ with non-negative weight function w, and vertex $s \in V$: if $distance[v] = p(v)$ for all vertices $v \in S$, then $distance[u] = p(u)$ for u, the next vertex added to S.

Proof

First we prove two useful properties



Lemma (Triangle Lemma)

Let $G = (V, E)$ be a weighted, directed graph with weight function w, and source vertex s. If (u, v) is an edge in E, then $p(v) \leq p(u) + w(u, v)$.

Proof.

If there is no path $s \rightsquigarrow u$, then $p(u) = \infty$, so $p(v) \leq p(u)$ and the lemma holds. If there is a path $s \rightsquigarrow u$, then $s \rightsquigarrow u \rightarrow v$ is a path to v. The length of one such path to v is $p(u) + w(u, v)$. The *shortest* path to v cannot be *longer* than this, so the lemma also holds in this case. □

Proof

First we prove two useful properties

Lemma (Triangle Lemma)

Let $G = (V, E)$ be a weighted, directed graph with weight function w, and source vertex s. If (u, v) is an edge in E, then $p(v) \leq p(u) + w(u, v)$.

Proof.

If there is no path $s \rightsquigarrow u$, then $p(u) = \infty$, so $p(v) \leq p(u)$ and the lemma holds. If there is a path $s \rightsquigarrow u$, then $s \rightsquigarrow u \rightarrow v$ is a path to v. The length of one such path to v is $p(u) + w(u, v)$. The *shortest* path to v cannot be *longer* than this, so the lemma also holds in this case. $\square$

Proof

First we prove two useful properties

Lemma (Triangle Lemma)

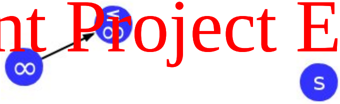
Let $G = (V, E)$ be a weighted, directed graph with weight function w, and source vertex s. If (u, v) is an edge in E, then $p(v) \leq p(u) + w(u, v)$.

Proof.

If there is no path $s \rightsquigarrow u$, then $p(u) = \infty$, so $p(v) \leq p(u)$ and the lemma holds. If there is a path $s \rightsquigarrow u$, then $s \rightsquigarrow u \rightarrow v$ is a path to v. The length of one such path to v is $p(u) + w(u, v)$. The *shortest* path to v cannot be *longer* than this, so the lemma also holds in this case. □

Proof

First we prove two useful properties



Lemma (Triangle Lemma)

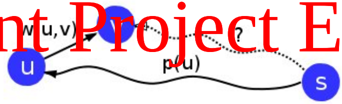
Let $G = (V, E)$ be a weighted, directed graph with weight function w, and source vertex s. If (u, v) is an edge in E, then $p(v) \leq p(u) + w(u, v)$.

Proof.

If there is no path $s \rightsquigarrow u$, then $p(u) = \infty$, so $p(v) \leq p(u)$ and the lemma holds. If there is a path $s \rightsquigarrow u$, then $s \rightsquigarrow u \rightarrow v$ is a path to v. The length of one such path to v is $p(u) + w(u, v)$. The *shortest* path to v cannot be *longer* than this, so the lemma also holds in this case. □

Proof

First we prove two useful properties



Lemma (Triangle Lemma)

Let $G = (V, E)$ be a weighted, directed graph with weight function w, and source vertex s. If (u, v) is an edge in E, then $p(v) \leq p(u) + w(u, v)$.

Proof.

If there is no path $s \rightsquigarrow u$, then $p(u) = \infty$, so $p(v) \leq p(u)$ and the lemma holds. If there is a path $s \rightsquigarrow u$, then $s \rightsquigarrow u \rightarrow v$ is a path to v. The length of one such path to v is $p(u) + w(u, v)$. The *shortest* path to v cannot be *longer* than this, so the lemma also holds in this case. □

Proof

This lemma shows that *distance*$[u]$ is always an upper bound for $p(u)$

Lemma (Upper Bound Lemma)

Let $G = (V, E)$ be a weighted, directed graph with weight function w, and source vertex s. If distance$[s]$ is initialised to 0 and distance$[v]$, for all $v \in V$ where $v \neq s$, is initialised to ∞, then distance$[u] \geq p(u)$, for all $u \in V$, after relaxing any sequence of edges in G.

Proof.

Firstly, consider a sequence of 0 relaxed edges

- *distance*$[u] = \infty$, for $u \neq s$
- *distance*$[s] = 0$

If s is part of a negative weight cycle, then $p(s) = -\infty$, otherwise $p(s) = 0$. So, *distance*$[u] \geq p(u)$ for all $u \in V$ in this case. $\square$

Proof

Proof (continued).

Now consider the relaxation of edge (x, y) within some sequence of relaxations.

- Assume $distance[u] \geq p(u)$ for all $u \in V$, prior to relaxing (x, y)

When (x, y) is relaxed either all $distance[u]$ are unchanged, or $distance[y] = distance[x] + w(x, y)$. In the latter case:

- $distance[y] = distance[x] + w(x, y)$, so
- $distance[y] \geq p(x) + w(x, y)$, by the assumption, and
- $distance[y] \geq p(y)$, by the Triangle Lemma

So after relaxing (x, y), $distance[u] \geq p(u)$ still holds for all vertices in G, and by the principle of induction $distance[u] \geq p(u)$ is always true for any sequence of edge relaxations. □

Proof

Theorem (Correctness of Dijkstra)

At the start of the while loop of Dijkstra's algorithm, run on weighted, directed graph $G = (V, E)$ with non-negative weight function w, and vertex $s \in V$: if $distance[v] = p(v)$ for all vertices $v \in S$, then $distance[u] = p(u)$ for u, the next vertex added to S.

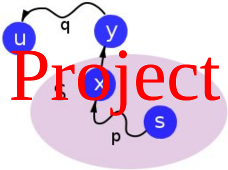
Proof.

If there is no path $s \rightsquigarrow u$ then $p(u) = \infty$. Since:

- $distance[u] \geq p(u)$, by the Upper Bound Lemma, then
- $distance[u] = \infty$, so
- $distance[u] = p(u)$.

and the theorem is true. □

Proof



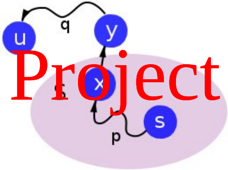
Proof (continued).

If there is a path $s \rightsquigarrow u$, then consider the shortest such path. Let this path be $s \rightsquigarrow^p x \rightarrow y \rightsquigarrow^q u$, where y is the first vertex on the path not in S. First, it is shown that $distance[y] = p(y)$, as follows. $s \rightsquigarrow^p x \rightarrow y$ must be a shortest path from s to y. (Or there would be a shorter path to u.) Then,

- $distance[x] = p(x)$
- $distance[y] = distance[x] + w(x, y) = p(x) + w(x, y)$

since x is in S and (x, y) was relaxed when x was added to S. □

Proof



Proof (continued).

And, since $s \rightsquigarrow^{p} x \rightarrow y$ is a shortest path from s to y, then:

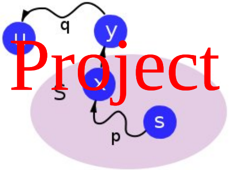
- $p(y) = p(x) + w(x, y) = distance[y]$

Next we show that $distance[u] = distance[y] = p(y) = p(u)$ using the observations that

- (1) $distance[u] \leq distance[y]$, since u is added next to S
- (2) $p(y) \leq p(u)$, since all edges are non-negative.

□

Proof



Proof (continued).

So, beginning with Observation (1):

- $distance[u] \leq distance[y]$, and therefore
- $distance[u] \leq p(y)$, and
- $distance[u] \leq p(u)$, by Observation (2).

But $distance[u] \geq p(u)$ by the Upper Bound Lemma, so $distance[u] = p(u)$ and the theorem is true. □

Dijkstra's Algorithm

Discussion

What is the time complexity of Dijkstra's algorithm?

Dijkstra (Input: weighted graph $G = (V, E)$, vertex s)

```
distance[v] = infinity for all vertices
distance[s] = 0
S = {}
while V - S != {}
  u is vertex in V - S with least distance[u]
  for v in G.adj[u]
    relax (u,v)
  S = S + {u}
```

Dijkstra's Algorithm

Discussion

What is the time complexity of Dijkstra's algorithm?

Dijkstra (Input: weighted graph $G = (V, E)$, vertex s)

```
distance[v] = infinity for all vertices
distance[s] = 0
S = {}
while V - S != {}
  u is vertex in V - S with least distance[u]
  for v in G.adj[u]
    relax (u,v)     affects ordering of vertices
  S = S + {u}
```

Performance

The running time of Dijkstra's algorithm depends on the way in which the ordering of the vertices is managed

- Implement $V - S$ as a priority queue
- There is one iteration through the graph vertices
- Each edge is relaxed once, giving an aggregate of $|E|$

With a binary-heap-based priority queue adding, removing and updating (changing key) all run in $O(\log_2 V)$ time.

- Overall running time is then $O(E \log_2 V)$