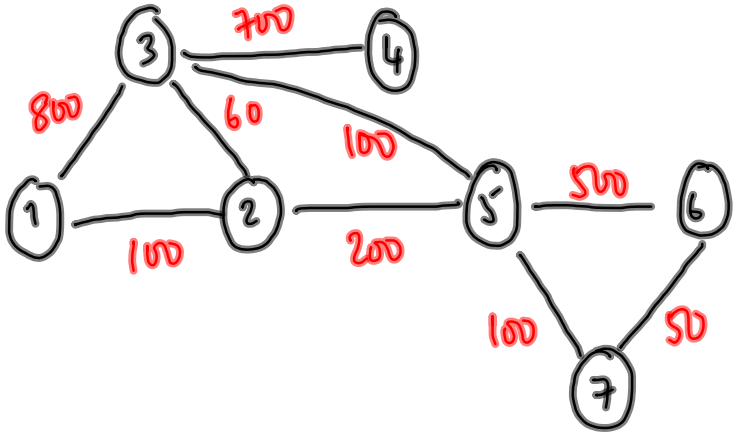
Shortest paths in weighted graphs

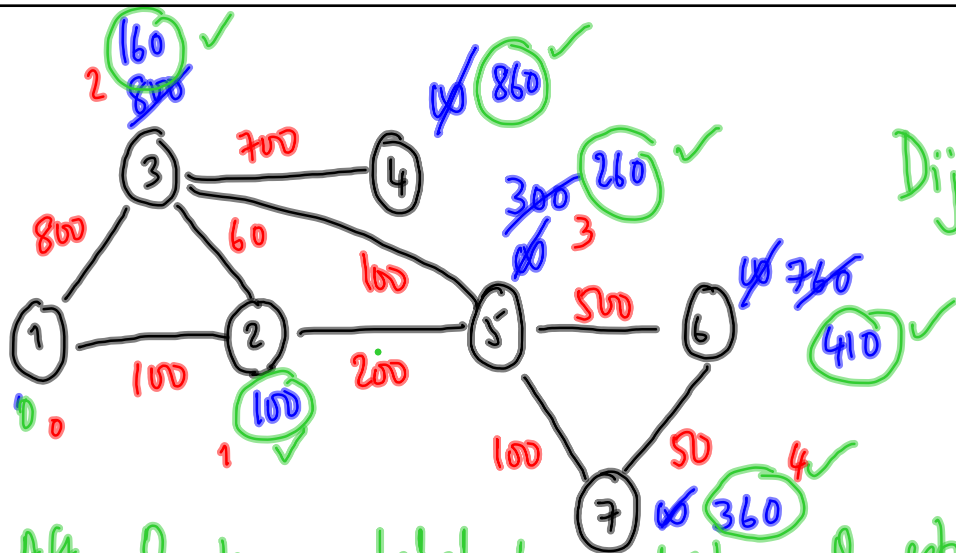


Want shortest path from 1 to every other vertex

Edge weights $w: E \to \mathbb{N} = \{1, 2, \ldots\}$

- represent "costs" - money, distance, time
  - replace 1 by w in adj matrix
  - in an adj list - pairs $i \to \{(j, w), (k, w') \ldots\}$

Dijkstra's algorithm



After 0 steps – label 1 as distance 0, estimates to other nodes

After 1 step – label shortest distance to 2 as 100

recompute all estimates

2nd step: Have explored {1,2} Possible extension are 3, 5 (✓ ✗)

recompute estimates

Analogy:

Edges are ropes with length = cost

Set fire to initial vertex

Fire propagates at unit speed along all edges incident to a burnt vertex

Implementation:

burnt[v] – has v already been visited?

estimate[v] – best estimate of shortest distance to v

Initially: (assume we start at 1)

$\forall v.\ burnt[v] = 0$ $O(n)$

$estimate[1] = 0$

$\forall v \neq 1$ $estimate[v] = \infty$ $\Big\}\ O(n)$

$O(n^2)$ as written

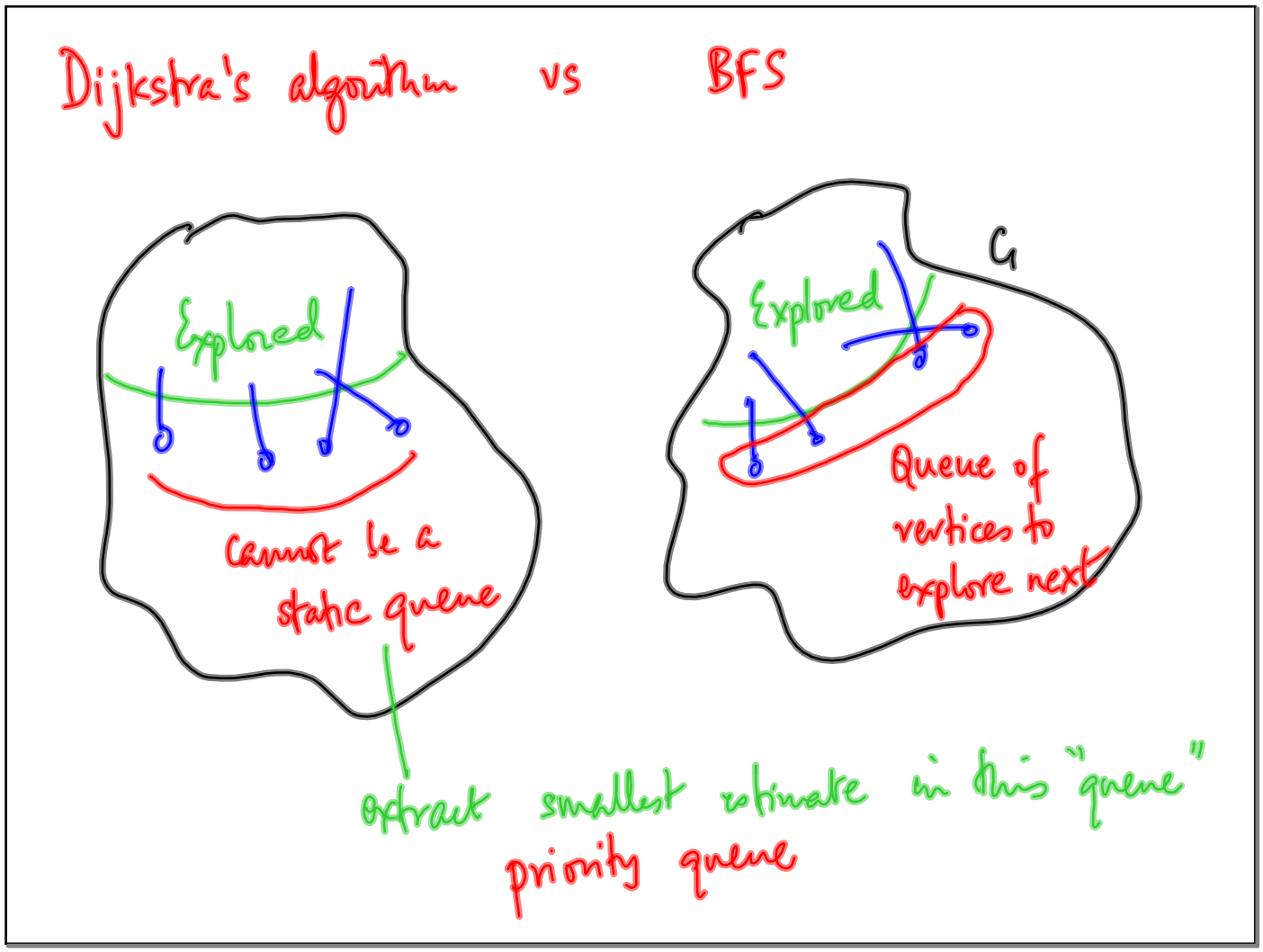
While there are unburnt vertices

n iterations

pick v s.t. $burnt[v] == 0$ & $estimate[v]$ is min — $O(n)$

$burnt[v] = 1$

$\forall w, (v,w) \in E$ update $estimate[w]$ as $\min(estimate[w], estimate[v] + weight(v,w))$


Dijkstra's algorithm vs BFS
Explored
cannot be a
static queue
extract smallest estimate in this "queue"
priority queue
G
Explored
Queue of
vertices to
explore next

Priority queue is an evolving/dynamic set of elements

Add an element with a given priority

Remove the element with "best" priority

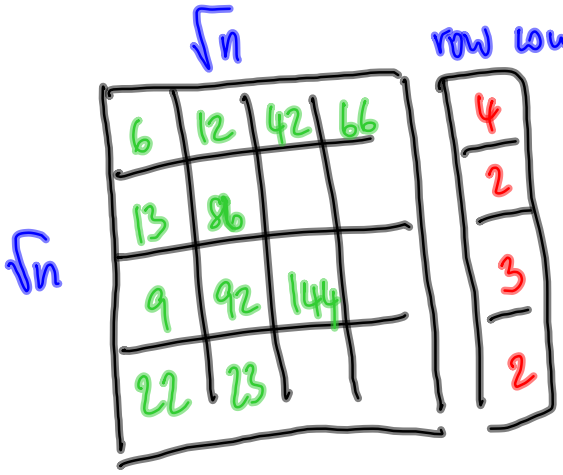
Update priority of elements already in queue (extension to standard priority queue)

Implementing priority queue

	Insert	Delete-best
Sorted list	$O(n)$	$O(1)$
Unsorted list	$O(1)$	$O(n)$

Way to go is to leave 1-dimensional data structures behind

Trivial 2D structure (assumes we have an upper bound n on size of p.q.)



Each row is sorted

delete-best: find min of col 1 + shift row $O(\sqrt{n})$

insert: Find first non-full row
Insert

$O(\sqrt{n})$