Dijkstra's Algorithm

UT Austin CS prof 84-99
Turing award in 1972
one of the founding fathers of CS.

Single Source Shortest Path

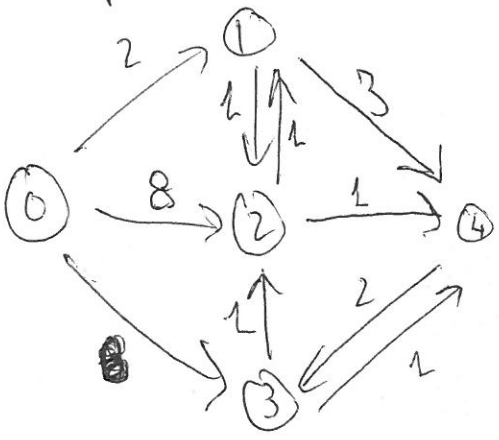
Input: Graph $G=(V,E)$
vertex $s \in V$

Output: For every vertex $v \in V$
shortest path from s to v.

We saw how to solve this problem for unweighted G (using BFS).

Today: Weighted case, where all weights are non-negative.

Example



	length
$0 \to 1$:	2
$0 \to 2$:	3
$0 \to 3$:	6
$0 \to 4$:	4

Algorithm

(1) Maintain $S \subseteq V$ = vertices for which we already found shortest paths.

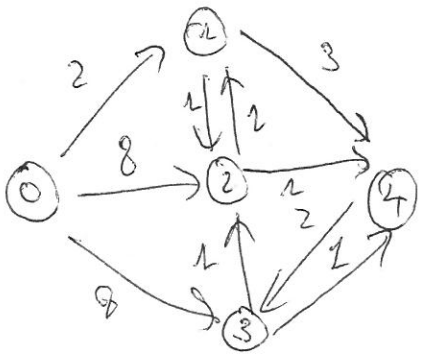
Initially $S = \{s\}$.
$d(s) = 0$.

(2) Repeat: pick $v \in V - S$ that minimizes $\min_{u \in S} d(u) + w(u,v)$

$S \leftarrow S \cup \{v\}$

$d(v) \leftarrow d(u) + w(u,v)$.

Run the algo on previous example: $s = 0$



$S = \{0\}$ $d(0) = 0$

$S = \{0,1\}$ $d(1) = 2$

$S = \{0,1,2\}$ $d(2) = 3$

$S = \{0,1,2,4\}$ $d(4) = 4$

$S = \{0,1,2,4,3\}$ $d(3) = 6$.

Correctness of Algorithm

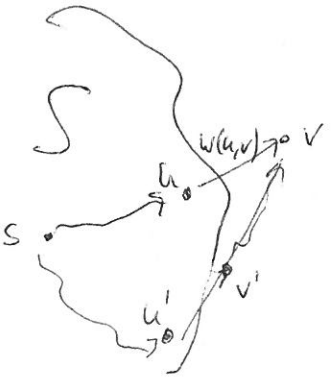
Lemma Every time a vertex is extracted from V and added to S its $d(v)$ is the length of the shortest path from s to v.

Pf By induction.

Base When s is added $d(s)=0$ which is indeed the length.

Hypothesis Suppose this is true for S so far.

Step now we add v to S.



(a) There is a path $S \rightsquigarrow v$ of length $d(u)+w(u,v)$ because there's a path $S \rightsquigarrow u$ of length $d(u)$.

(b) Suppose there's a path $S \rightsquigarrow v$ that is shorter than $d(u)+w(u,v)$. on way of contradiction

$s \in S$ and $v \notin S$ so there must be an edge (u',v') on the path so $u' \in S$, $v' \notin S$.

$$\underbrace{d(u')+w(u',v')}_{\text{part of a path that is shorter than } d(u)+w(u,v).} < d(u)+w(u,v)$$

$\Rightarrow$ Dijkstra would have picked v' & set $d(v') \leq d(u')+w(u',v') < d(u)+w(u,v)$

← Chapter 2.5 in the book

Priority Queue: maintain a set Q of elements where each element has a value "key".

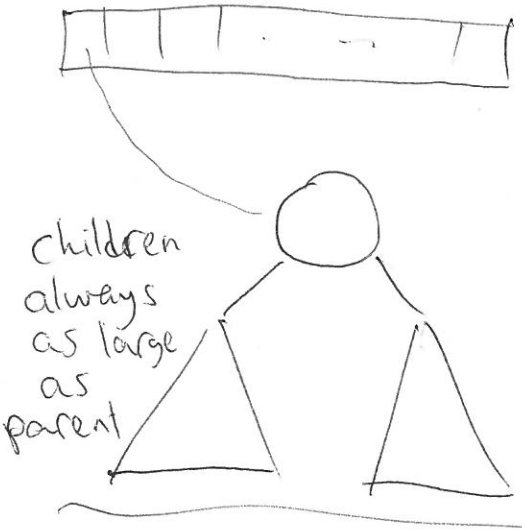
The following operations are supported:

- Insert(Q,x): insert element x with value key(x) to Q.
- Minimum(Q): returns a pointer to element in Q with min key. report if $Q=\emptyset$.
- Extract-Min(Q): returns an element with min key & extracts it from Q (i.e. deletes)
- Decrease-key(Q,x,k): given a pointer to element x in S change key(x) to k if key$(x) > k$.

Dijkstra (G,s)

1 For each $v \in V$ do $d[v] \leftarrow \infty$.
2. $d[s] \leftarrow 0$
3. Q - a binary heap with all $v \in V$, each with key $d[v]$.
4. While $Q \neq \emptyset$ do
5. $\quad u \leftarrow$ Extract-Min(Q)
6. $\quad$ For each neighbor v of u do
7. $\quad\quad$ If $v \in Q$ then Decrease-key$(Q, v, d[u]+w(u,v))$.

Run-time



Binary heap implementation:

- Insert $O(\lg n)$ time.
- Min $O(1)$ time.
- Extract-Min $O(\lg n)$ time.
- Decrease-key $O(\lg n)$ time.

Note: A priority queue allows sorting so n inserts + n extract-min must take $\Omega(n \lg n)$ time

Steps 1-3 $O(n \lg n)$. $n = |V|$.

Step 5 $O(n \lg n)$

Steps 6-7 $O(m \lg n)$

Overall: $O((m+n) \lg n)$.