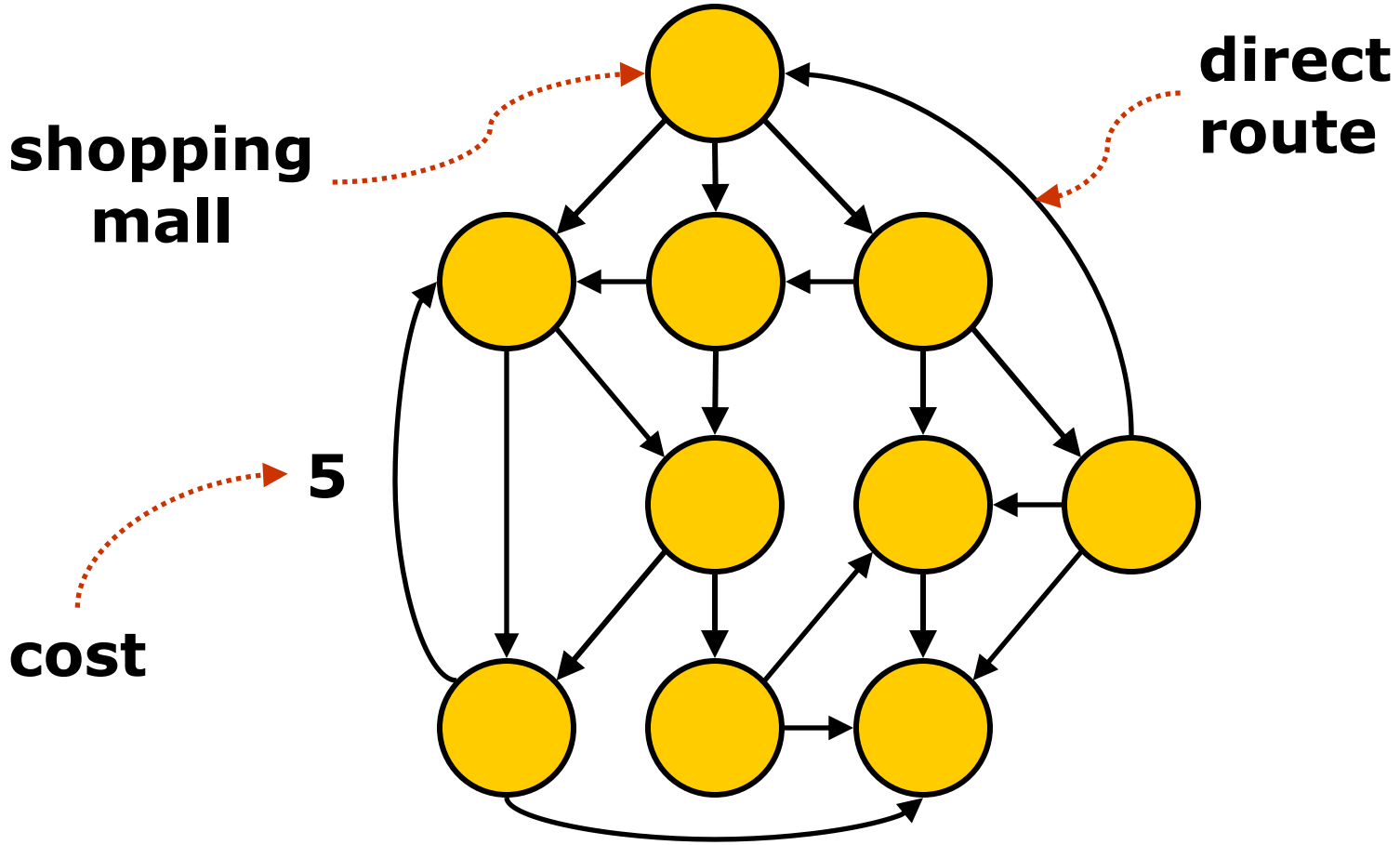
CS2040 Data Structures and Algorithms
Lecture Note #12

Graphs
Part 3: Shortest Path Algorithms

Travel Planning



Question

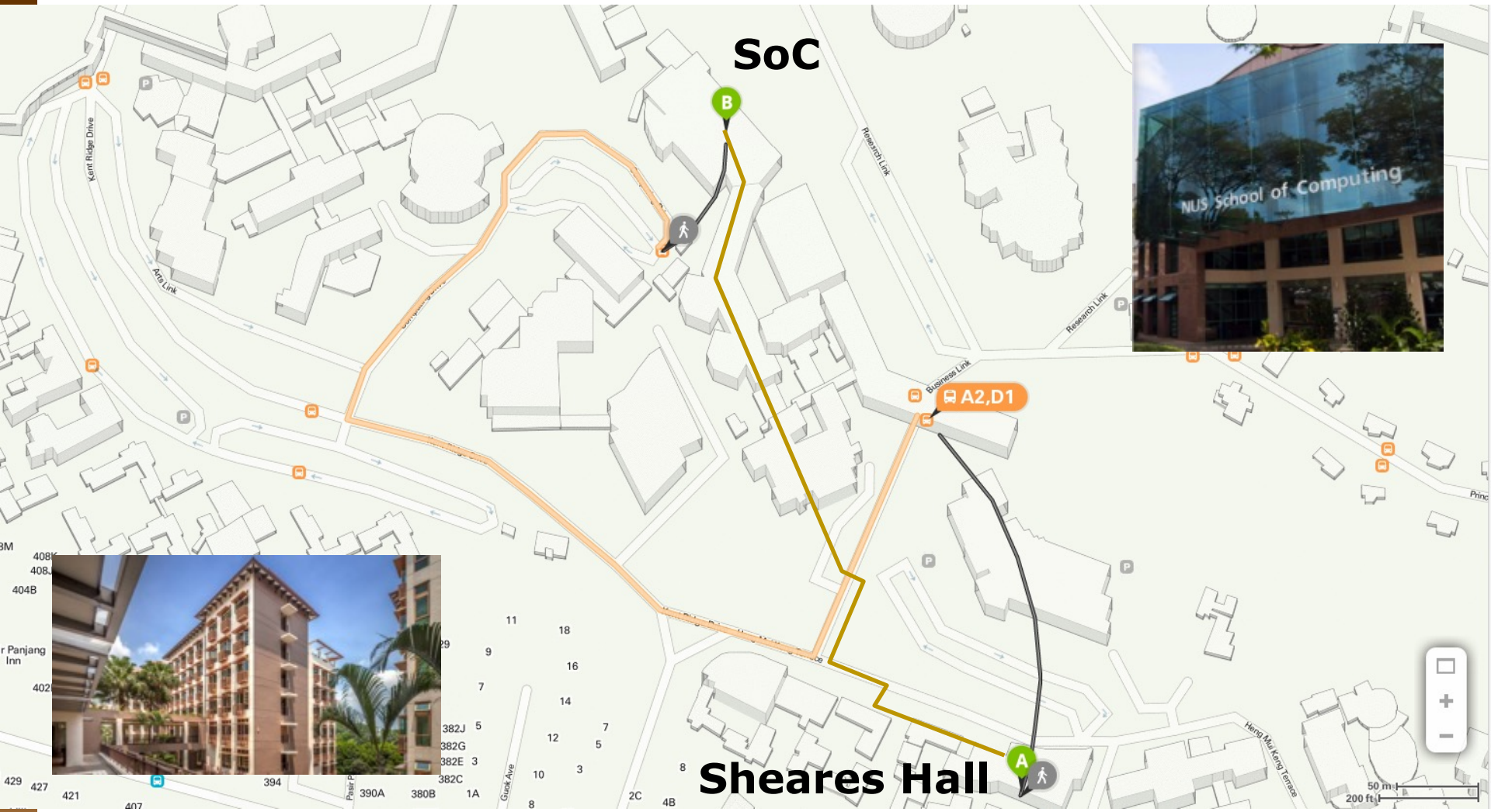
- What is the shortest way to travel between A and B?

  "**SHORTEST PATH PROBLEM**"

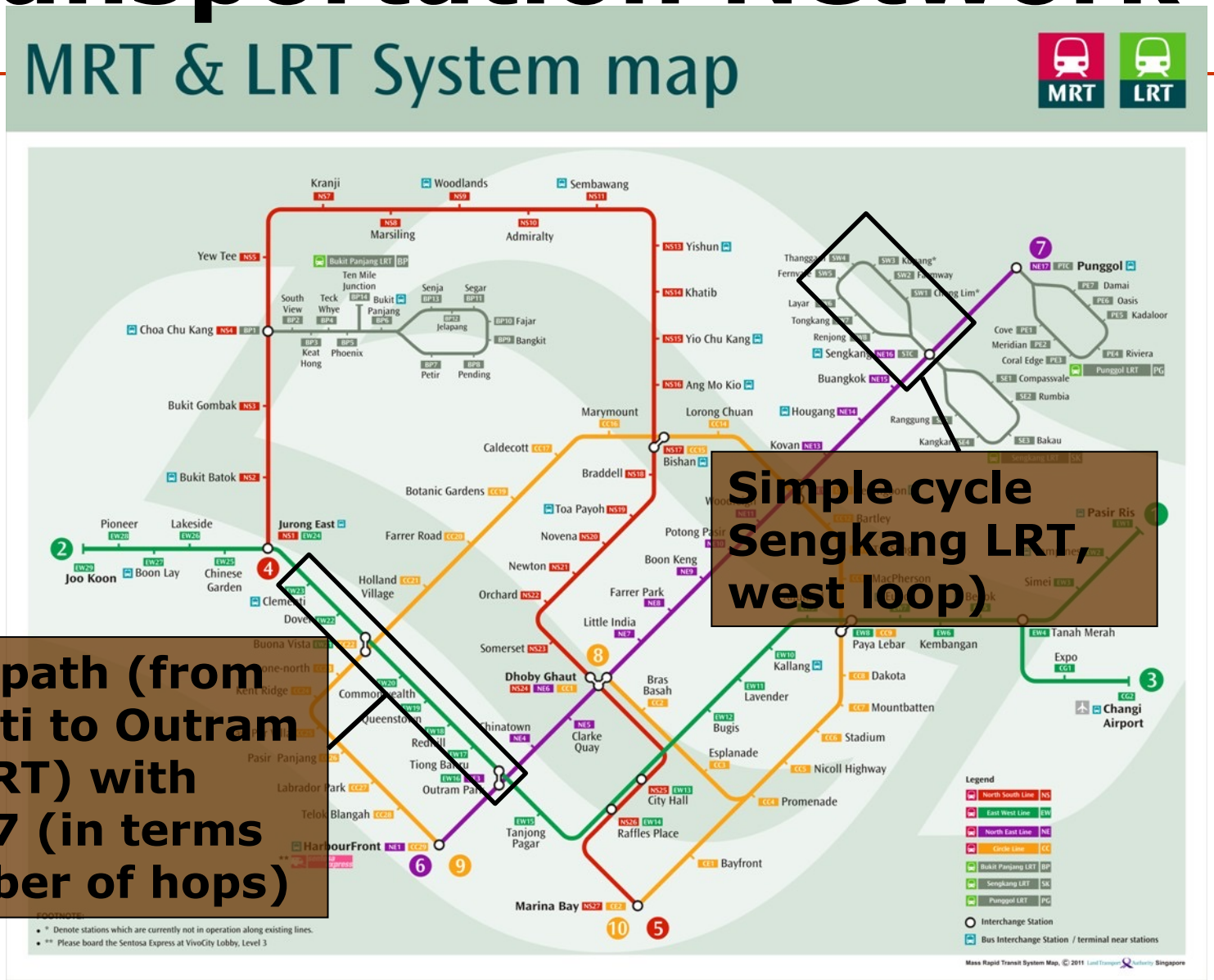
- How to minimize the cost of visiting n cities such that we visit each city exactly once, and finishing at the city where we start from?

  "TRAVELING SALESMAN PROBLEM (TSP)"

Motivating Example



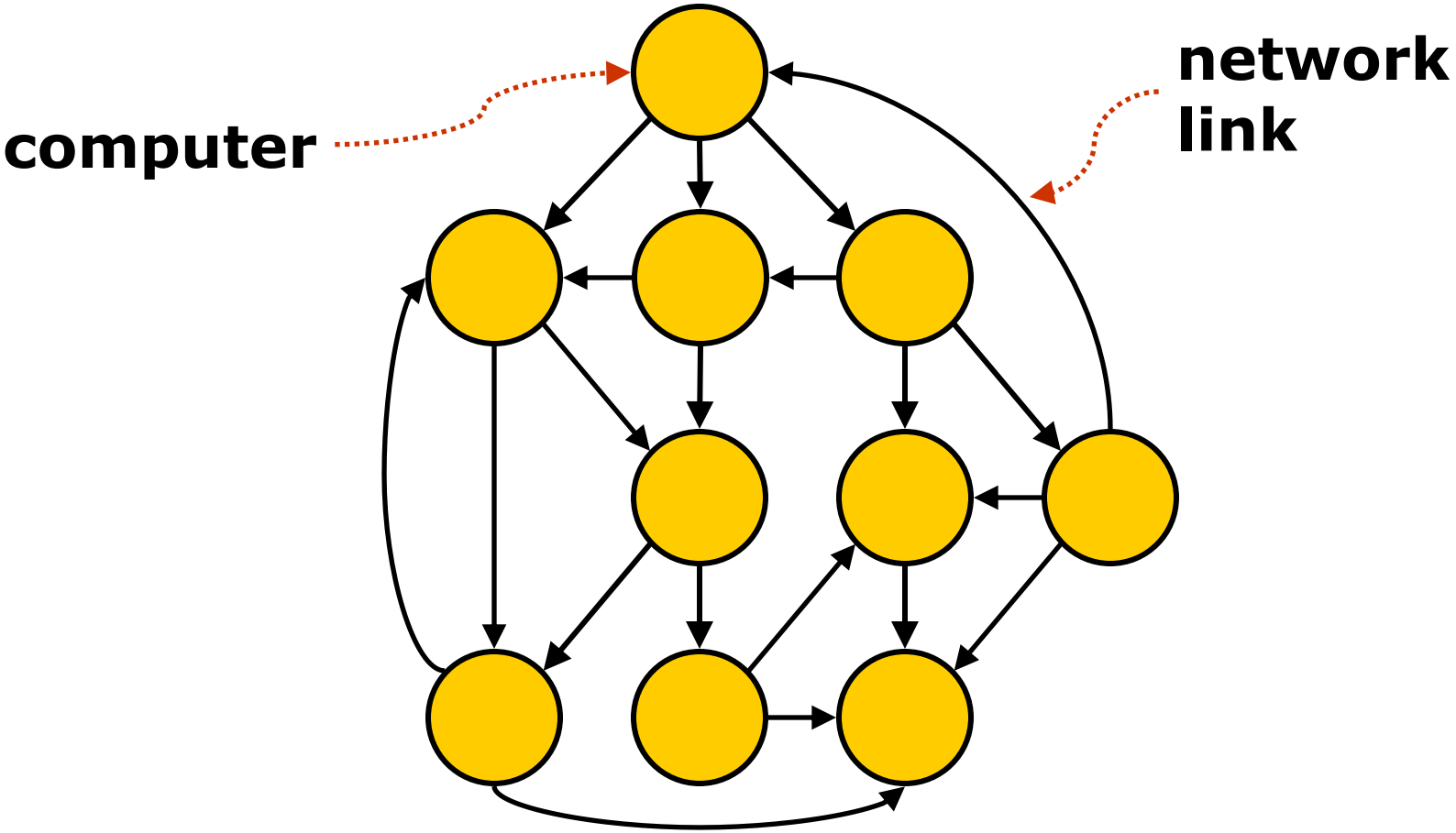
Transportation Network



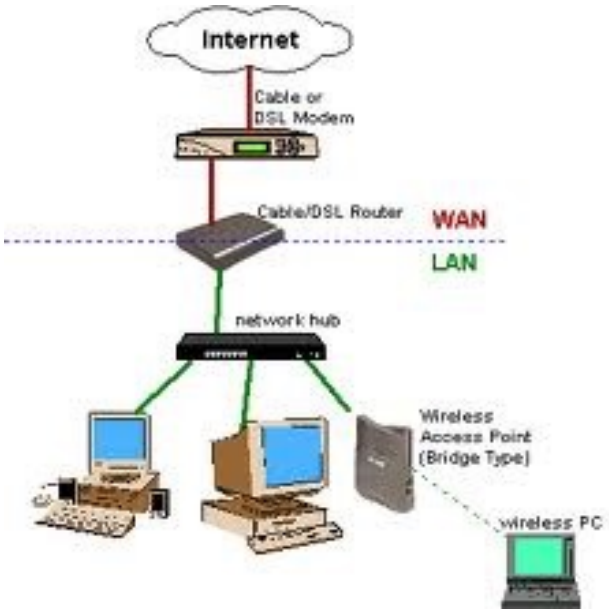
Simple path (from Clementi to Outram Park MRT) with length 7 (in terms of number of hops)

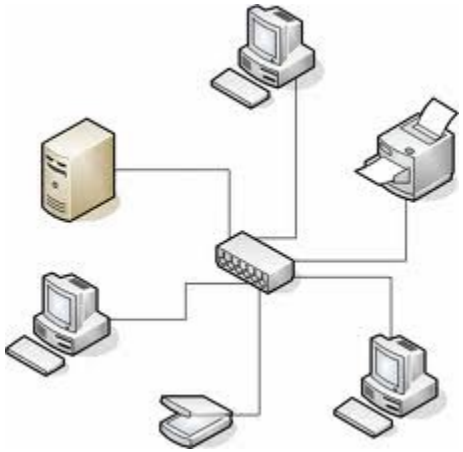
Simple cycle Sengkang LRT, west loop)

Internet

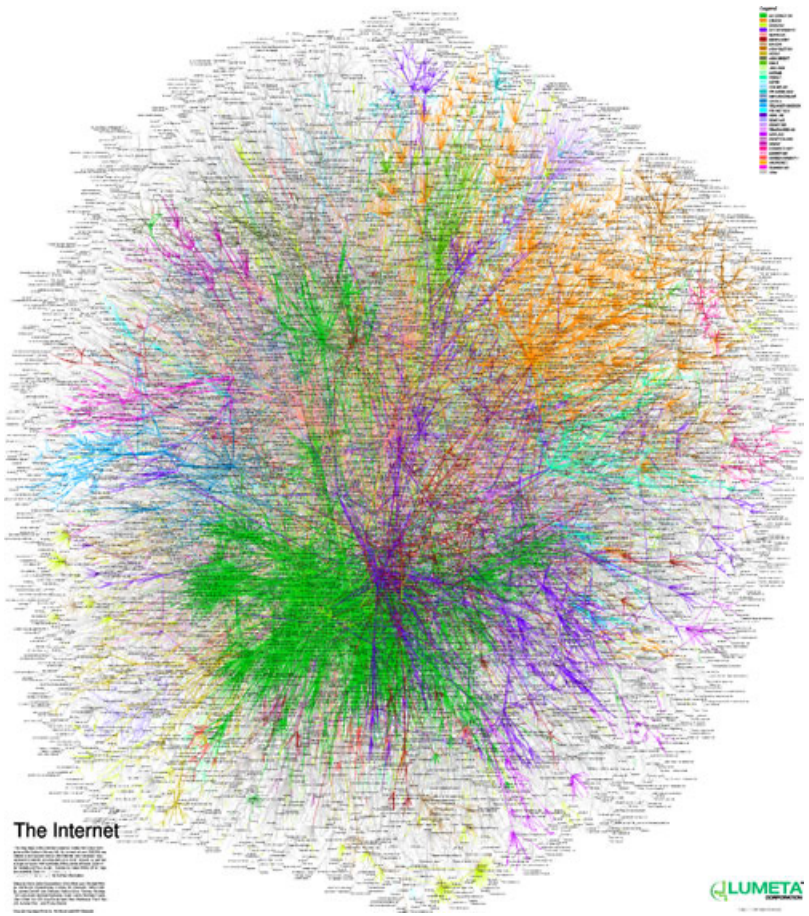


Internet / Computer Networks





Internet / Computer Networks



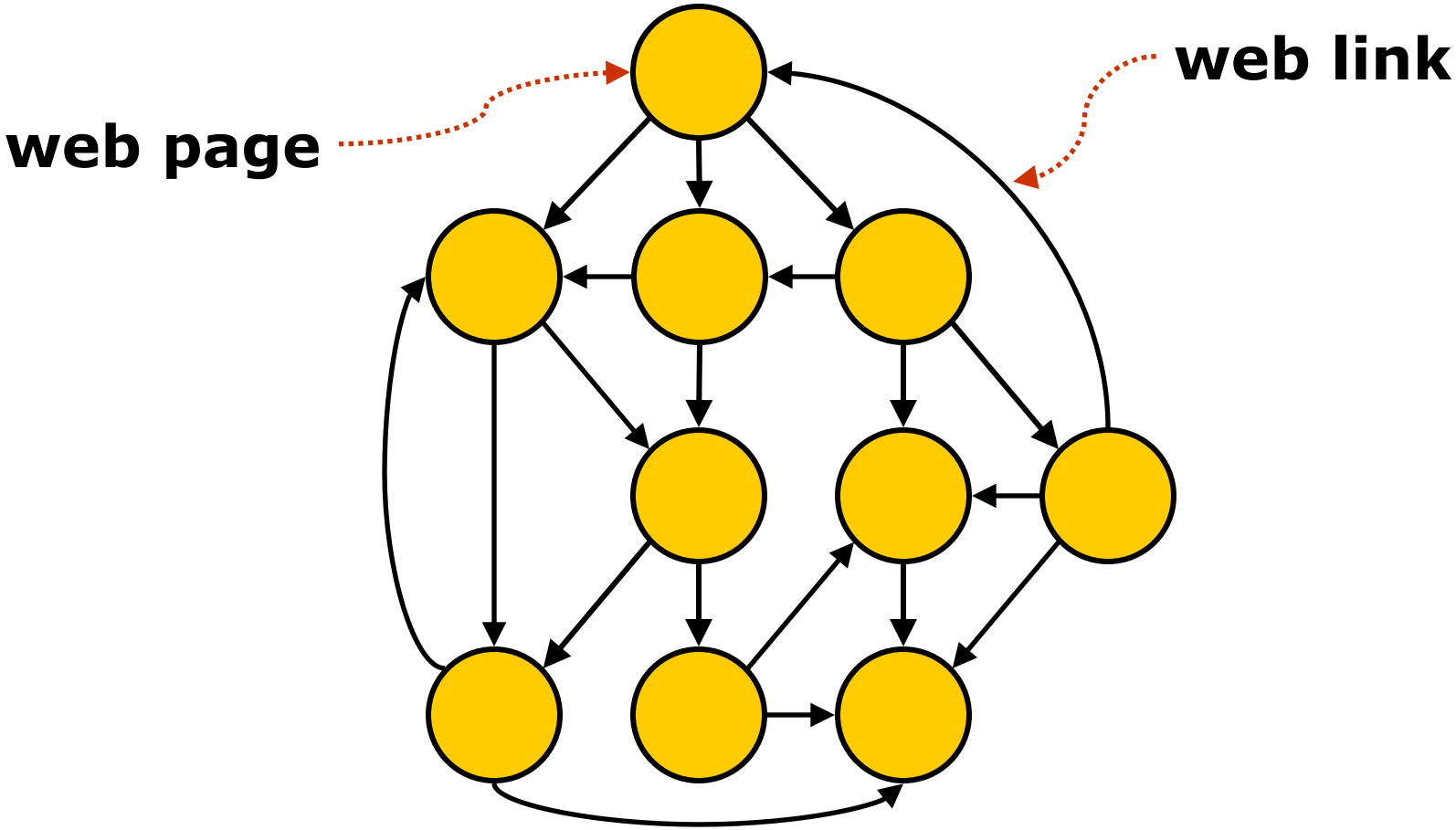
Question

- What is the shortest route to send a packet from A to B?

"SHORTEST PATH PROBLEM"

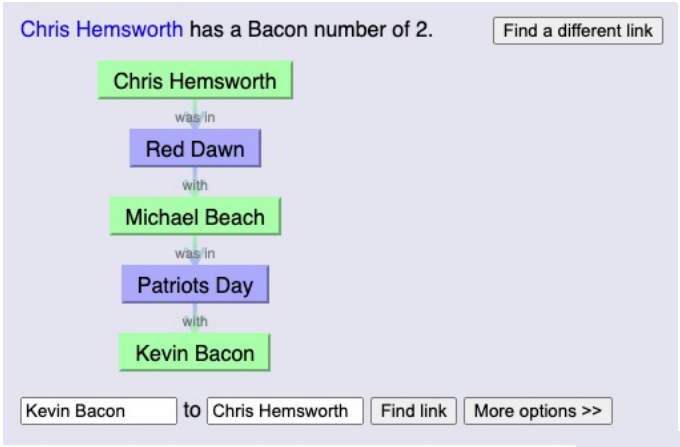
Communication Network

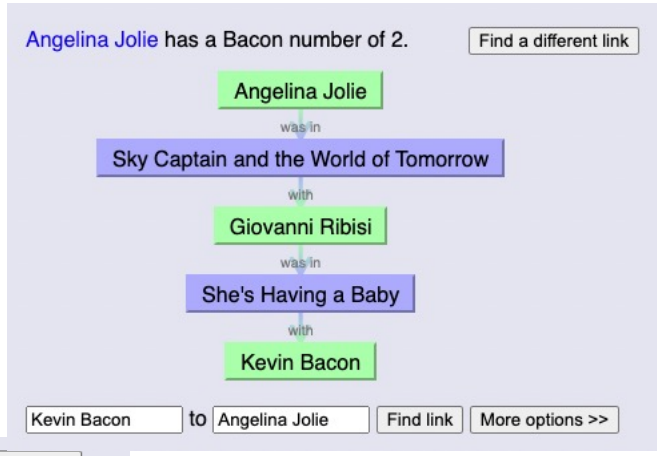
The Web

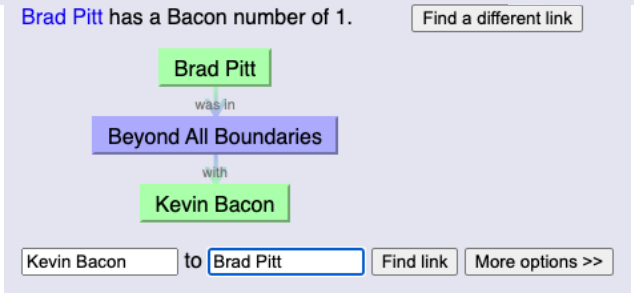


Social Networks (1)

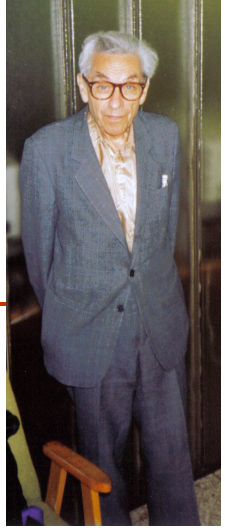
- Facebook, Instagram, etc.
- "Six-degrees of separation"
- How is any actor connected to Kevin Bacon ("The Oracle of Bacon"), e.g.:



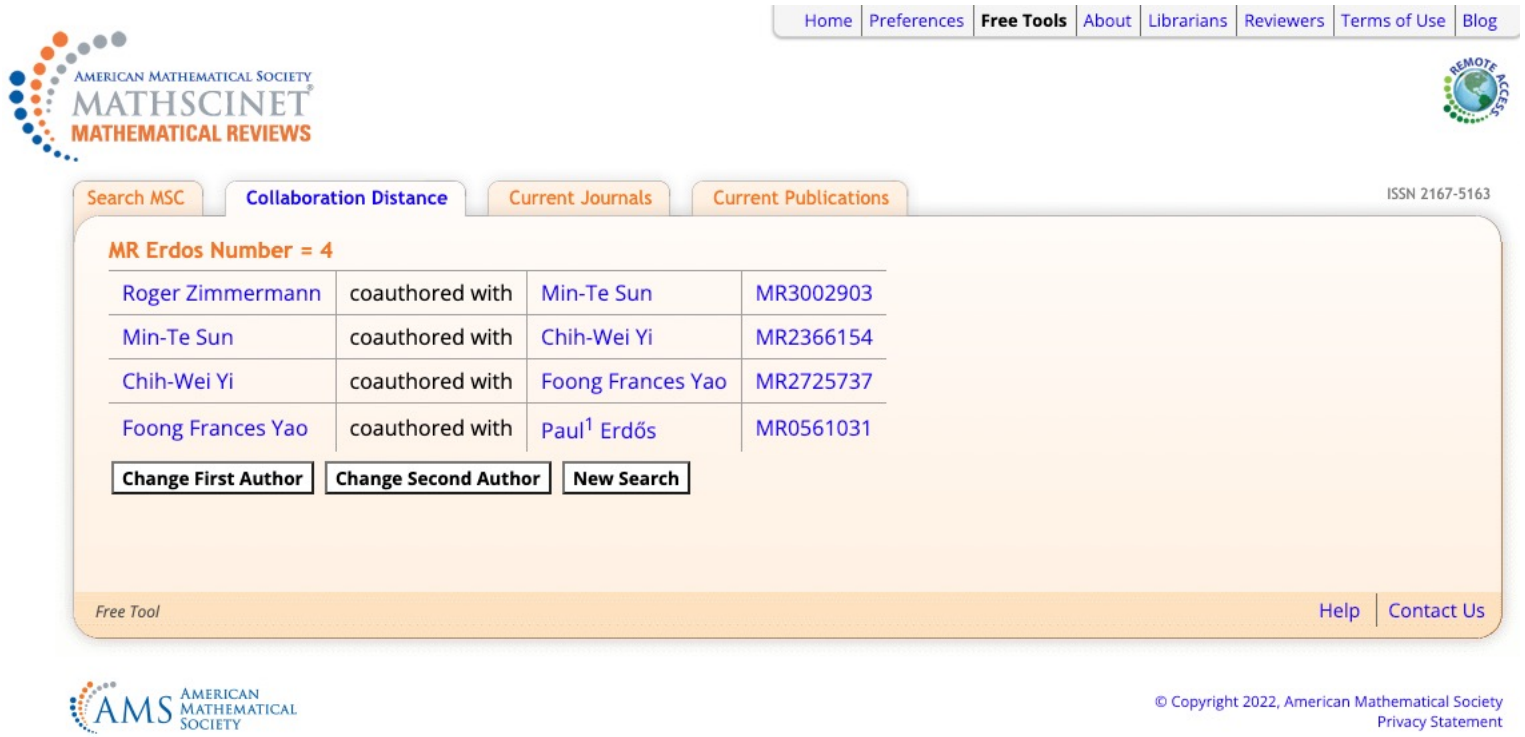




Social Networks (2)


Mathematician Paul Erdös

- Co-authors of scientific publications also form a social network, e.g., Erdös Number:

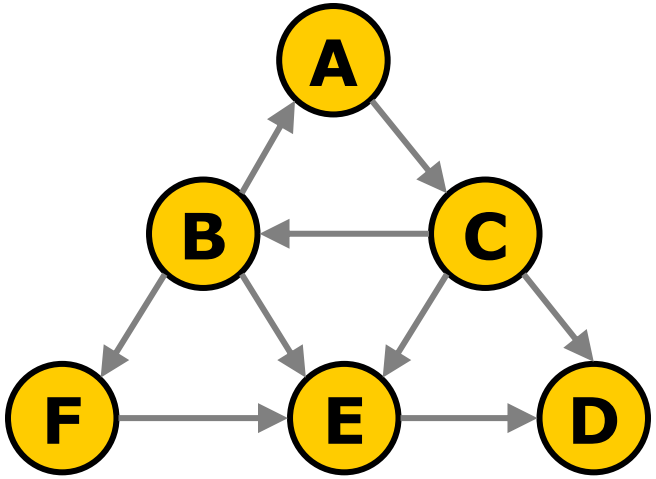


https://mathscinet.ams.org/mathscinet/freeTools.html?version=2

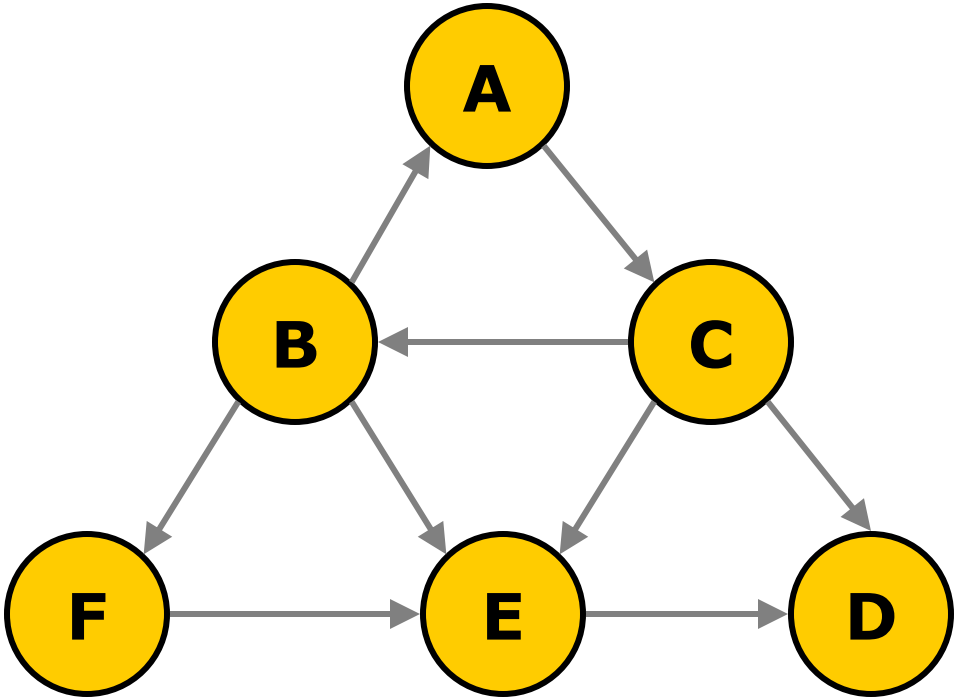
Definition

A path on a graph G is a sequence of vertices v_0, v_1, v_2, .. v_n where $(v_i, v_{i+1}) \in E$

The cost of a path is the sum of the cost of all edges in the path.



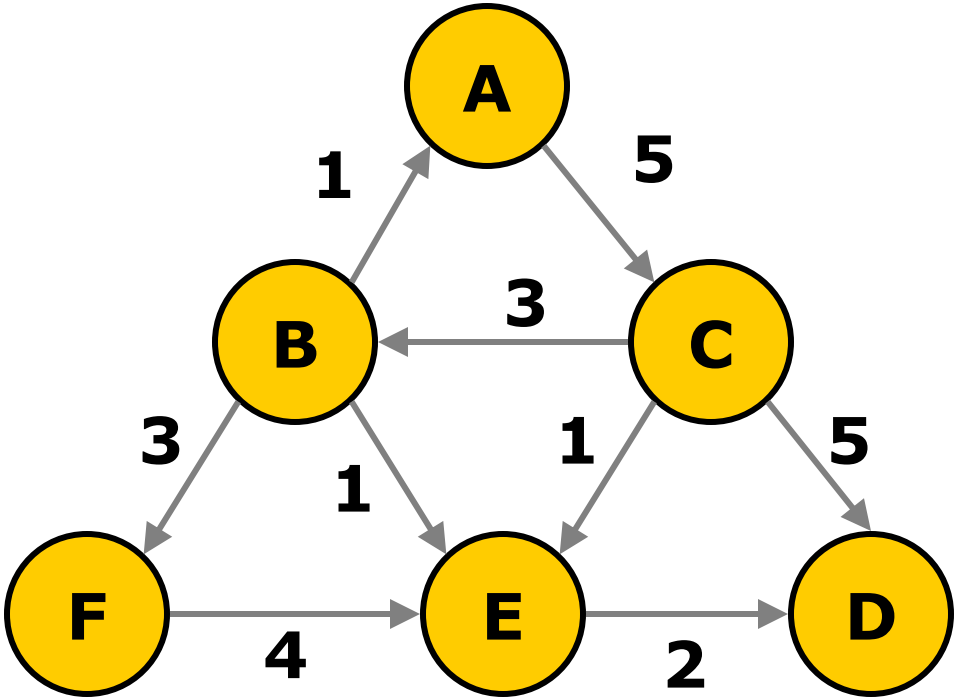
Unweighted shortest path



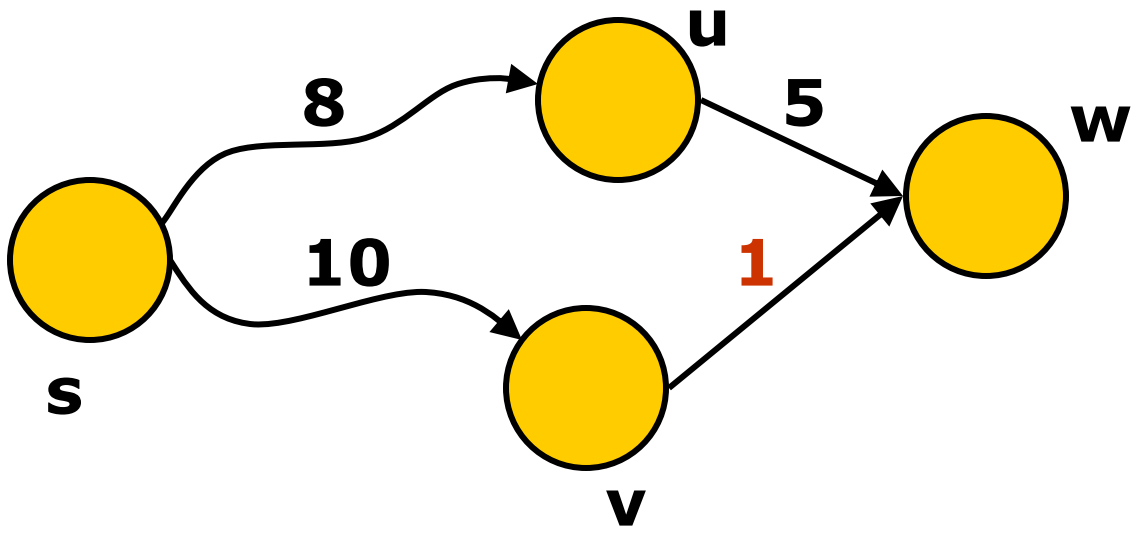
ShortestPath(s)

- Run BFS(s)
  - w.level: shortest distance from s
  - w.parent: shortest path from s

Positive weighted shortest path

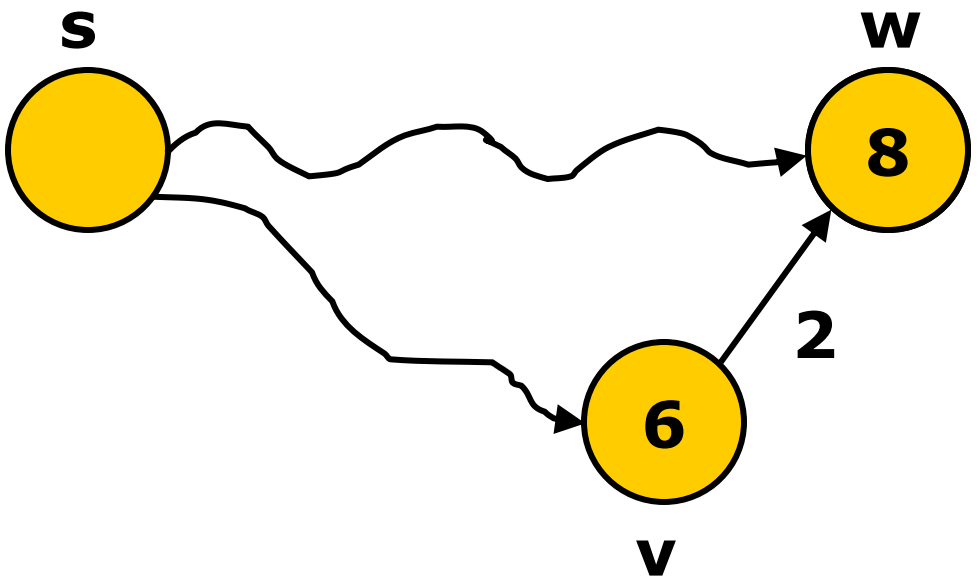


BFS(s) does not work

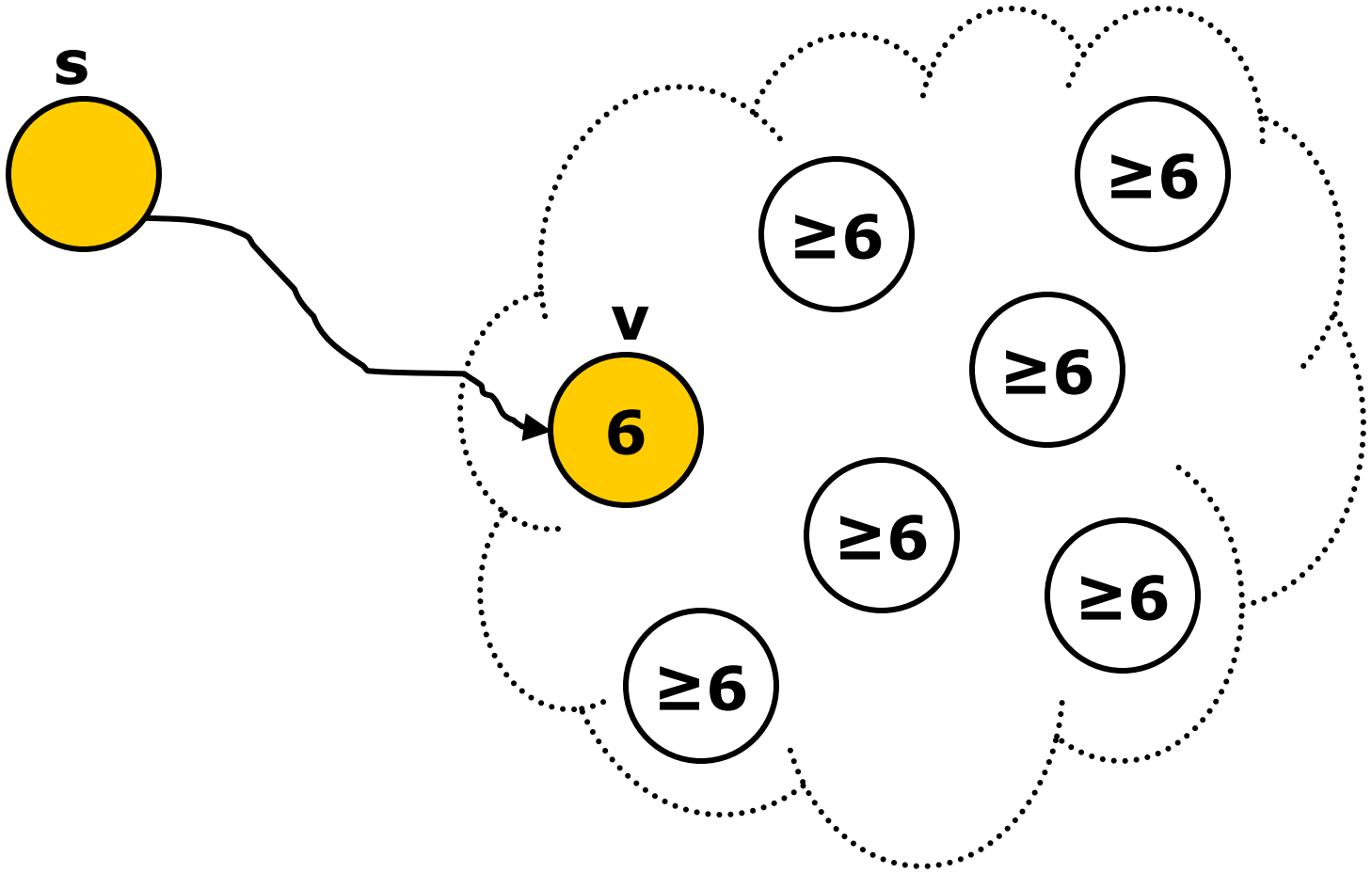


- Must keep track of smallest distance so far
- If we found a new, shorter path, update the distance.

Observation 1



Observation 2



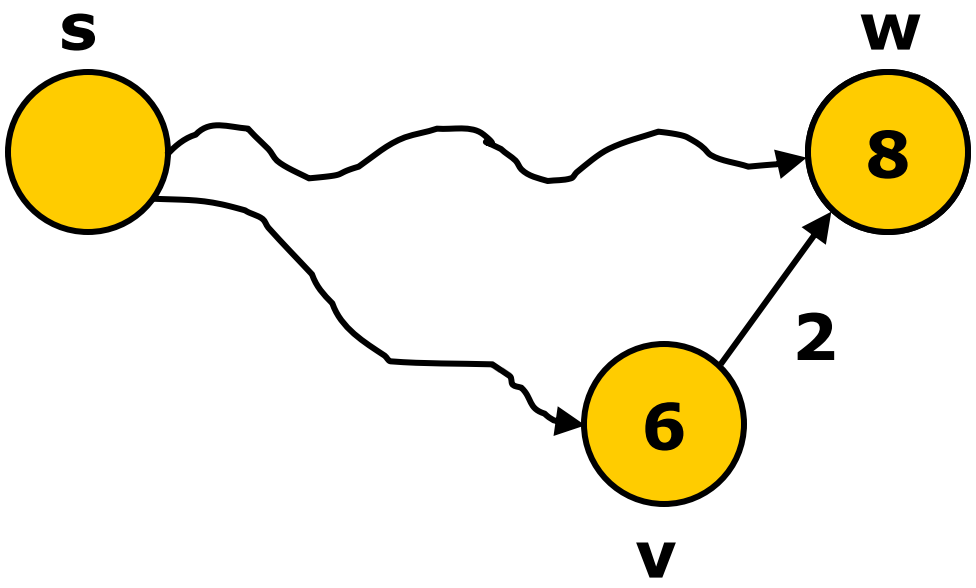
Definition

distance(v) : shortest distance so far from s to v

parent(v) : previous node on the shortest path so far from s to v

weight(u, v) : the weight of edge from u to v

Example



distance(w) = 8
weight(v,w) = 2
parent(w) = v

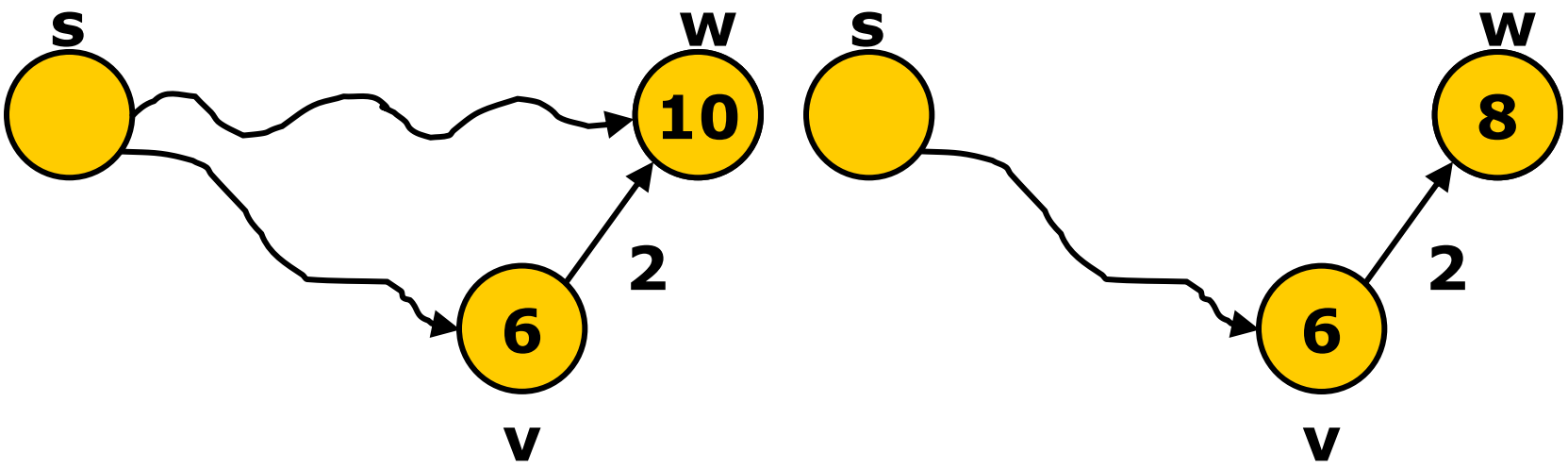
Relax(v,w)

d = distance(v) + weight(v,w)

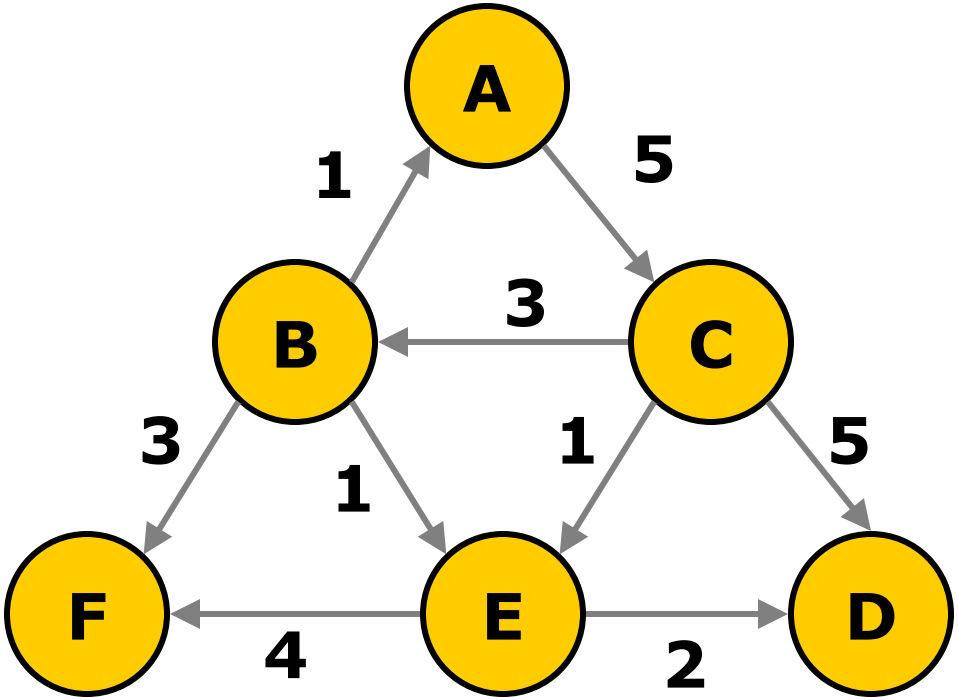
if distance(w) > d **then**

distance(w) = d

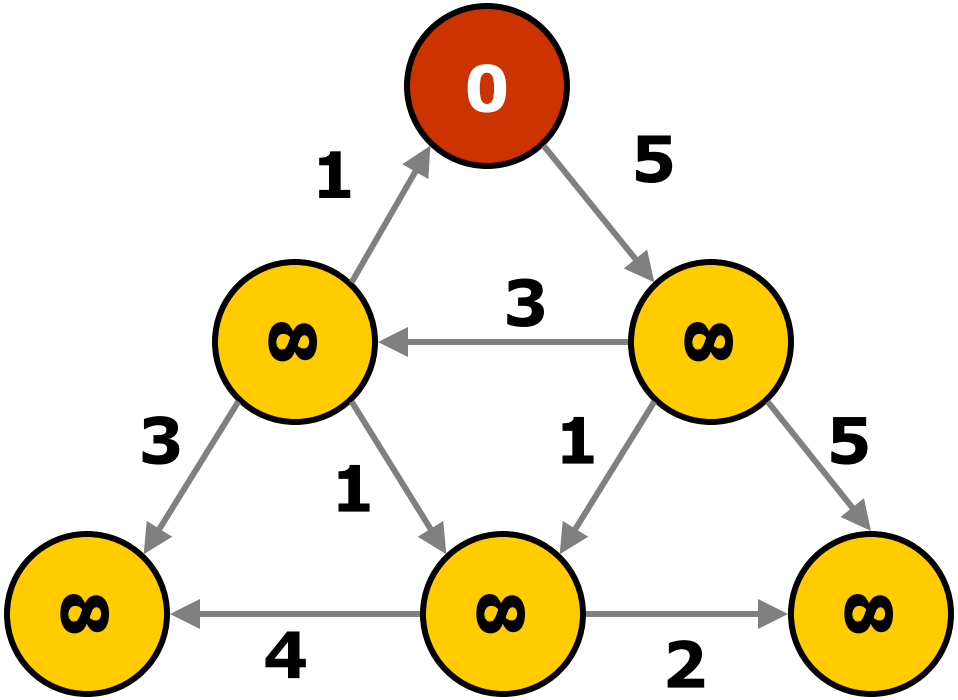
parent(w) = v



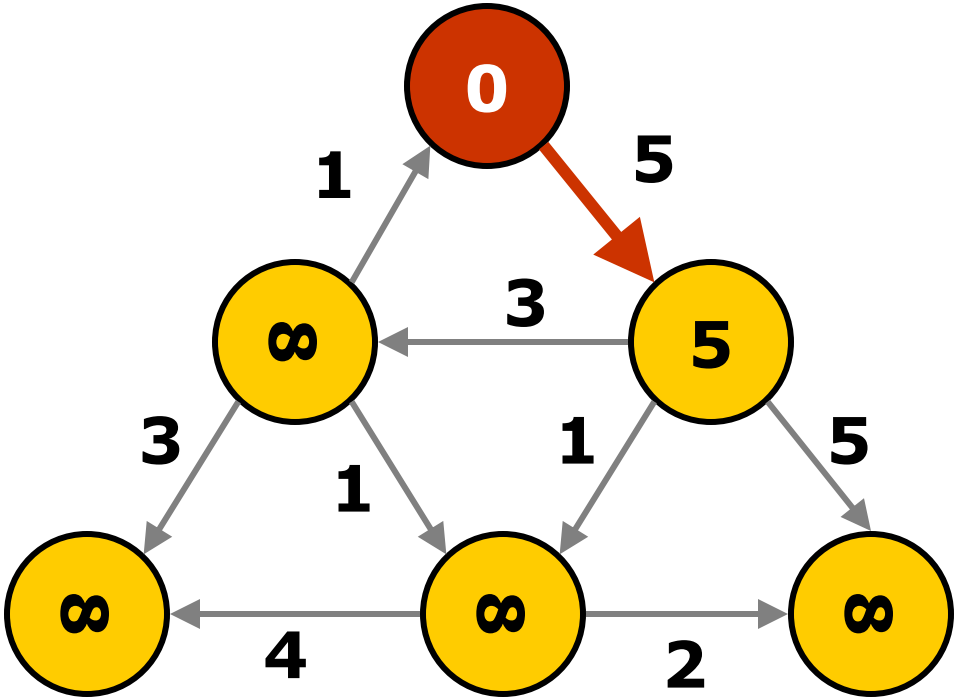
Dijkstra's algorithm



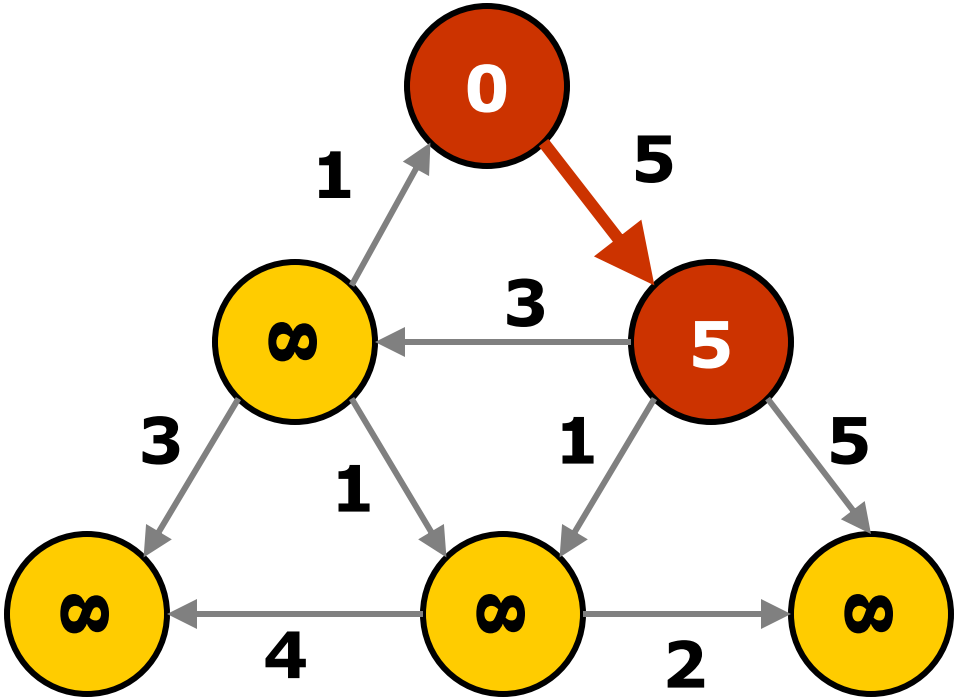
Dijkstra's algorithm



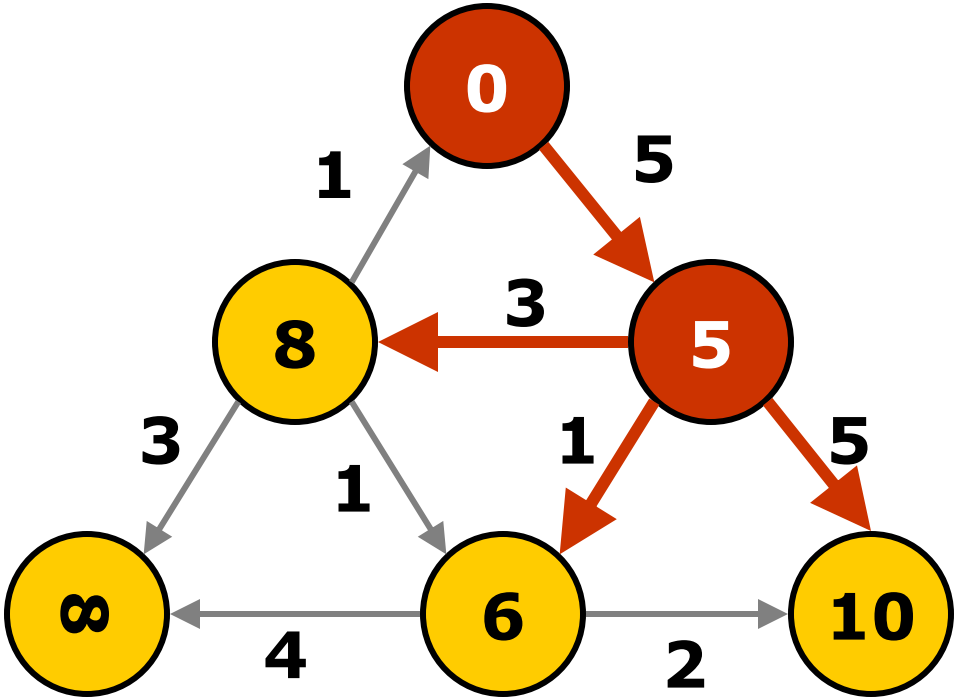
Dijkstra's algorithm



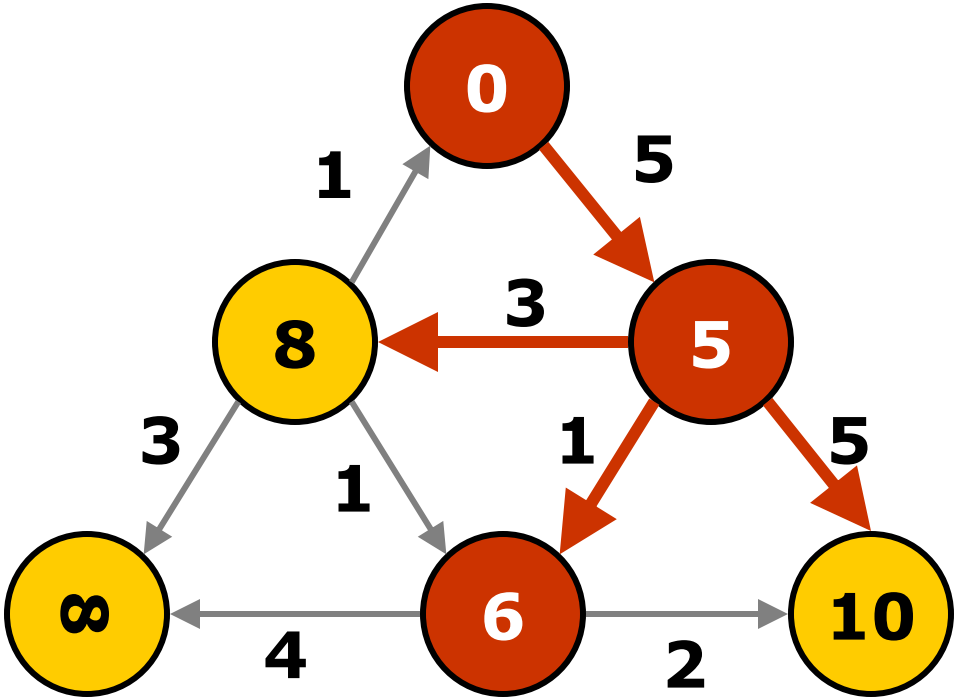
Dijkstra's algorithm



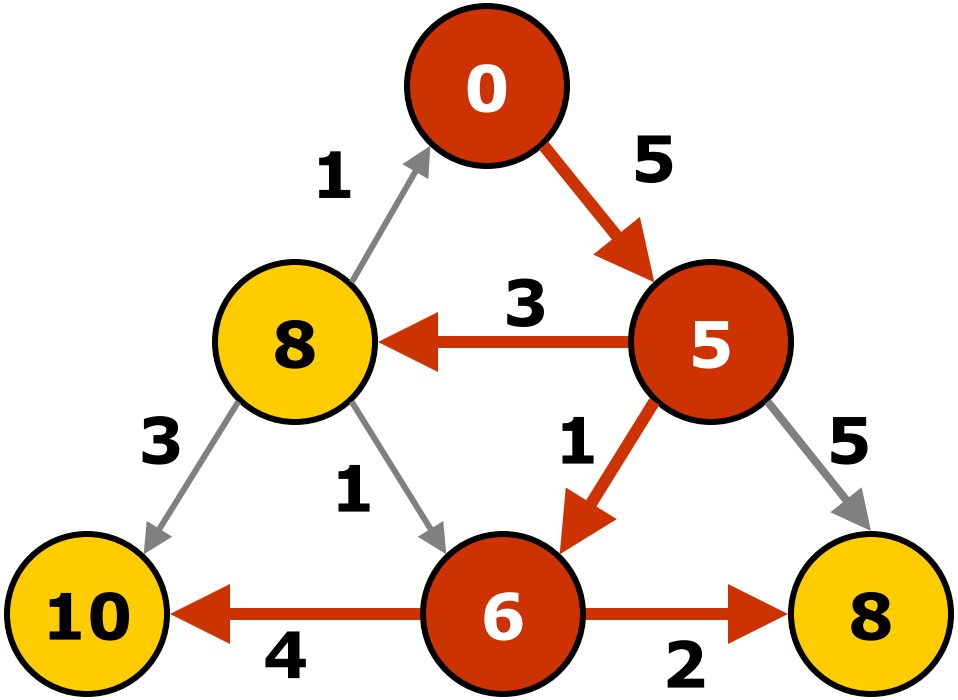
Dijkstra's algorithm



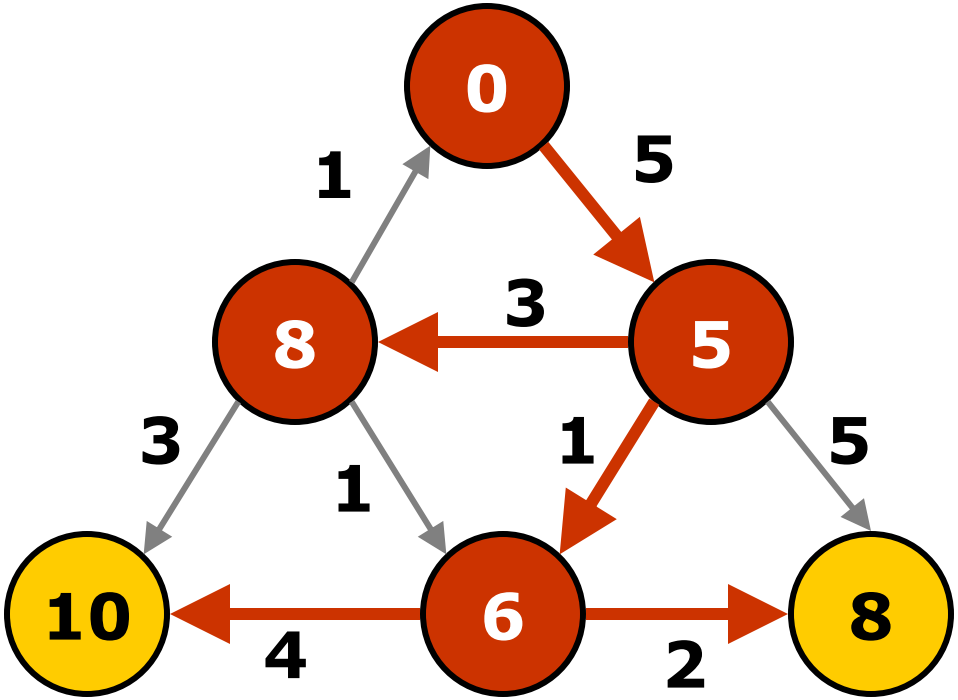
Dijkstra's algorithm



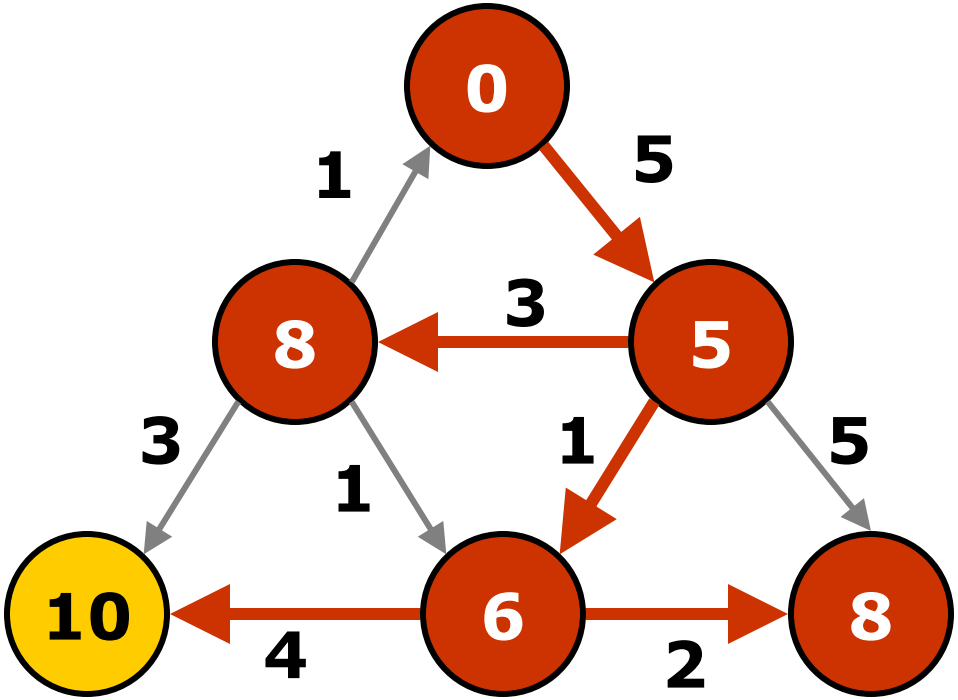
Dijkstra's algorithm



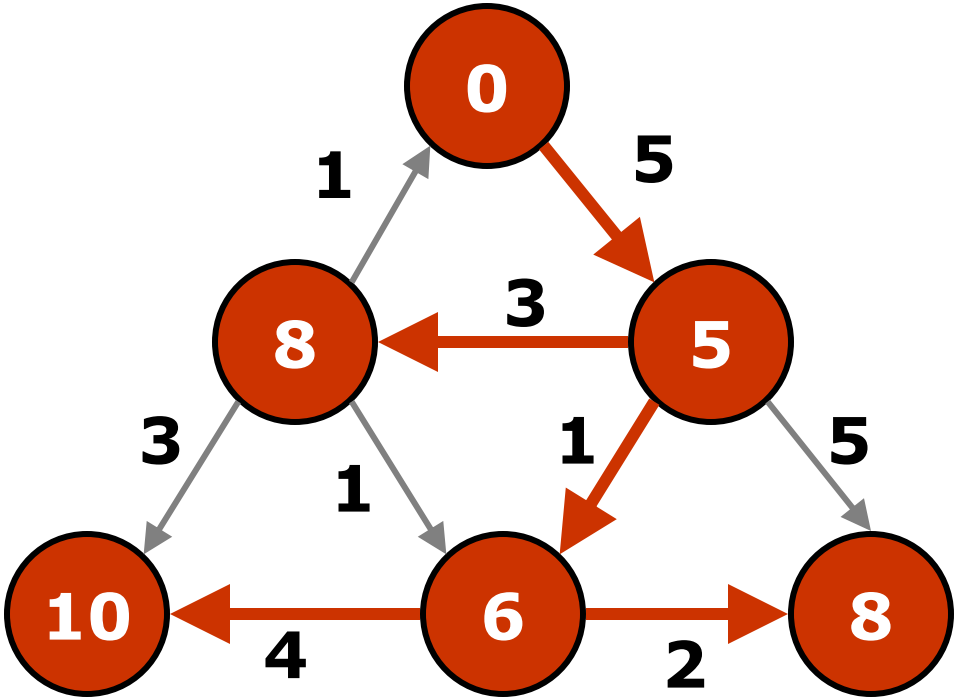
Dijkstra's algorithm



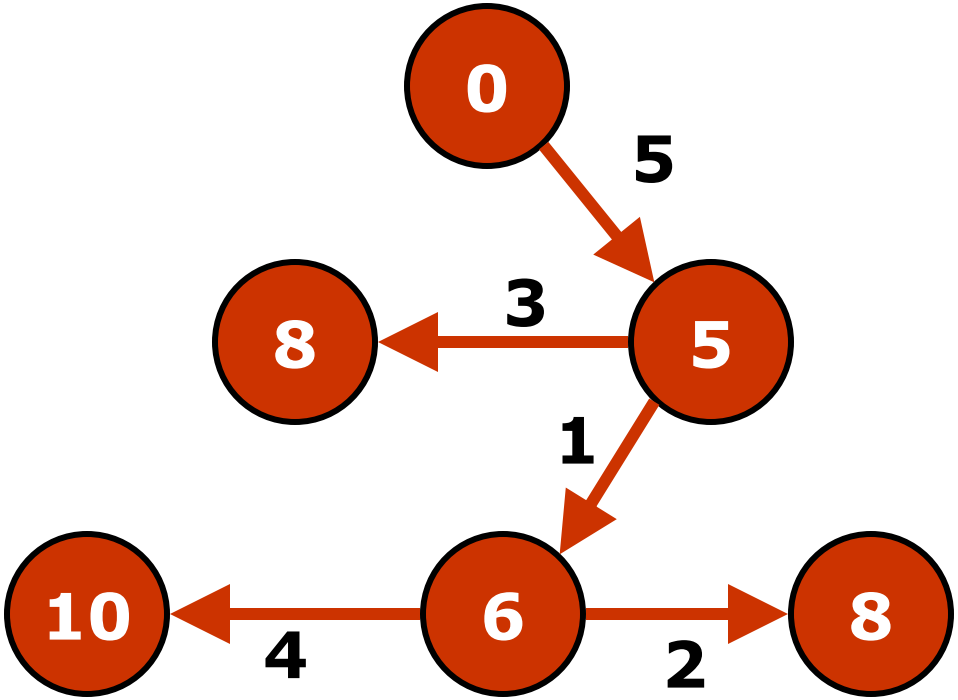
Dijkstra's algorithm



Dijkstra's algorithm



Dijkstra's algorithm



Dijkstra's algorithm

color all vertices yellow

foreach vertex w

  distance(w) = INFINITY

distance(s) = 0

Dijkstra's algorithm

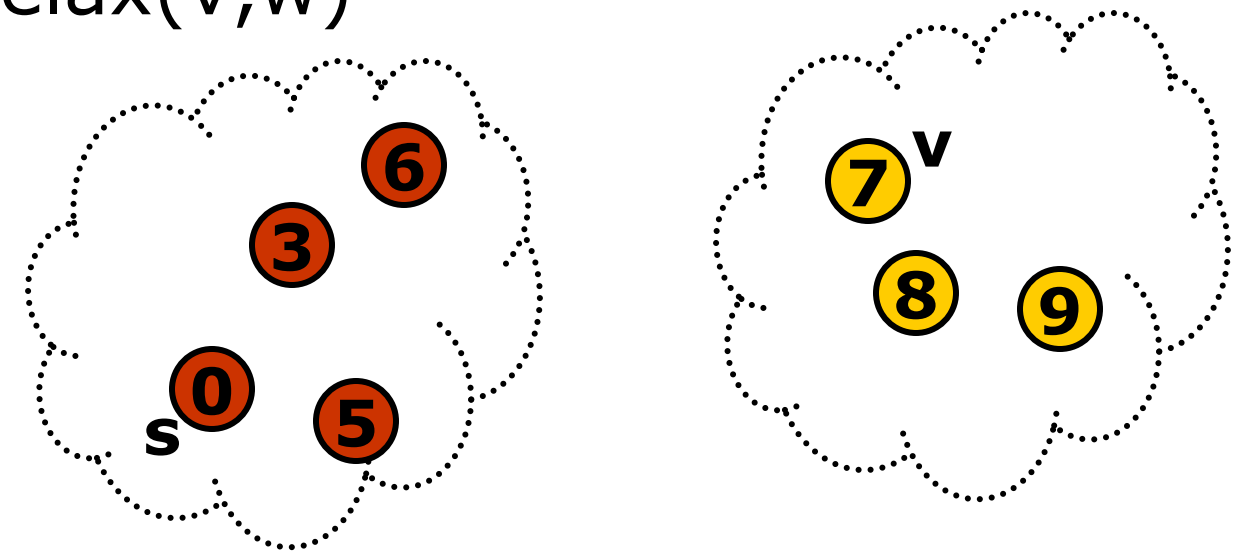
while there are yellow vertices

v = yellow vertex with min distance(v)

color v red

foreach neighbour w of v

relax(v,w)



Complexity


DILBERT
BY SCOTT ADAMS
HOW MANY DAYS WILL IT TAKE TO FINISH THE TESTS?
THREE.
YOU HAVE TWO.
I CAN'T DO IT IN TWO DAYS. THAT'S WHY I SAID THREE.
THAT WAS BEFORE I USED MY LEADERSHIP SKILLS TO TELL YOU TO DO IT IN TWO DAYS.
LEADERSHIP DOESN'T CHANGE THE LAWS OF PHYSICS. THE TEST TAKES THREE DAYS.
YOU HAVE TWO.
LEADERSHIP!
THESE TEST RESULTS LOOK INCOMPLETE.
JUST LIKE MY SOUL.
DilbertCartoonist@gmail.com
www.dilbert.com

Running time $O(V^2 + E)$

color all vertices yellow
foreach vertex w
  distance(w) = INFINITY
distance(s) = 0
while there are yellow vertices
  v = yellow vertex with min distance(v)
  color v red
  foreach neighbour w of v
    relax(v,w)

Using priority queue

```
foreach vertex w
   distance(w) = INFINITY
distance(s) = 0
pq = new PriorityQueue(V)

while pq is not empty
   v = pq.deleteMin()
   foreach neighbour w of v
        relax(v,w)
```

Initialization O(V)

foreach vertex w

distance(w) = INFINITY

distance(s) = 0

pq = new PriorityQueue(V)

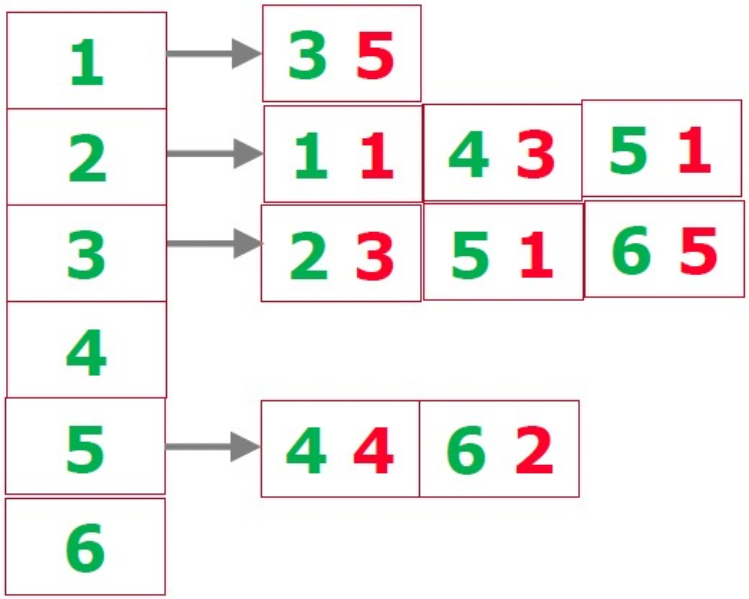
Main loop

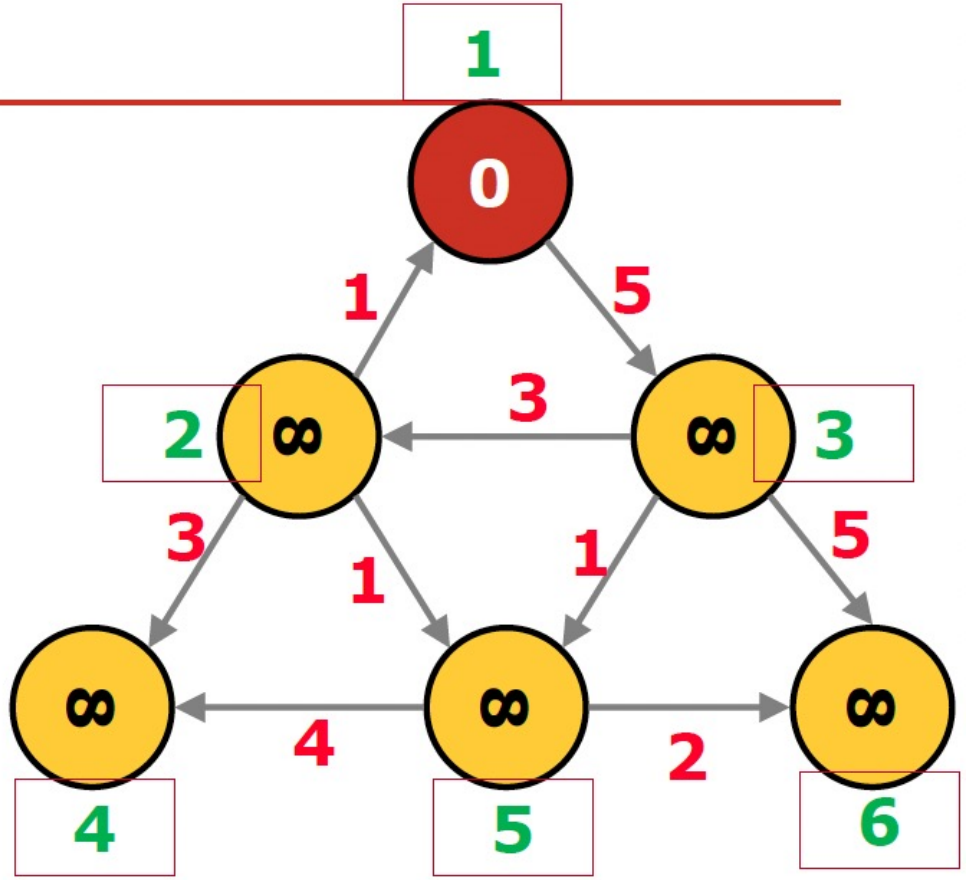
```
while pq is not empty
    v = pq.deleteMin()
    foreach neighbour w of v
        relax(v,w)
```

Main loop O((V+E) log V)

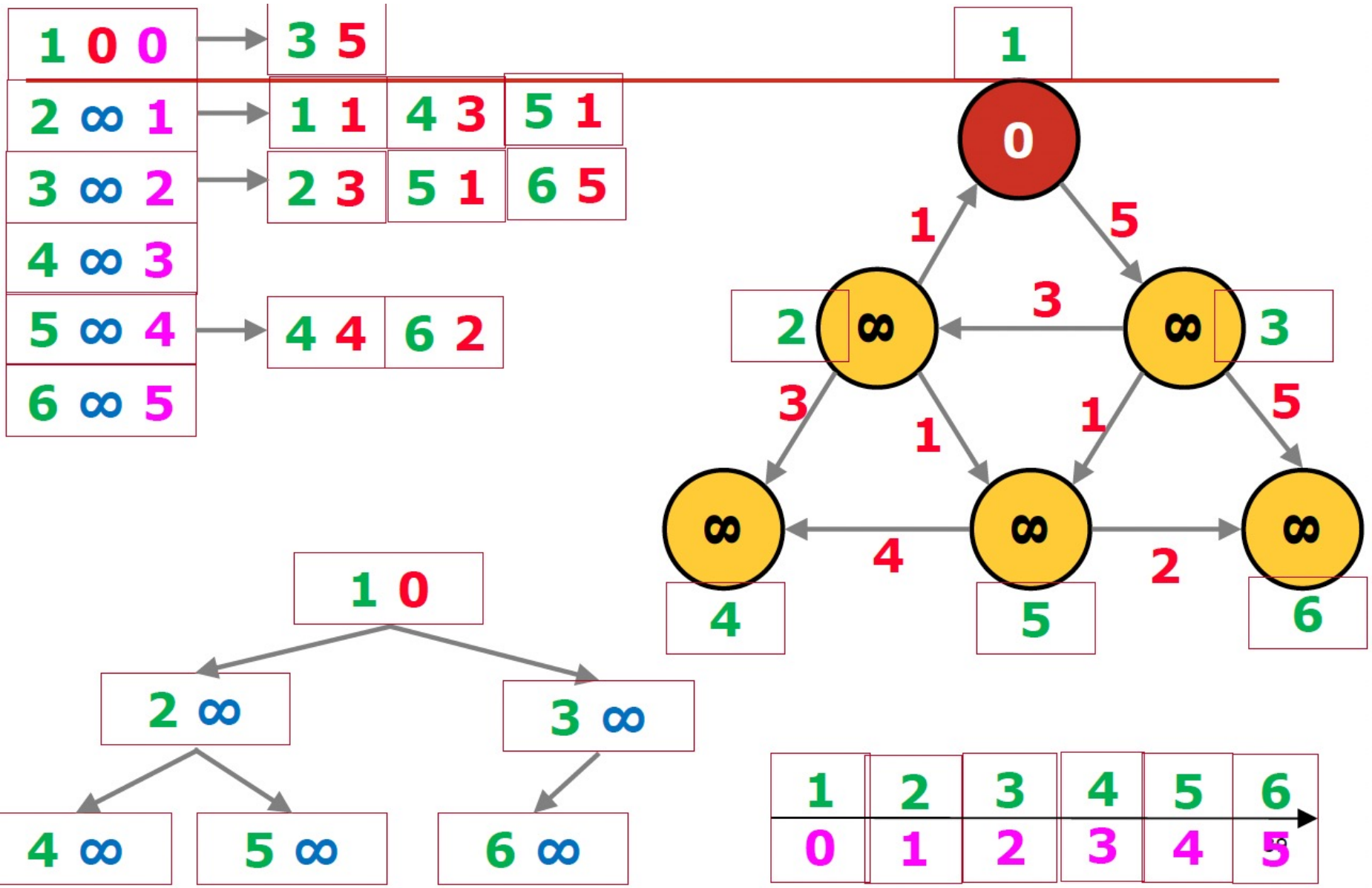
```
while pq is not empty
   v = pq.deleteMin()     // log(V)
   foreach neighbour w of v          //(log(v) *
        d = distance(v) + cost(v,w)   //  adj(v)
        if distance(w) > d then
              // distance(w) = d
              pq.decreaseKey(w, d)
              parent(w) = v

// sum(log(V) + log(V) * adj(v))
```

Dijkstra's algorithm

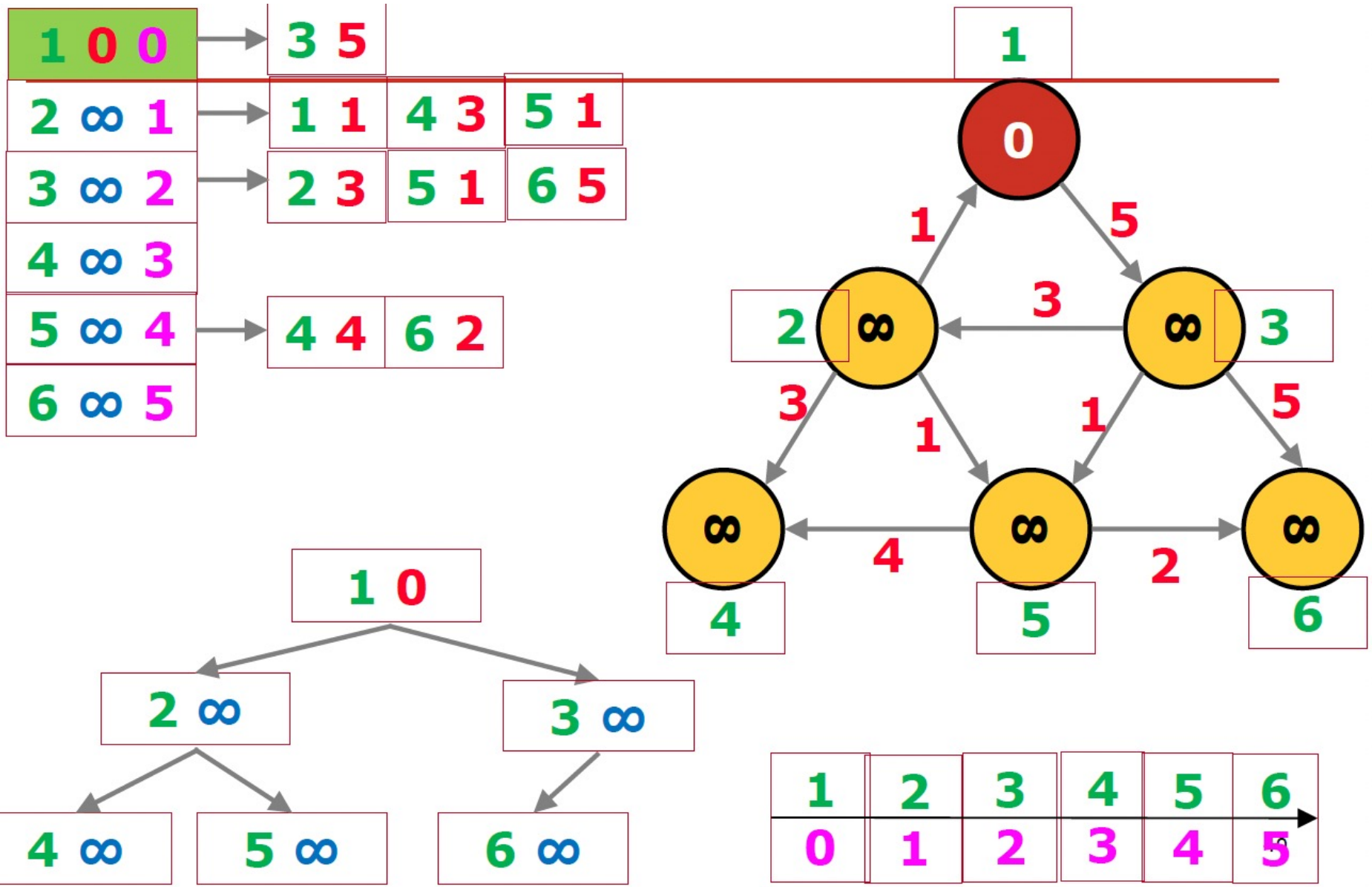




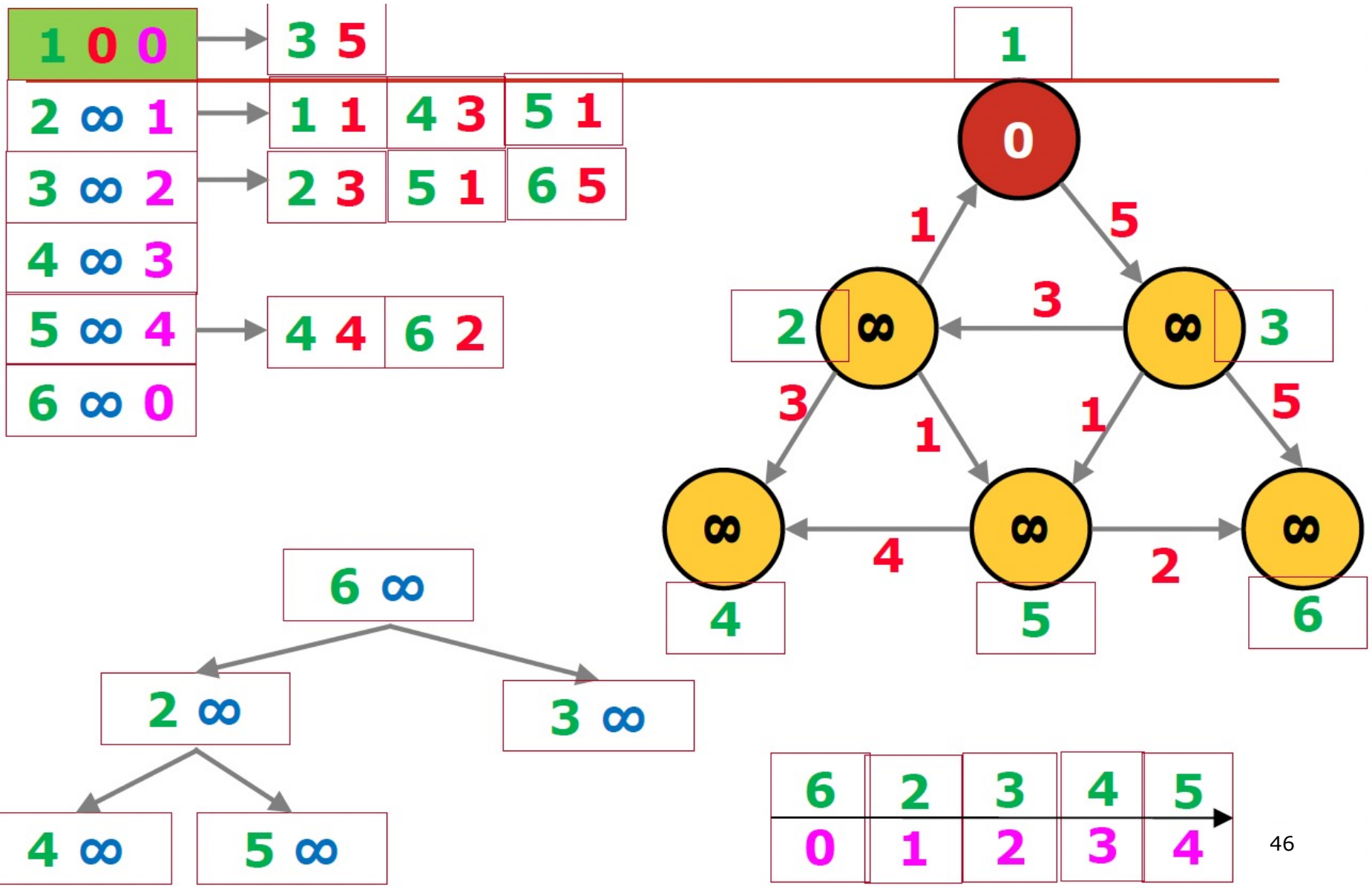
Dijkstra's algorithm



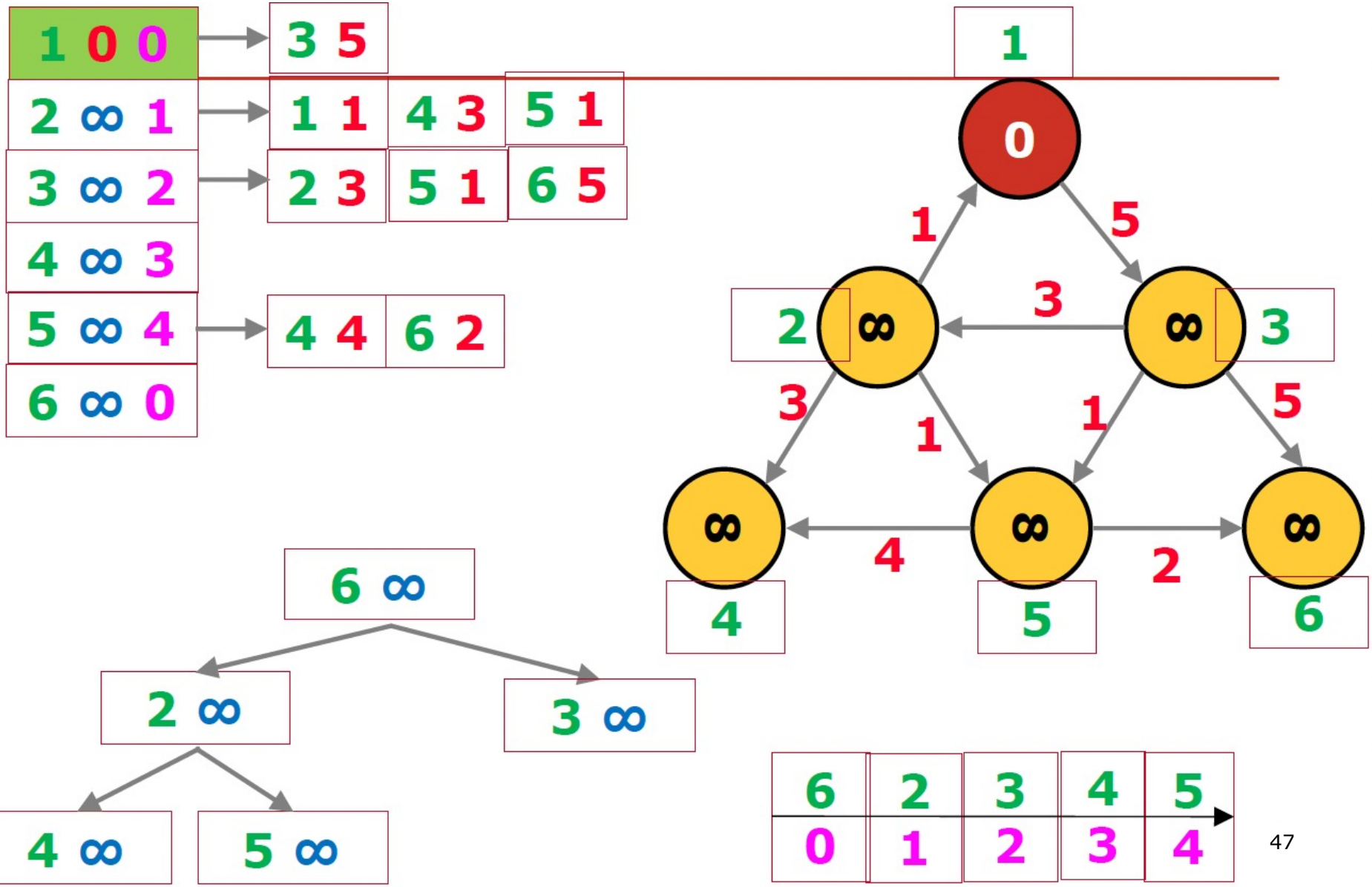
pq.deleteMin()



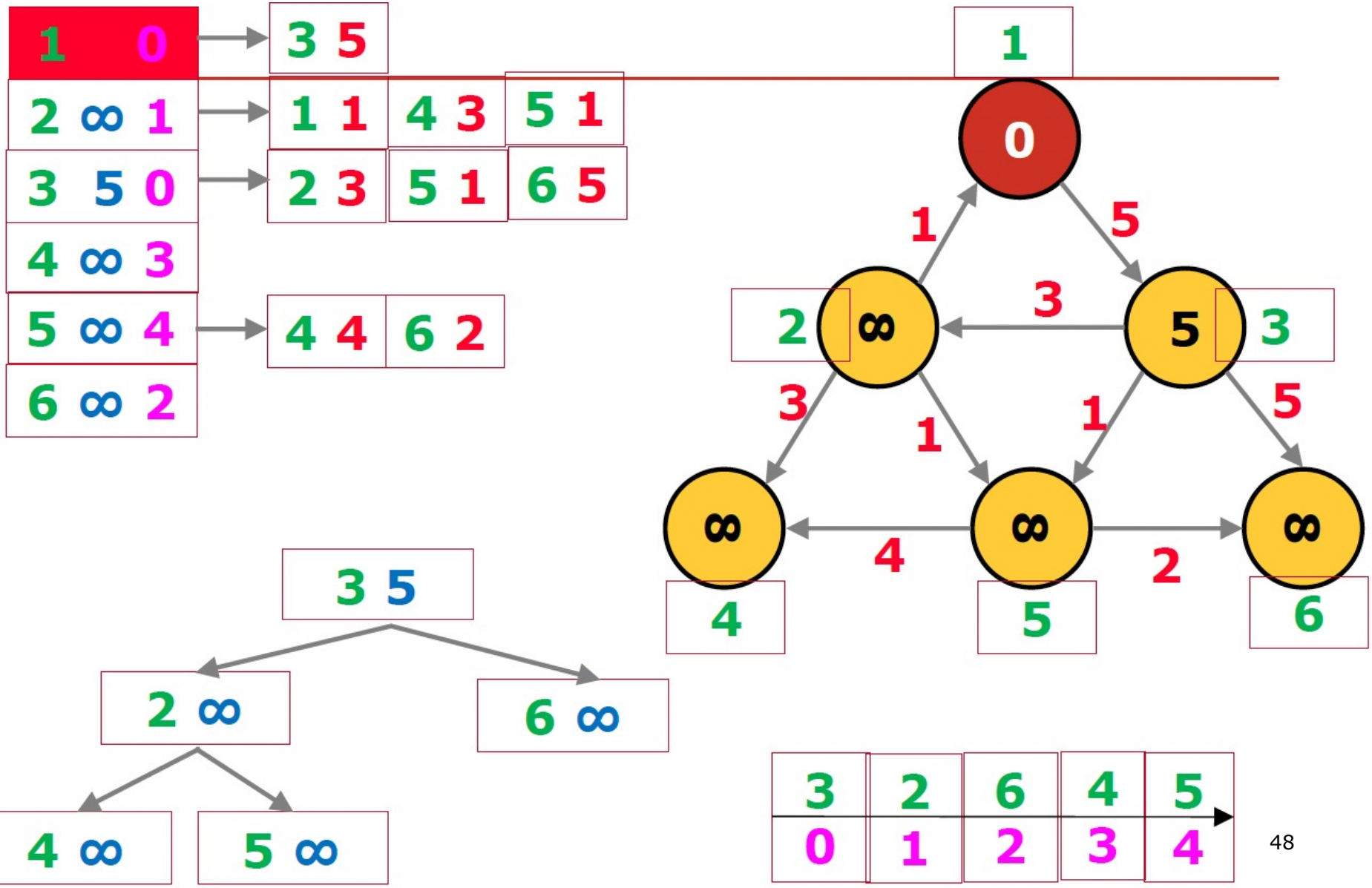
pq.deleteMin()



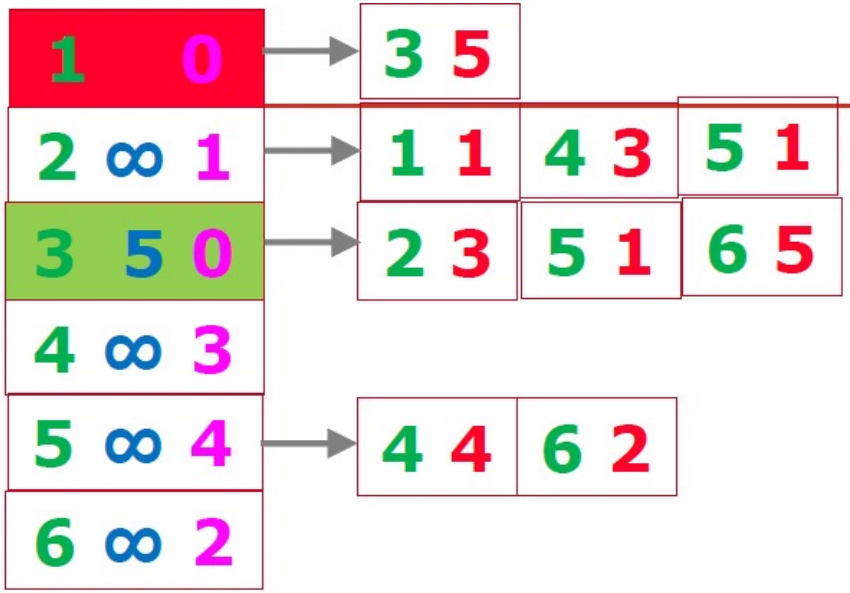
pq.decreaseKey(3,5)

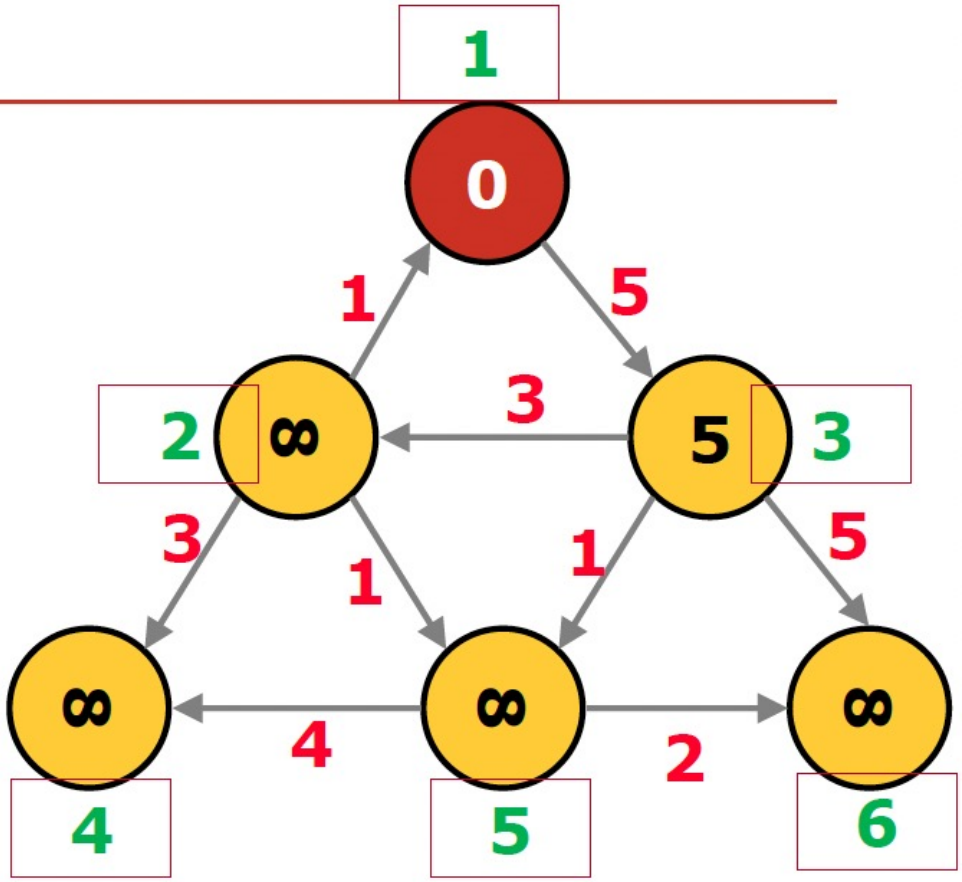


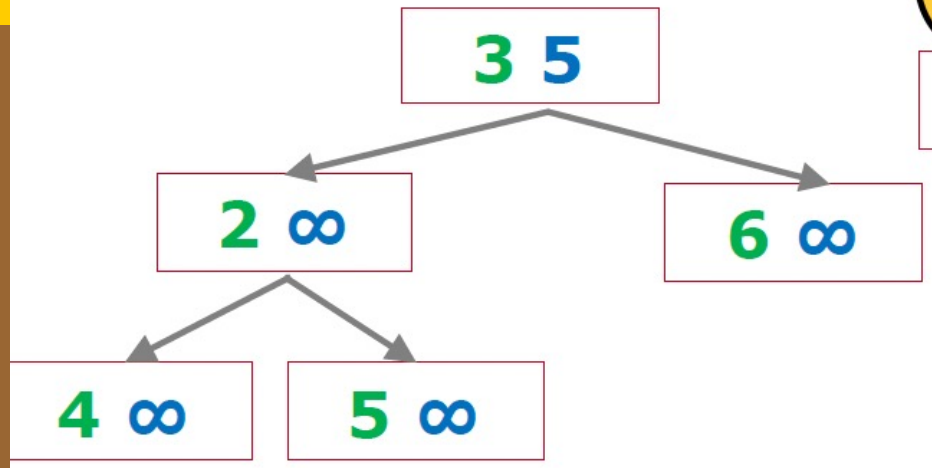
pq.decreaseKey(3,5)



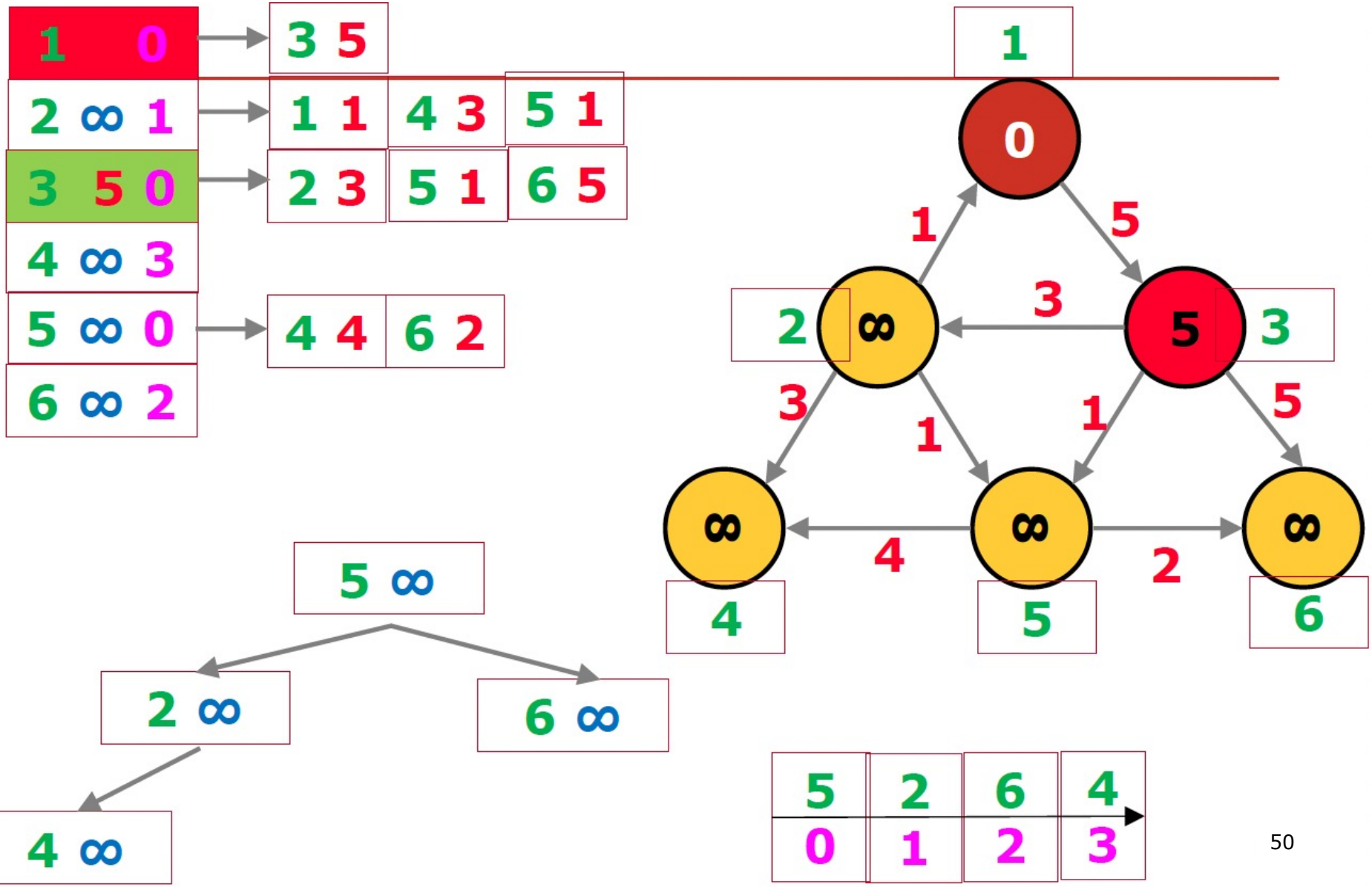
pq.deleteMin()



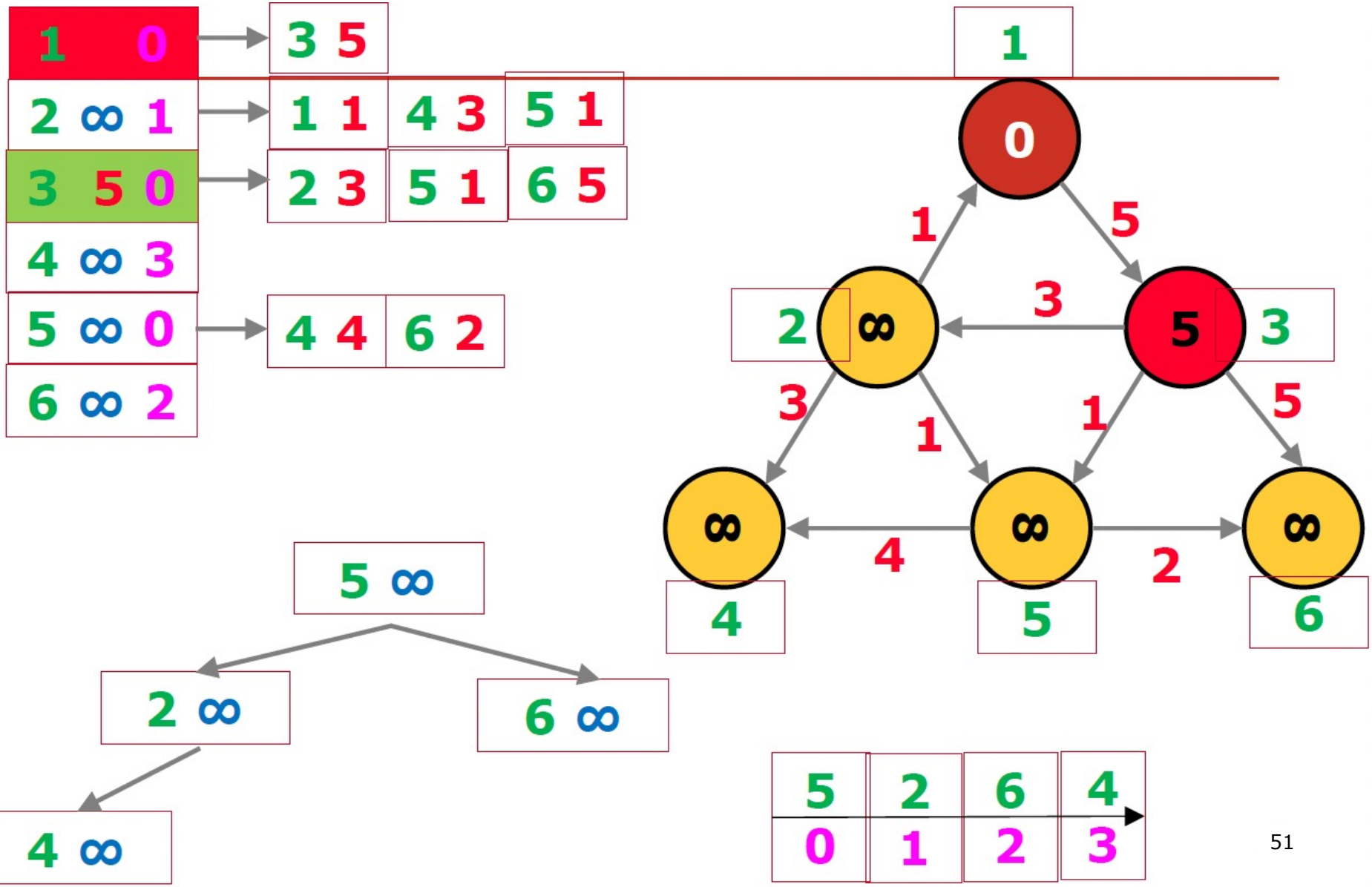




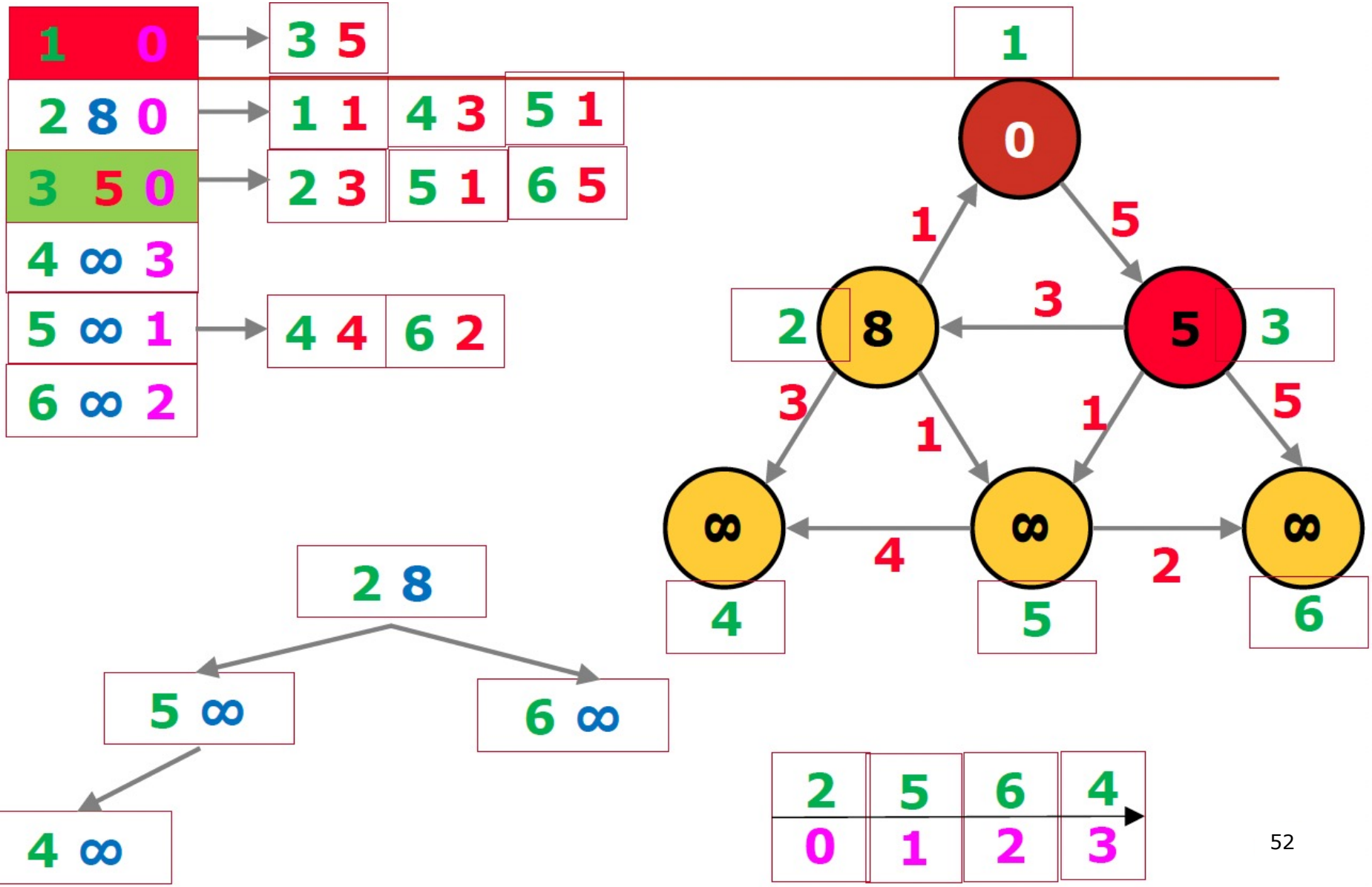
pq.deleteMin()



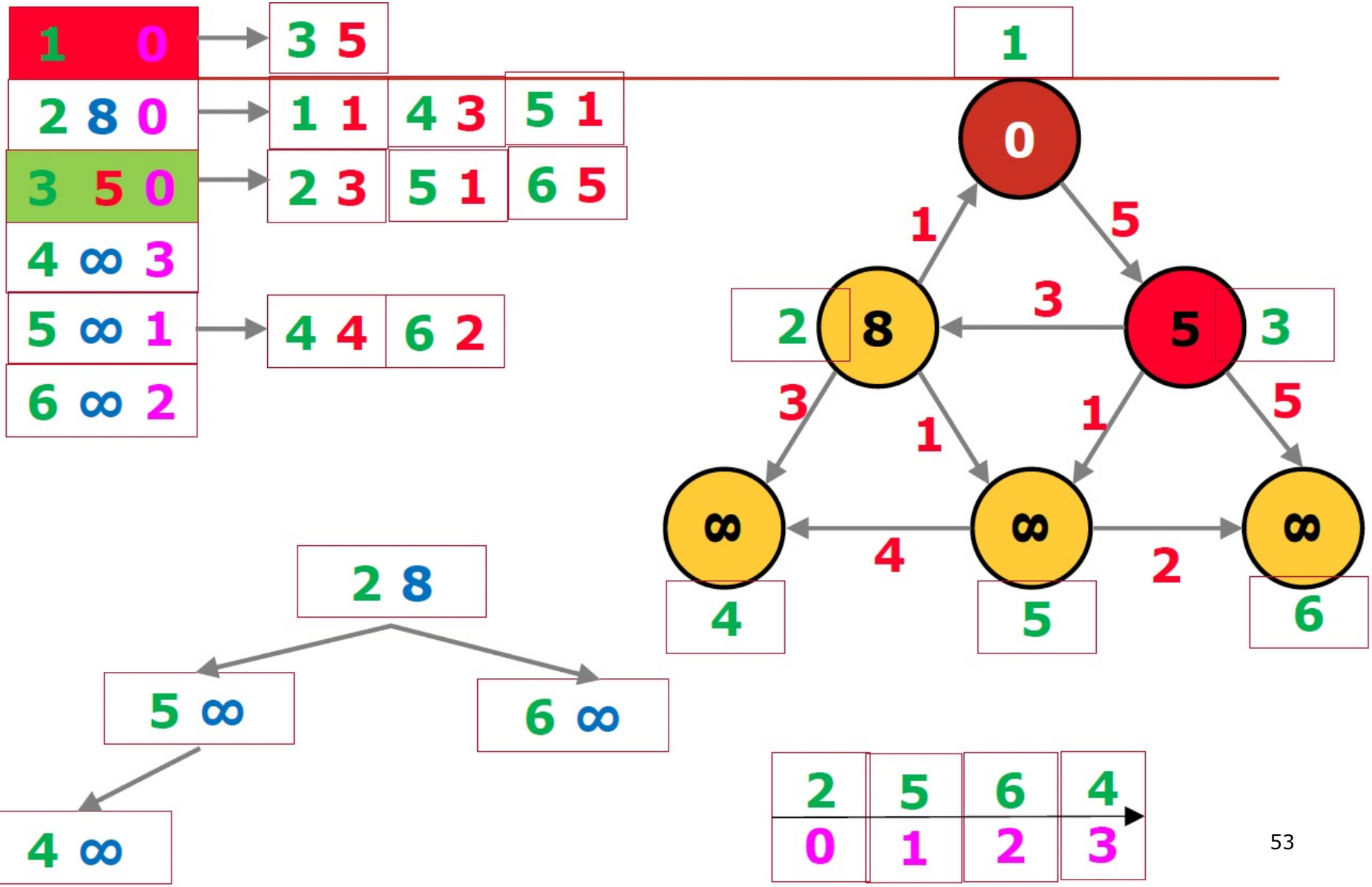
pq.decreaseKey(2,8)



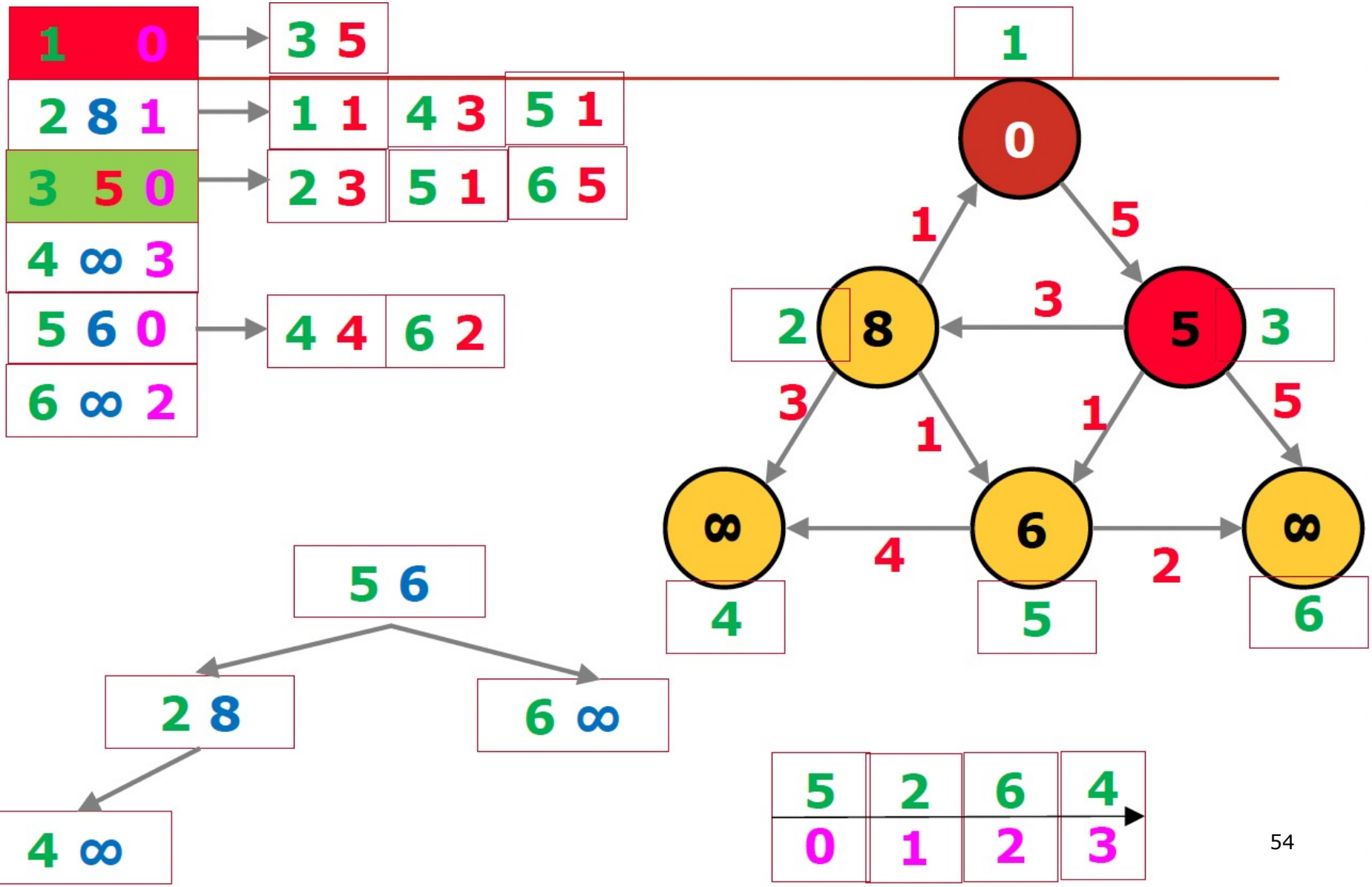
pq.decreaseKey(2,8)



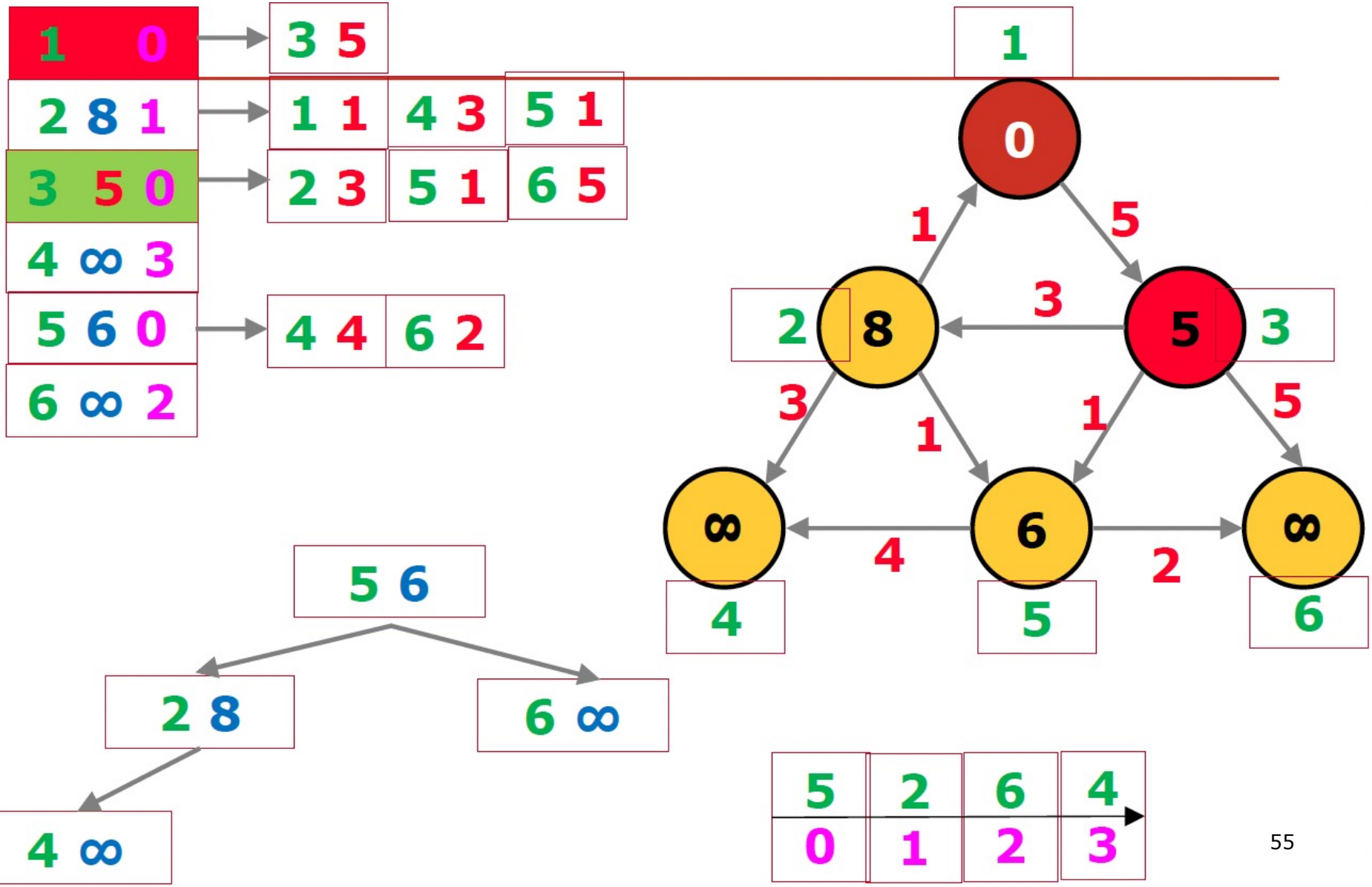
pq.decreaseKey(5,6)



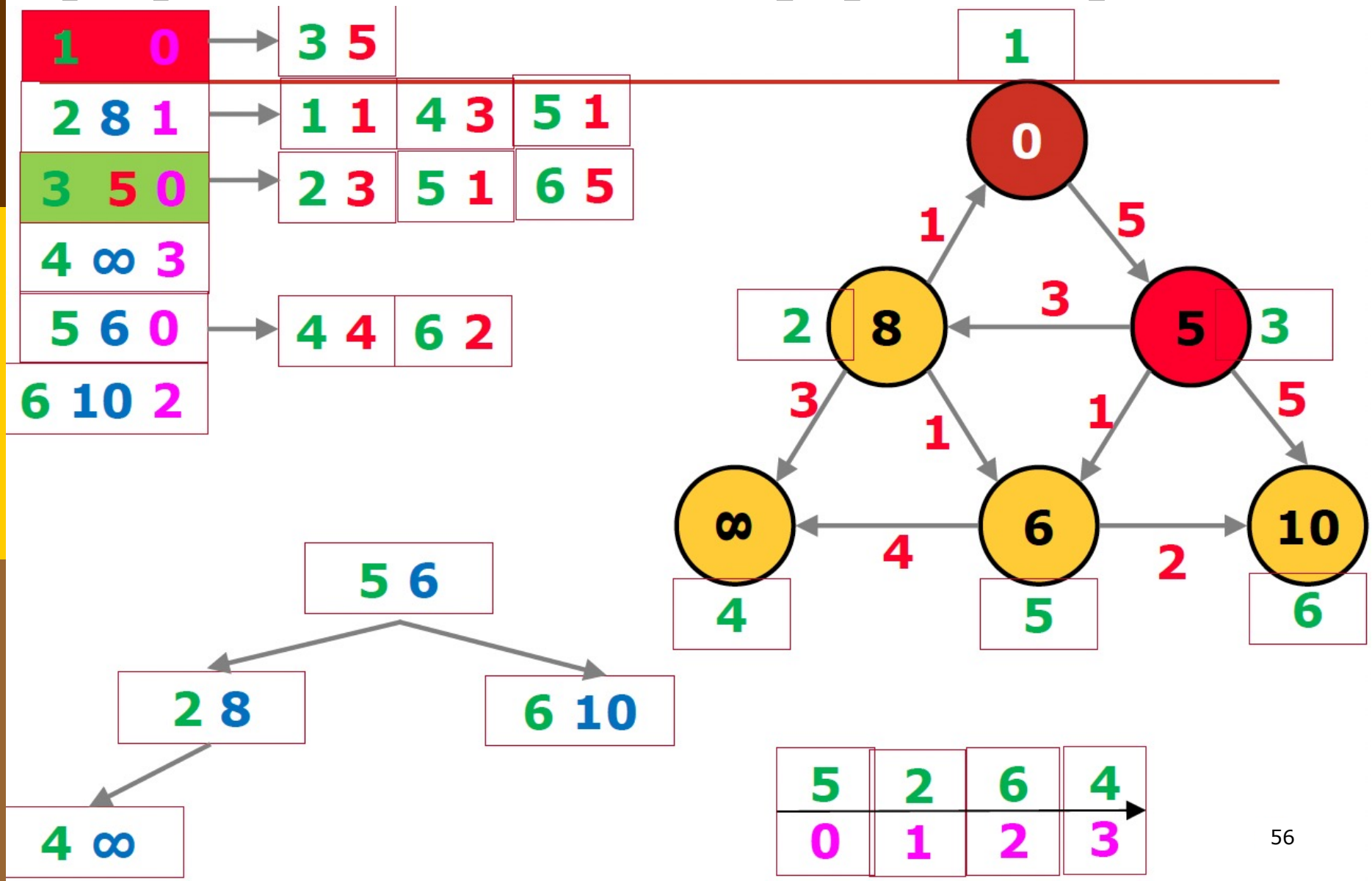
pq.decreaseKey(5,6)



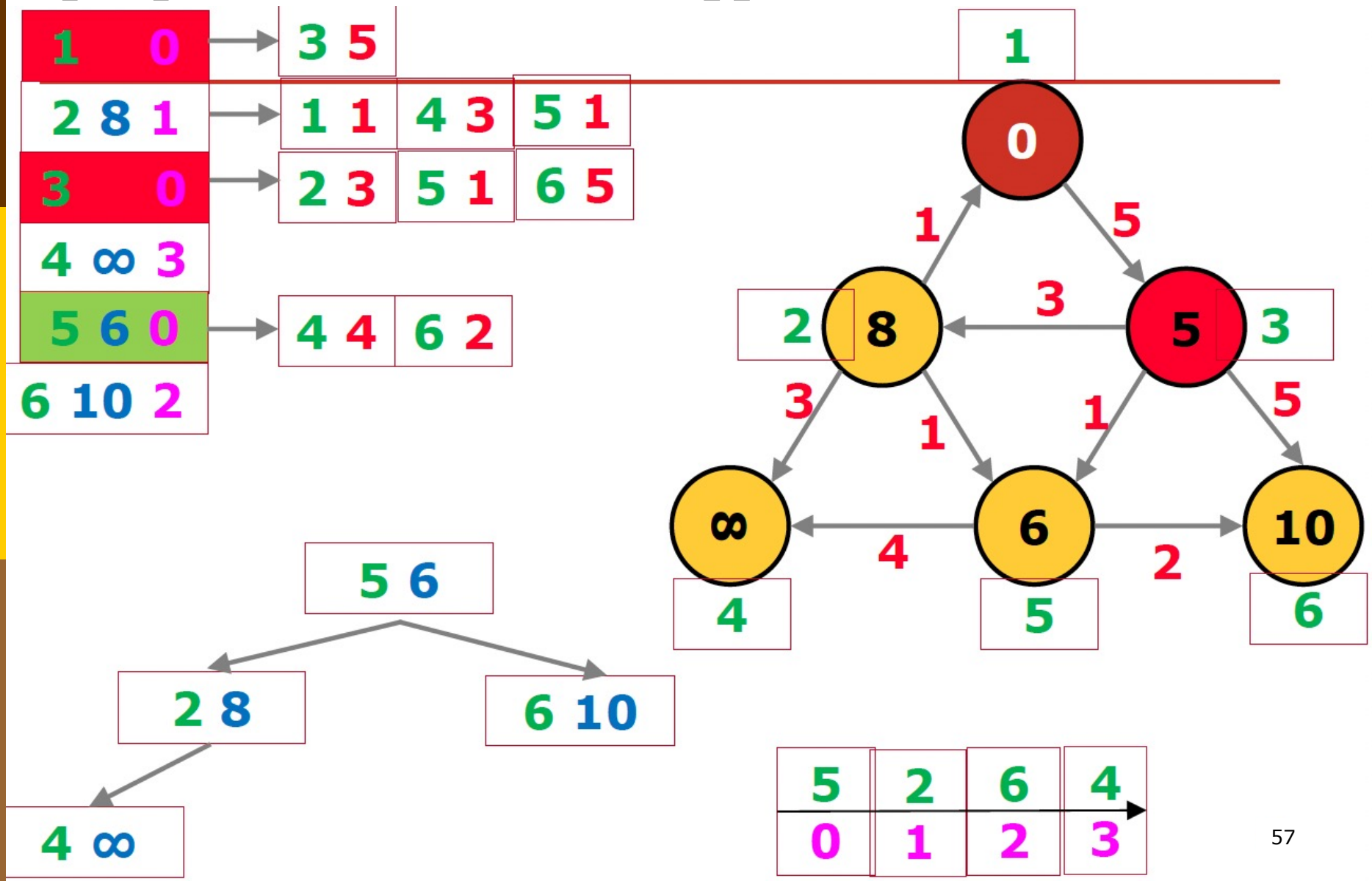
pq.decreaseKey(6,10)



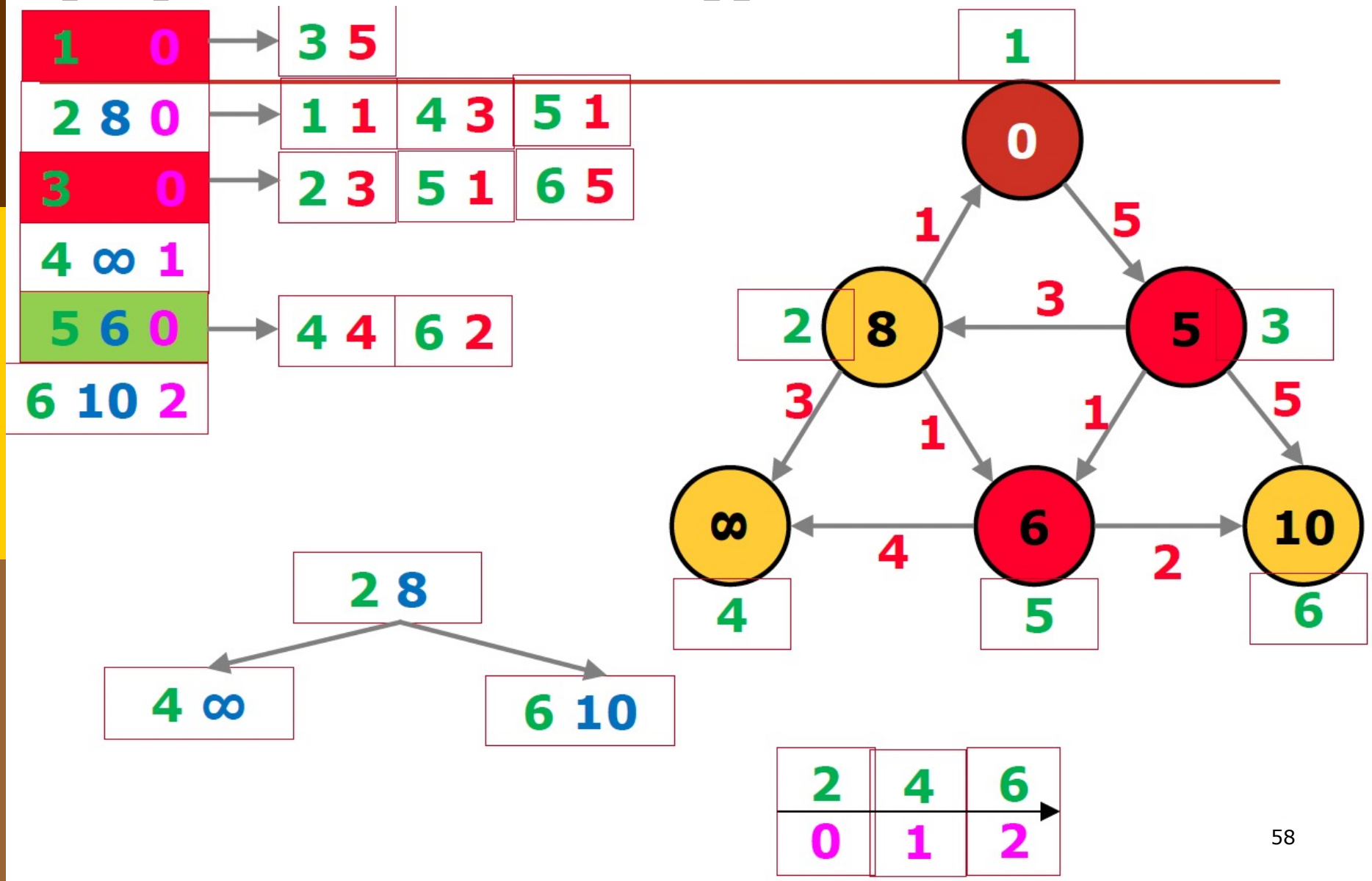
pq.decreaseKey(6,10)



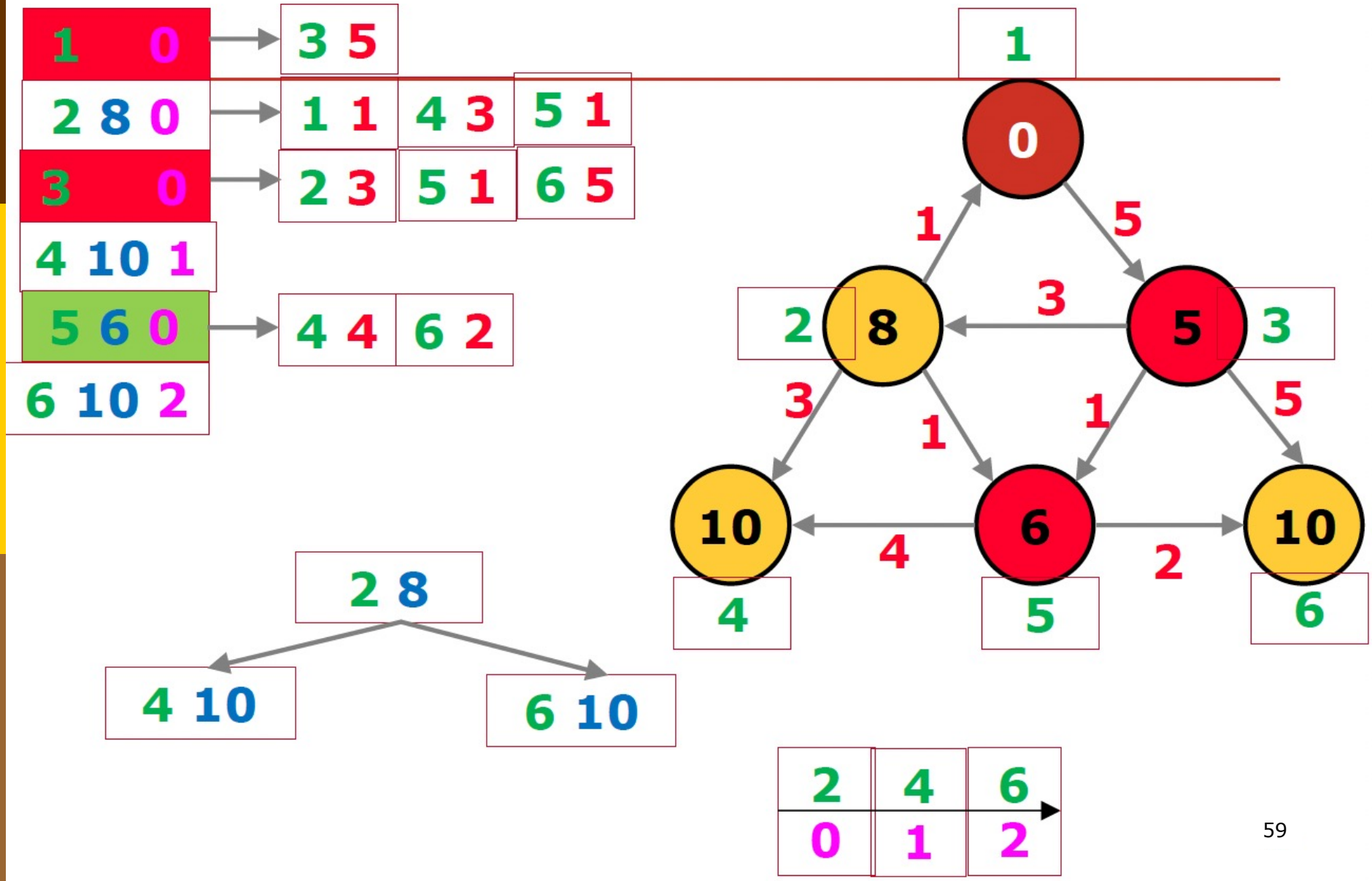
pq.deleteMin()



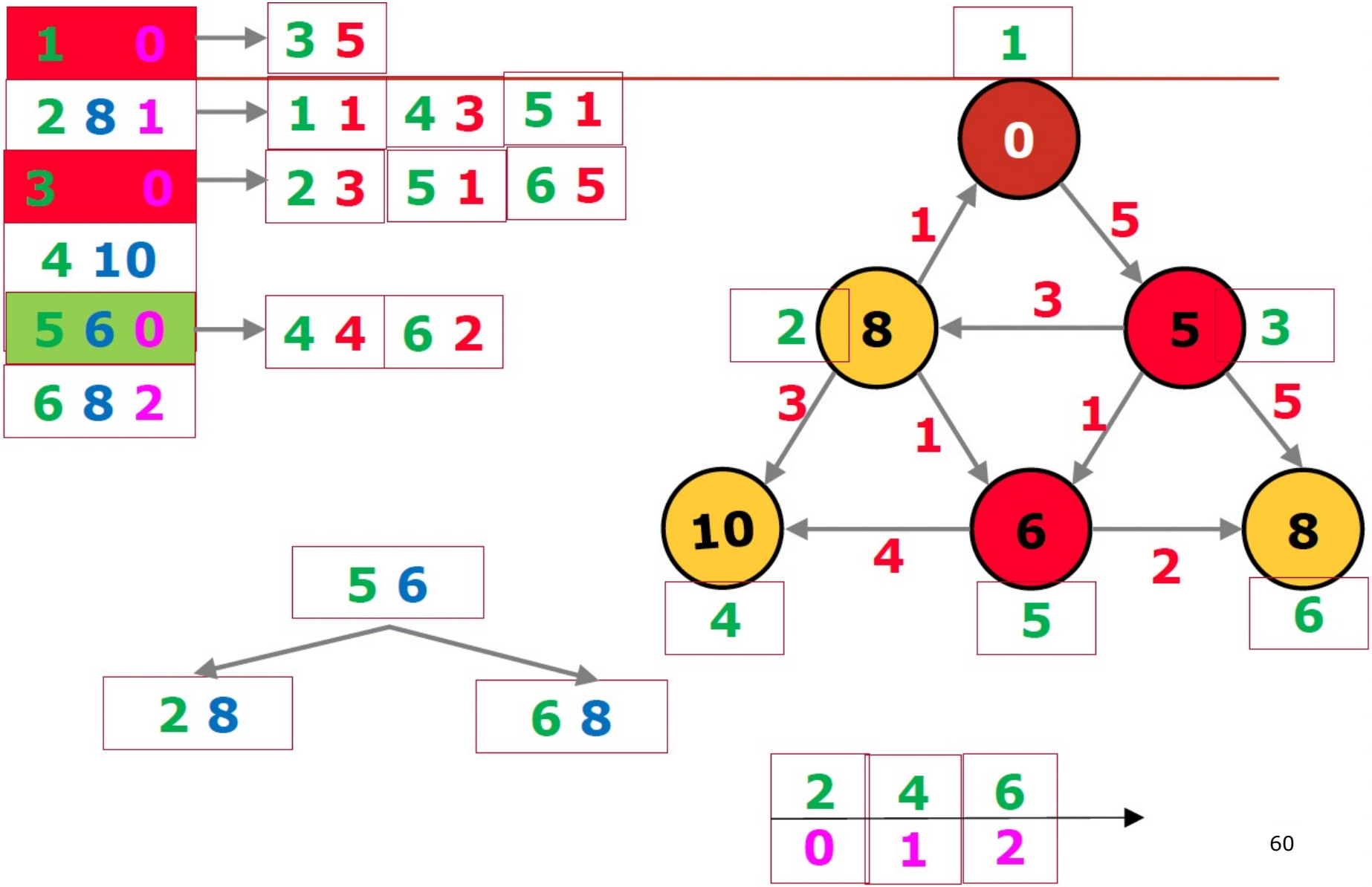
pq.deleteMin()



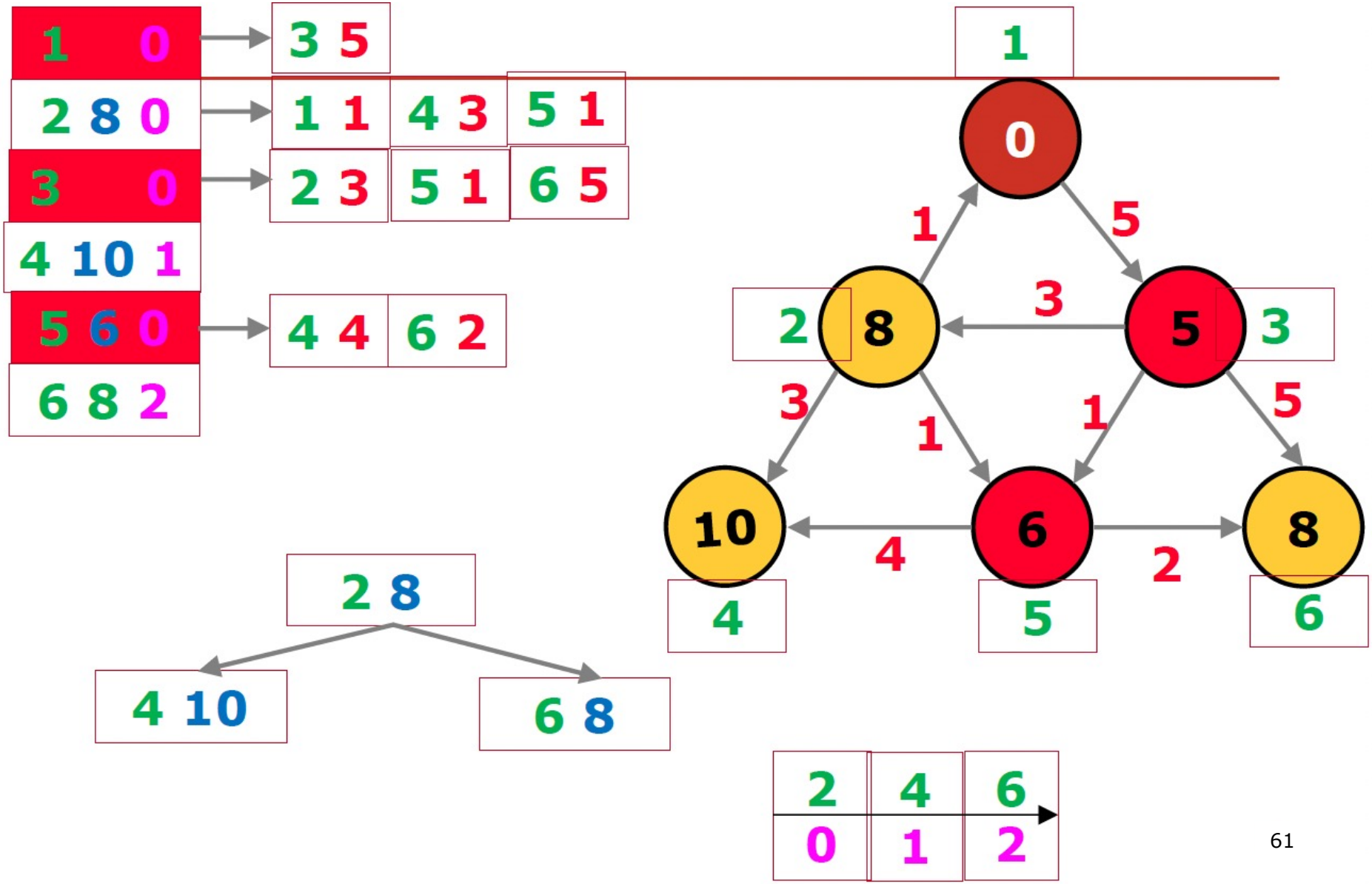
pq.decreaseKey(4,10)



pq.decreaseKey(6,8)



pq.deleteMin()



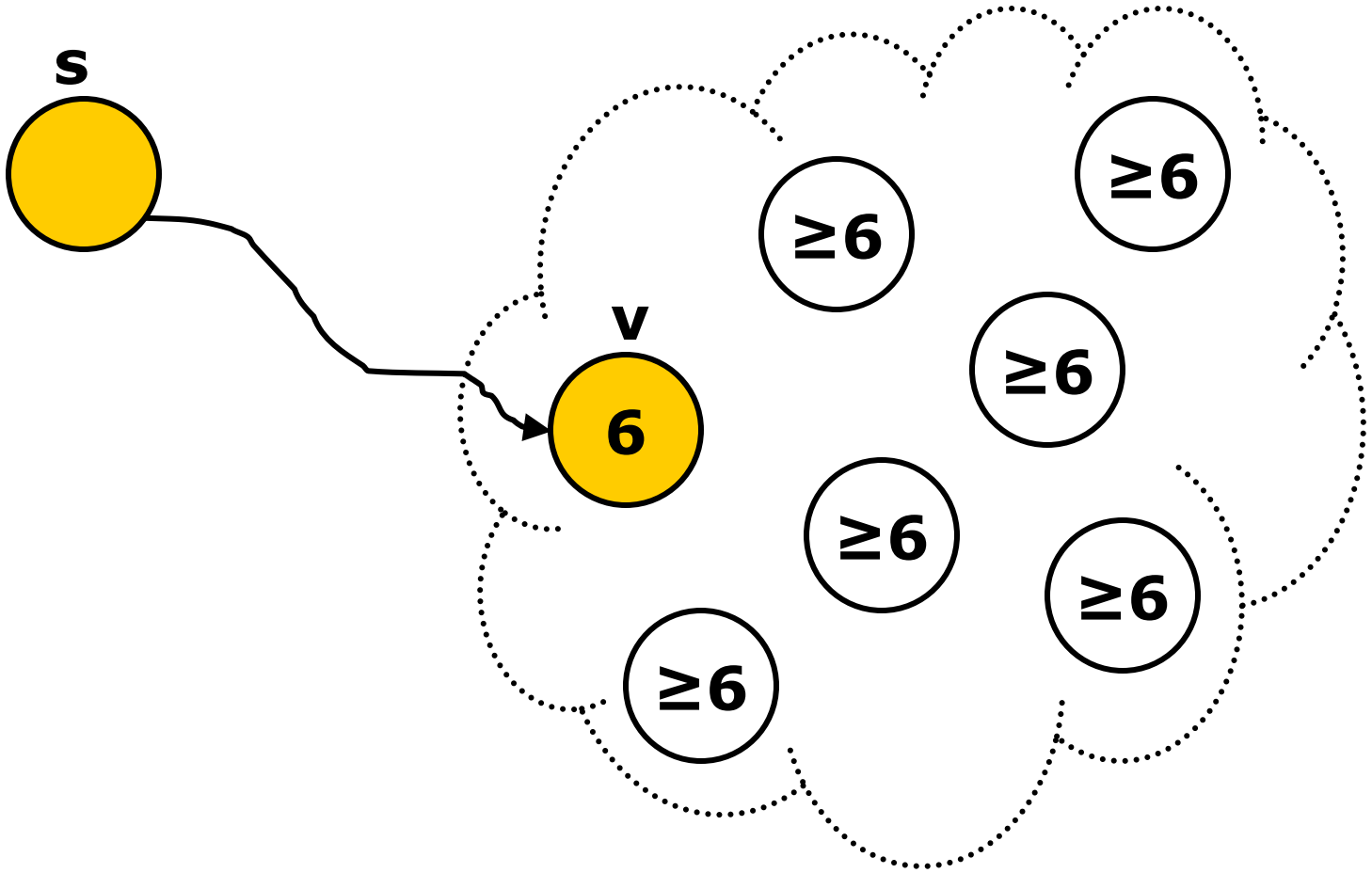
Bellman-Ford

Shortest path algorithm

What if the graph has negative weight?

- Dijkstra's shortest path algorithm will not work.
- **Why**?

Observation 2



The problem

- Given a graph and a source vertex *src* in graph, find shortest paths from *src* to all vertices in the given graph.
- The graph may contain **negative weight edges.**

The problem

- If there is a negative weight cycle, then shortest distances are not calculated, negative weight cycle is reported.
- *Bellman-Ford is also simpler than Dijkstra*
- *But time complexity of Bellman-Ford is O(VE), which is more than Dijkstra.*

Data structure

```
class Edge {
    int src, dest, weight;
    Edge() {
        src = dest = weight = 0;
    }
}

int V, E;
Edge [] edge;
```

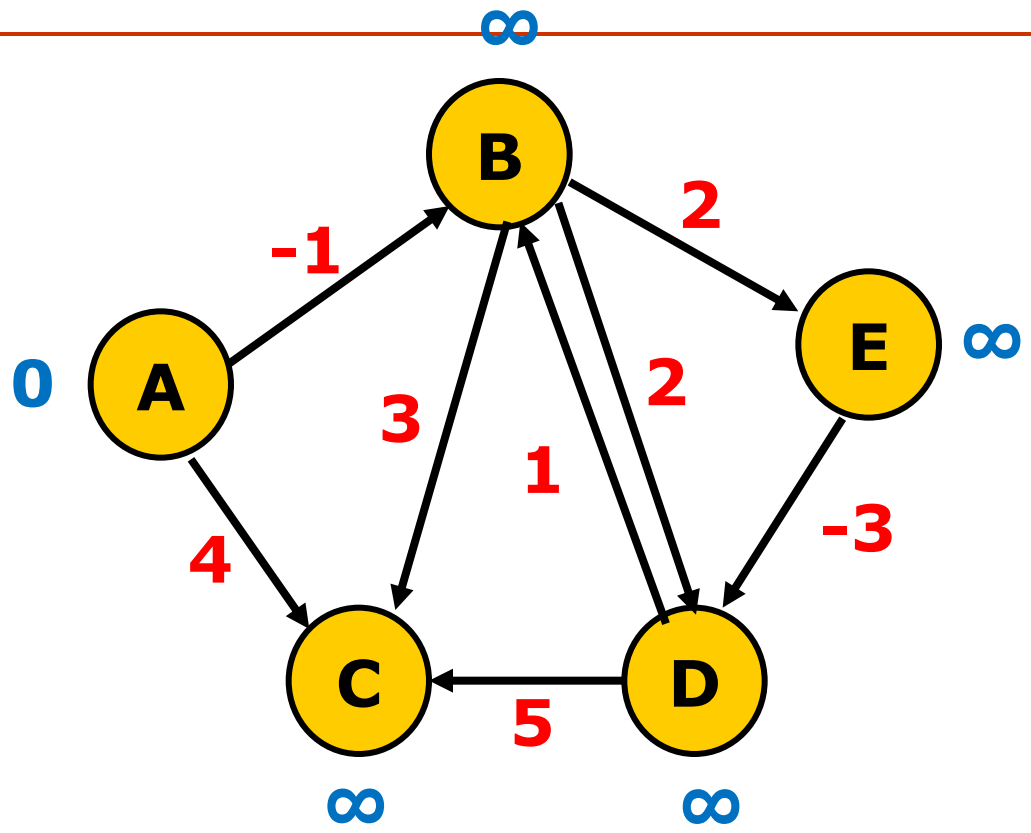
The algorithm

- Initialization
  - Initializes distances from the source to all vertices as infinite and distance to the source itself as 0.
  - Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.

  - int [] dist = new int[V];

```
for (int i = 0; i < V; ++i) {
    dist[i] = Integer.MAX_VALUE;
dist[src] = 0;
```

- Example, for a graph with 5 vertics.



Vertex	A	B	C	D	E
dist	0	∞	∞	∞	∞

The algorithm

- calculates shortest distances.

  Do following |V|-1 times where |V| is the number of vertices in the graph.

  Do following for each edge u-v

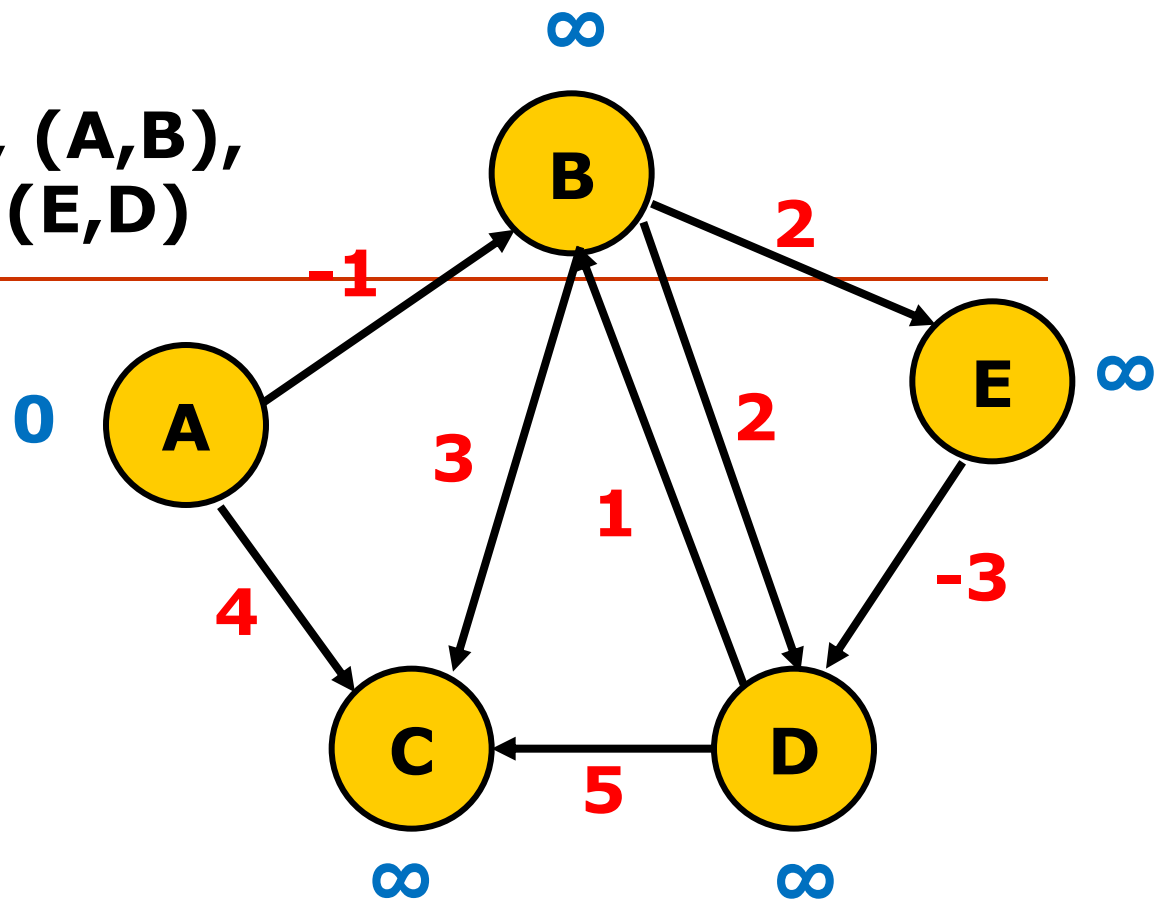
  If dist[v] > dist[u] + weight of edge uv,

  then update dist[v]

  dist[v] = dist[u] + weight of edge uv

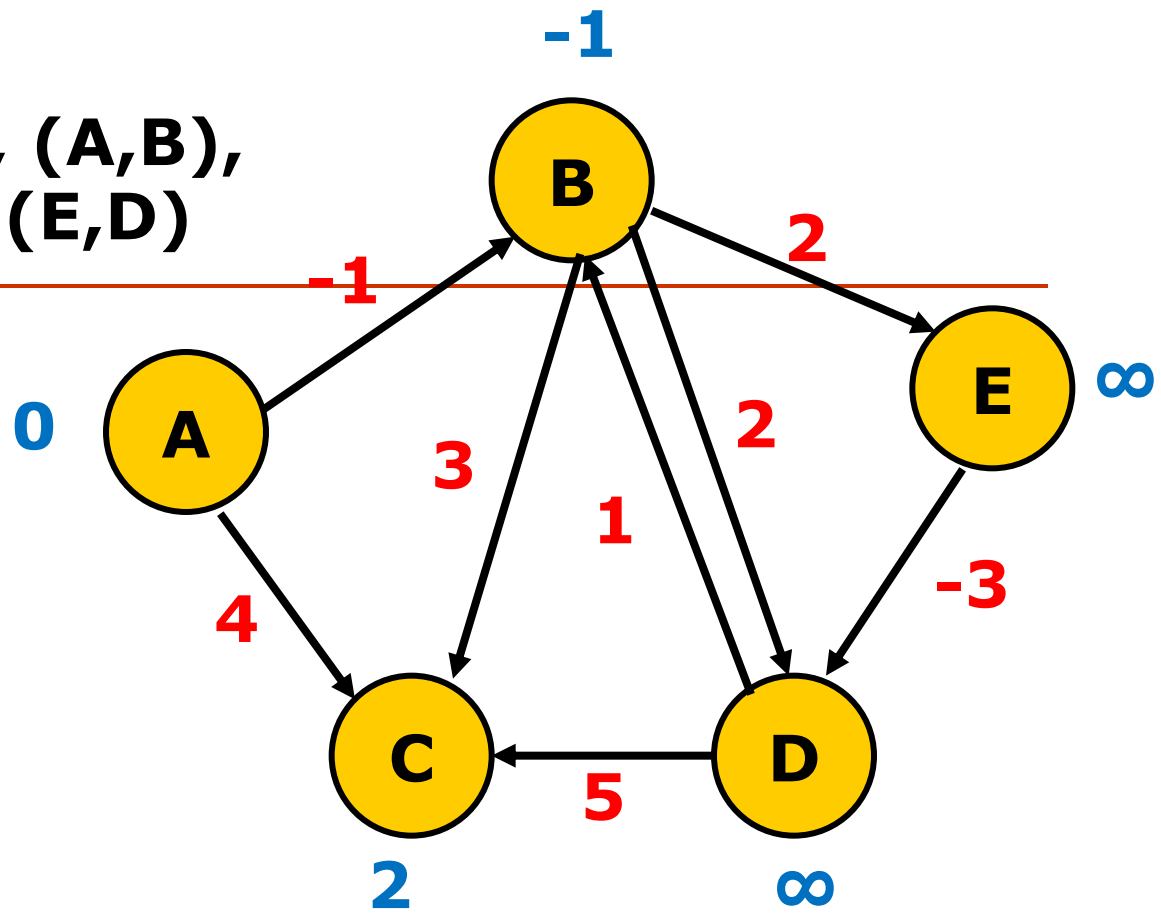
- Note that the sequence in which the edges are processed are not important

Process sequence:
(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C),(E,D)



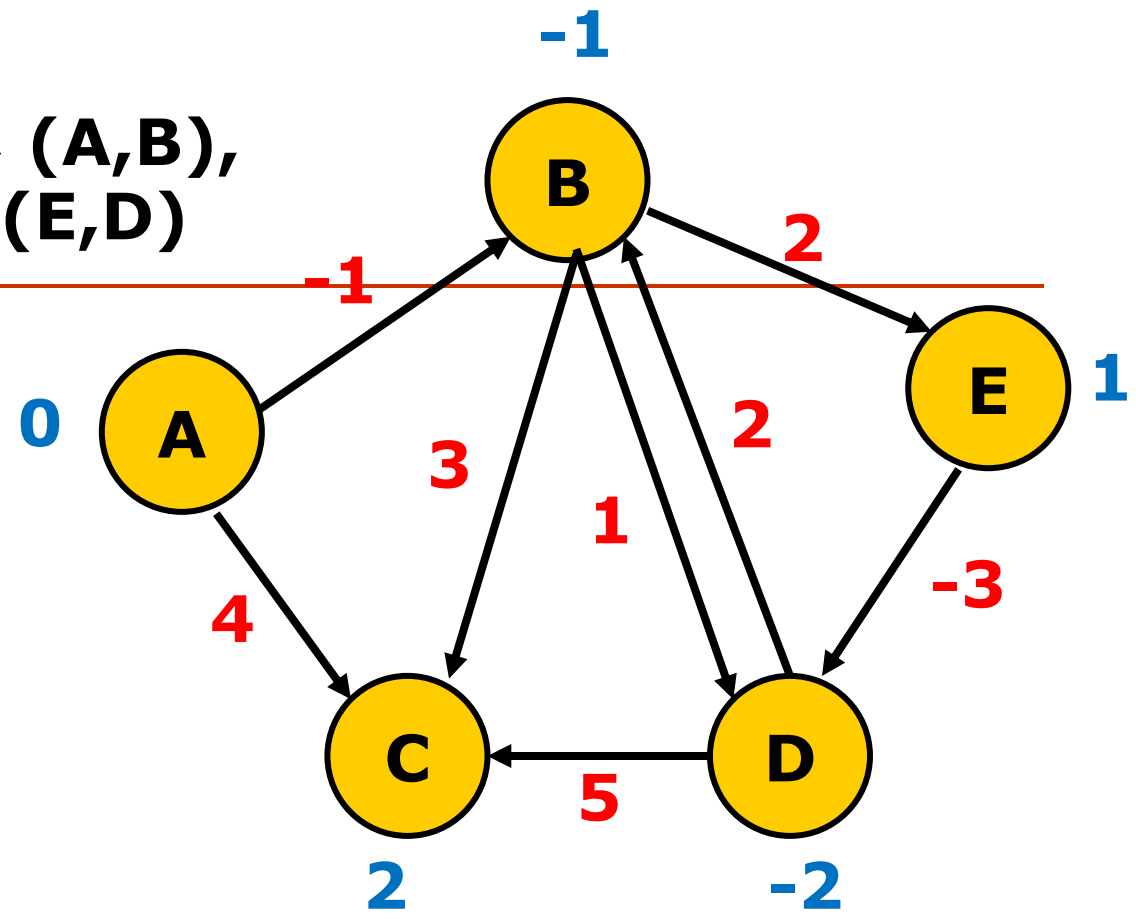
Vertex	A	B	C	D	E
dist	0	∞	∞	∞	∞

Process sequence:
(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C),(E,D)



Vertex	A	B	C	D	E
dist	0	-1	2	∞	∞

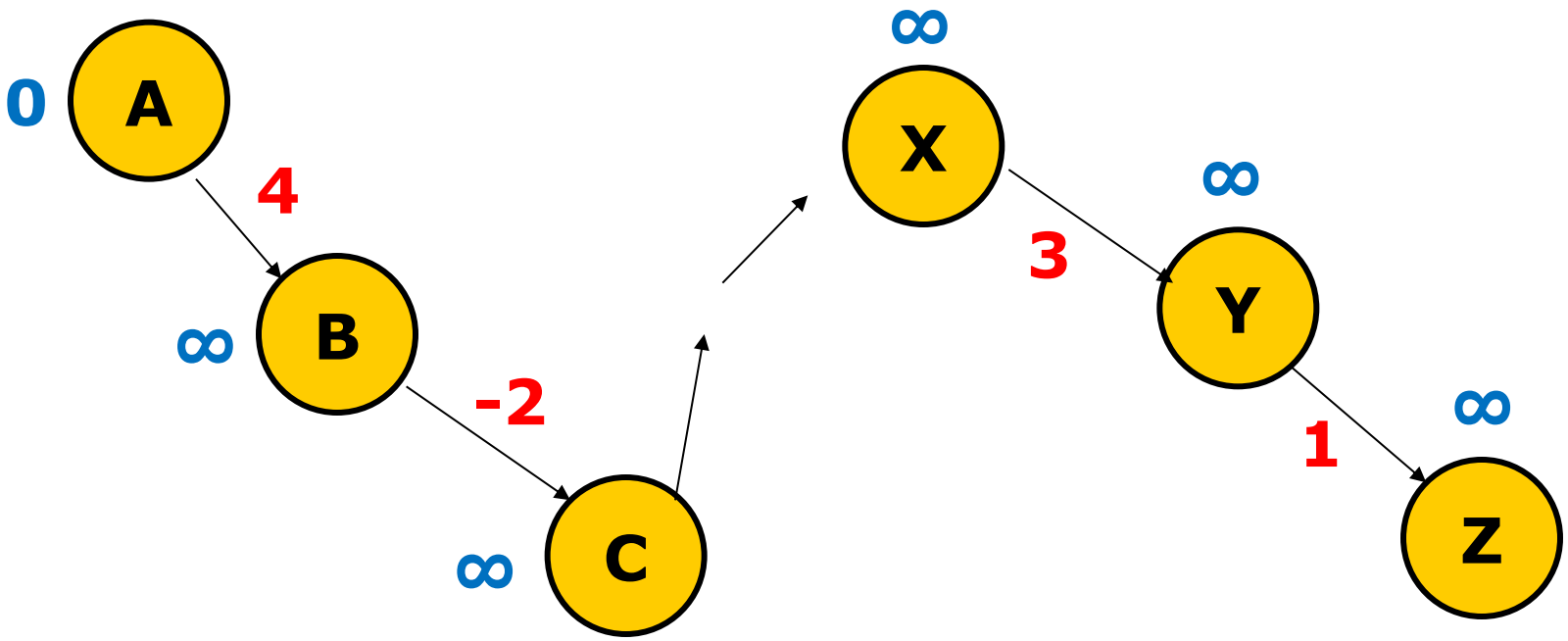
Process sequence:
(B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C),(E,D)



Vertex	A	B	C	D	E
dist	0	-1	2	∞	∞
	0	-1	2	1	1
	0	-1	2	-2	1

Why need to do |V-1| times?

- As the sequence that you process the edges are random, you might end up only updating one node in each pass.



The algorithm

- Report if there is a negative weight cycle
  - Do following for each edge u-v
    ……If dist[v] > dist[u] + weight of edge uv, then "Graph contains negative weight cycle"

- The idea is, the previous step guarantees the shortest distances if the graph doesn't contain a negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

Complexity

- Initialisation O(V)
  - int [] dist = new int[V];
    for (int i = 0; i < V; ++i) {
        dist[i] = Integer.MAX_VALUE;
    dist[src] = 0;
- calculates shortest distances. O(VE)
  - |V-1| * |E|