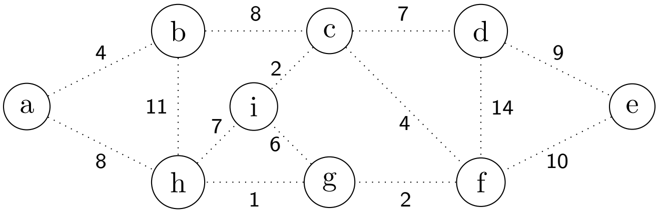
CSCB63 – Design and Analysis of Data Structures

Anya Tafliovich[1]

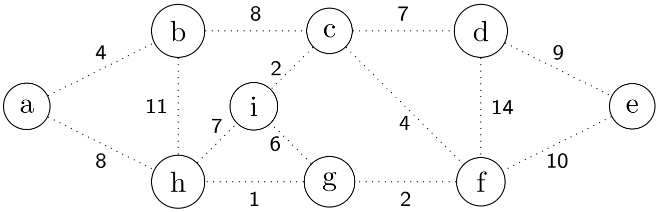
[1] with huge thanks to Anna Bretscher and Albert Lai

finding the shortest paths



- Given an (edge-)weighted graph and two vertices in it,
- find the cheapest (minimum possible weight) path between them, or
- report that one does not exist.

finding the shortest paths



Even better:

- Given an (edge-)weighted graph and a vertex s in it,
- find the cheapest (minimum possible weight) paths from s to all other vertices.

Dijkstra's algorithm: idea

Dijkstra's algorithm finds shortest paths by a BFS with a twist

- the queue is replaced with a minimum priority queue
- with an additional operation `decrease-priority(vertex, new-priority)`

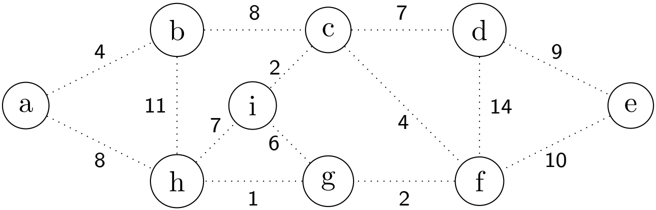
Keep unvisited vertices in the priority queue:

$$priority(v) = distance(start, v) \text{ via finished vertices only}$$
$$priority(v) = \infty \text{ if no such path}$$

The algorithm grows paths by one edge at a time.

Correctness idea: every time we `extract-min`, we get the next vertex closest to `start`.

Dijkstra's algorithm: example



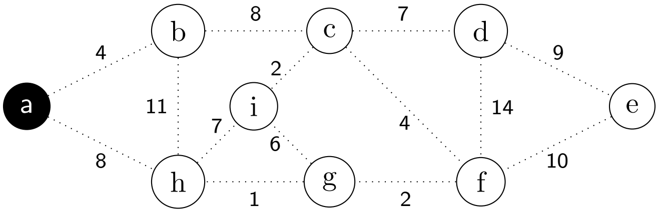
Priority queue contains vertices *not* in tree:

vertex	a	b	c	d	e	f	g	h	i
priority	0	∞	∞	∞	∞	∞	∞	∞	∞
pred									

Distance tree:

$\{ \qquad \}$

Dijkstra's algorithm: example



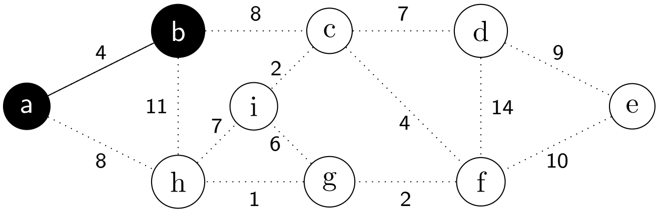
Priority queue contains vertices *not* in tree:

vertex	b	h	c	d	e	f	g	i
priority	4	8	∞	∞	∞	∞	∞	∞
pred	a	a						

Distance tree:

$\{ \qquad \}$

Dijkstra's algorithm: example

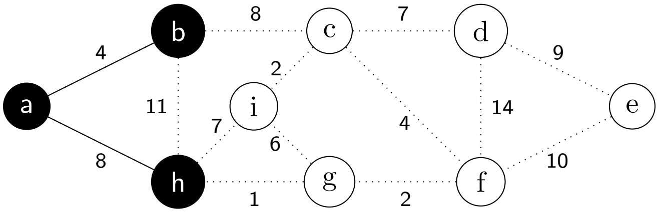


Priority queue contains vertices *not* in tree:

vertex	h	c	d	e	f	g	i
priority	8	12	∞	∞	∞	∞	∞
pred	a	b					

Distance tree:
$\{$ (a,b,4), $\}$

Dijkstra's algorithm: example



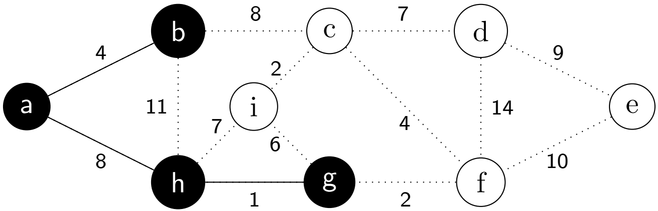
Priority queue contains vertices *not* in tree:

vertex	g	c	i	d	e	f
priority	9	12	15	∞	∞	∞
pred	h	b	h			

Distance tree:

{ (a,b,4), (a,h,8), }

Dijkstra's algorithm: example



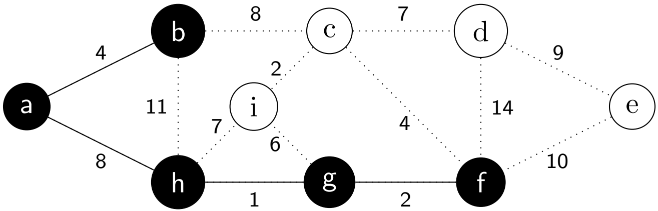
Priority queue contains vertices *not* in tree:

vertex	f	c	i	d	e
priority	11	12	15	∞	∞
pred	g	b	h		

Distance tree:

{ (a,b,4), (a,h,8), (h,g,9), }

Dijkstra's algorithm: example



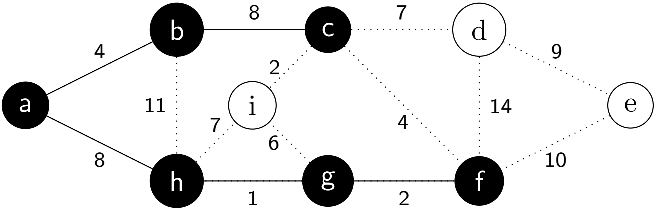
Priority queue contains vertices *not* in tree:

vertex	c	i	e	d
priority	12	15	21	25
pred	b	h	f	f

Distance tree:

{ (a,b,4), (a,h,8), (h,g,9), (g,f,11), }

Dijkstra's algorithm: example

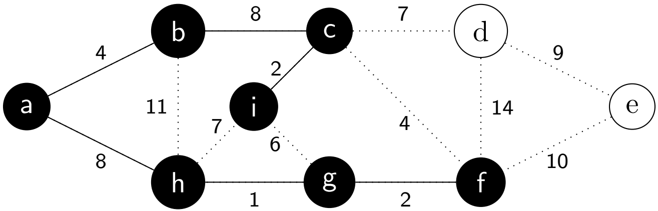


Priority queue contains vertices *not* in tree:

vertex	i	d	e
priority	14	19	21
pred	c	c	f

Distance tree:
{ (a,b,4), (a,h,8), (h,g,9), (g,f,11), (b,c,12), }

Dijkstra's algorithm: example



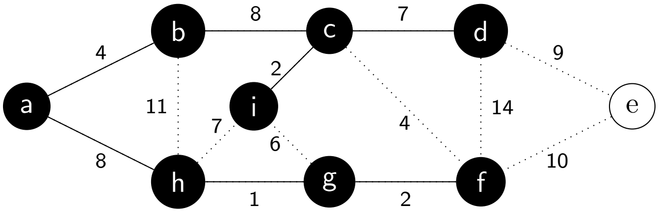
Priority queue contains vertices *not* in tree:

vertex	d	e
priority	19	21
pred	c	f

Distance tree:

{ (a,b,4), (a,h,8), (h,g,9), (g,f,11), (b,c,12), (c,i,14), }

Dijkstra's algorithm: example

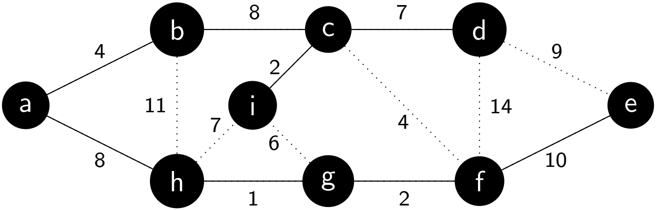


Priority queue contains vertices *not* in tree:

vertex	e
priority	21
pred	f

Distance tree:
{ (a,b,4), (a,h,8), (h,g,9), (g,f,11), (b,c,12), (c,i,14), (c,d,19), }

Dijkstra's algorithm: example



Priority queue contains vertices *not* in tree:

$\boxed{}$

Distance tree:
{ (a,b,4), (a,h,8), (h,g,9), (g,f,11), (b,c,12), (c,i,14), (c,d,19), (f,e,21) }

Dijkstra's algorithm

```
PQ := new min-heap()
PQ.insert(0, start)
start.d := 0
for each vertex v != start:
  PQ.insert(inf, v)
  v.d := inf
while not PQ.is-empty():
  u := PQ.extract-min()
  for each v in u's adjacency list:
    d' := u.d + weight(u, v)
    if d' < v.d:
      PQ.decrease-priority(v, d')
      v.d := d'
      v.pred := u
```

Dijkstra's algorithm: time

Let $n = |V|$ and $m = |E|$. Then:

- every vertex enters and leaves min-heap once
  - enters in the beginning only; continue until heap is empty
  - $\mathcal{O}(\log n)$ each, for a total of $\mathcal{O}(n \log n)$
- with every edge may call `decrease-priority`
  - $\mathcal{O}(\log n)$ each, for a total of $\mathcal{O}(m \log n)$
- the rest can be done in $\Theta(1)$ per vertex or per edge

Total time worst case: $\mathcal{O}((n+m)\log n)$