

Pathfinding with BFS

Lecture Question

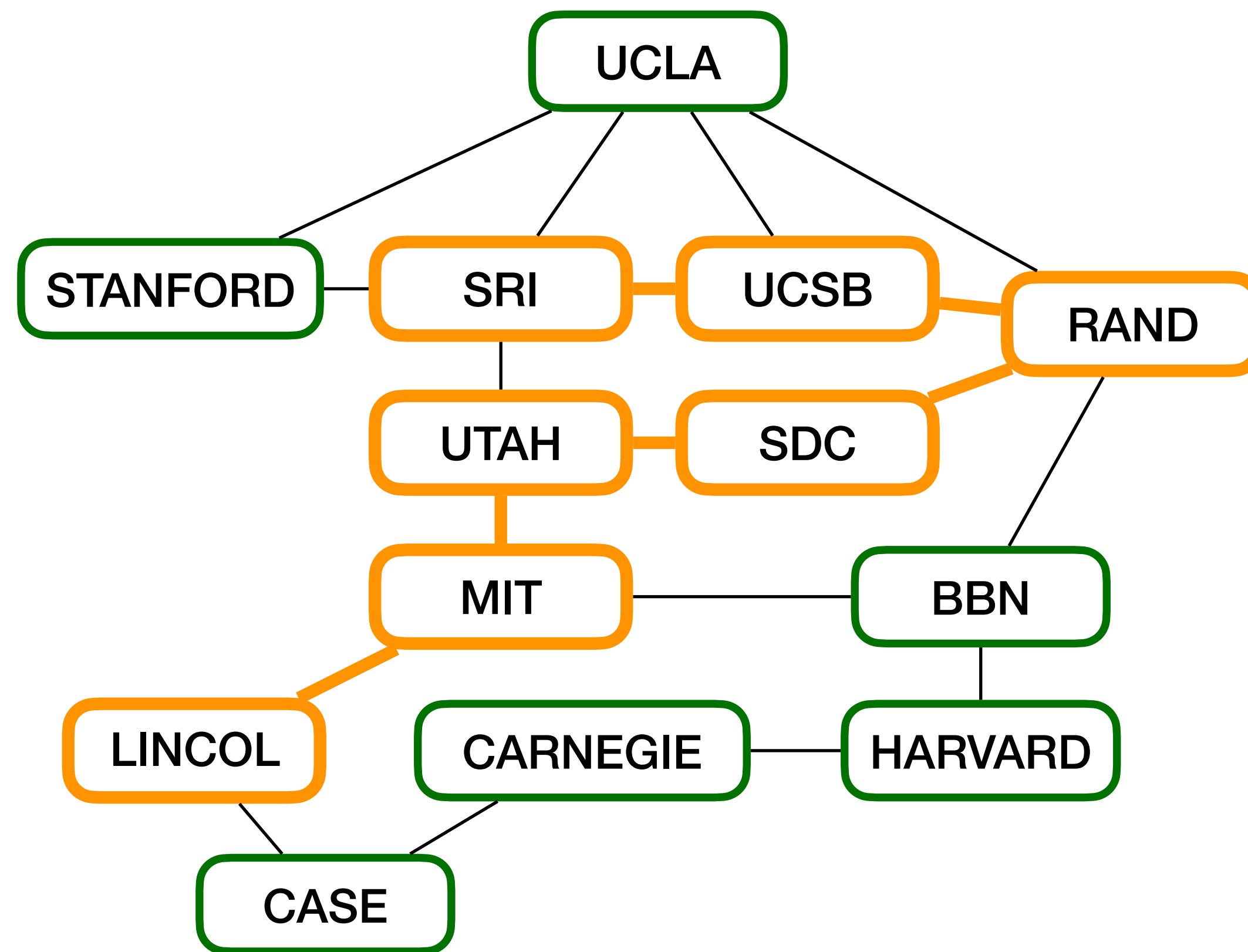
Task: Find the distance between two nodes in a graph

- In the week9.Graph class
 - Write a method named distance that takes two node indices (Ints) and returns the distance between the two nodes
 - You may assume the two input nodes are connected

Testing: In a package named "tests" create a class named "TestDistance" as a test suite that tests the functionality listed above

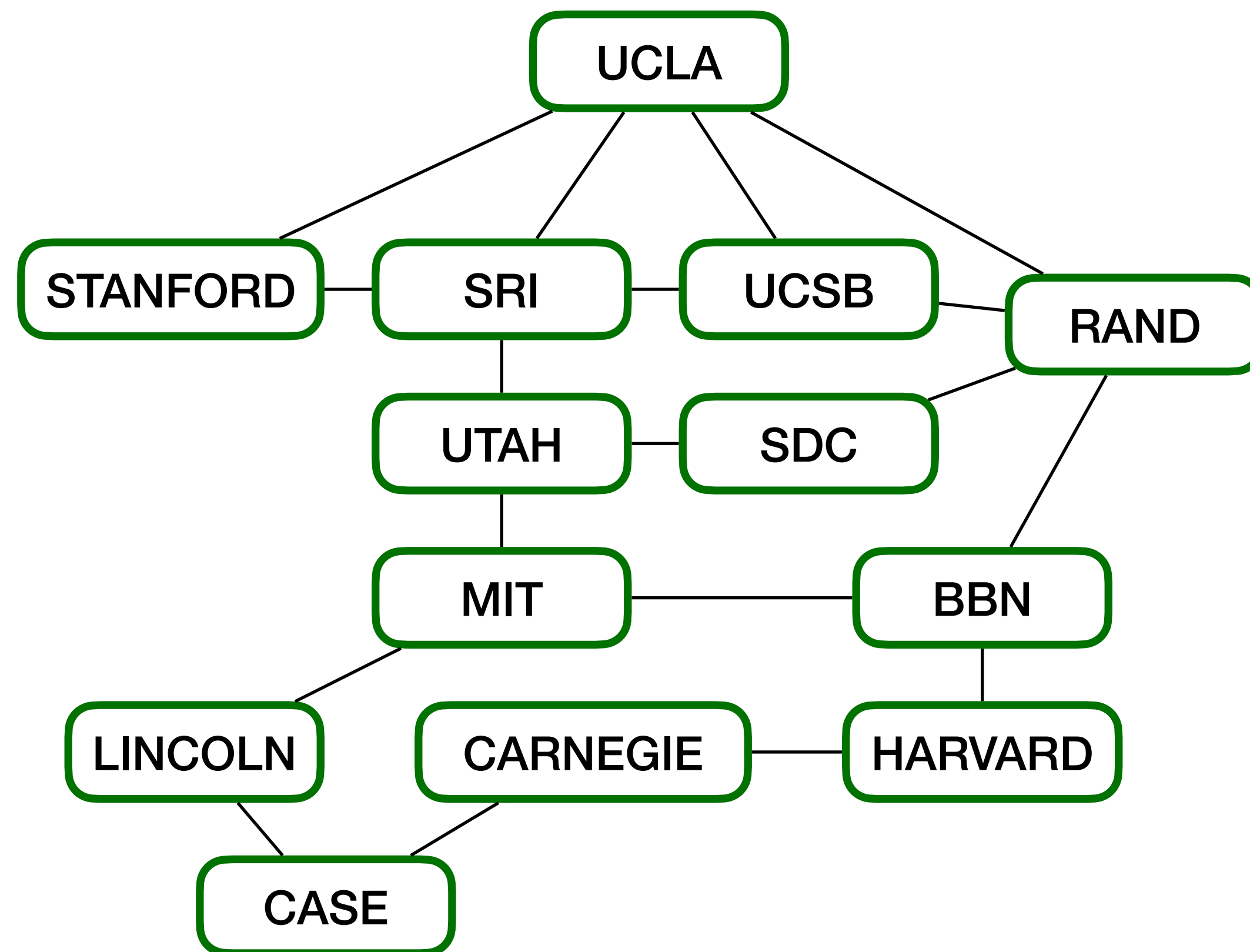
Paths

- Path: A sequence of nodes with each adjacent pair of nodes connected by an edge
- The length of a path is the number of edges it contains (number of nodes - 1)
- [LINCOLN, MIT, UTAH, SDC, RAND, UCSB, SRI] <-- Path of length 6



Distance

- Distance between two nodes: The length of the shortest path between the nodes
- Distance between LINCOLN and SRI == 3
- Distance between RAND and BBN == 1

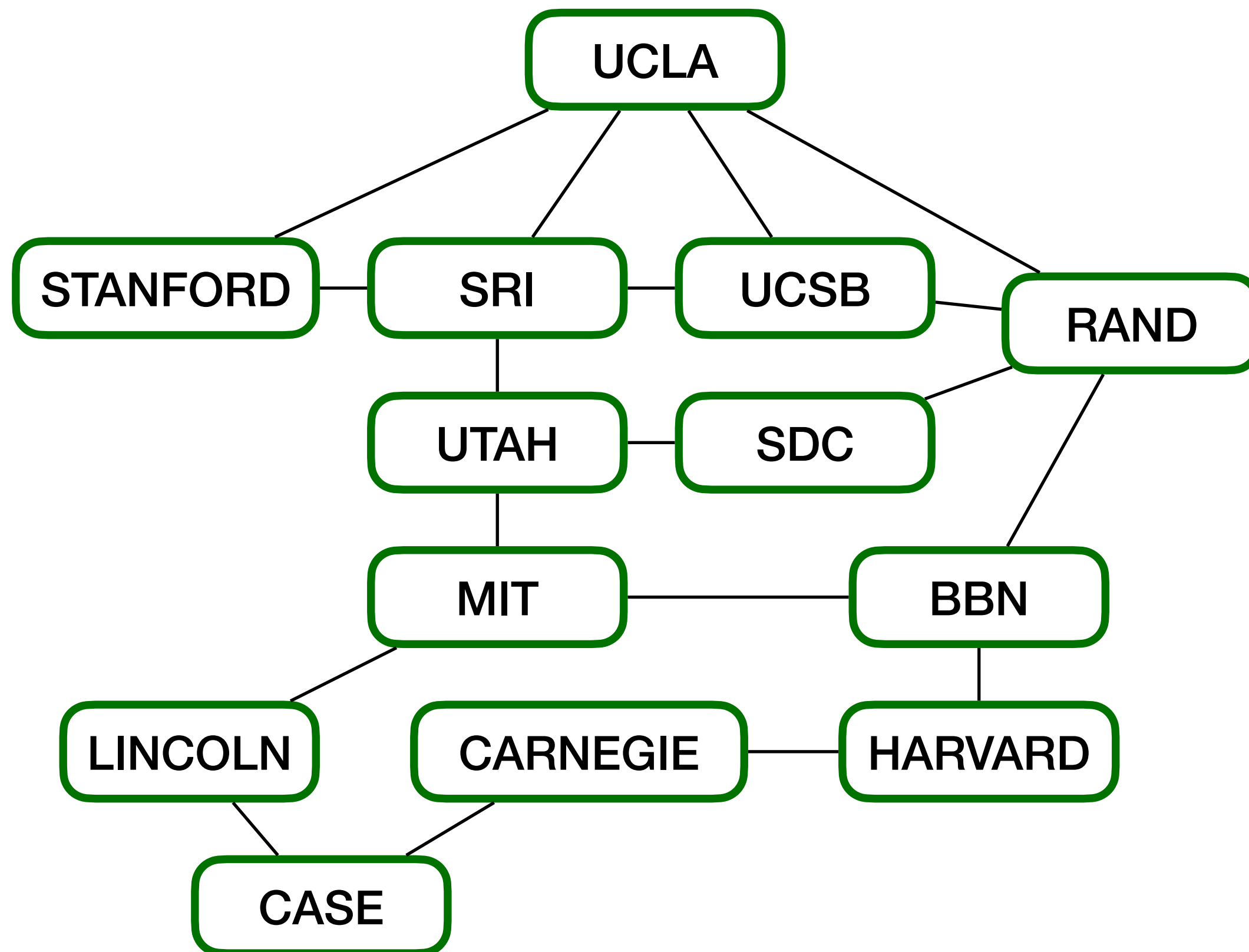


Use BFS to find the distance between nodes

Track the shortest path for pathfinding

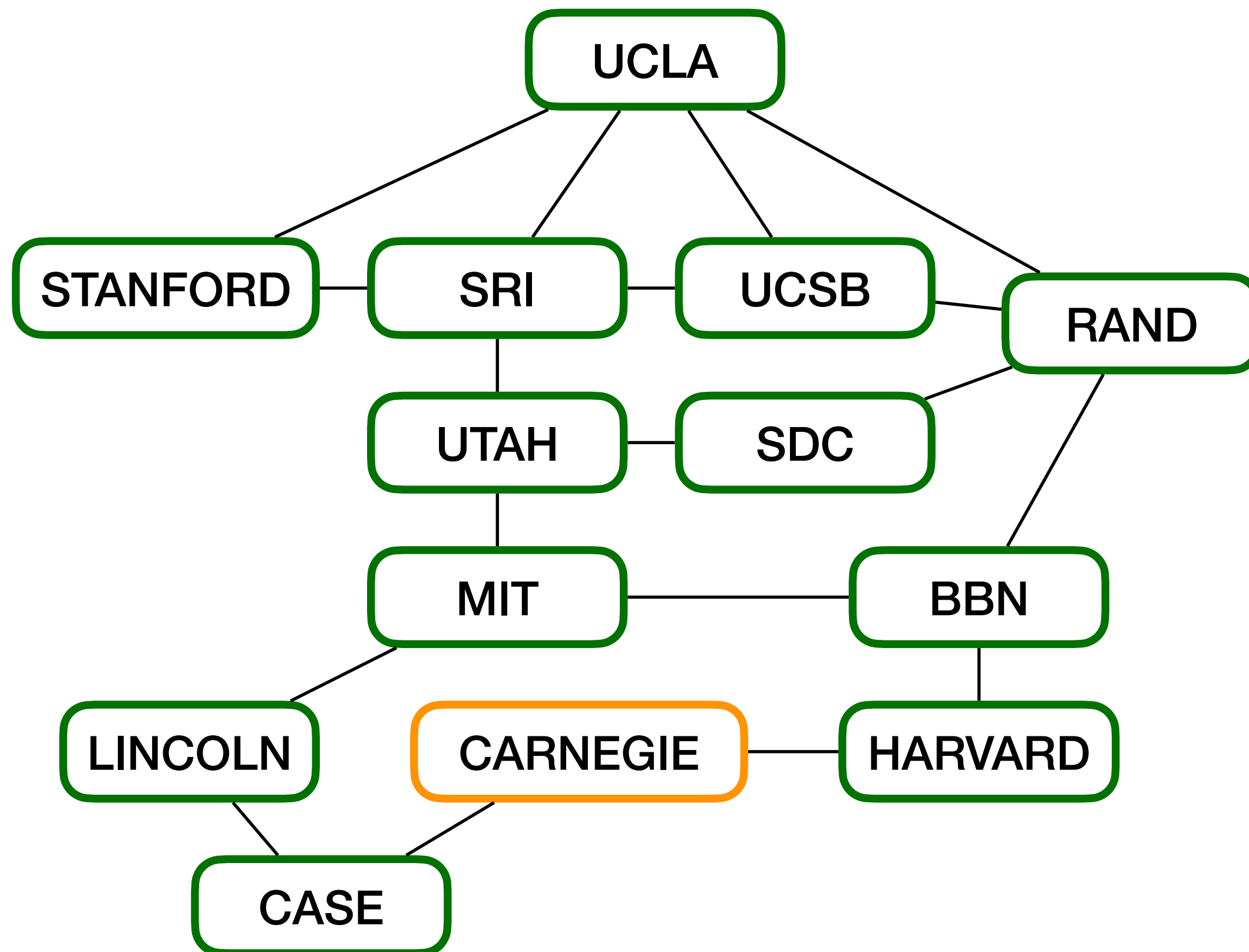
There's Levels to This

- Let's run through BFS again
 - Instead of just finding the connected component, let's track the paths taken to explore each node



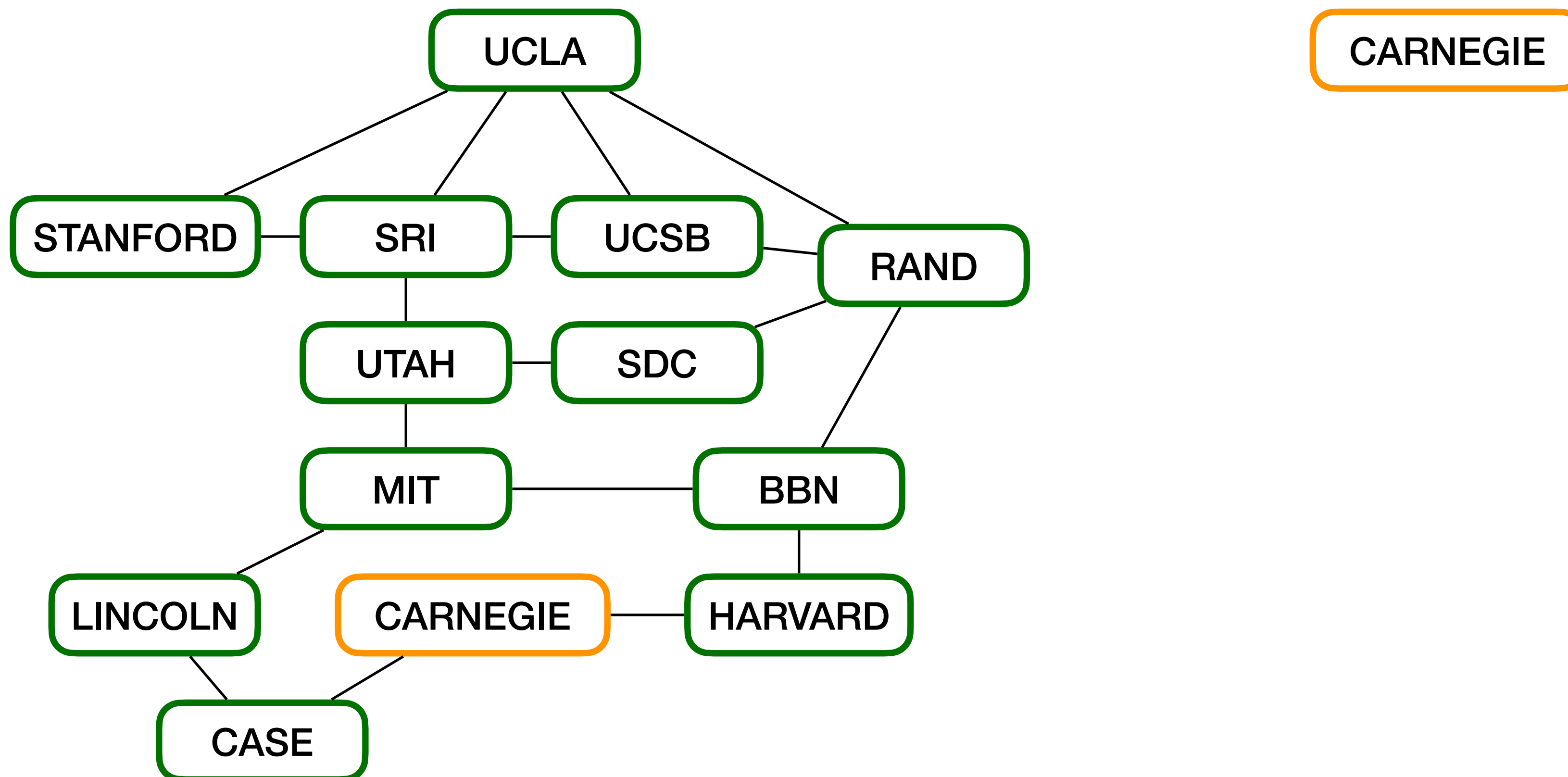
There's Levels to This

- Let's start at CARNEGIE this time



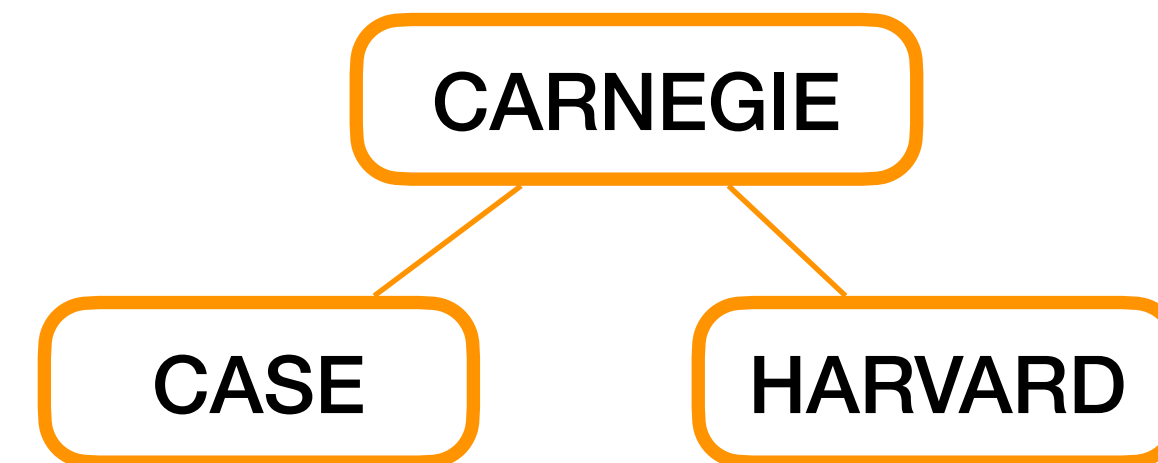
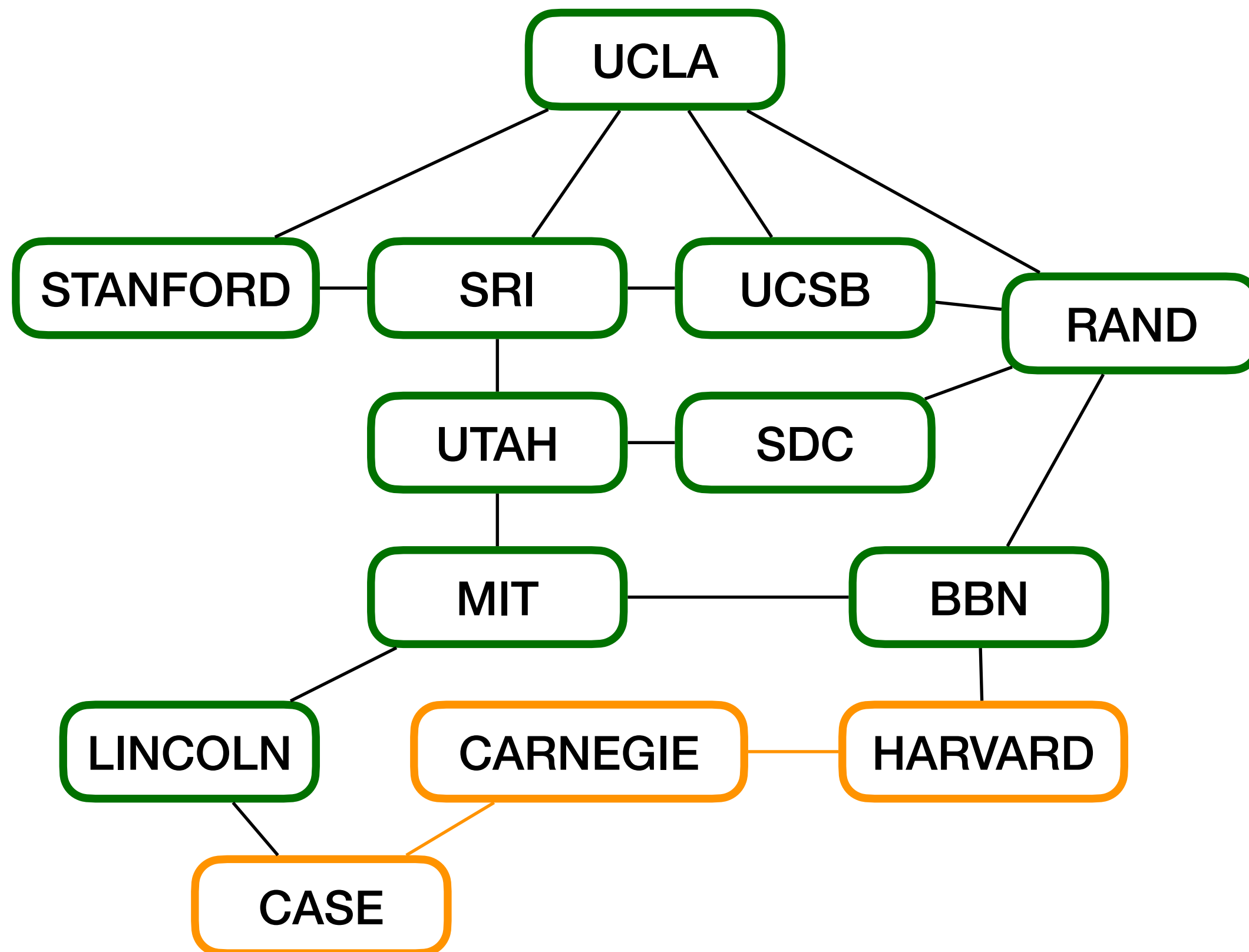
There's Levels to This

- Keep track of all edges used to explore new nodes
- Redraw the graph with only these edges



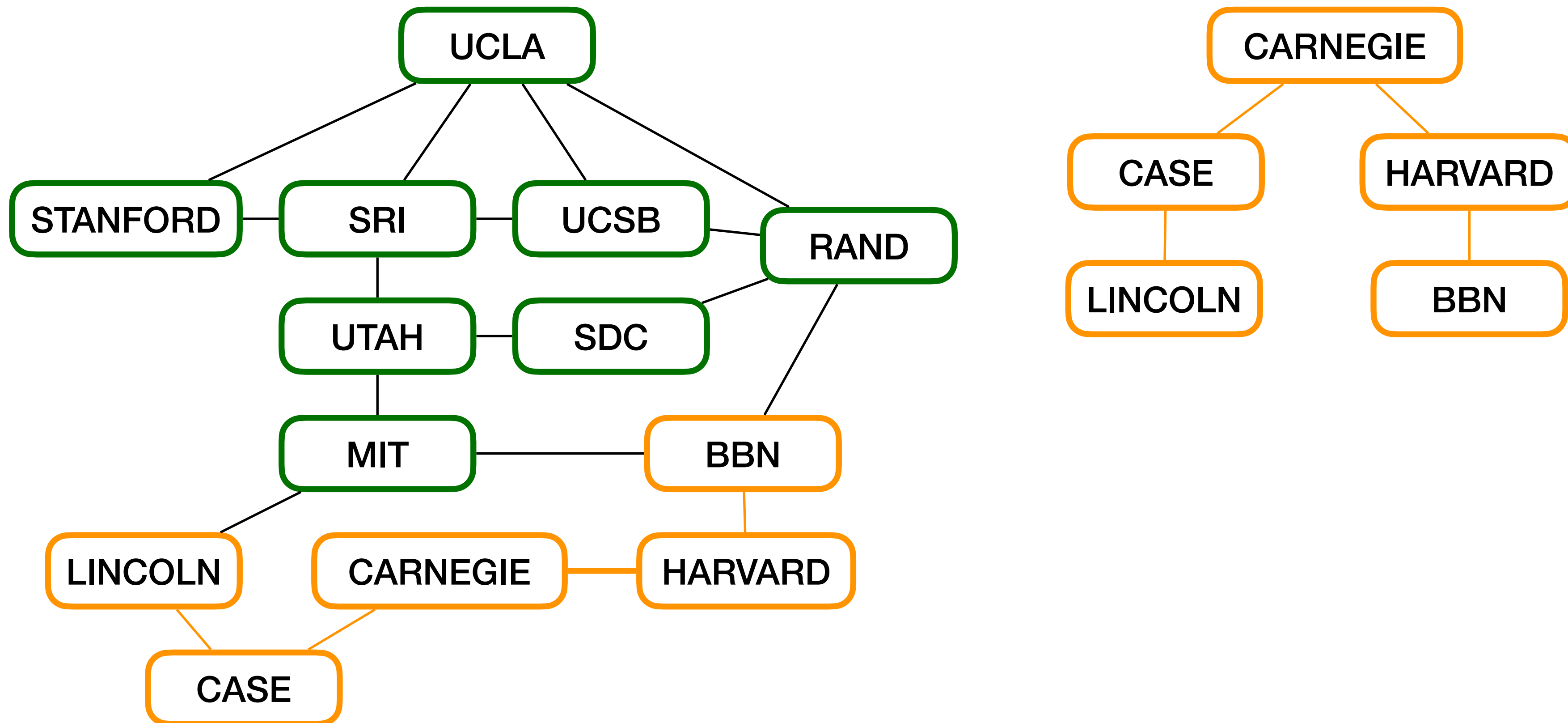
There's Levels to This

- Explore all neighbors of the starting node



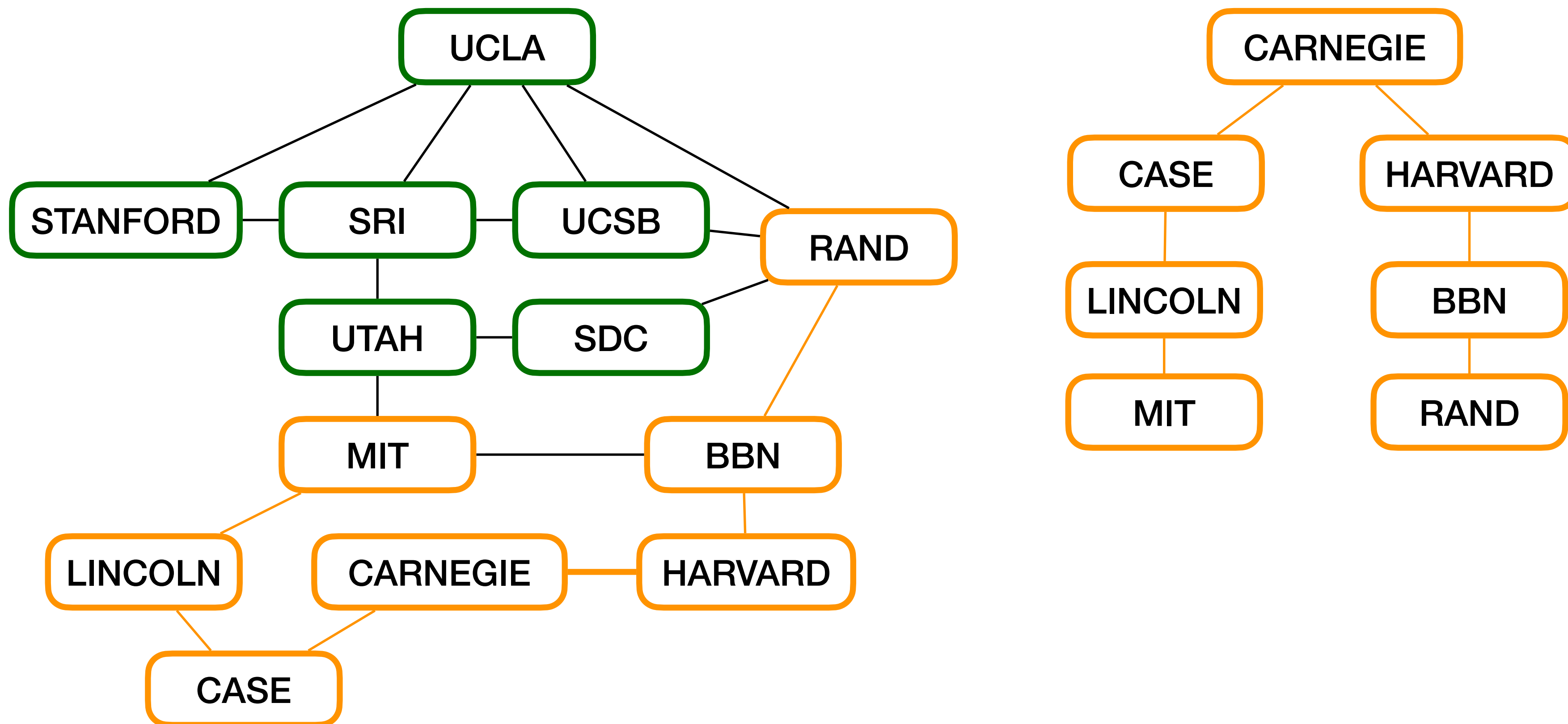
There's Levels to This

- Explore all neighbors of the nodes explored in the last step



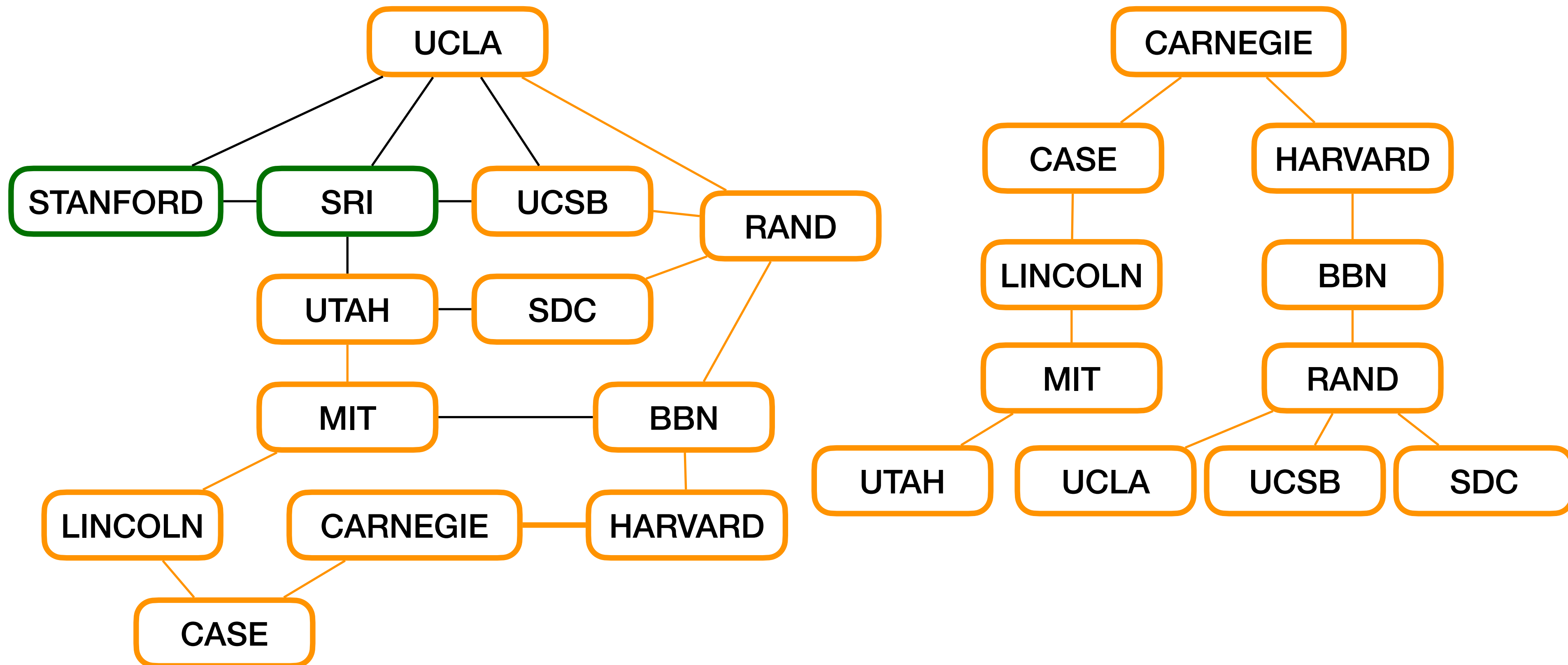
There's Levels to This

- Repeat
- Choose edge to use for MIT arbitrarily



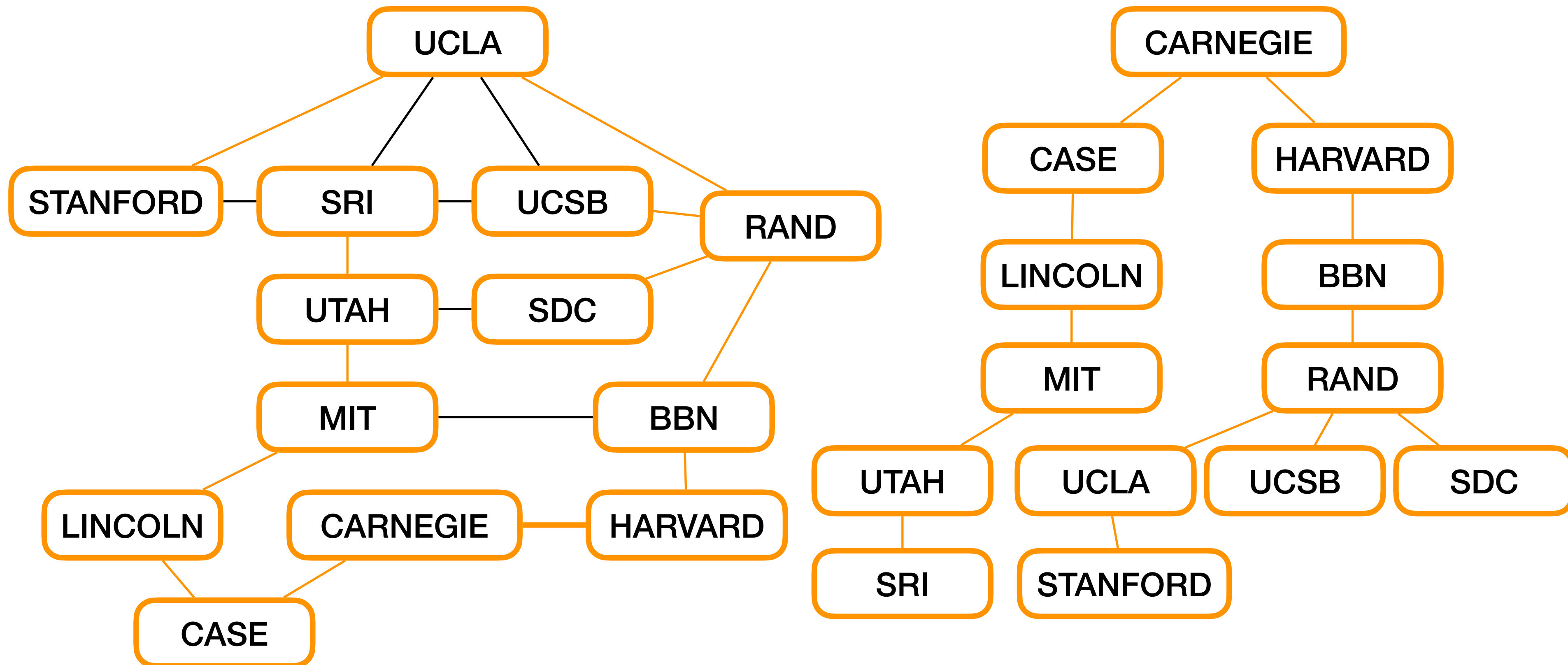
There's Levels to This

- Each step we explore all nodes that can be reached from the nodes added in the previous step



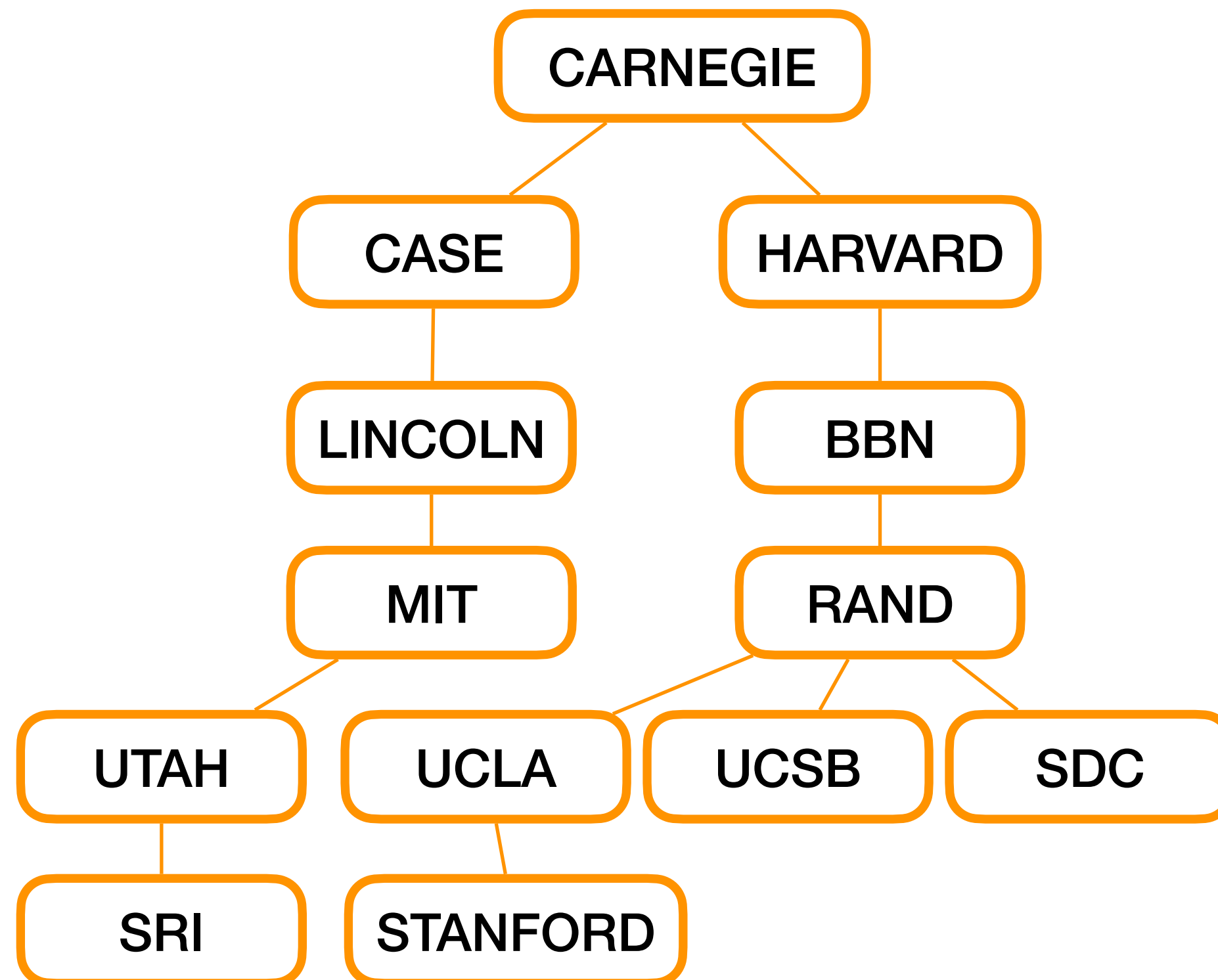
There's Levels to This

- Each step we explore all nodes that can be reached from the nodes added in the previous step



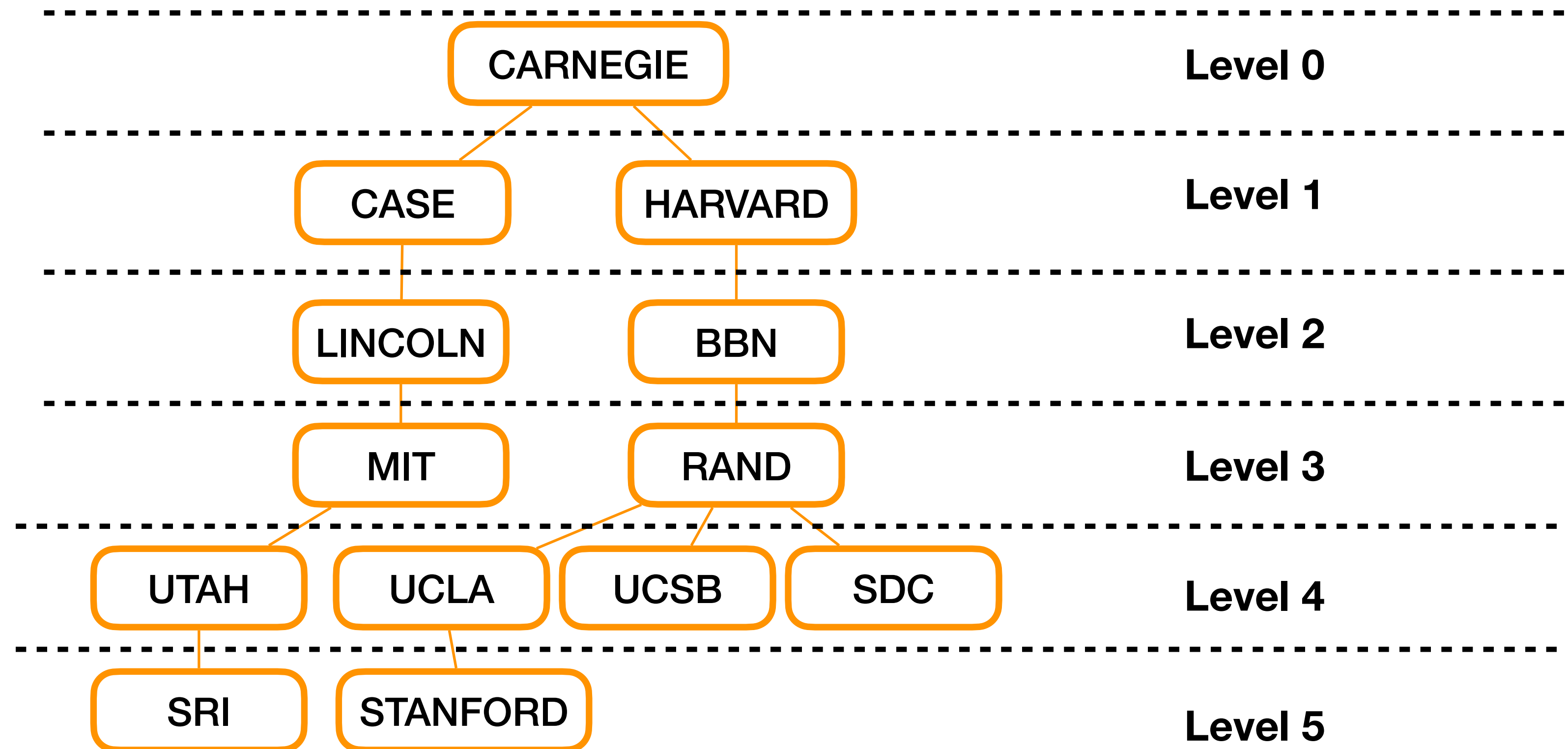
There's Levels to This

- We have a new graph with a few edges removed
- This graph is a tree (no cycles)



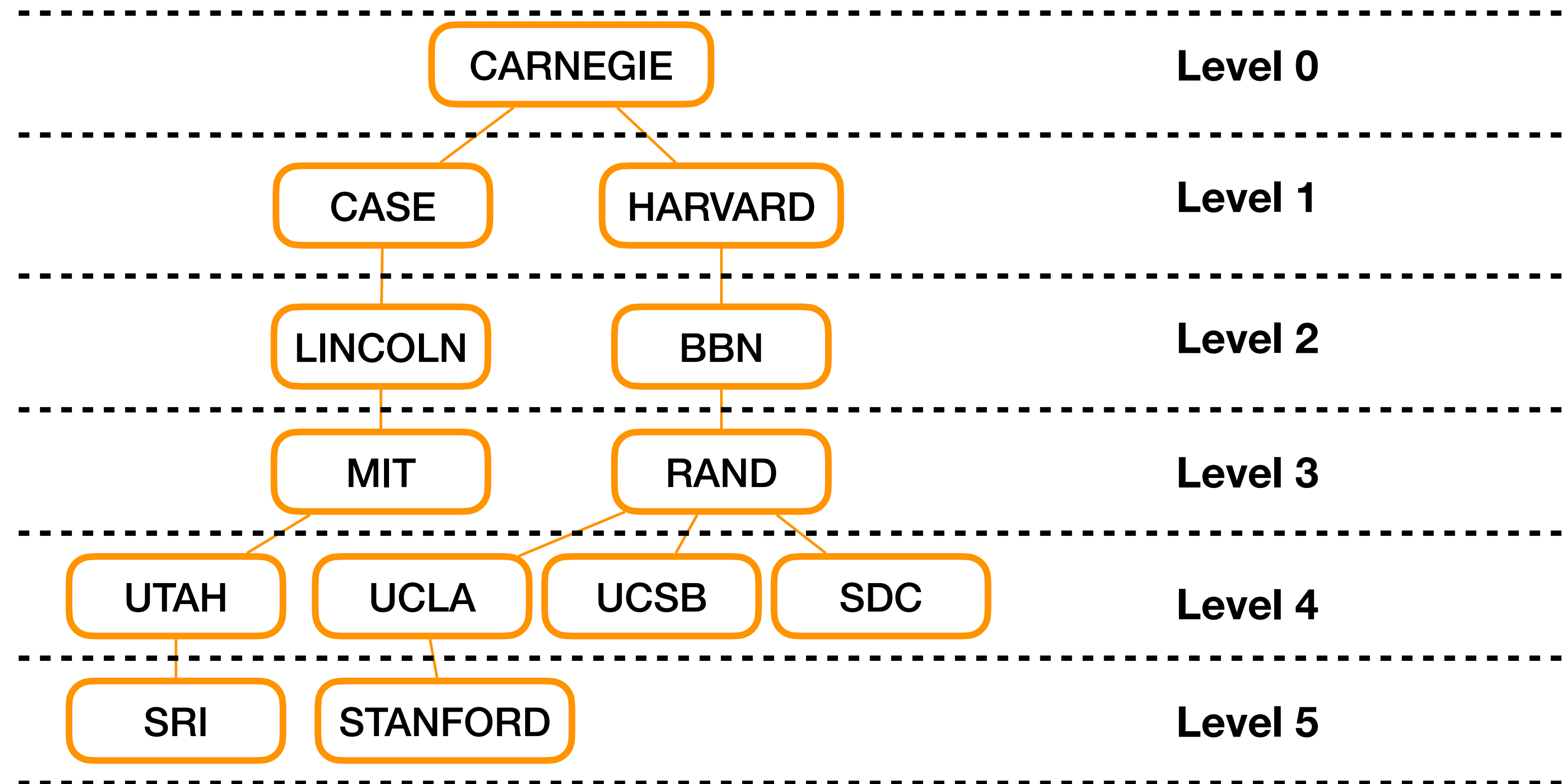
There's Levels to This

- And it has levels!



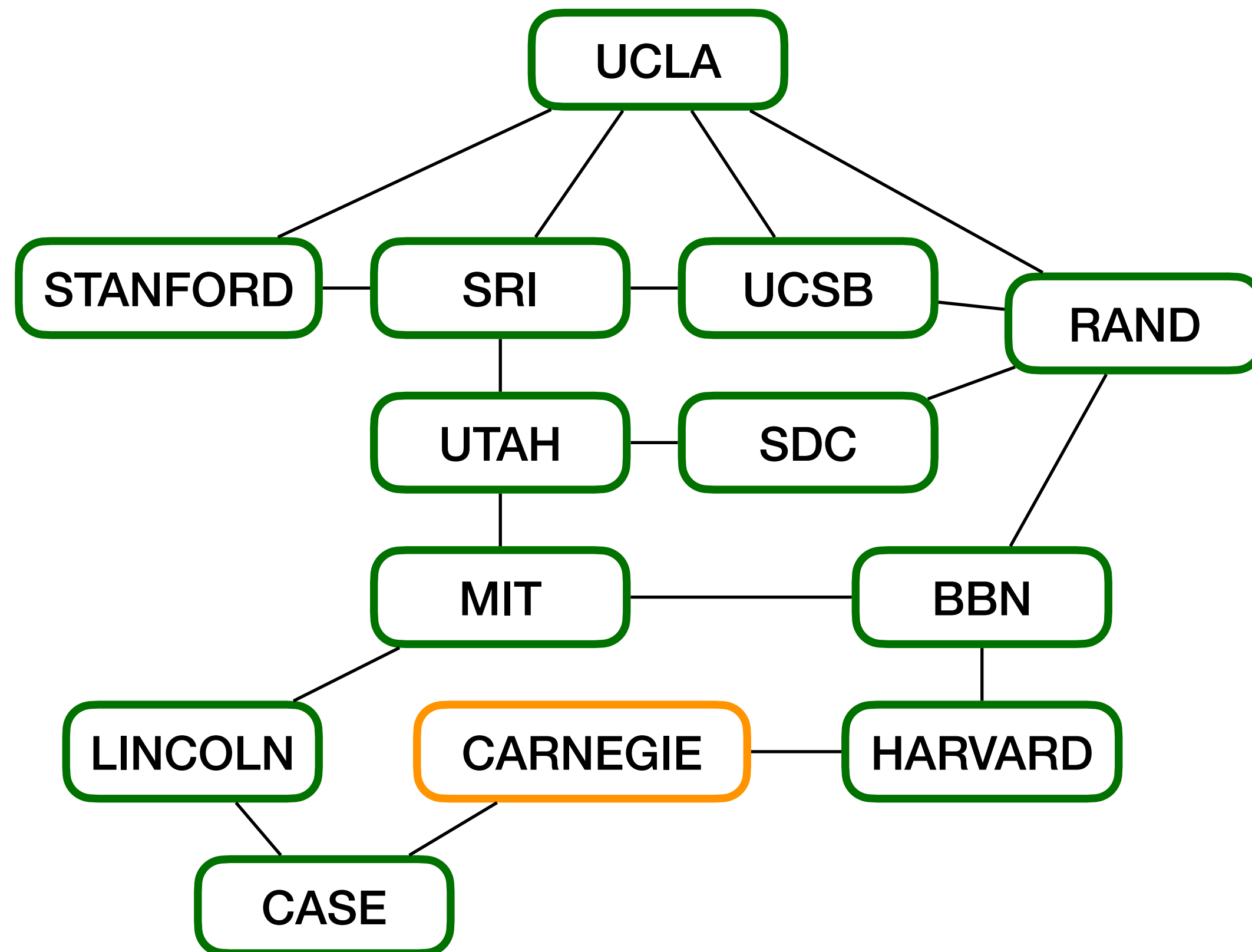
There's Levels to This

- Number the levels starting with 0
- The level number == the distance from the starting node to any node in that level



BFS and Distance

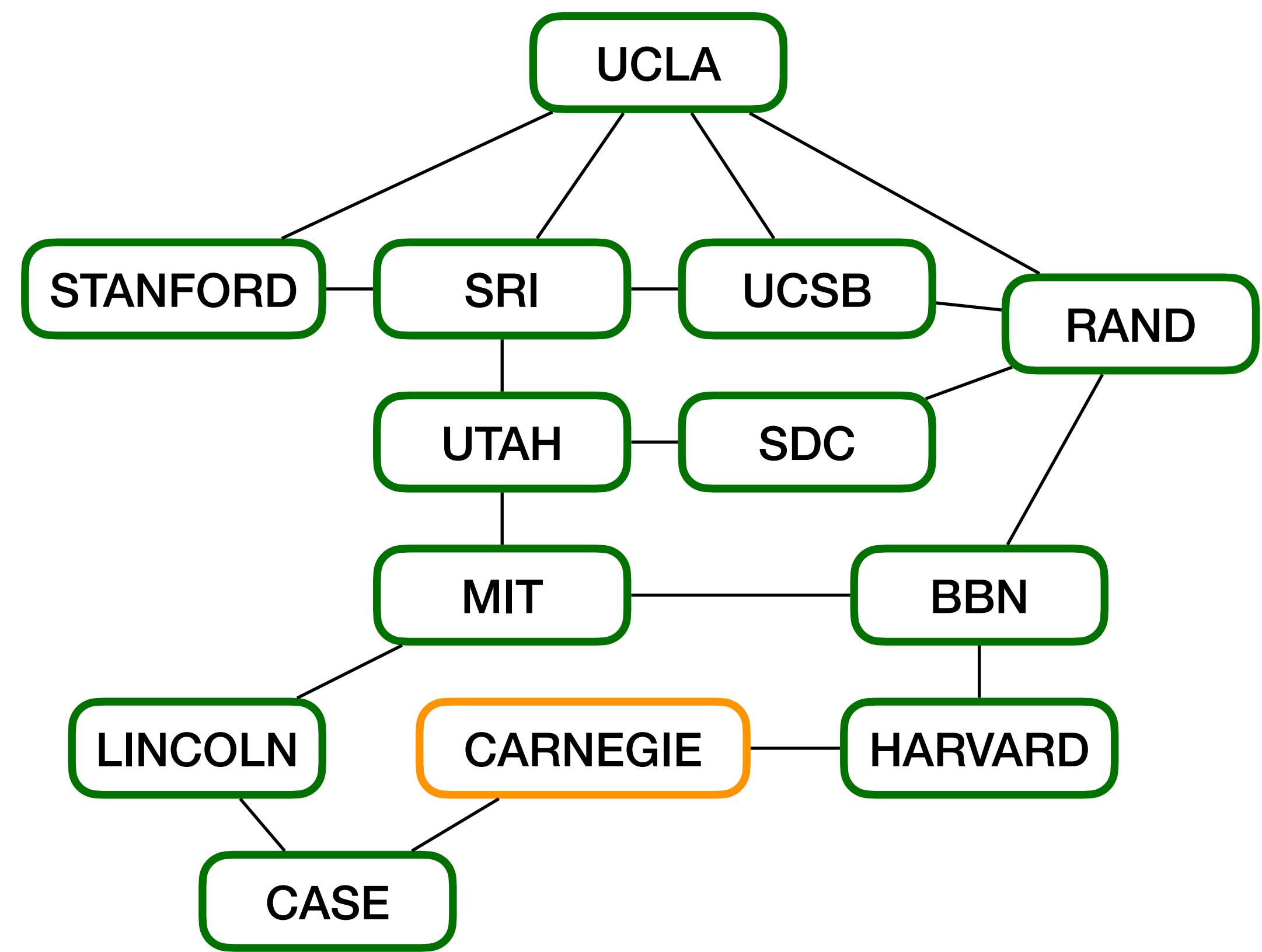
- But how do we track the levels?
- Track levels in a data structure



UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	∞
BBN	∞
LINCOLN	∞
CARNEGIE	0
HARVARD	∞
CASE	∞

BFS and Distance

CARNEGIE

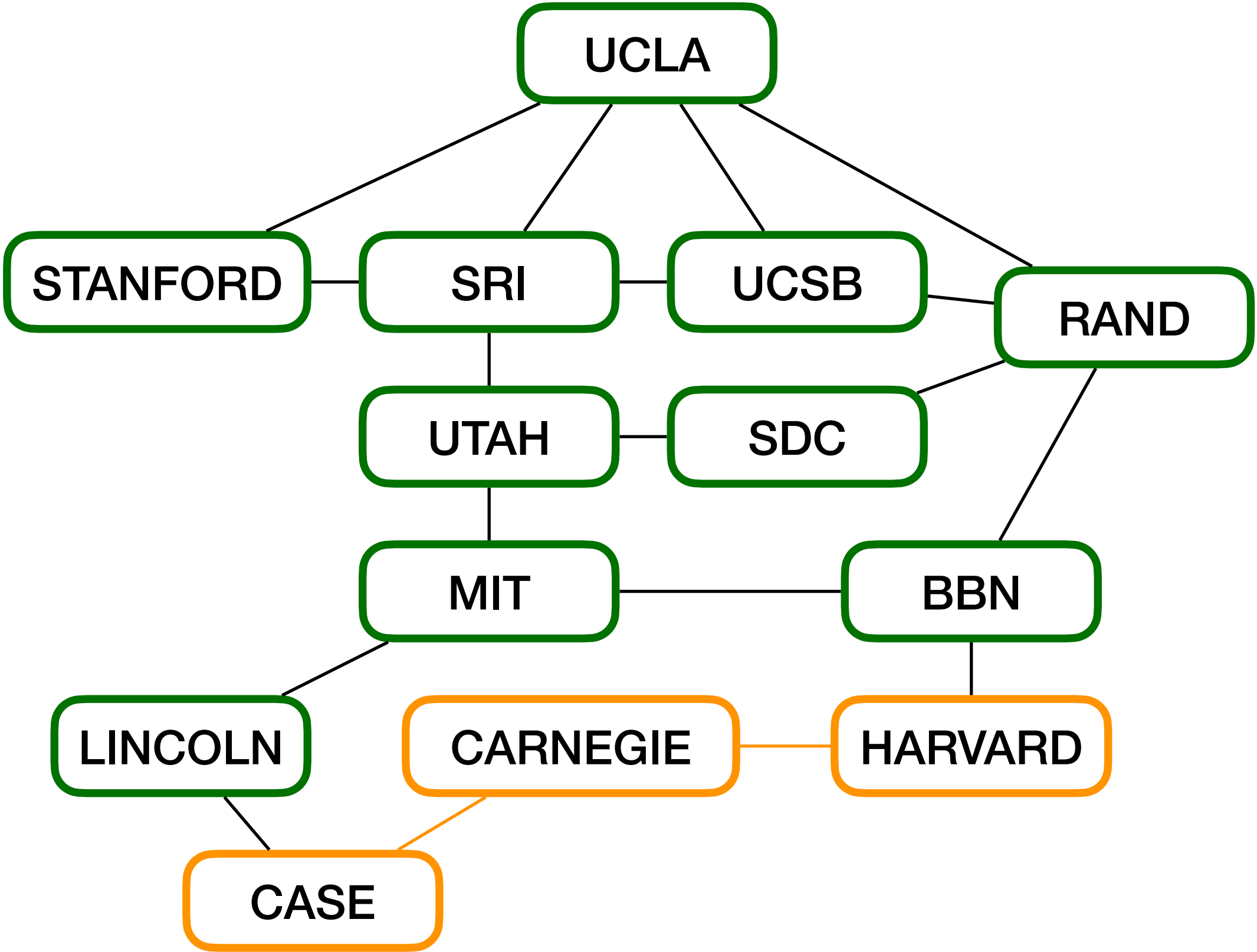


UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	∞
BBN	∞
LINCOLN	∞
CARNEGIE	0
HARVARD	∞
CASE	∞

BFS and Distance

CASE

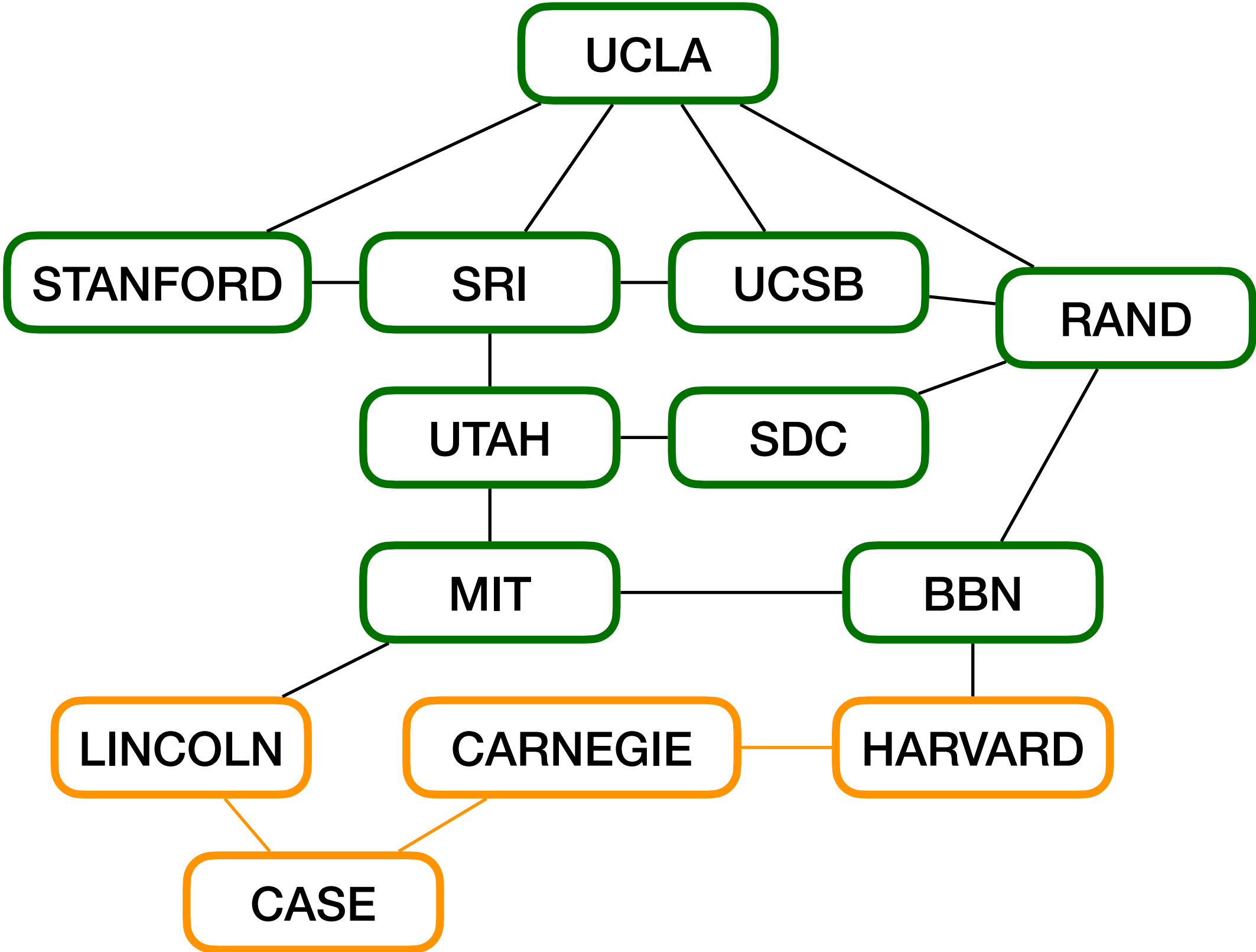
HARVARD



UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	∞
BBN	∞
LINCOLN	∞
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

HARVARD LINCOLN

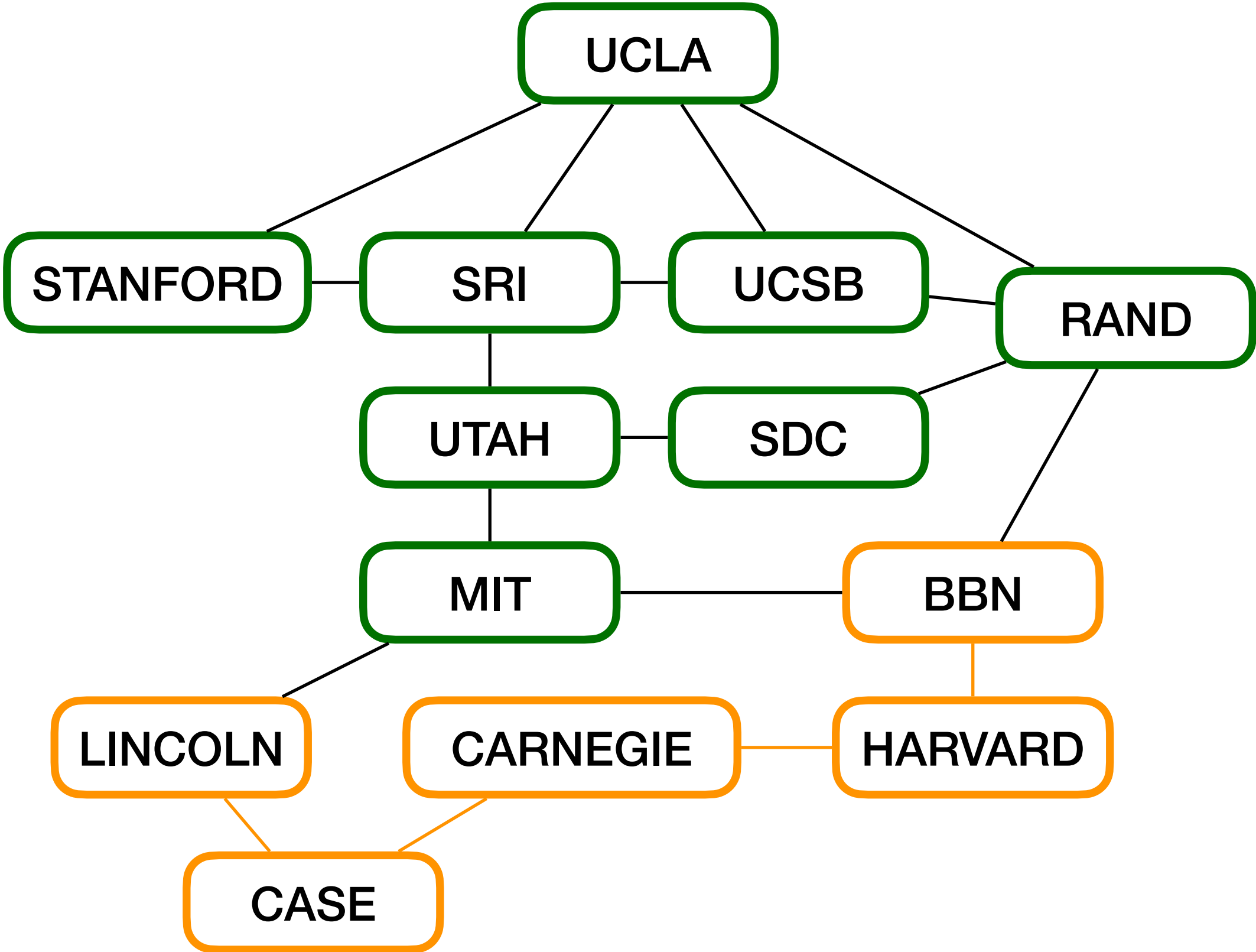


UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	∞
BBN	∞
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

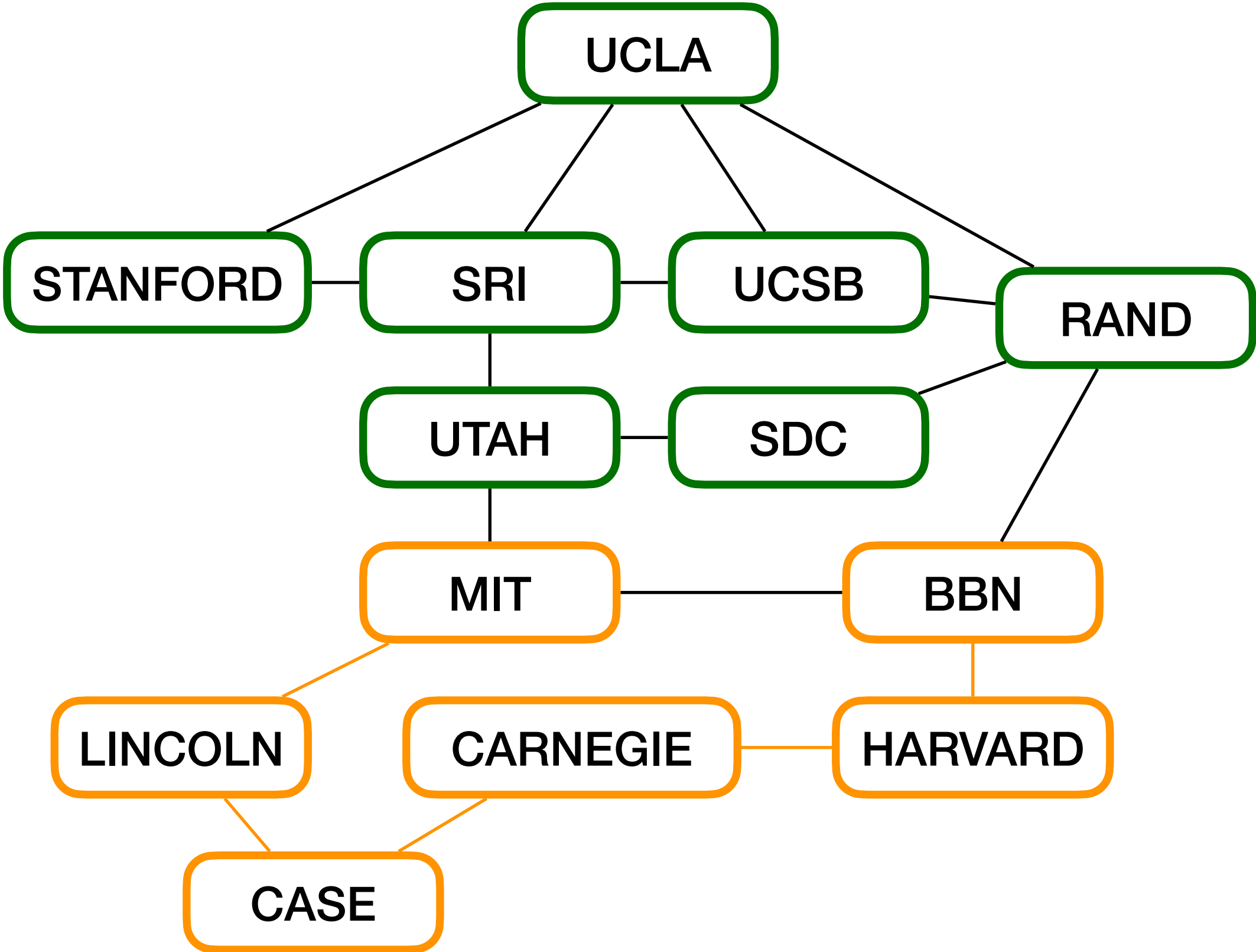
LINCOLN

BBN



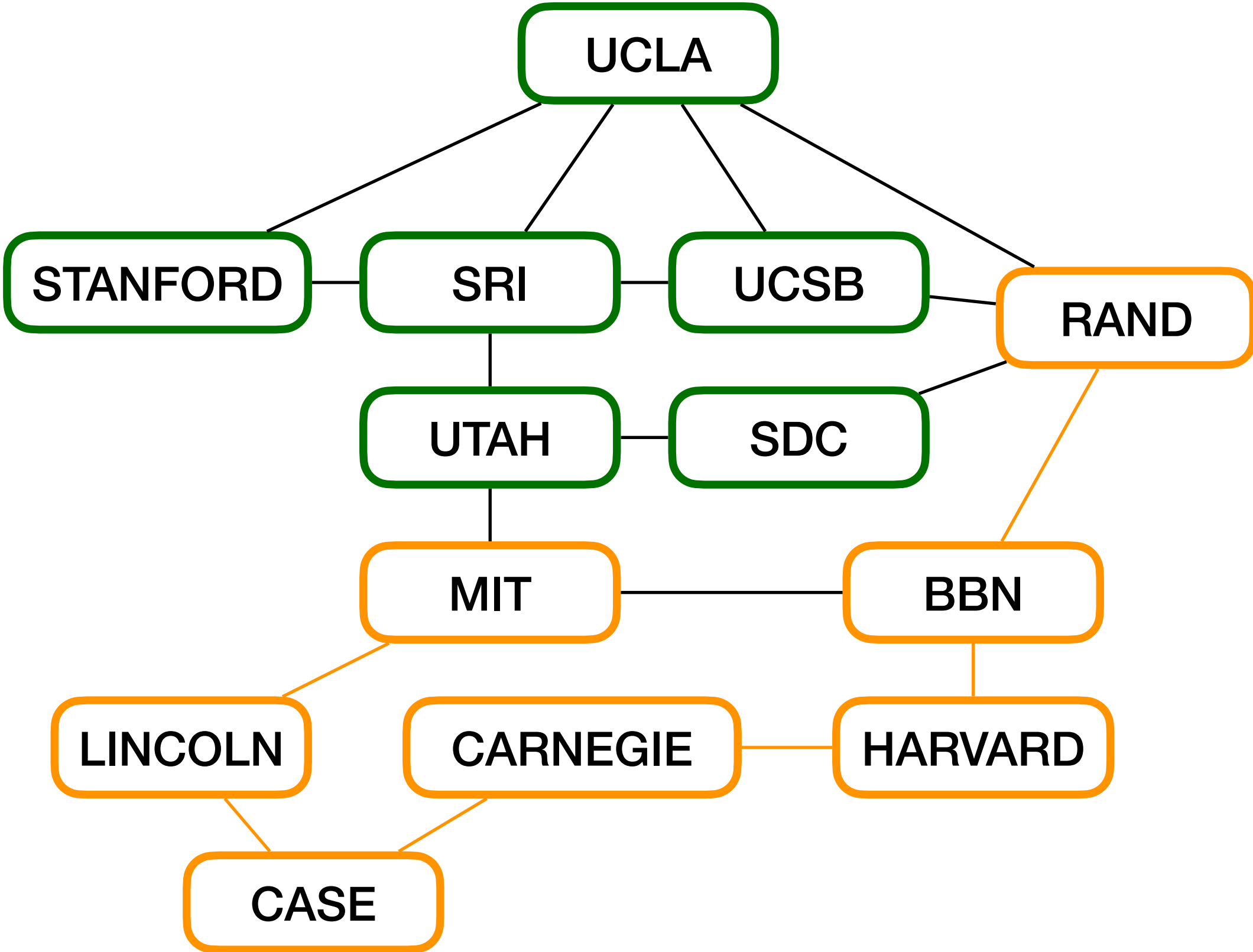
UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	∞
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance



UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

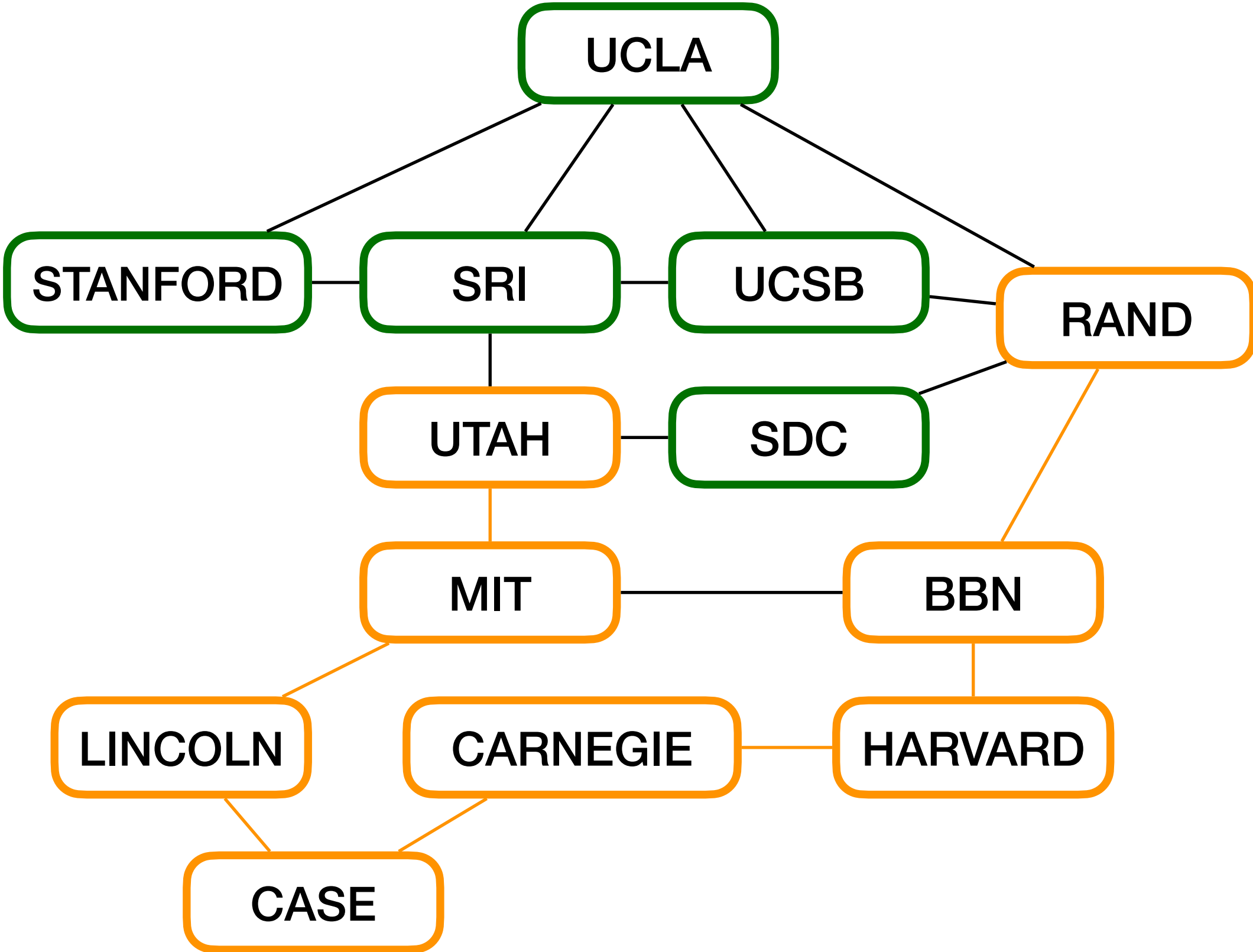
BFS and Distance



UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	3
UTAH	∞
SDC	∞
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

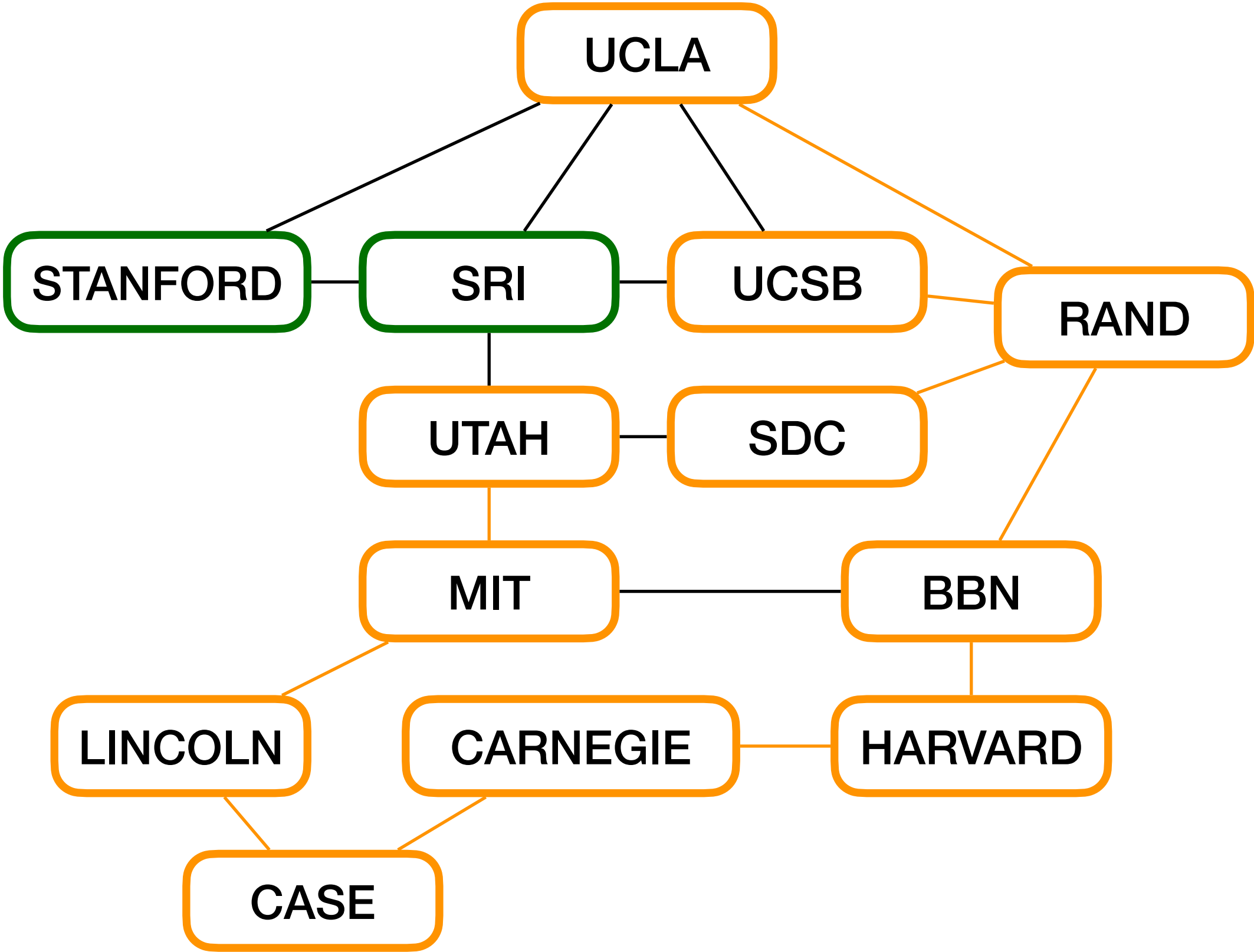
RAND	UTAH
------	------



UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	3
UTAH	4
SDC	∞
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

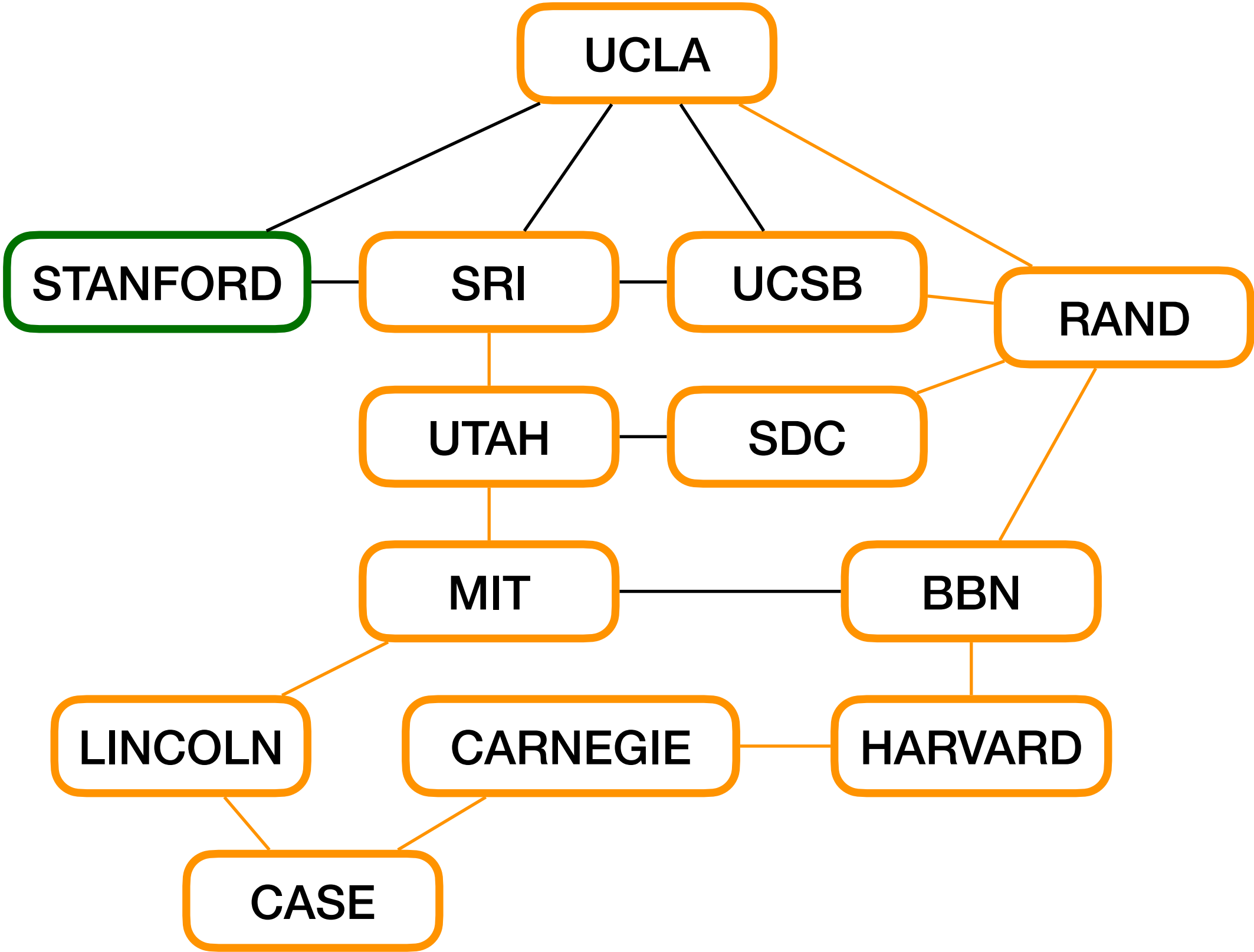
UTAH	UCLA	UCSB	SDC
------	------	------	-----



UCLA	4
STANFORD	∞
SRI	∞
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

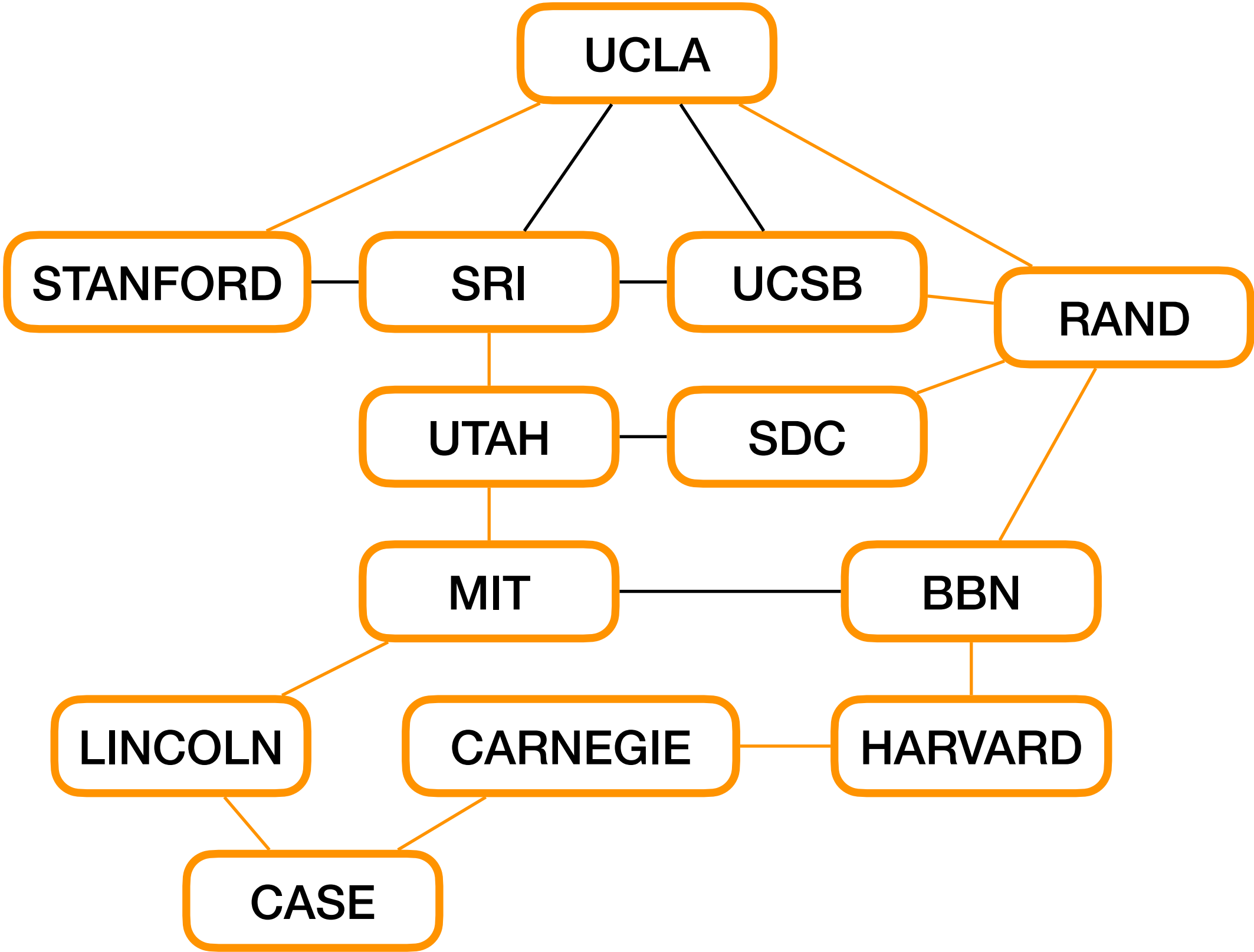
UCLA	UCSB	SDC	SRI
------	------	-----	-----



UCLA	4
STANFORD	∞
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

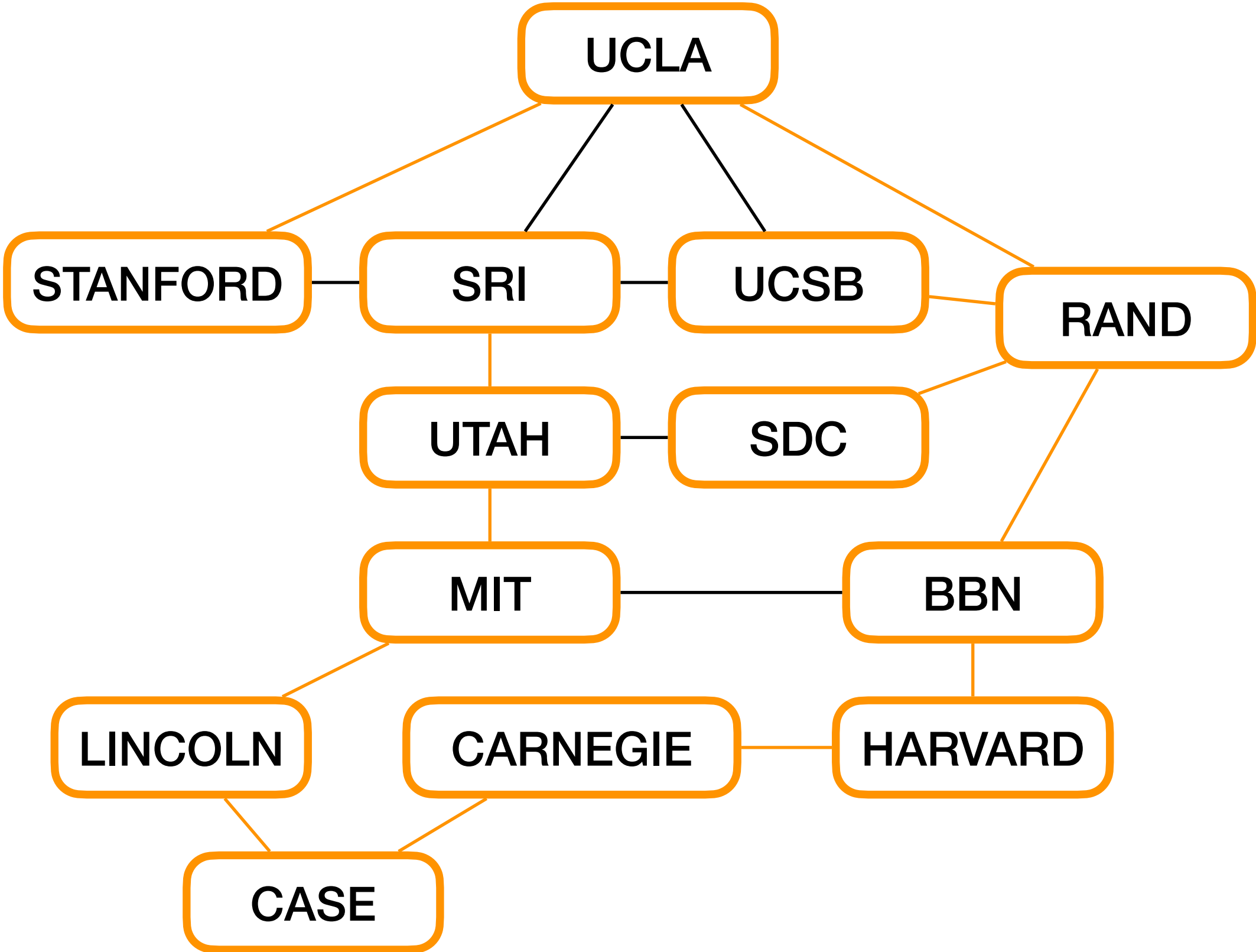
UCSB	SDC	SRI	STANFORD
------	-----	-----	----------



UCLA	4
STANFORD	5
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

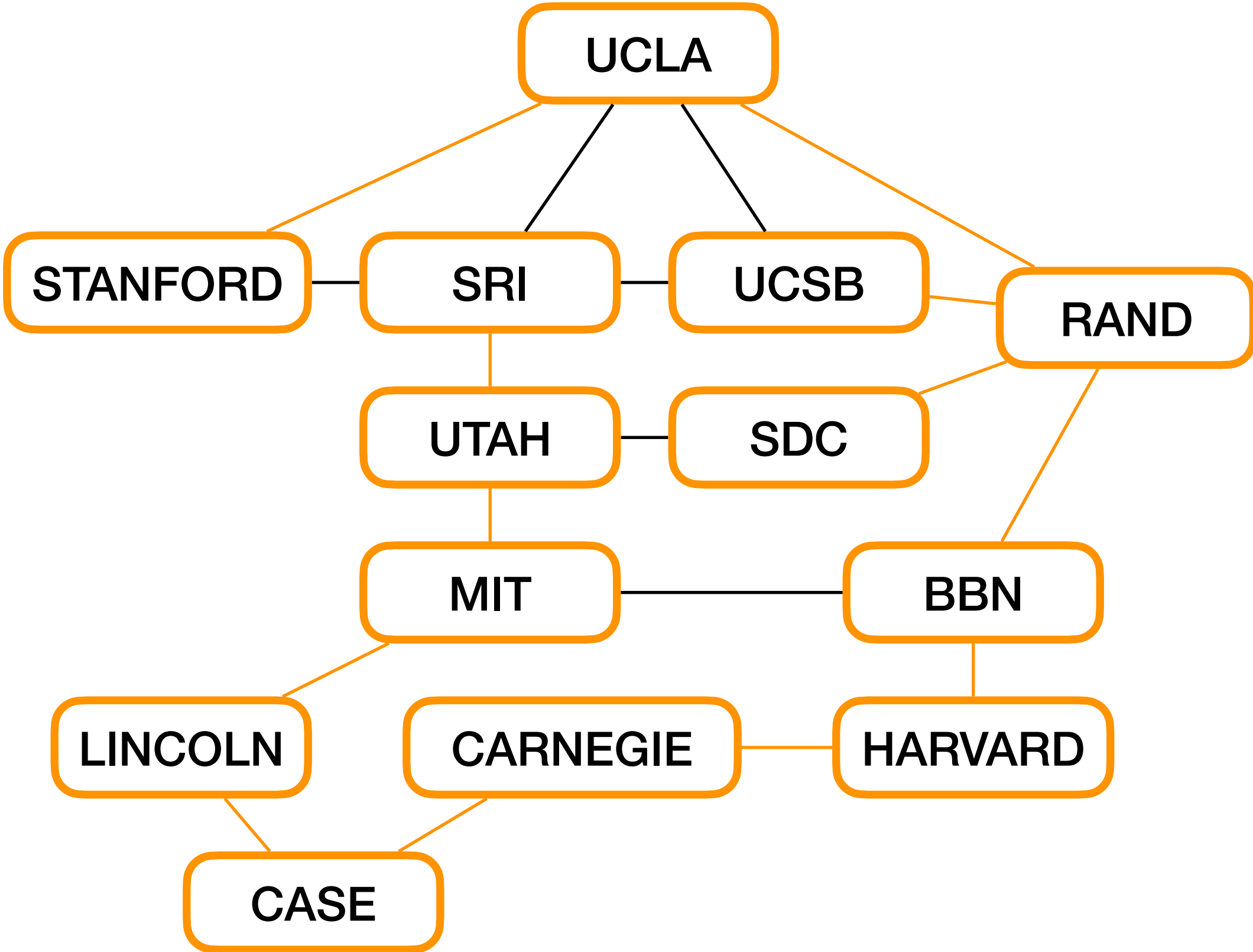
SDC	SRI	STANFORD
-----	-----	----------



UCLA	4
STANFORD	5
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

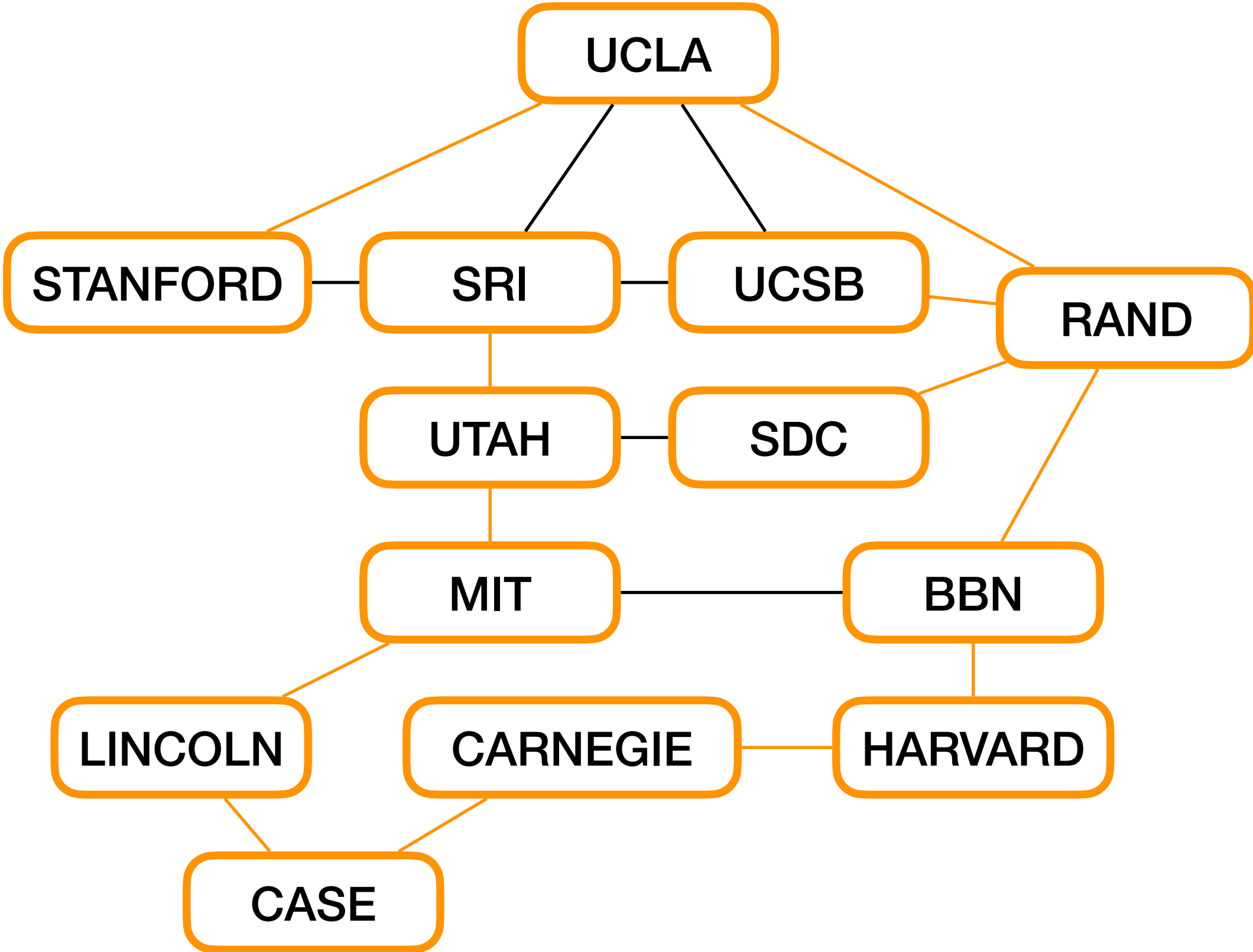
SRI	STANFORD
-----	----------



UCLA	4
STANFORD	5
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

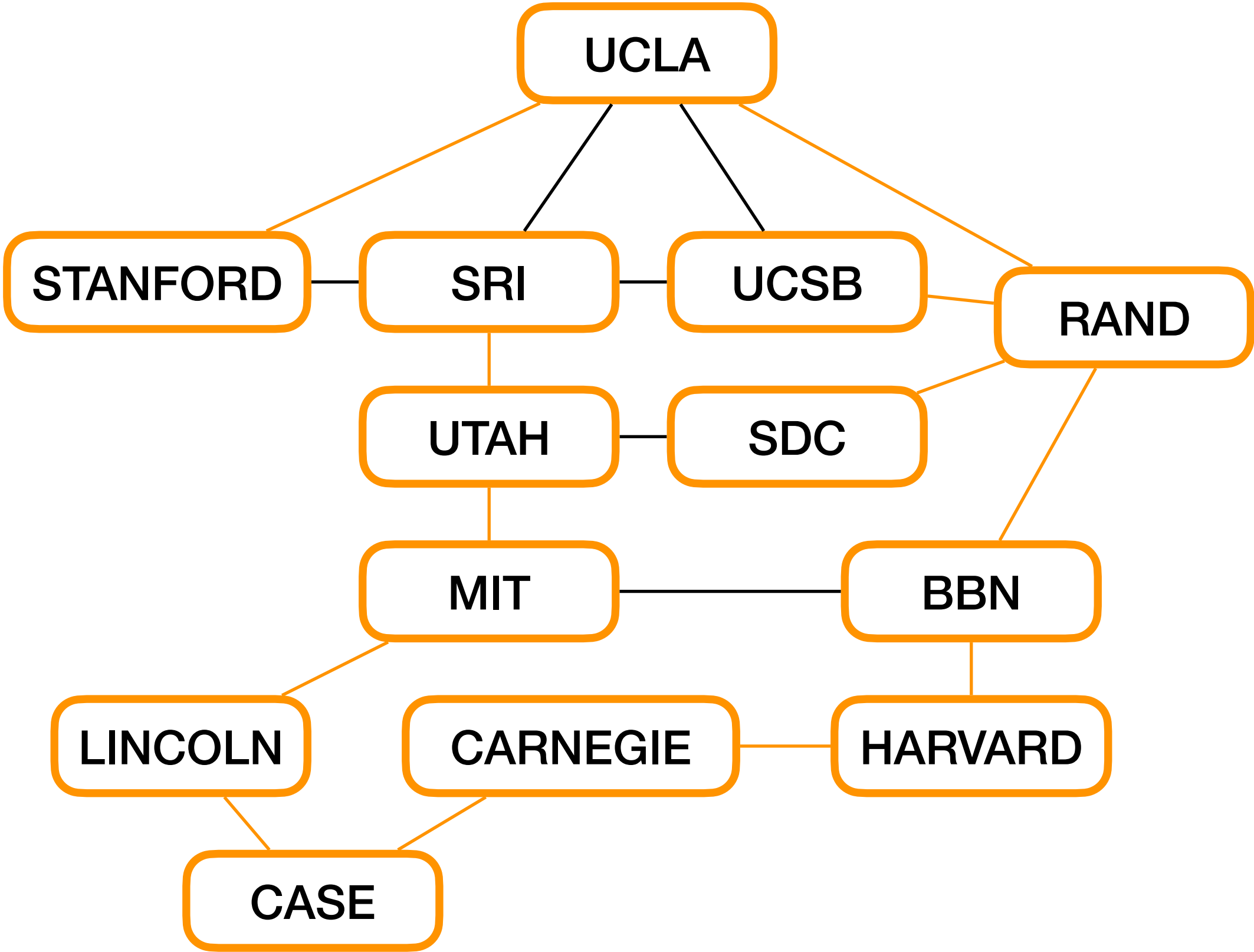
BFS and Distance

STANFORD



UCLA	4
STANFORD	5
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

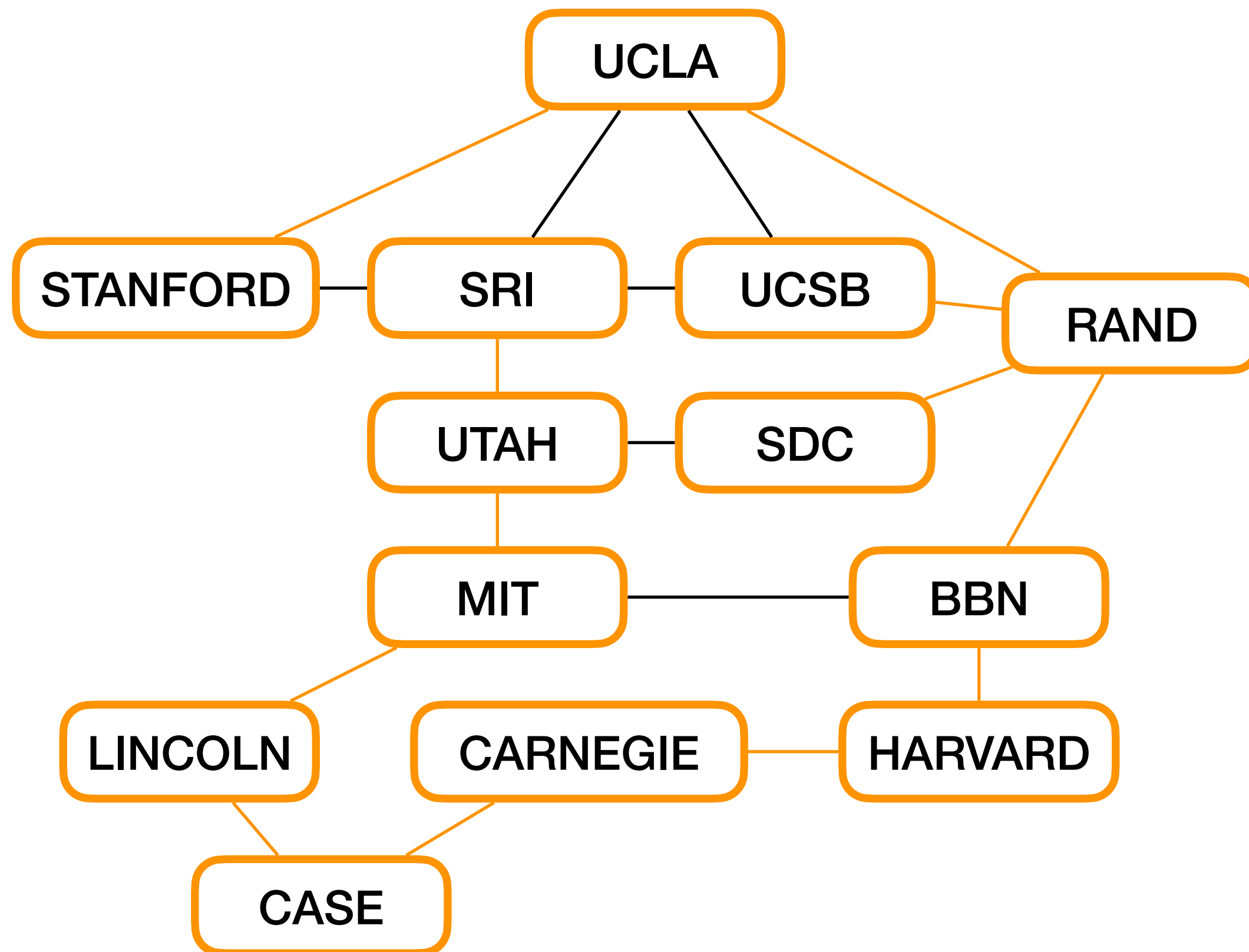
BFS and Distance



UCLA	4
STANFORD	5
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Distance

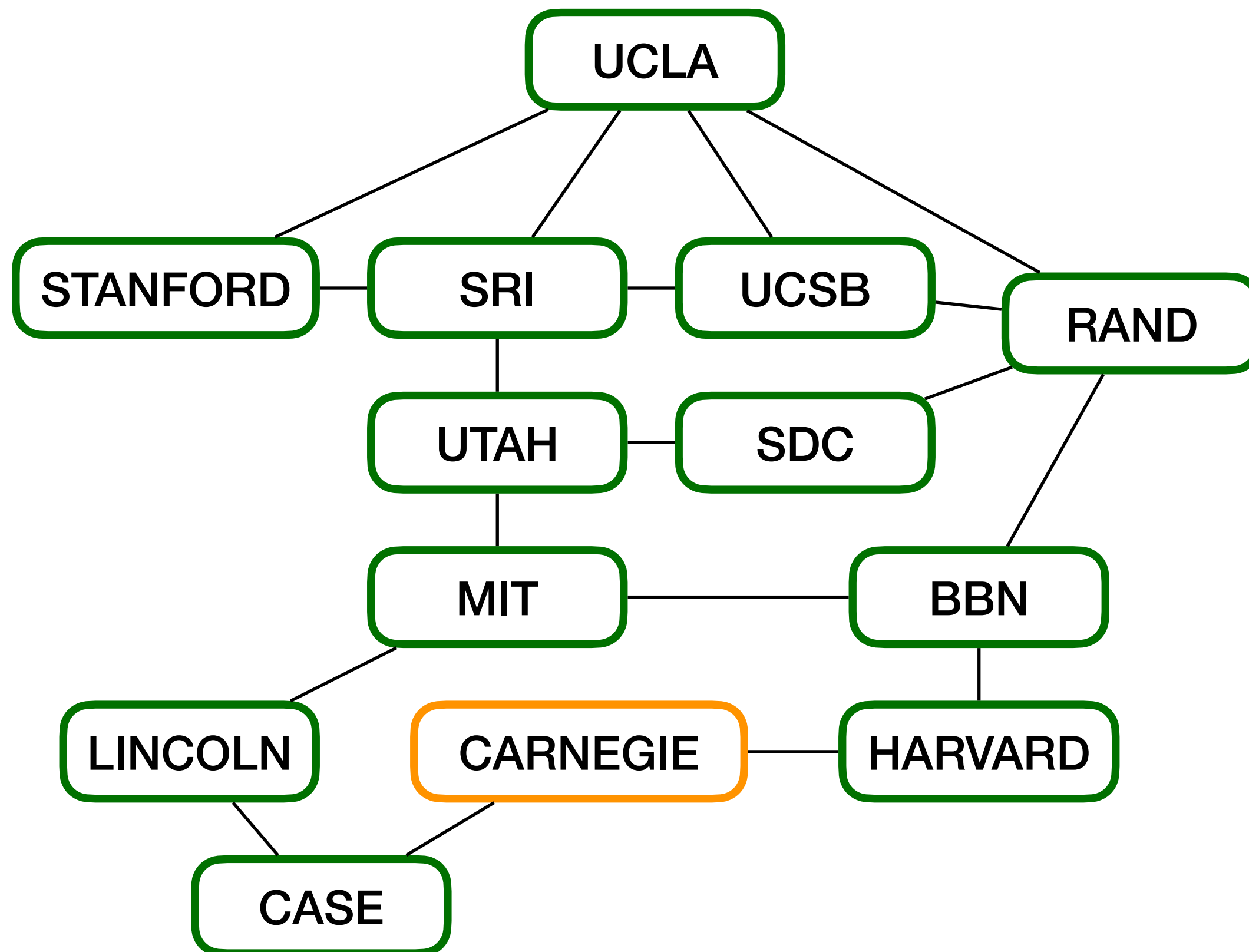
- And we have the distance from the start node to all other nodes in the graph



UCLA	4
STANFORD	5
SRI	5
UCSB	4
RAND	3
UTAH	4
SDC	4
MIT	3
BBN	2
LINCOLN	2
CARNEGIE	0
HARVARD	1
CASE	1

BFS and Pathfinding

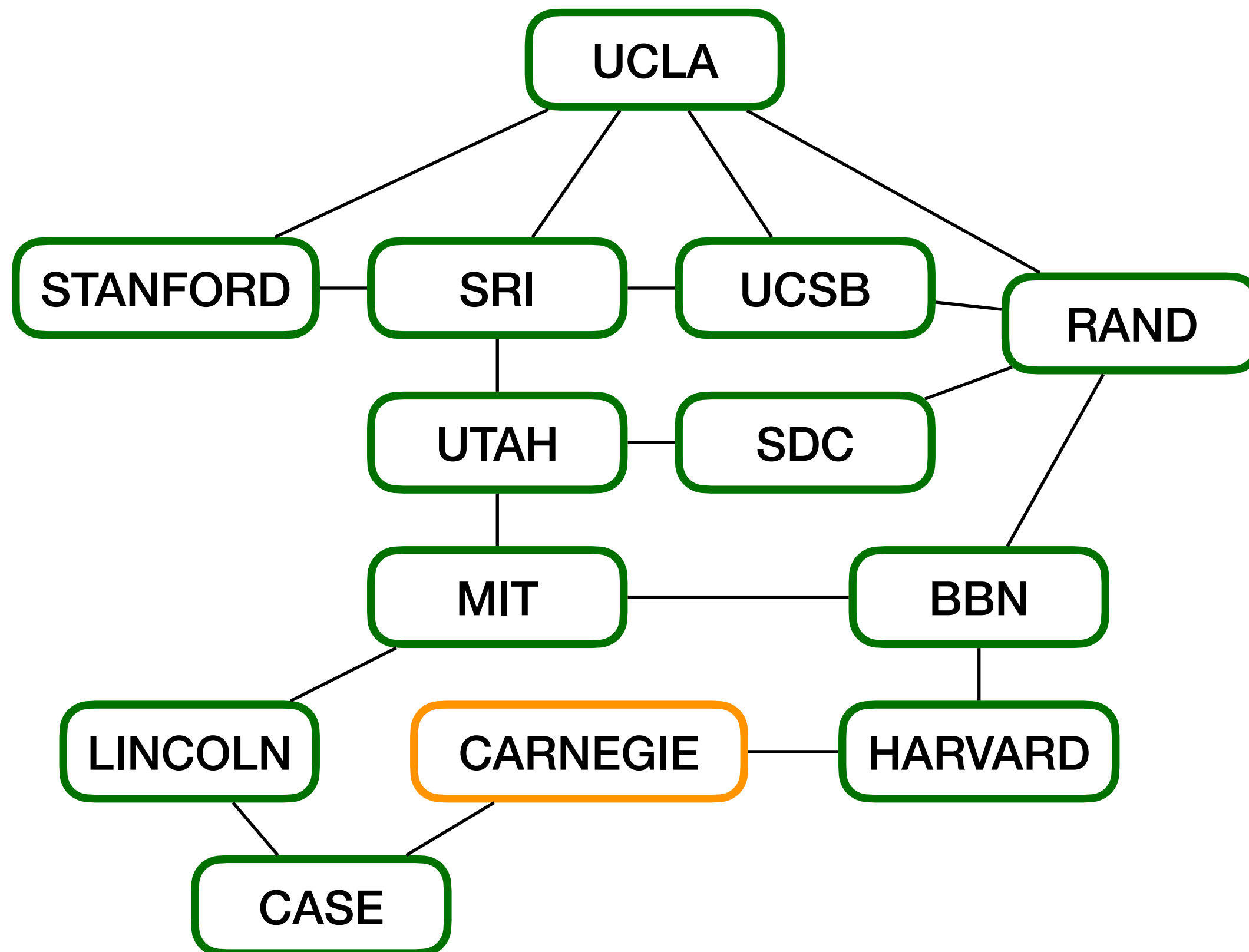
- But don't we want to find the shortest path for the Maze HW?
- Not just the length of the shortest path



UCLA	∞
STANFORD	∞
SRI	∞
UCSB	∞
RAND	∞
UTAH	∞
SDC	∞
MIT	∞
BBN	∞
LINCOLN	∞
CARNEGIE	0
HARVARD	∞
CASE	∞

BFS and Pathfinding

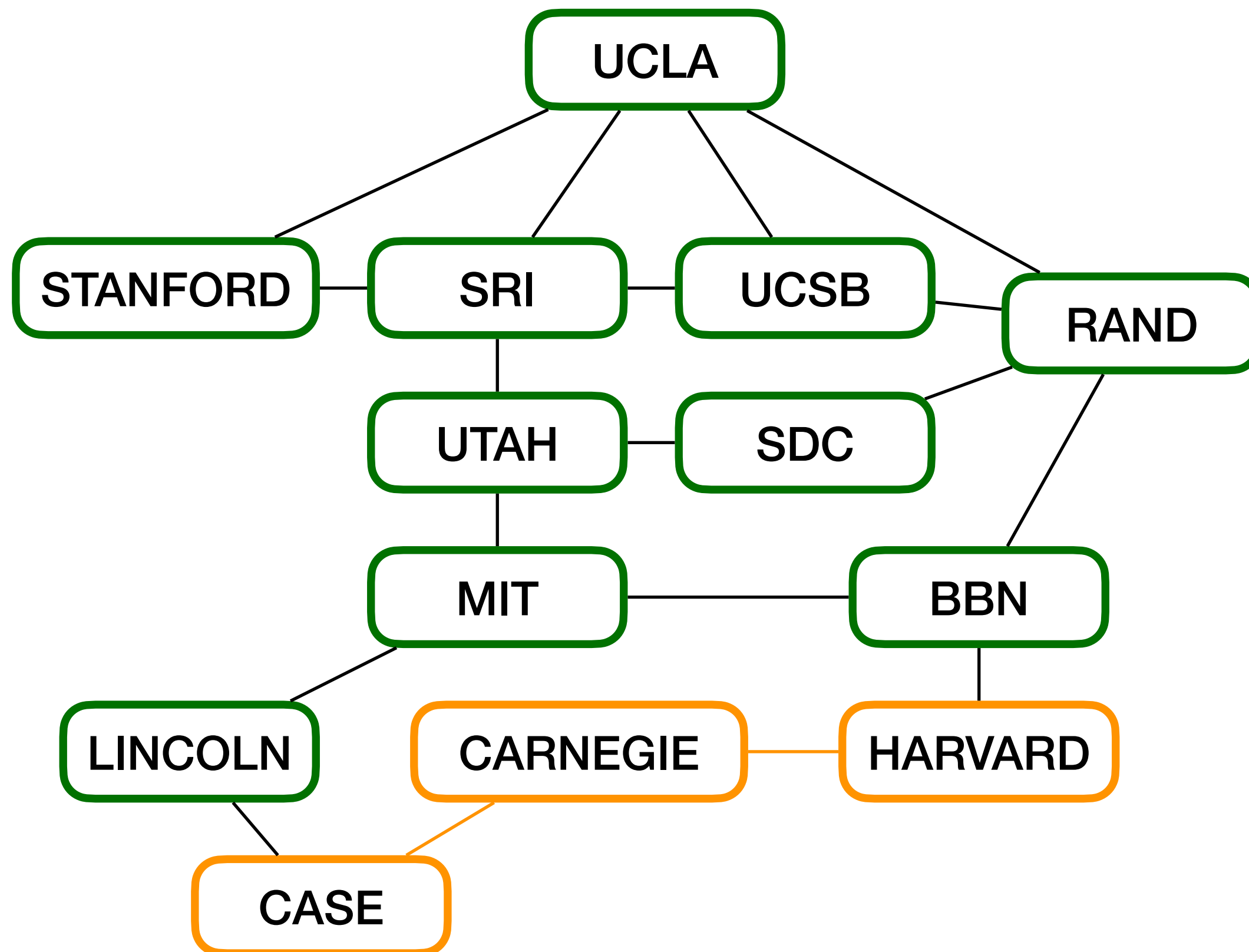
- Instead of tracking the distance, track the node that discovered each node



UCLA	unexplored
STANFORD	unexplored
SRI	unexplored
UCSB	unexplored
RAND	unexplored
UTAH	unexplored
SDC	unexplored
MIT	unexplored
BBN	unexplored
LINCOLN	unexplored
CARNEGIE	<START>
HARVARD	unexplored
CASE	unexplored

BFS and Pathfinding

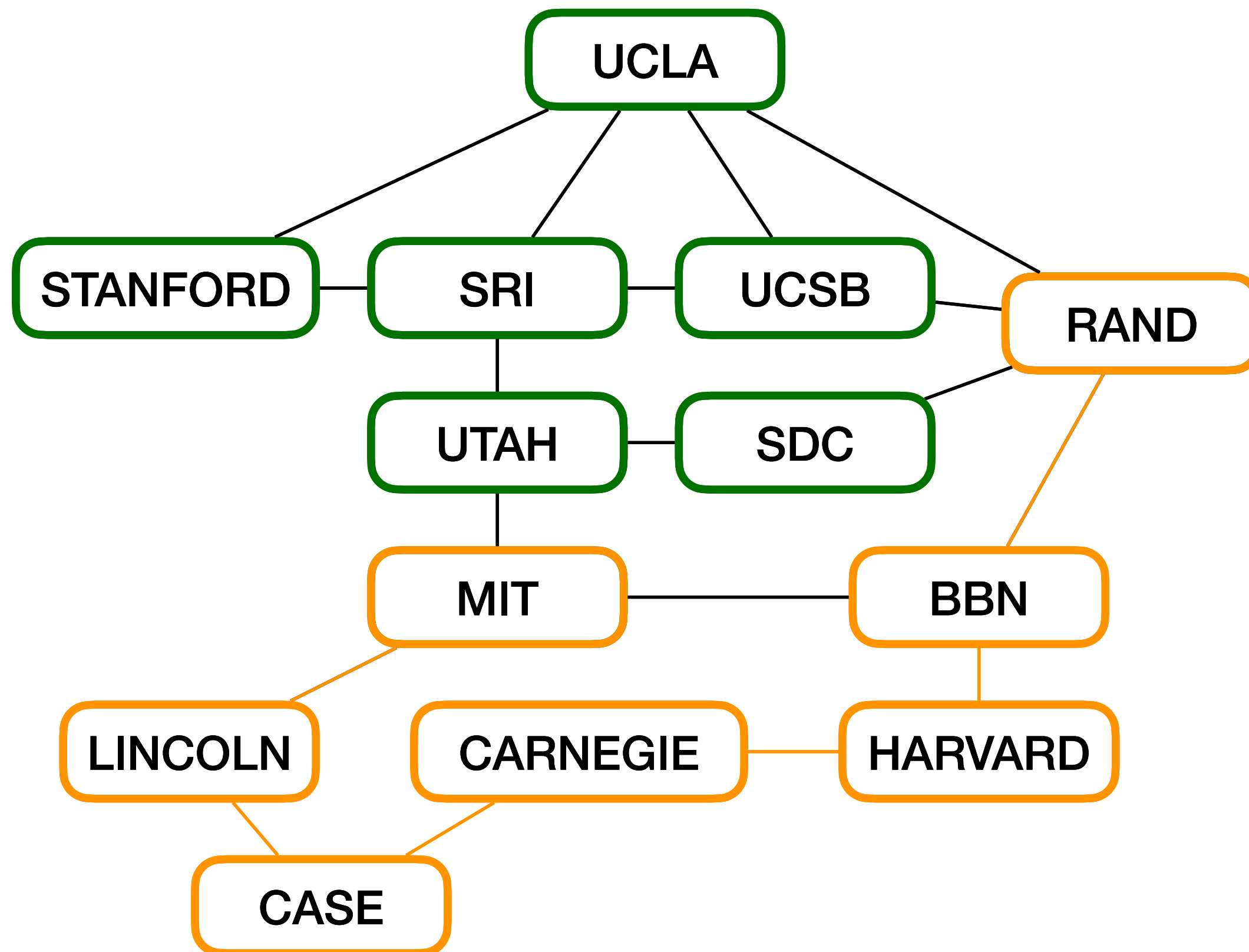
- Now each node remembers how it was reached



UCLA	unexplored
STANFORD	unexplored
SRI	unexplored
UCSB	unexplored
RAND	unexplored
UTAH	unexplored
SDC	unexplored
MIT	unexplored
BBN	unexplored
LINCOLN	unexplored
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

BFS and Pathfinding

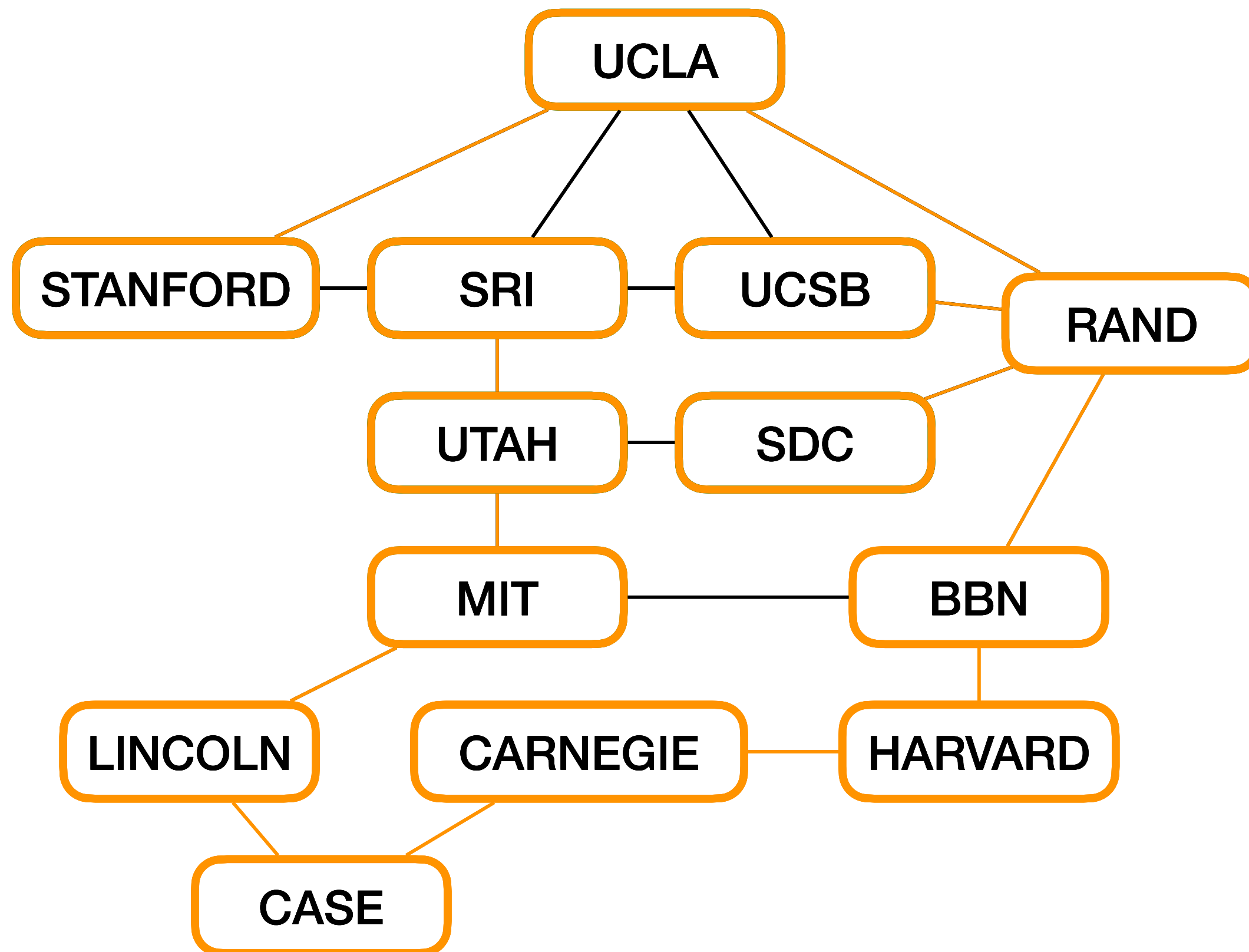
- Repeat at each step



UCLA	unexplored
STANFORD	unexplored
SRI	unexplored
UCSB	unexplored
RAND	BBN
UTAH	unexplored
SDC	unexplored
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

BFS and Pathfinding

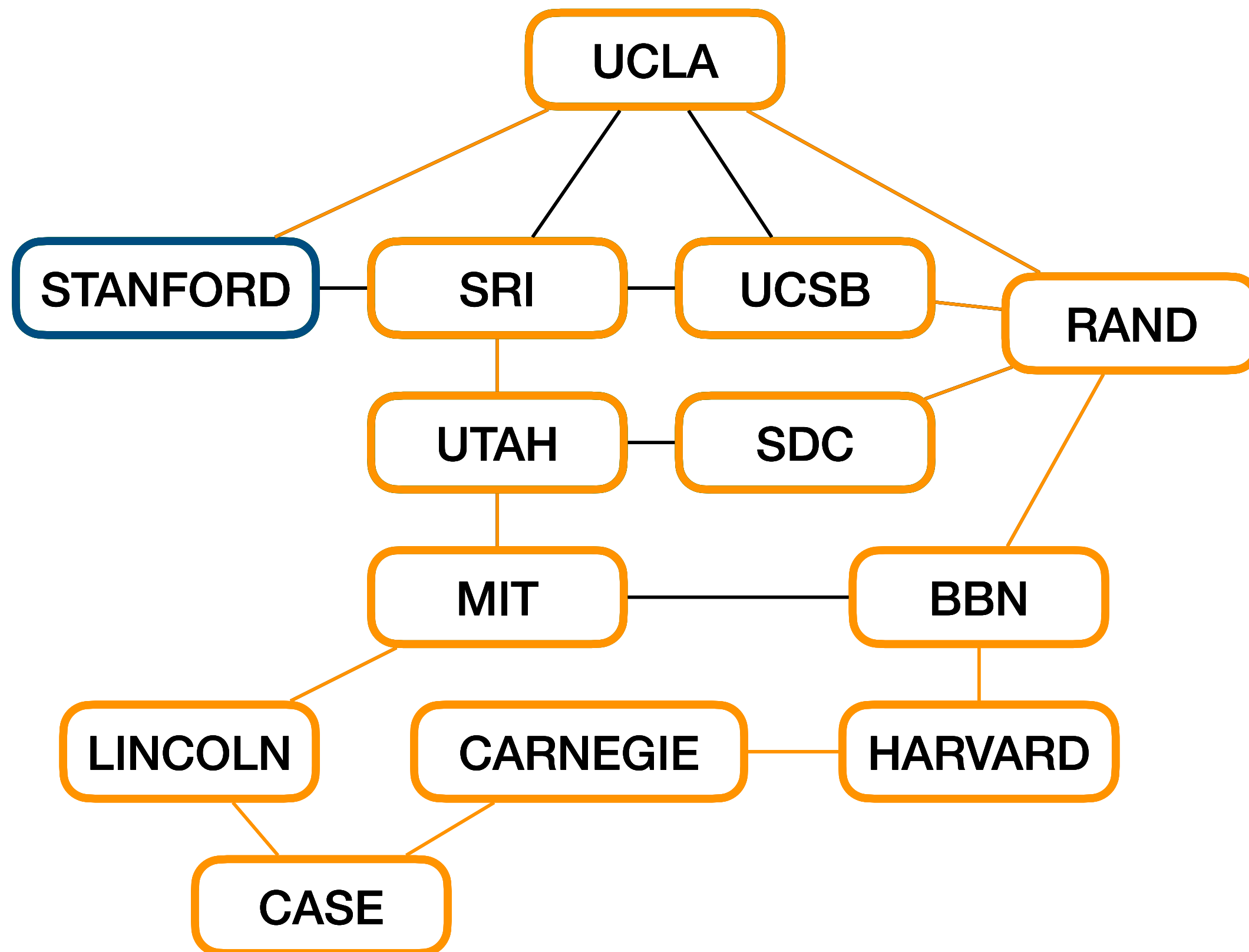
- At the end of the algorithm you'll know how each node was discovered



UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

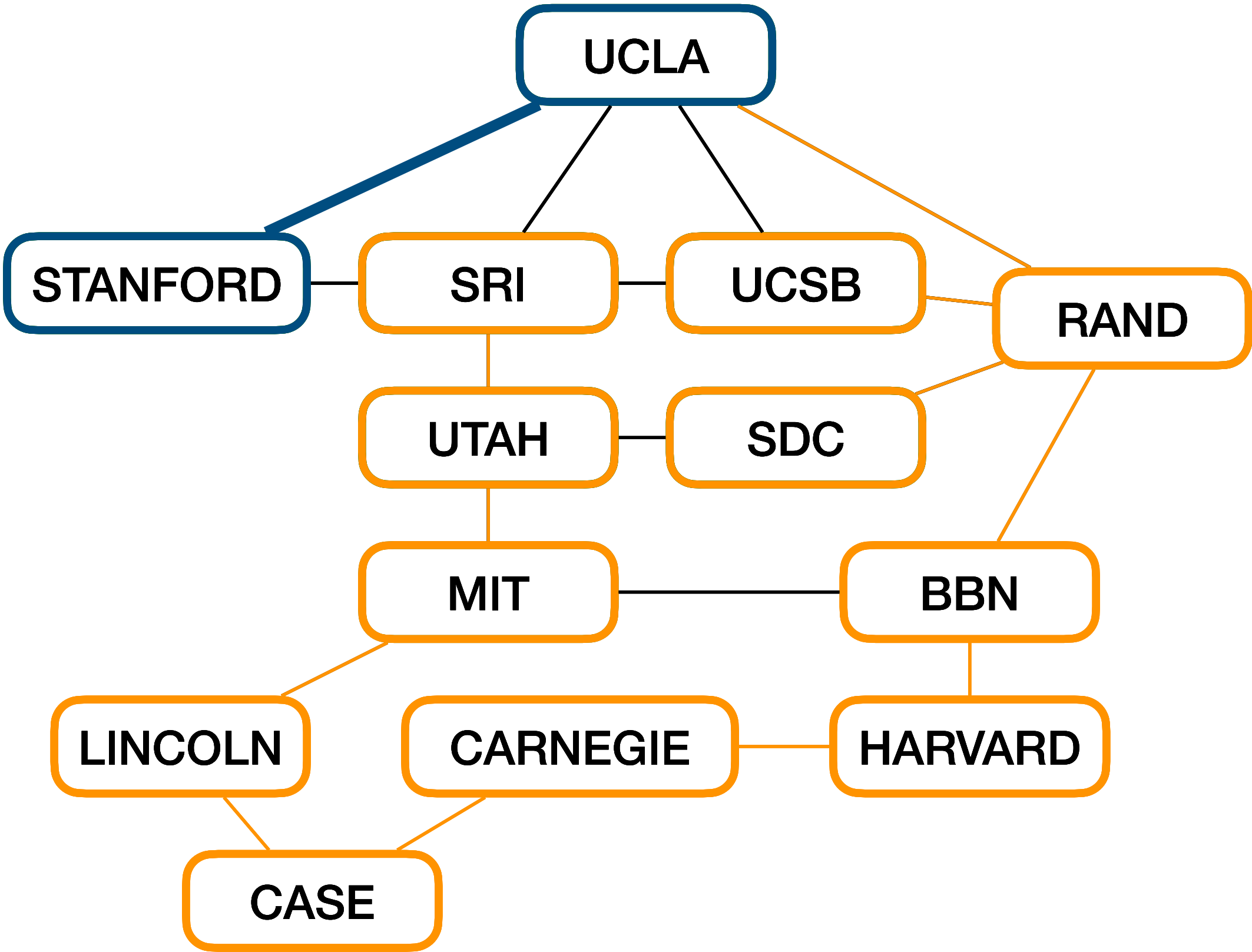
BFS and Pathfinding

- Work backwards to build the shortest path
- Find path from CARNEGIE to STANFORD



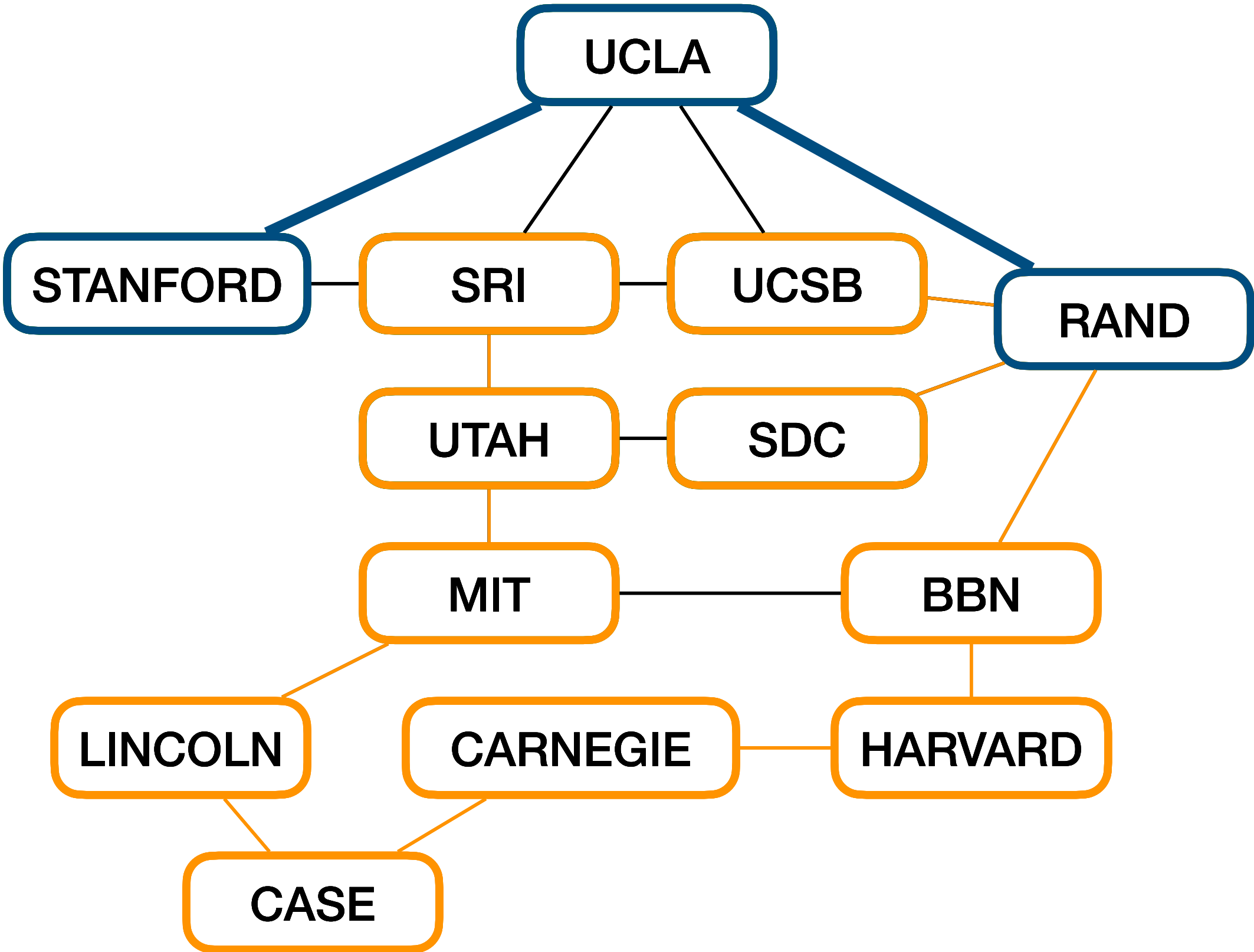
UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

BFS and Pathfinding



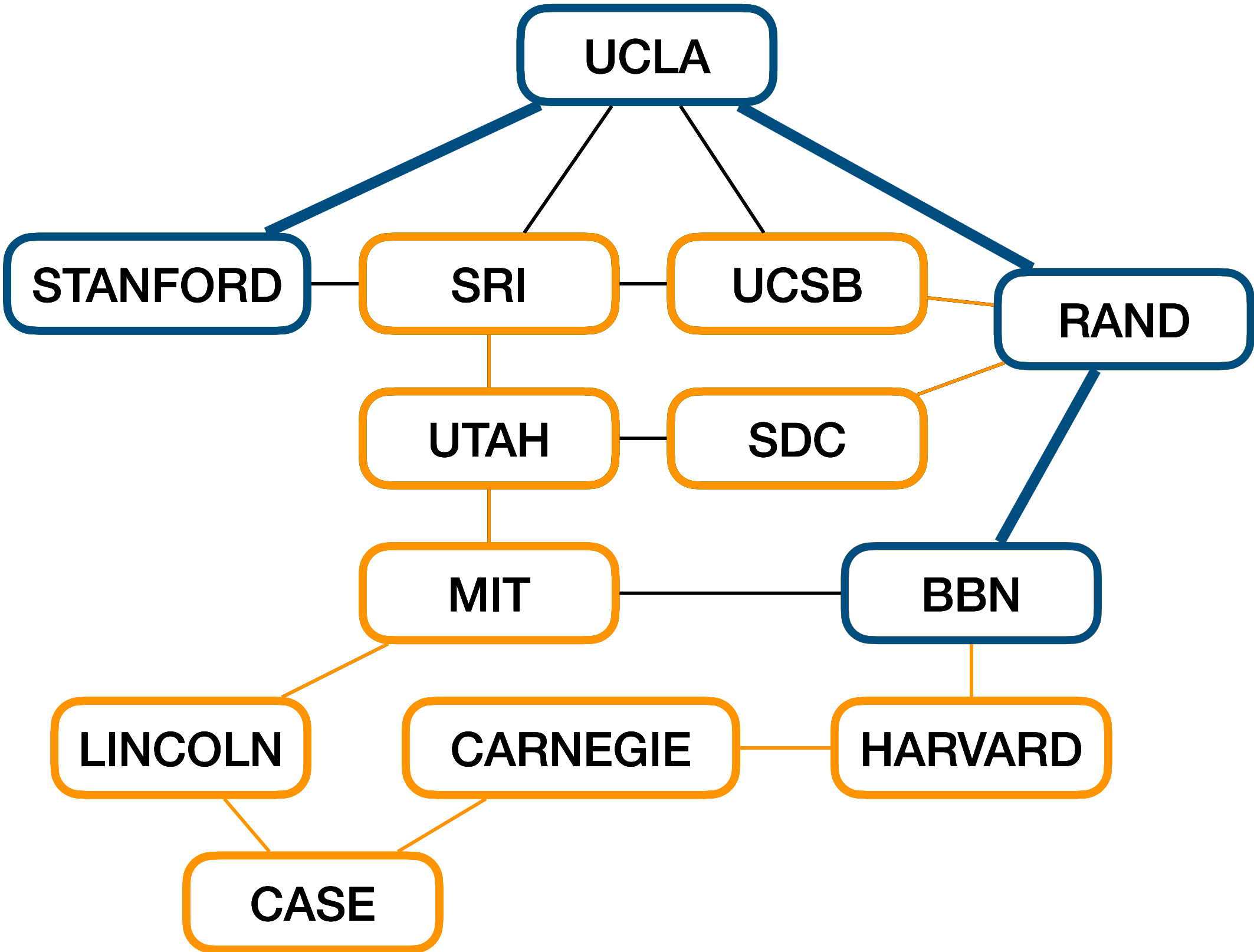
UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

BFS and Pathfinding



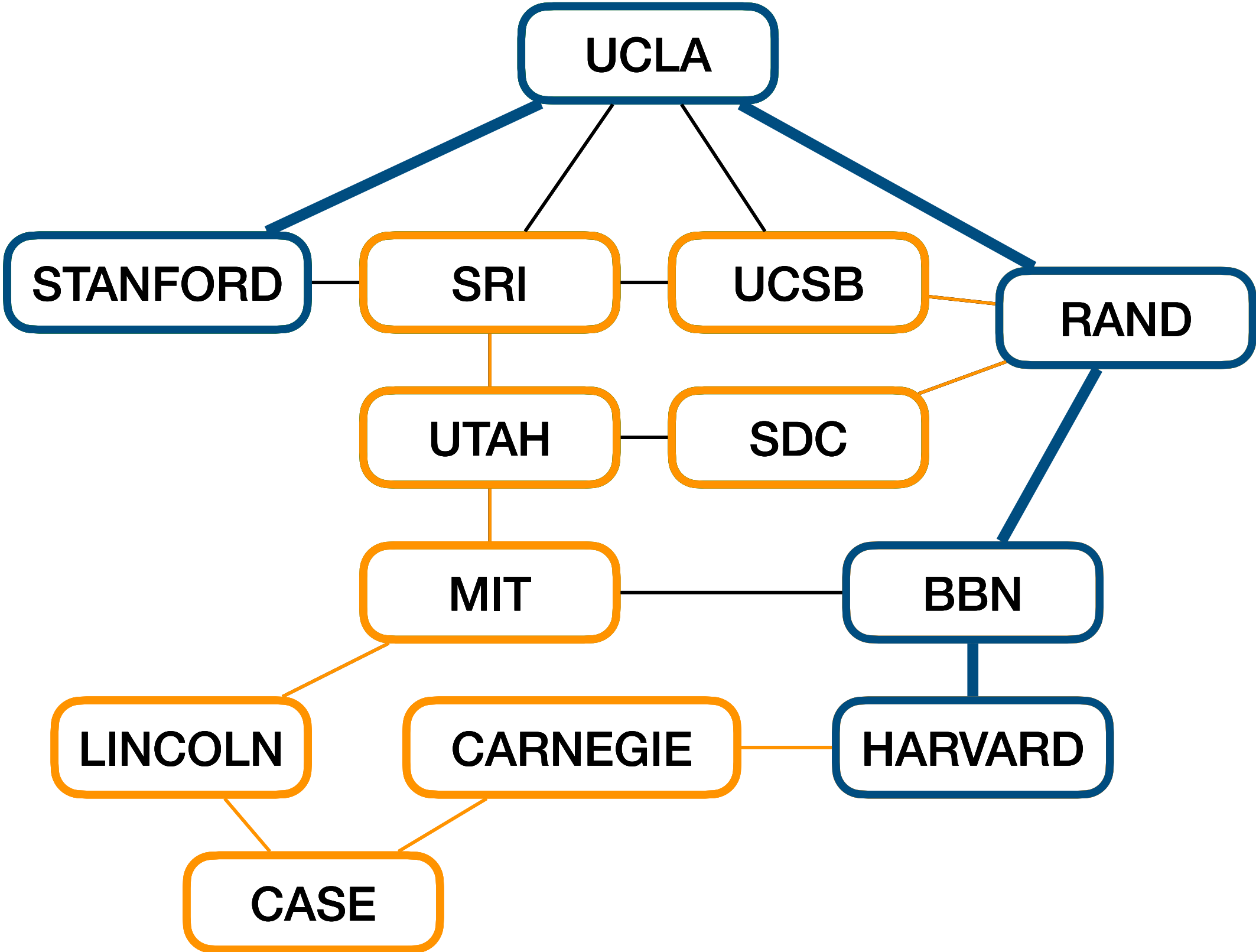
UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

BFS and Pathfinding



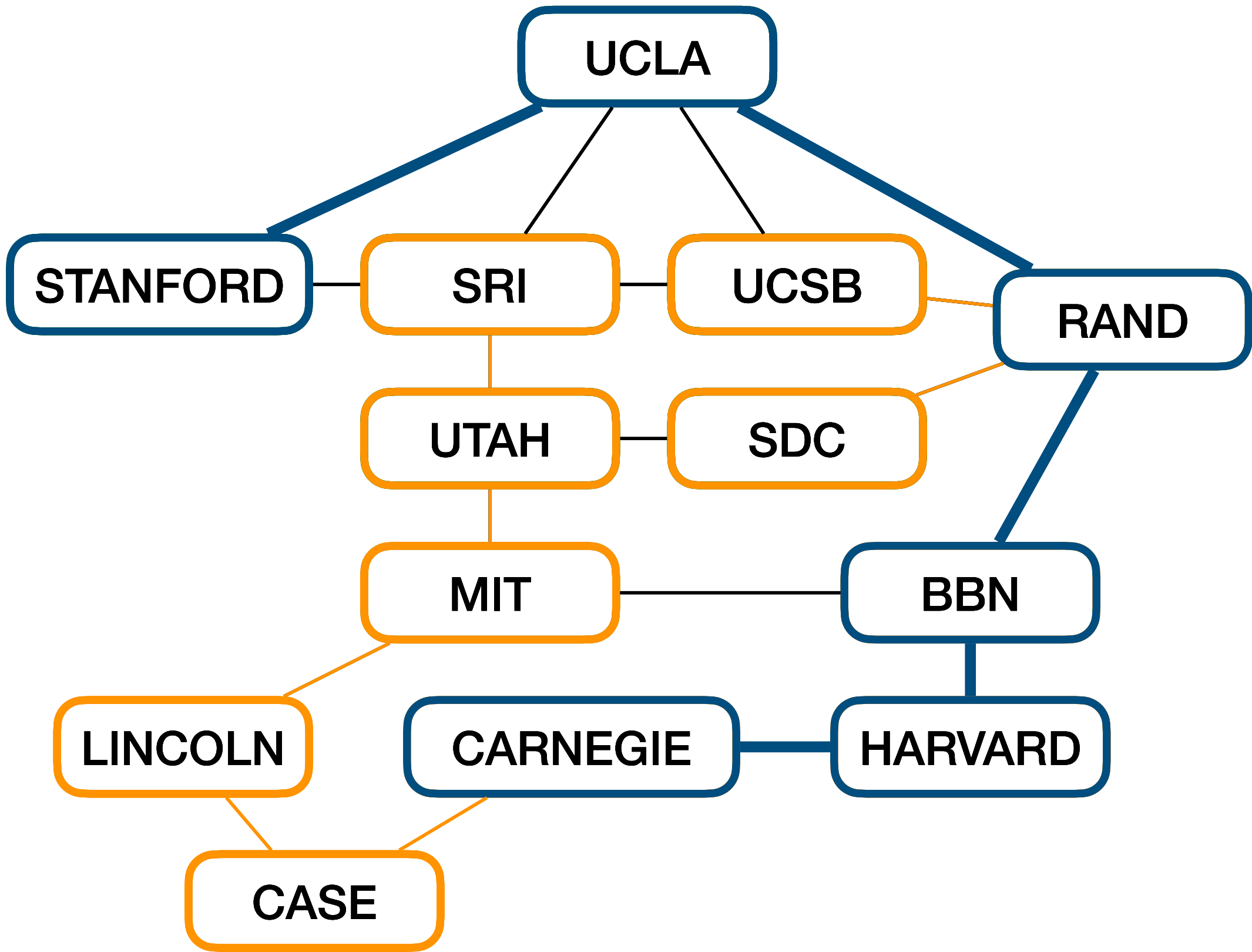
UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

BFS and Pathfinding



UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

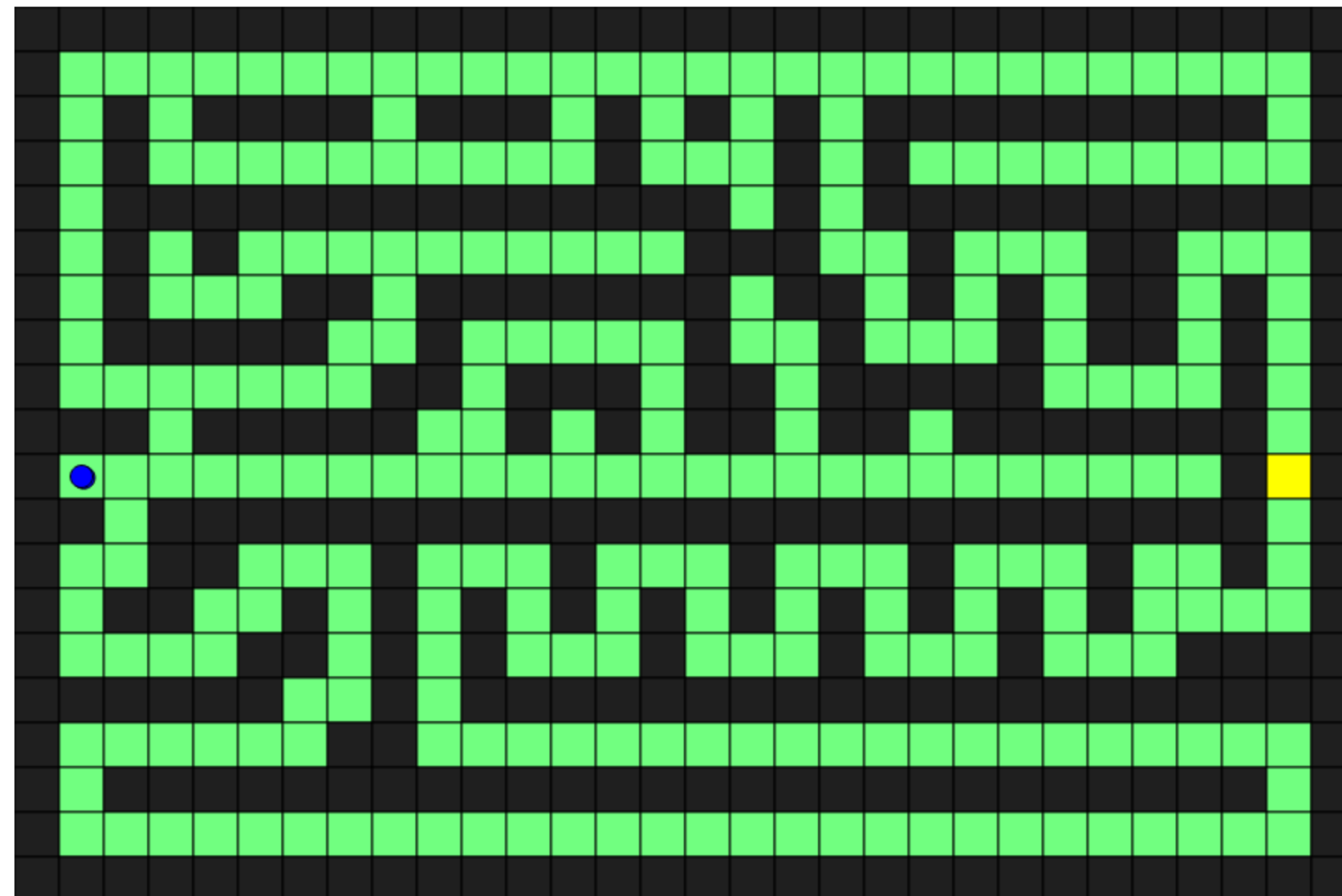
BFS and Pathfinding



UCLA	RAND
STANFORD	UCLA
SRI	UTAH
UCSB	RAND
RAND	BBN
UTAH	MIT
SDC	RAND
MIT	LINCOLN
BBN	HARVARD
LINCOLN	CASE
CARNEGIE	<START>
HARVARD	CARNEGIE
CASE	CARNEGIE

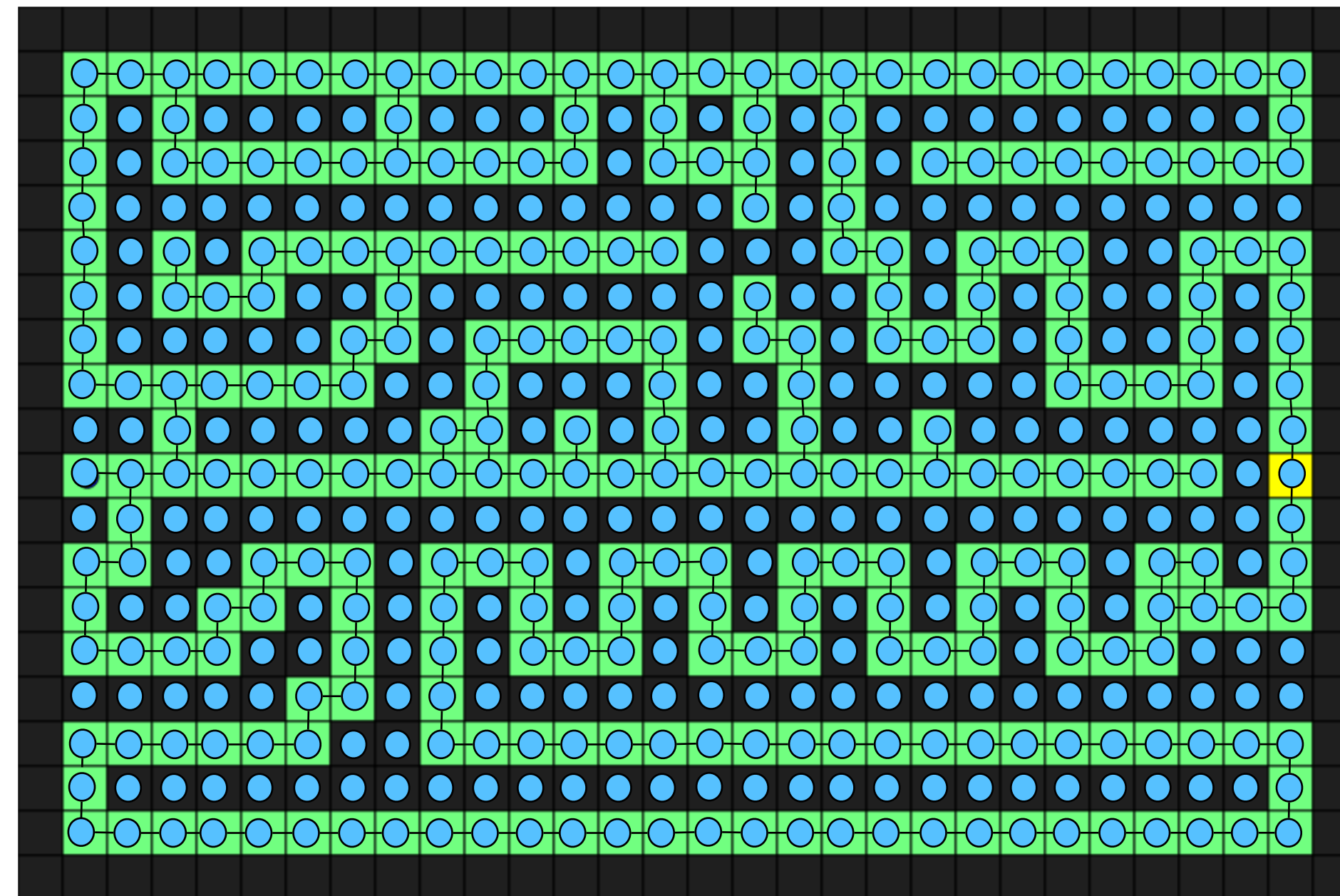
But we have to find paths in a maze

How do graphs help with this?

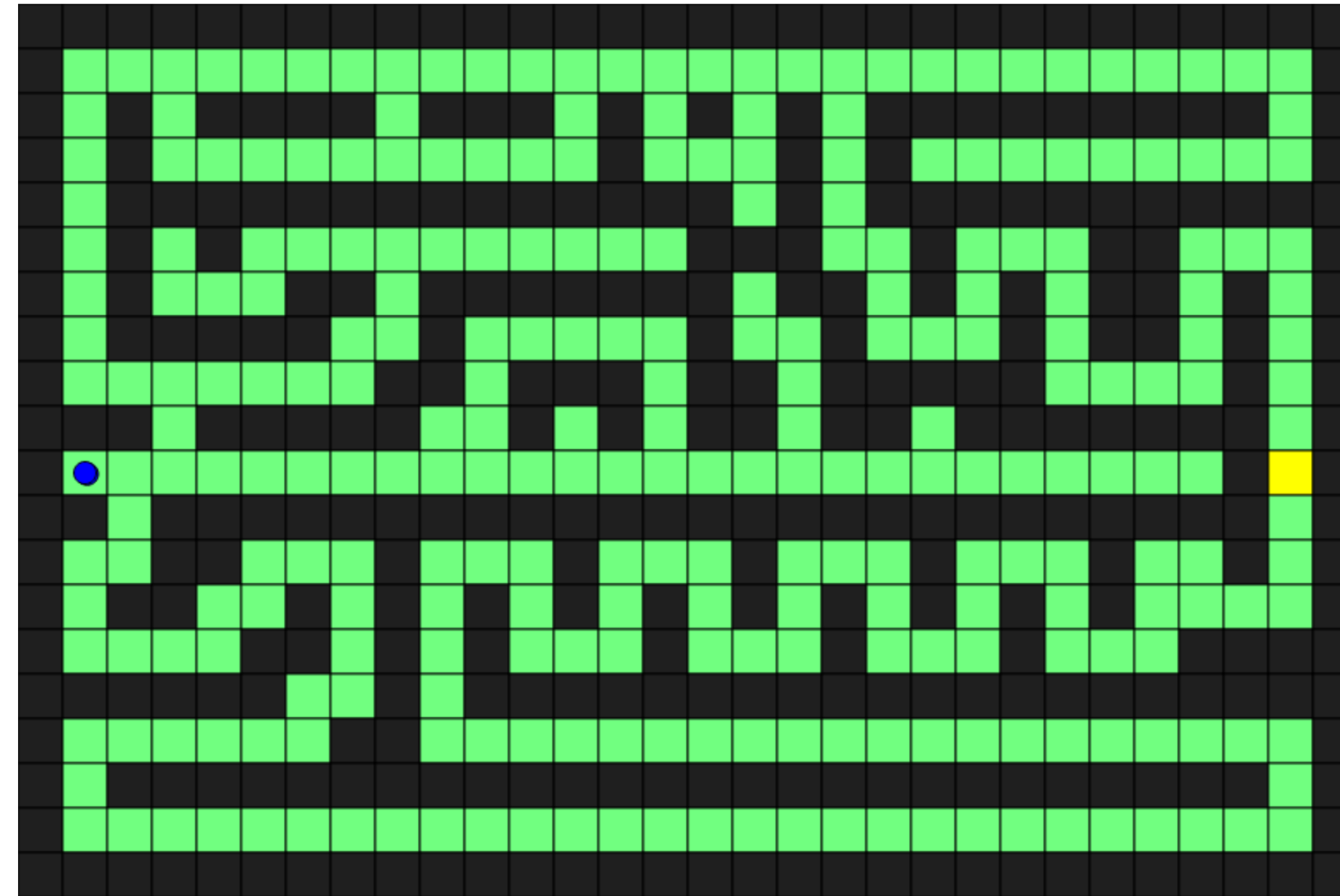


Pathfinding on a Grid

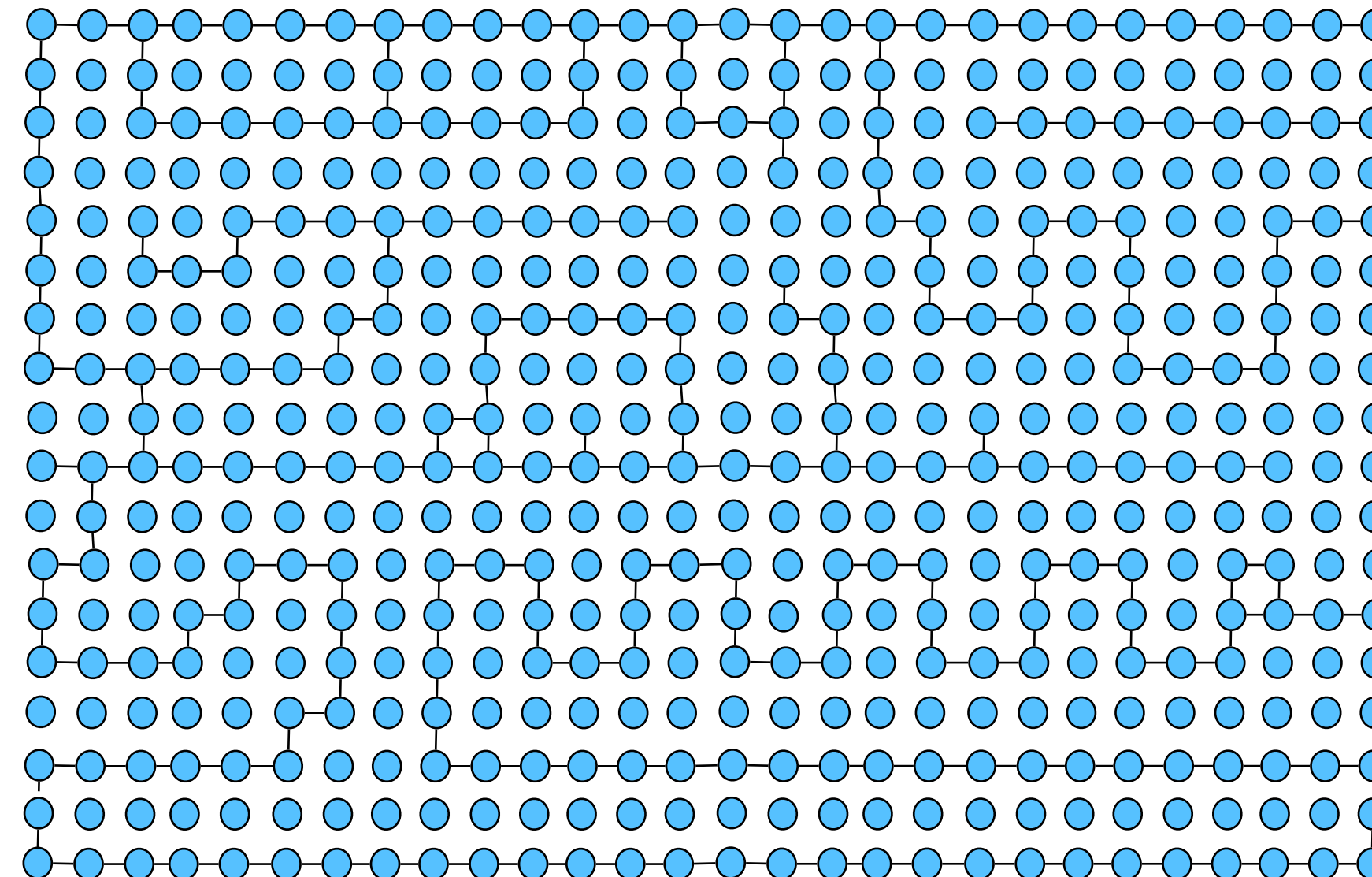
- Convert the maze to a graph
- Run BFS starting at the tile containing the maze runner
- Backtrack from the goal tile to build the path



We see:



Computer sees:



Lecture Question

Task: Find the distance between two nodes in a graph

- In the week9.Graph class
 - Write a method named distance that takes two node indices (Ints) and returns the distance between the two nodes
 - You may assume the two input nodes are connected

Testing: In a package named "tests" create a class named "TestDistance" as a test suite that tests the functionality listed above