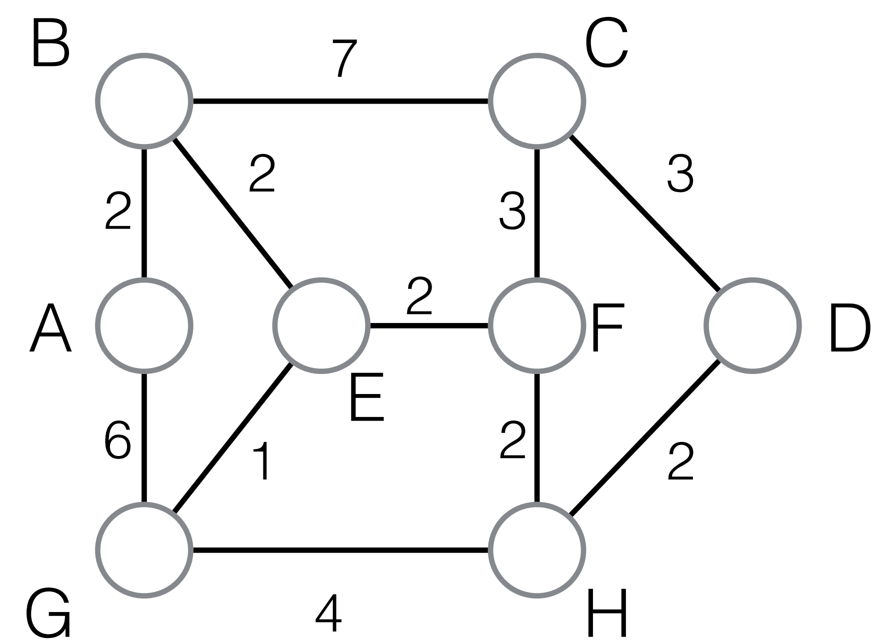
Finding the Shortest Path
Dijkstra's Algorithm

A.Gerdelan <gerdela@scss.tcd.ie>

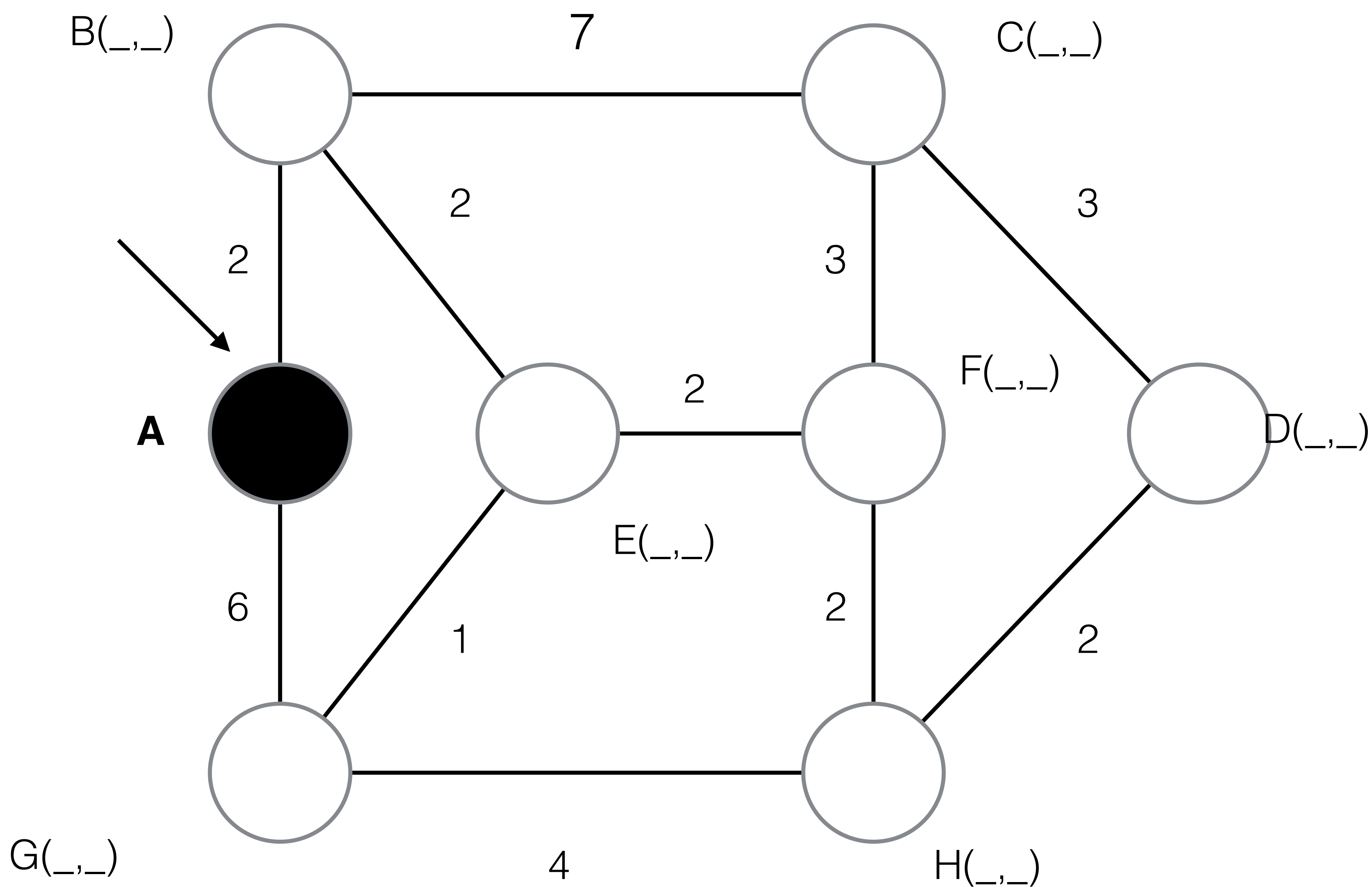
Dijkstra's Algorithm

- ~1959 - Edsger Dijkstra
  - a founder of computer science as an academic field
  - many contributions
- find the **shortest path** through a **weighted** network (**graph**)
- fastest known graph search (asymptotically)
  - original - $\mathbf{O(V^2)}$ - V is vertices
  - 1984 - using a Fibonacci Heap - $\mathbf{O(E + V \log V)}$ - E is edges
- each node updated *shortest known distance* info
- has some similarities to breadth-first search

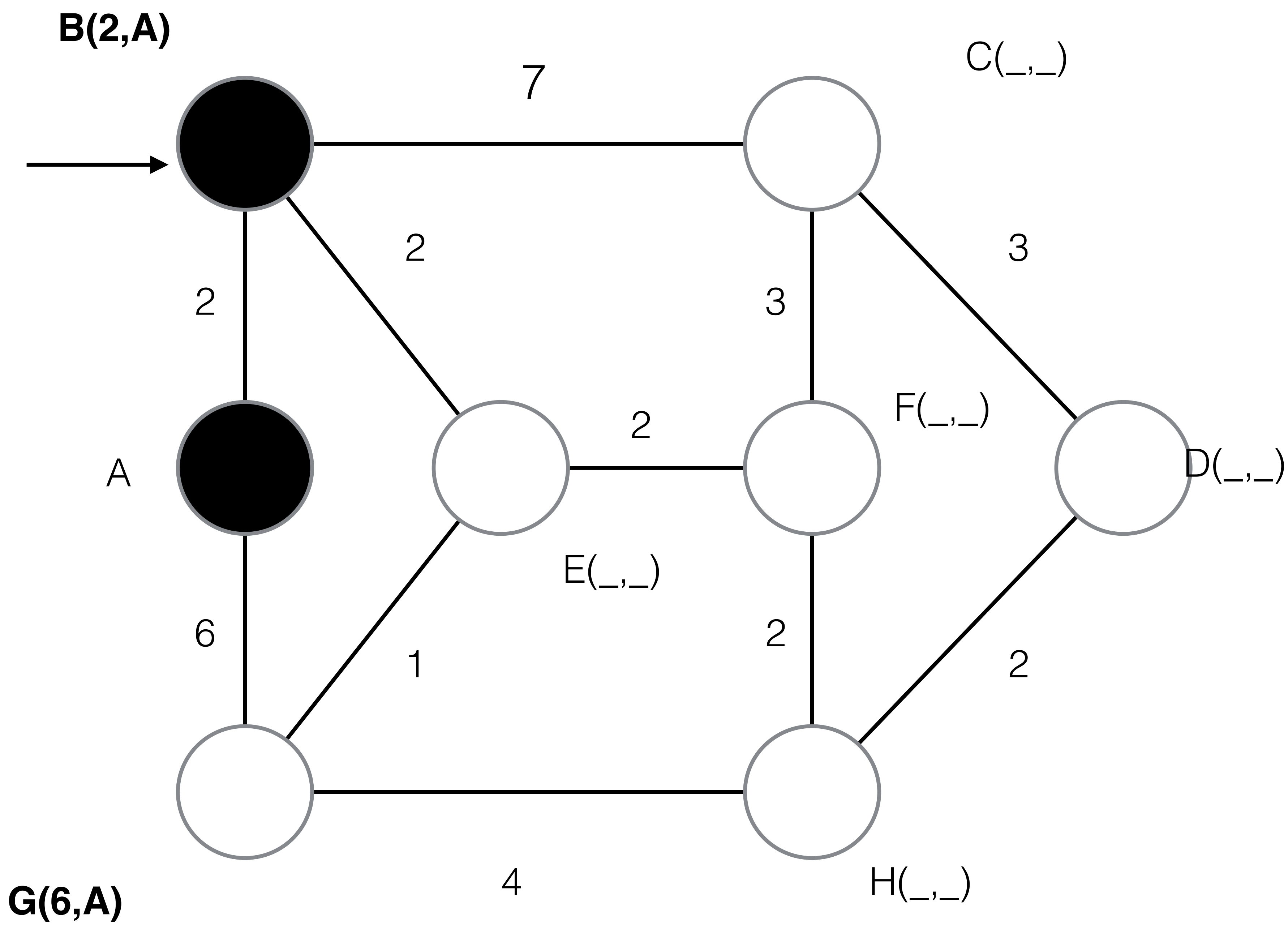
The Algorithm



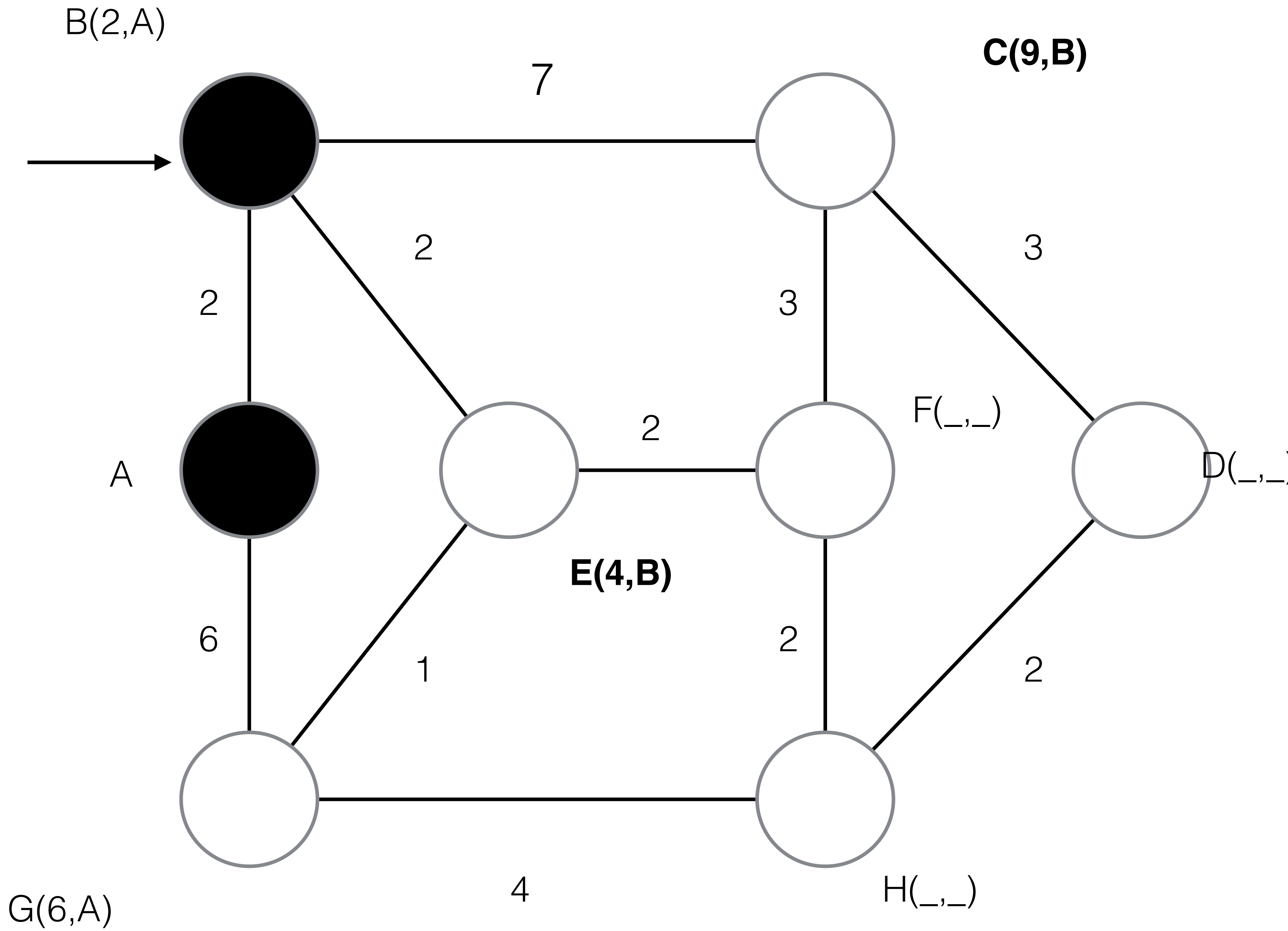
1. label each vertex
    - **tentative** (white circles)
    - distance infinity
2. mark start vertex as
    - **permanent** (black circles)
    - make it the **current vertex** (arrow)
3. calculate distance from current vertex to each of its tentative, adjacent neighbours
4. from all tentative vertices - mark the one with shortest distance as
    - permanent
    - make it the current vertex
5. repeat from 3 until all vertices are marked permanent



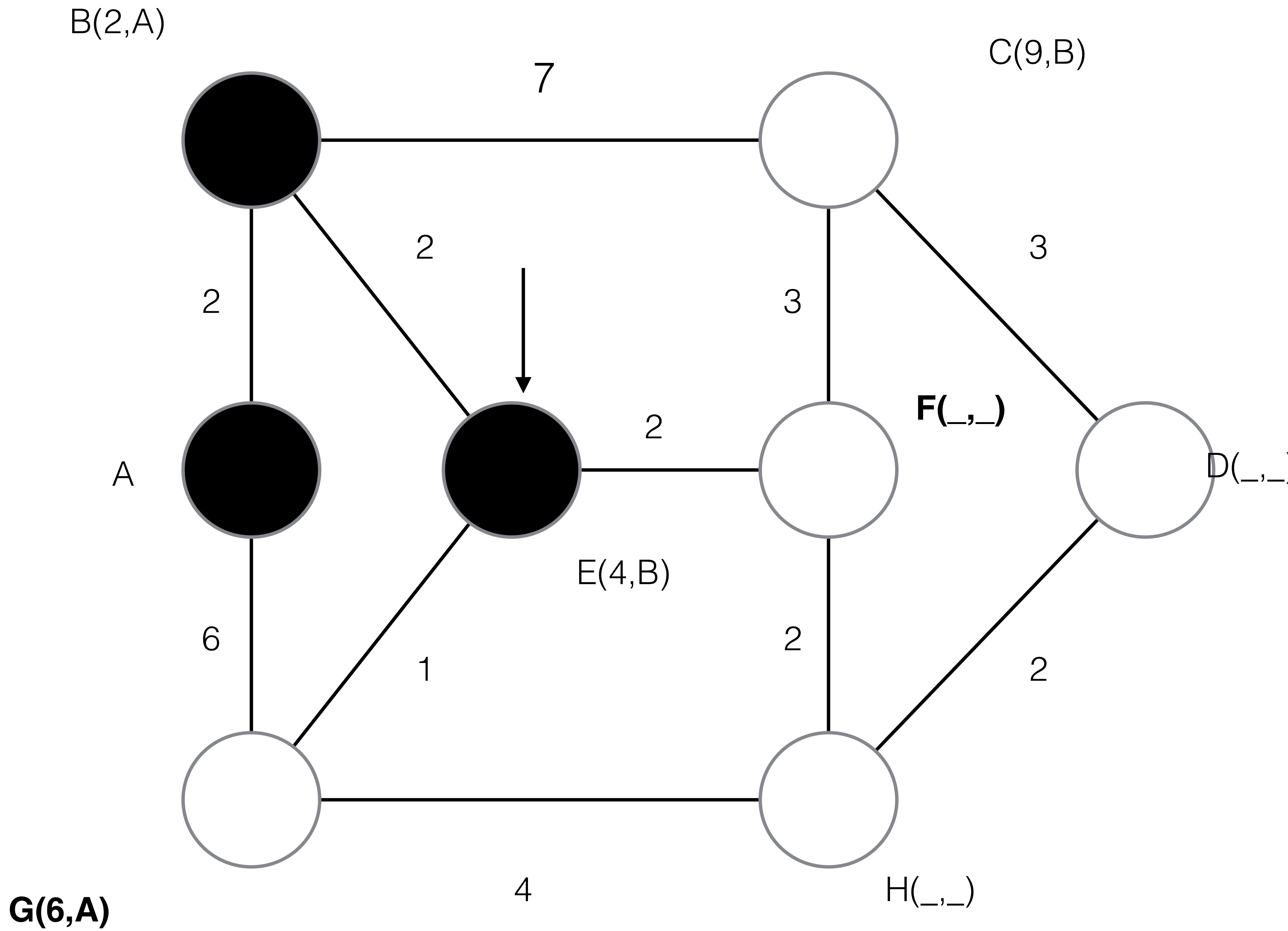
1) start at source node A
flag as *permanent*



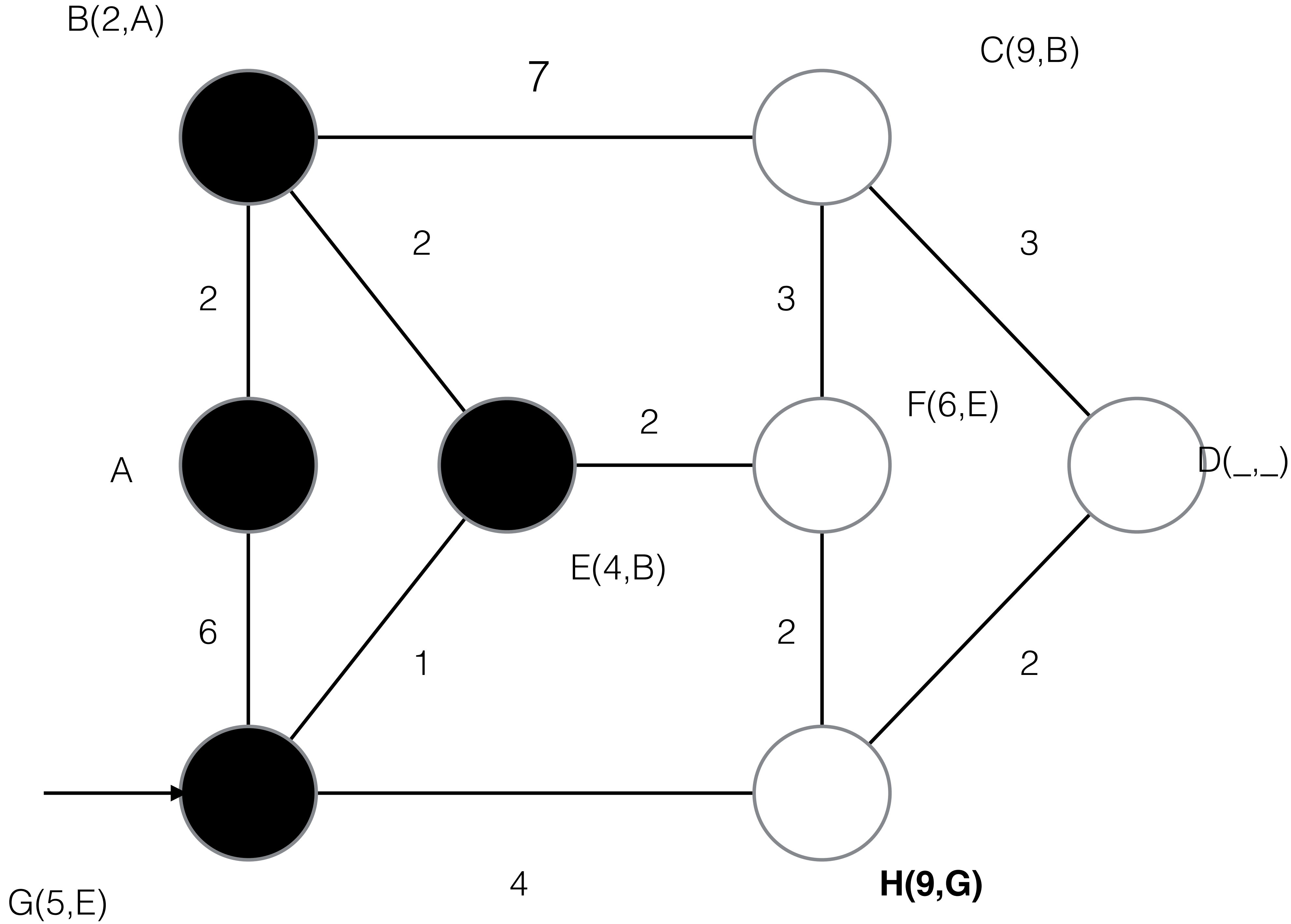
2) re-label adjacent nodes to A with shortest path (from A)
make shortest (B) permanent, current.



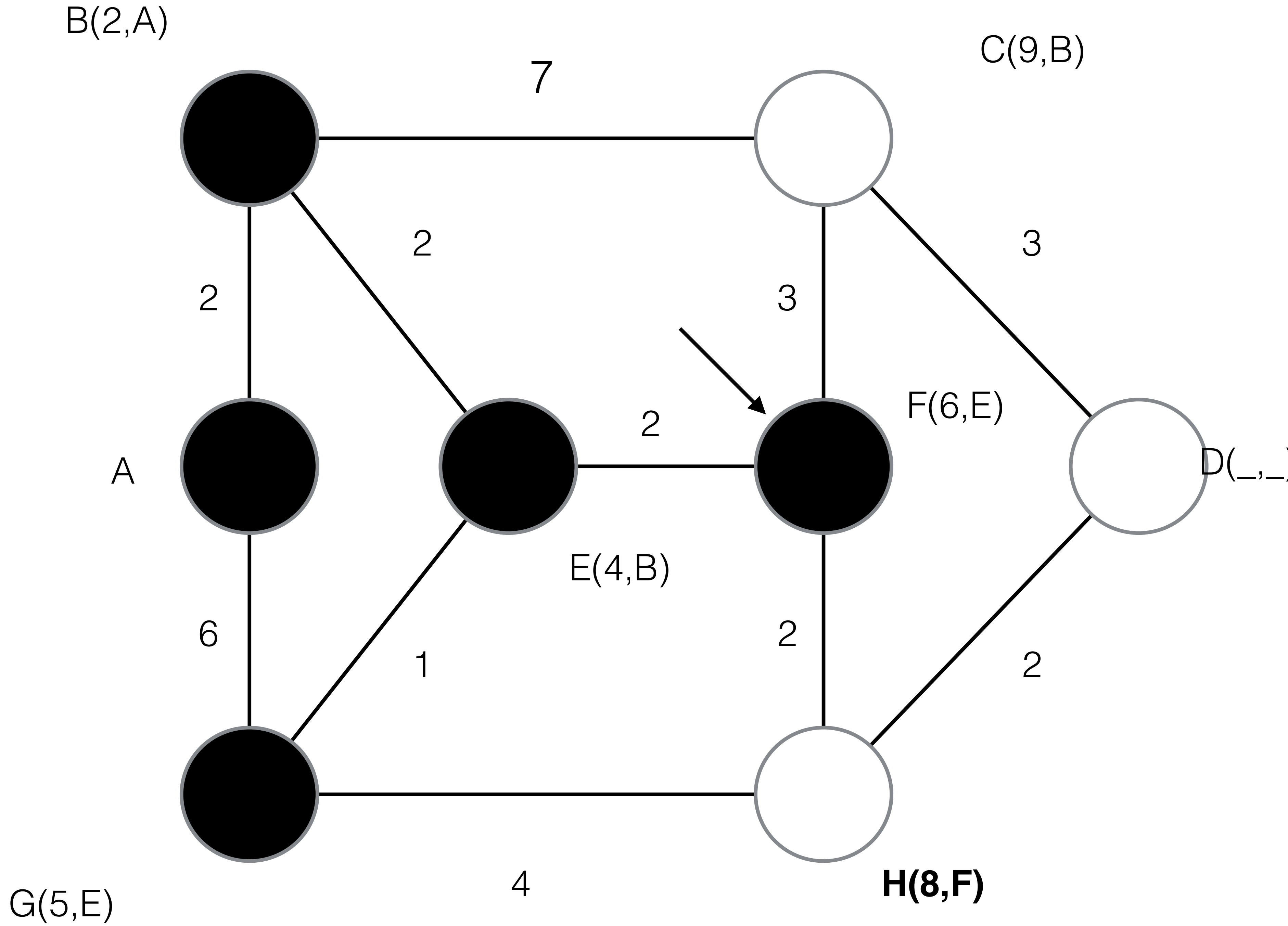
3) re-label adjacent nodes to B with shortest path (add own SP)
don't label A - it's already permanent



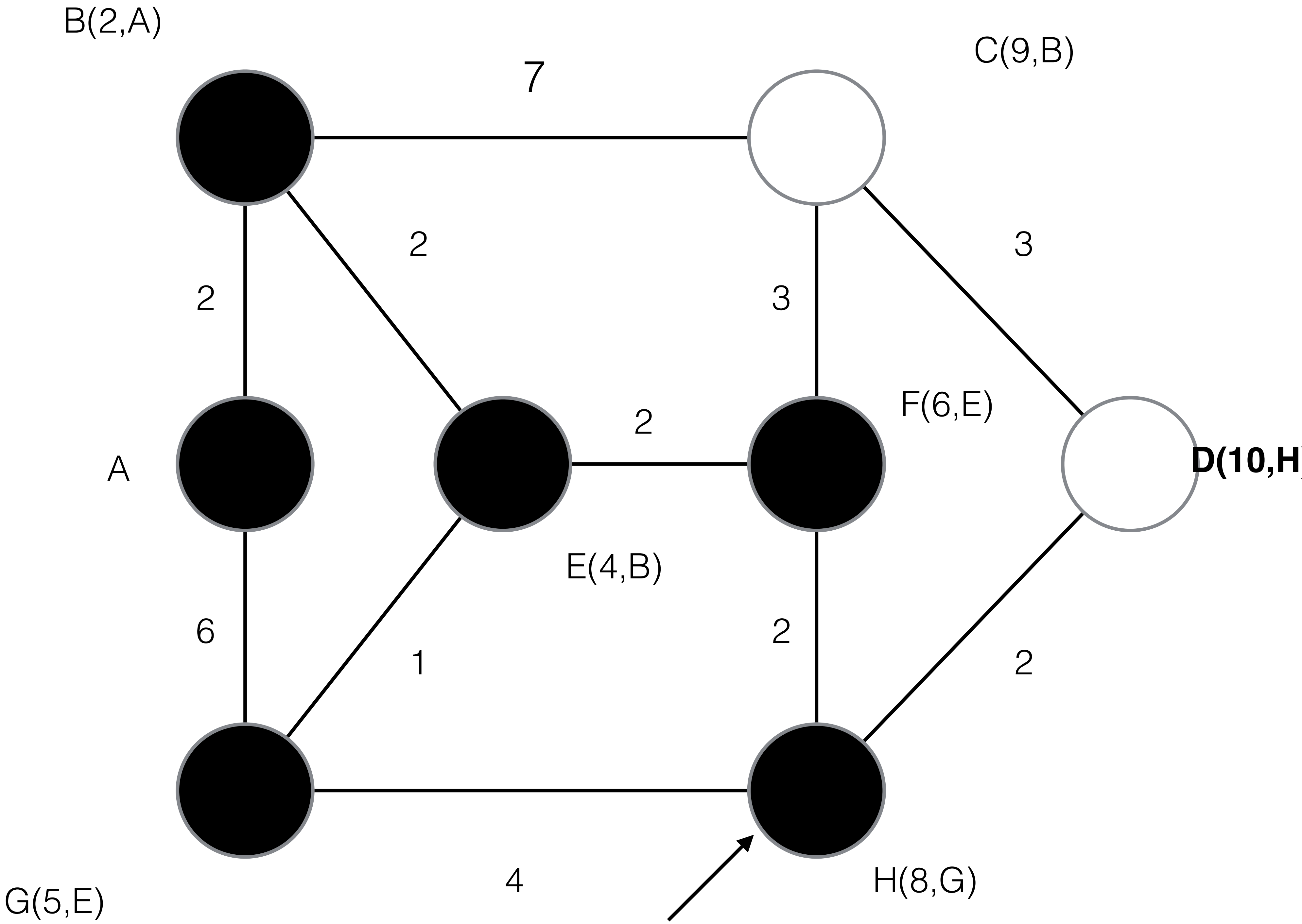
4) re-label G to … ? and F to … ?
new current is?



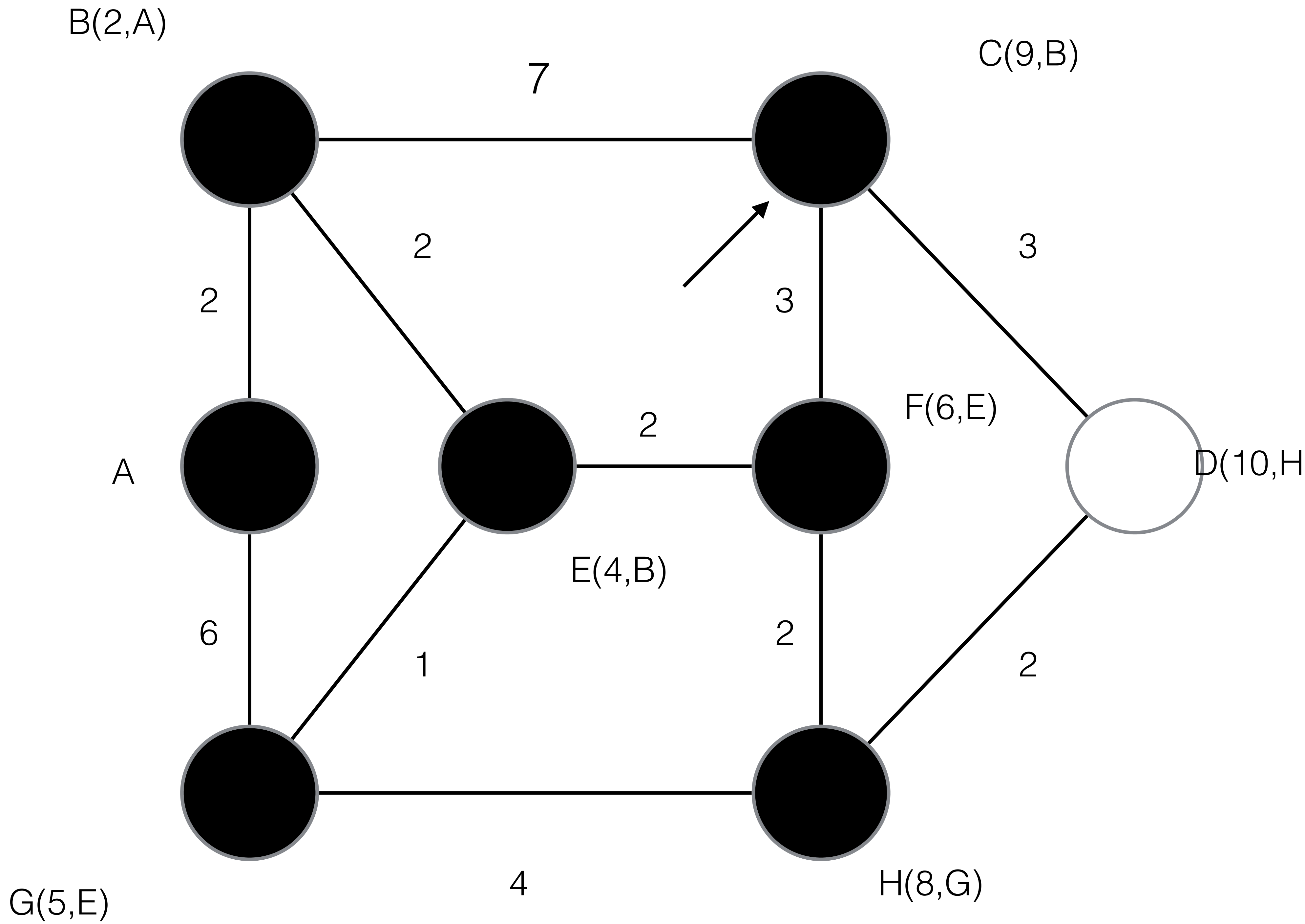
5) re-label H. what is the new current?
we have {9, 6, 9, inf} to choose from



6) we need to recalculate H
H is the new current



7) new node is … ?



8) mark last node as permanent
done!
we now have the shortest path <u>to any node from A</u>

Code for Dijkstra

- decide how to represent a vertex and edges
  - adjacency list?
  - sets?
  - does this graph have a predictable structure?
    - grids / checker board - edges are implied

Graph as a Grid

1.414	1	sqrt(1*1 + 1*1)	
1	start	1	
1.414	obstacle cost = 4	1.414	

Grid Implementation

- 2d array or sparse matrix?
- each element in array holds cost
- easy to look up cost of neighbours
  - grid[current_row + 1][current_col]
  - don't go off the edge (indices < 0 or >= max)
- often used to simplify representation of complex problem
  - solve the simple problem