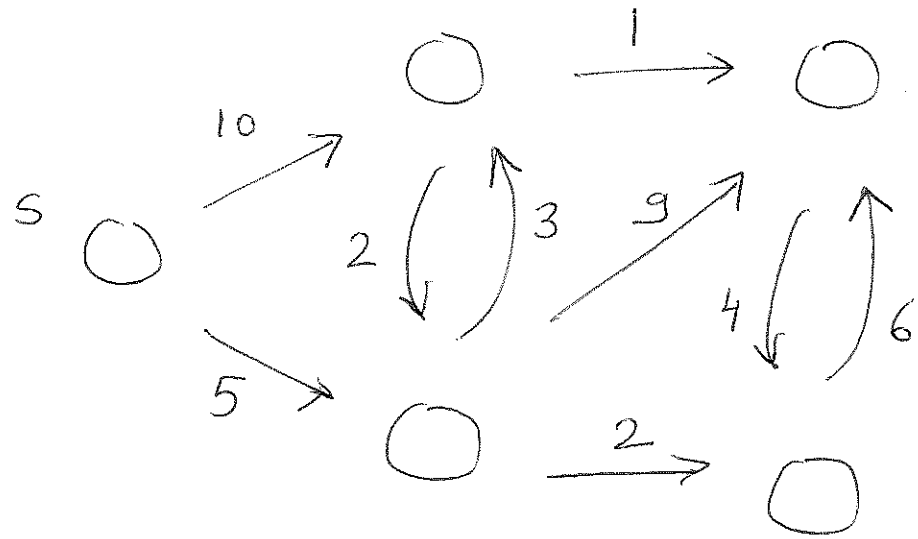
Dijkstra's Shortest Path



Given - directed graph $G(V,E)$

- weight $wt(u,v) \geq 0 \quad \forall (u,v) \in E$

- source $s \in V$.

Goal is to find (length of) shortest path from s to every other vertex.

<u>Def</u> $dist(u,v)$ = length of shortest path from u to v.

<u>Idea</u> - Maintain label $d[v] \quad \forall v \in V$.

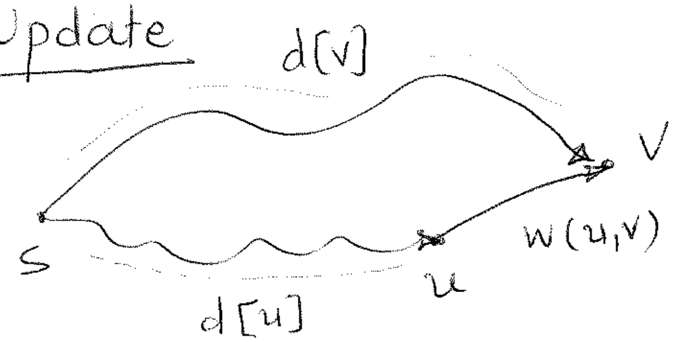
- $d[v]$ = "current estimate" of $dist(s,v)$, i.e. we have already found a $s \leadsto v$ path of length $d[v]$.

- Initially $d[s] = 0$, $d[v] = \infty \ \forall v \neq s$.

Fact: It always holds that

$$dist(s, v) \leq d[v] \quad \forall v \in V.$$

Edge-Update



$$d[v] \xleftarrow{\text{update}} \min\{d[v],\ d[u] + w(u,v)\}.$$

Relax (u)

- $\forall v$ such that $(u,v) \in E$, update
  - $d[v] \leftarrow \min\{d[v],\ d[u] + w(u,v)\}$.
  - If $d[v]$ got set to $d[u] + w(u,v)$ then set $parent(v) = u$.

When algorithm terminates, shortest $s \rightsquigarrow v$ path can be traced by tracing parent pointers backwards from v.

Naive Algorithm

$|V| = n, \quad |E| = m.$

Initialize $d[s] = 0$. $\quad d[v] = \infty \quad \forall v \neq s$.

Repeat n times.

{ For all $u \in V$,

Relax (u).

}. Phase.

<u>Claim</u>. The algorithm, when terminates, gives $d[v] = \text{dist}(s, v) \quad \forall v \in V$.

<u>Proof</u> Fix any $v \in V$. Let

$$s = u_0 \to u_1 \to u_2 \cdots u_{k-1} \to u_k = v.$$

be shortest $s \rightsquigarrow v$ path (hypothetical).

In Phase 1: $s = u_0$ relaxed. $\therefore d[u_1] = \text{dist}(s, u_1)$.

In Phase 2: u_1 // $\therefore d[u_2] = d[u_1] + wt(u_1, u_2)$

$= \text{dist}(s, u_1) + wt(u_1, u_2)$

$= \text{dist}(s, u_2)$.

⋮

Thus in Phase i, $d[u_i]$ gets set to $\text{dist}(s, u_i)$.

Noting that $k \leq n$, we are done. ∎

Dijkstra's Algorithm is clever implementation of the naive idea.

- Sequence of Relax (u) operations, one vertex at a time.
- Always pick vertex u with minimum value of $d[u]$ (among vertices not yet picked).

Algorithm

- $d[s] = 0$. $d[v] = \infty \quad \forall v \neq s$.
- $S = \phi$. (set of vertices relaxed so far.)

While ($V \setminus S \neq \phi$) {

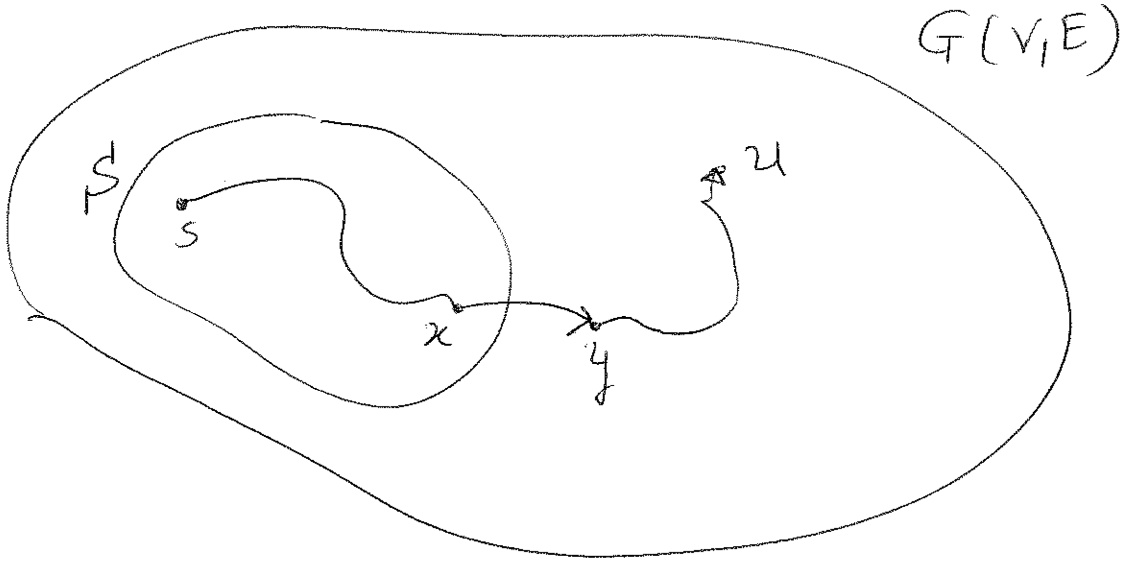
- Pick $u \in V \setminus S$ with minimum (*) value of $d[u]$.
- Relax (u). - Move u to S.

}

Output - $d[v]$ are the distances $dist(s,v)$.
- parent pointers give the shortest paths.

Claim. When a vertex u is picked in $(*)$ to relax, it is already the case that $d[u] = dist(s, u)$.

Proof



Let $s \rightsquigarrow x \to y \rightsquigarrow u$ be (hypothetical) shortest $s \rightsquigarrow u$ path where $x \to y$ is first edge that jumps outside S.

Note: $s \rightsquigarrow x \to y \rightsquigarrow u$ is also shortest path from s to every vertex on that path.

The claim follows as:

$\text{dist}(s,u) \geq \text{dist}(s,y)$ $\because s \rightsquigarrow x \to y$ is shortest path from s to y.

$= \text{dist}(s,x) + wt(x,y)$

$= d[x] + wt(x,y)$ $\because x \in S$, and by inductive hypothesis

$\geq d[y]$ $\because (x,y)$ has been relaxed

$\cancel{\geq \text{dist}(s,y)}$

$\geq d[u].$ $\because u$ had minimum value of $d[u]$ in $V \setminus S$.

Hence $\text{dist}(s,u) = d[u]$.

$\blacksquare$

Running Time of Dijkstra's Algorithm

One needs to maintain set of n numbers $\{d[v] \mid v \in V\}$ and

	# operations
— Find/Delete minimum	n
— Decrease key	m

Using Fibonacci heaps, Find/Delete Min takes $O(\log n)$ amortized time and Decrease key takes $O(1)$ amortized time.

$\therefore$ Overall $O(n \log n + m)$ time.