

A 40×40 Four-Neighbor Time-Based In-Memory Computing Graph ASIC Chip Featuring Wavefront Expansion and 2D Gradient Control

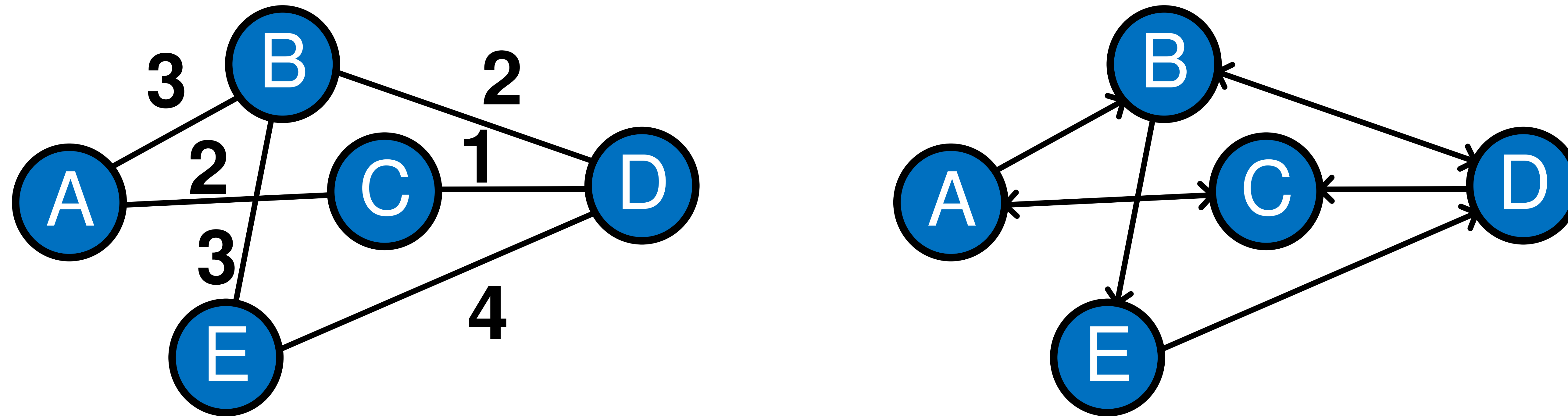
Luke R. Everson, Sachin S. Sapatnekar, Chris H. Kim
University of Minnesota, Minneapolis, MN



Outline

- **Background: Path Planning Algorithms**
- **Time-Based A* ASIC**
 - **Time-Based Primer**
 - **Top Level Design**
 - **Vertex Details**
 - **Edge Details**
- **65nm Test Chip Results**
- **Applications**
- **Conclusion**

Graph Definition



- **Graph: set of “objects” and their “connections”**
- **Formal definition:**
 - $G = (V, E)$, $V = \{v_1, v_2, \dots, v_n\}$, $E = \{e_1, e_2, \dots, e_m\}$
 - V : set of vertices (nodes), E : set of edges (links, arcs)
 - **Directed graph:** $e_k = (v_i, v_j)$
 - **Undirected graph:** $e_k = \{v_i, v_j\}$
 - **Weighted graph:** $w: E \rightarrow \mathbb{R}$, $w(e_k)$ is the “weight” of e_k .

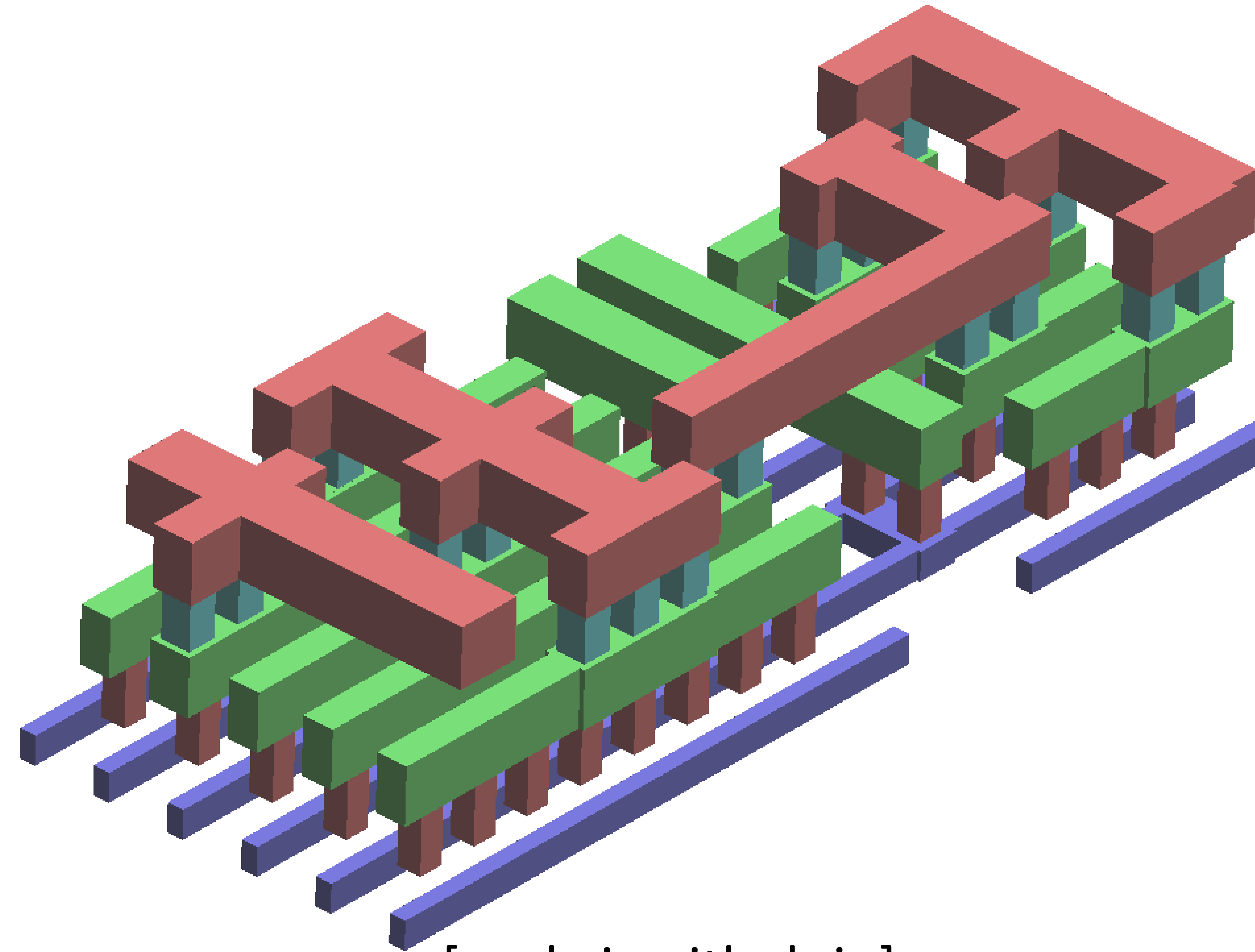
Why Graphs?

Network Analysis



[Wikipedia.org]

Routing/Path Planning



[yuchsia.github.io]

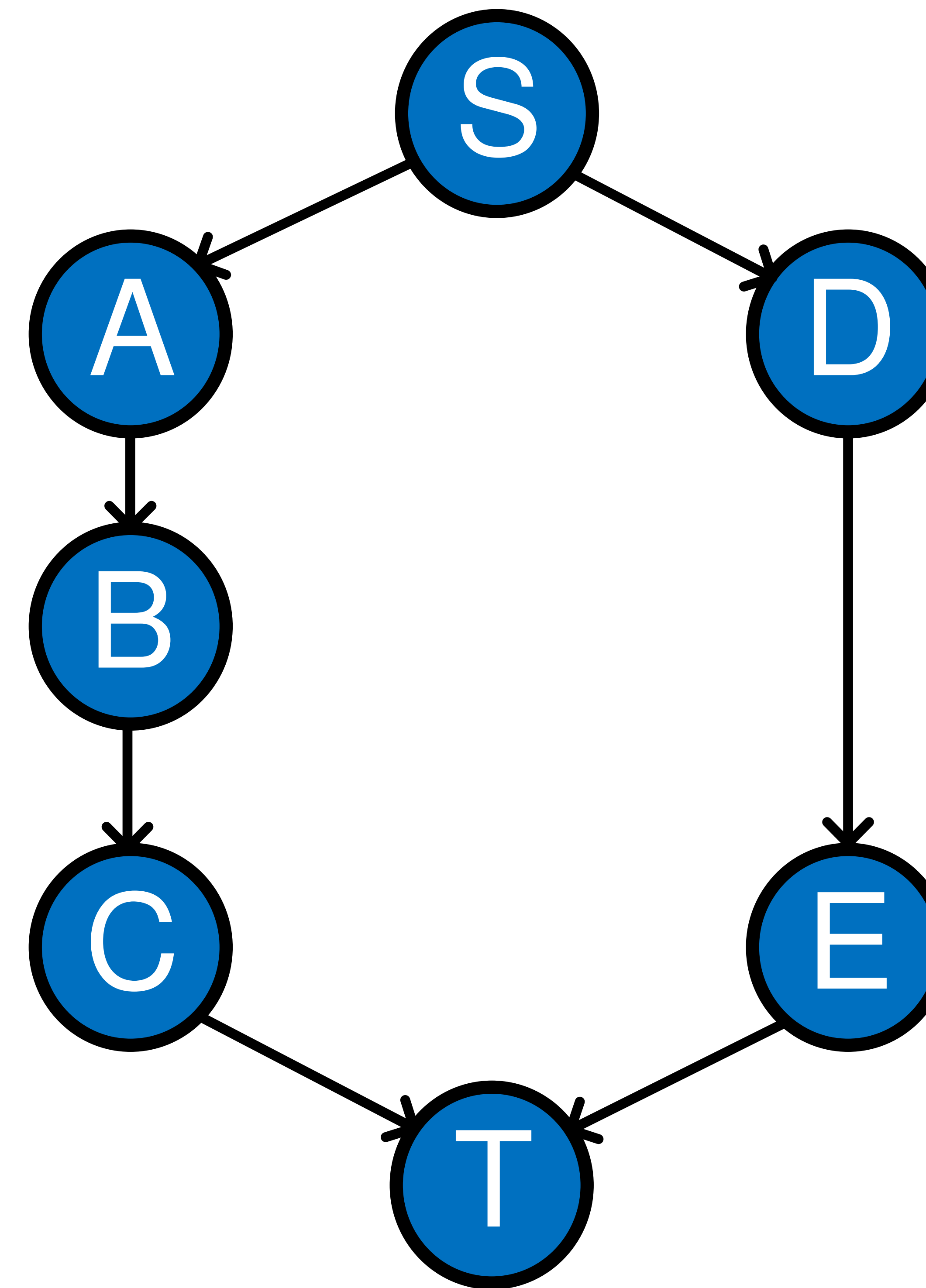
Self-Driving Cars



[techrepublic.com]

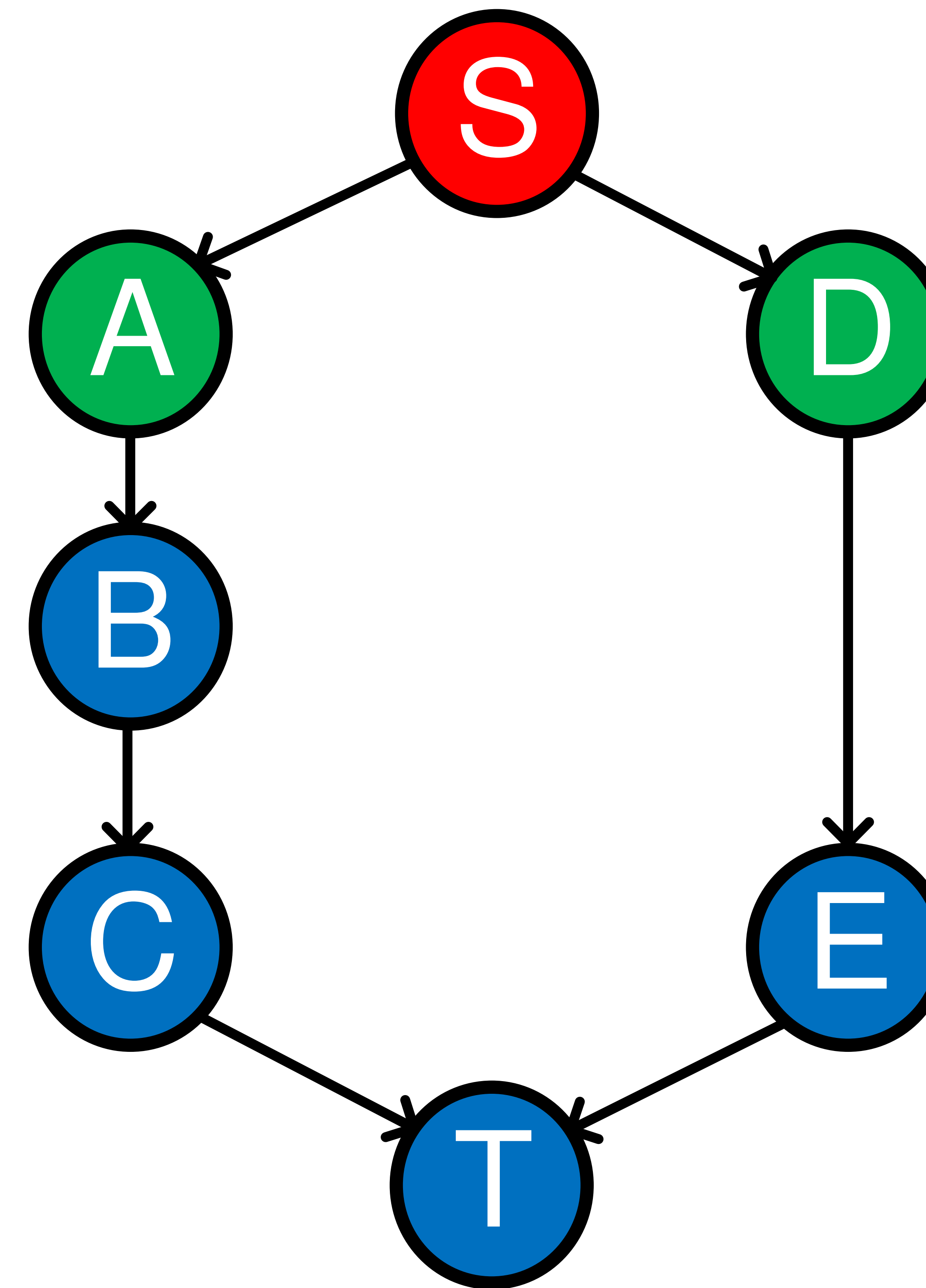
Graph Algorithms- Breadth First Search

- Explores search area equally in all directions
- Adds new neighbors to queue and visits in order added
- Simple to implement in software



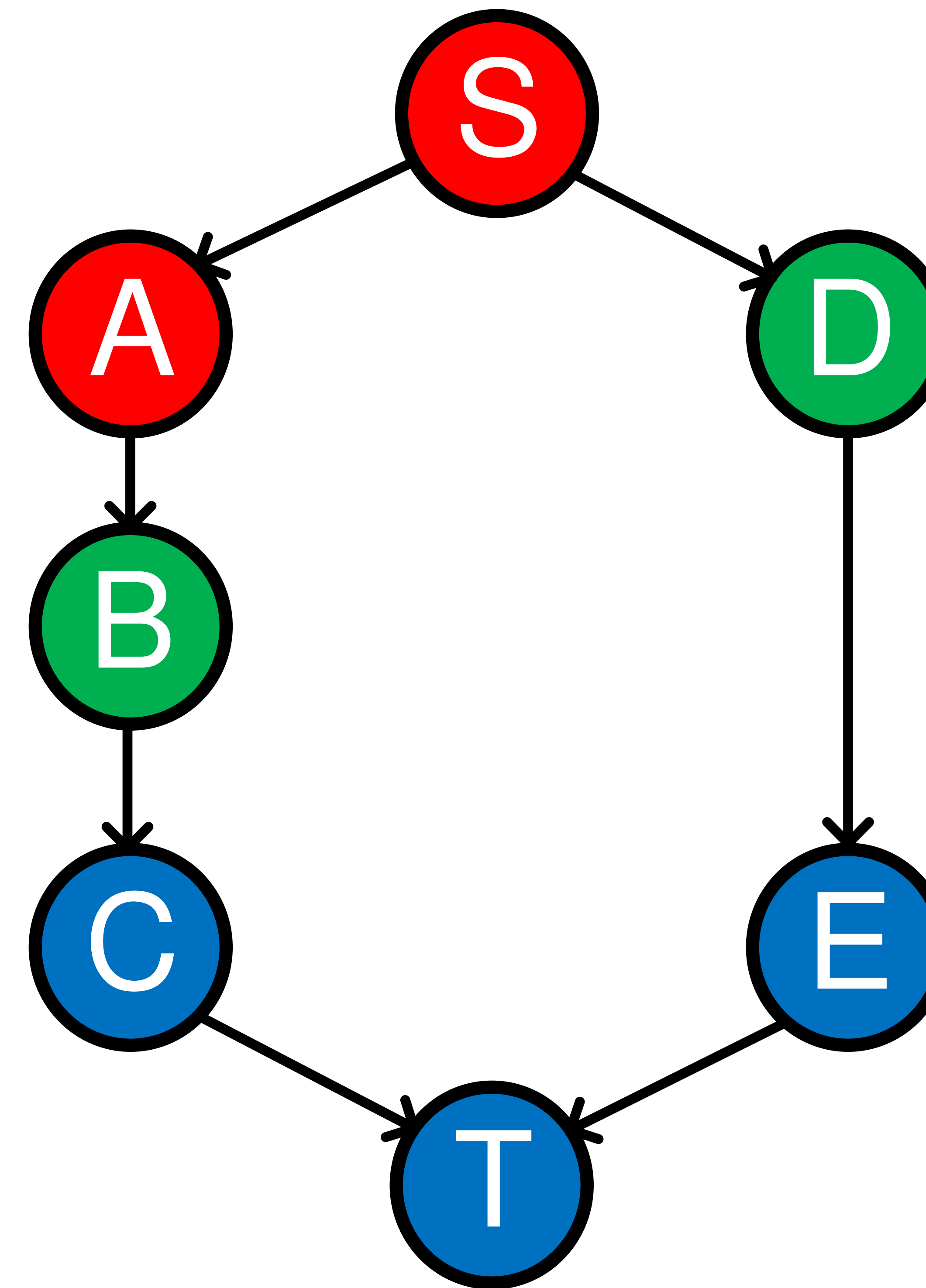
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S}
- Queue = {A,D}



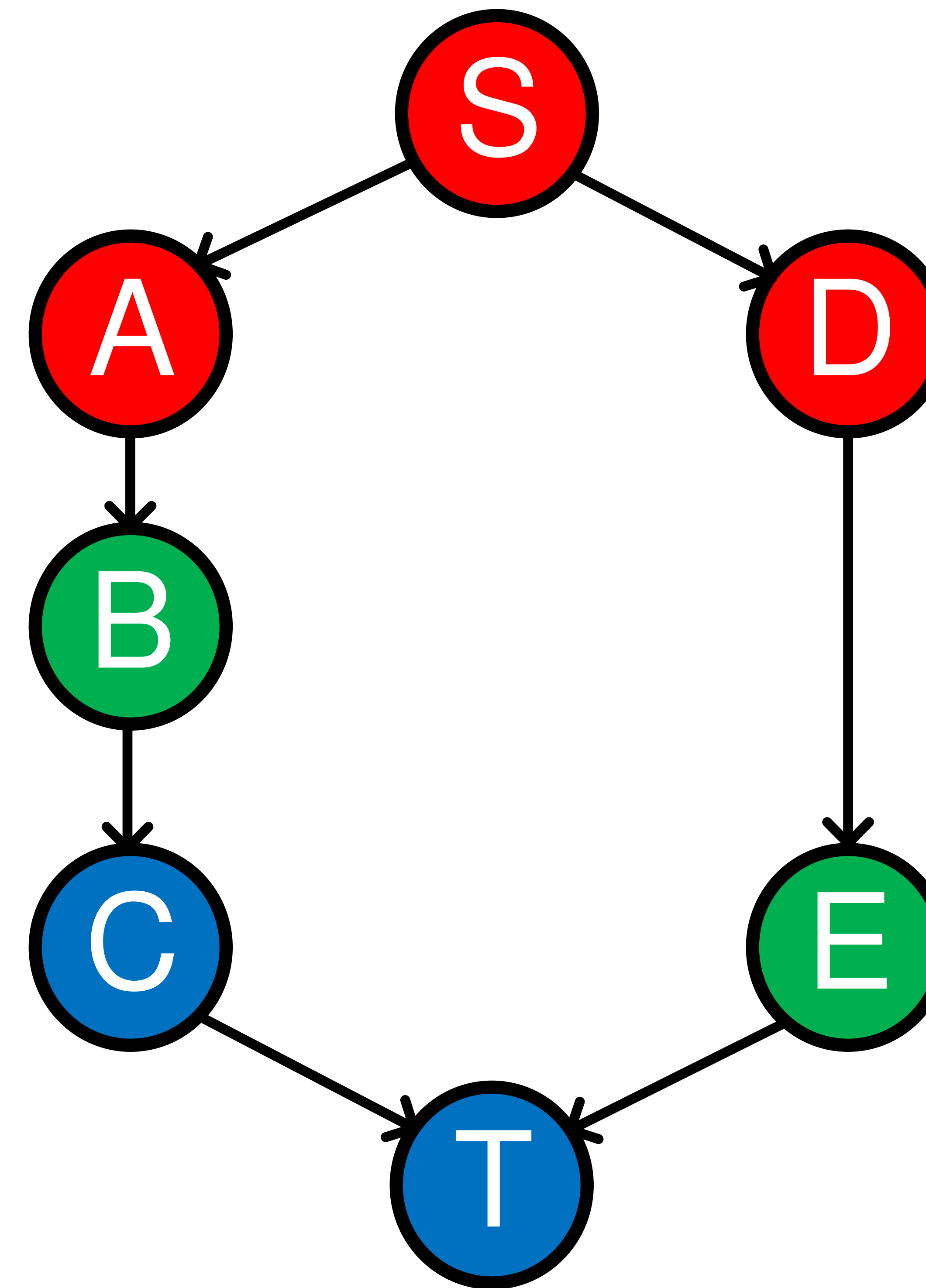
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S,A}
- Queue = {D,B}



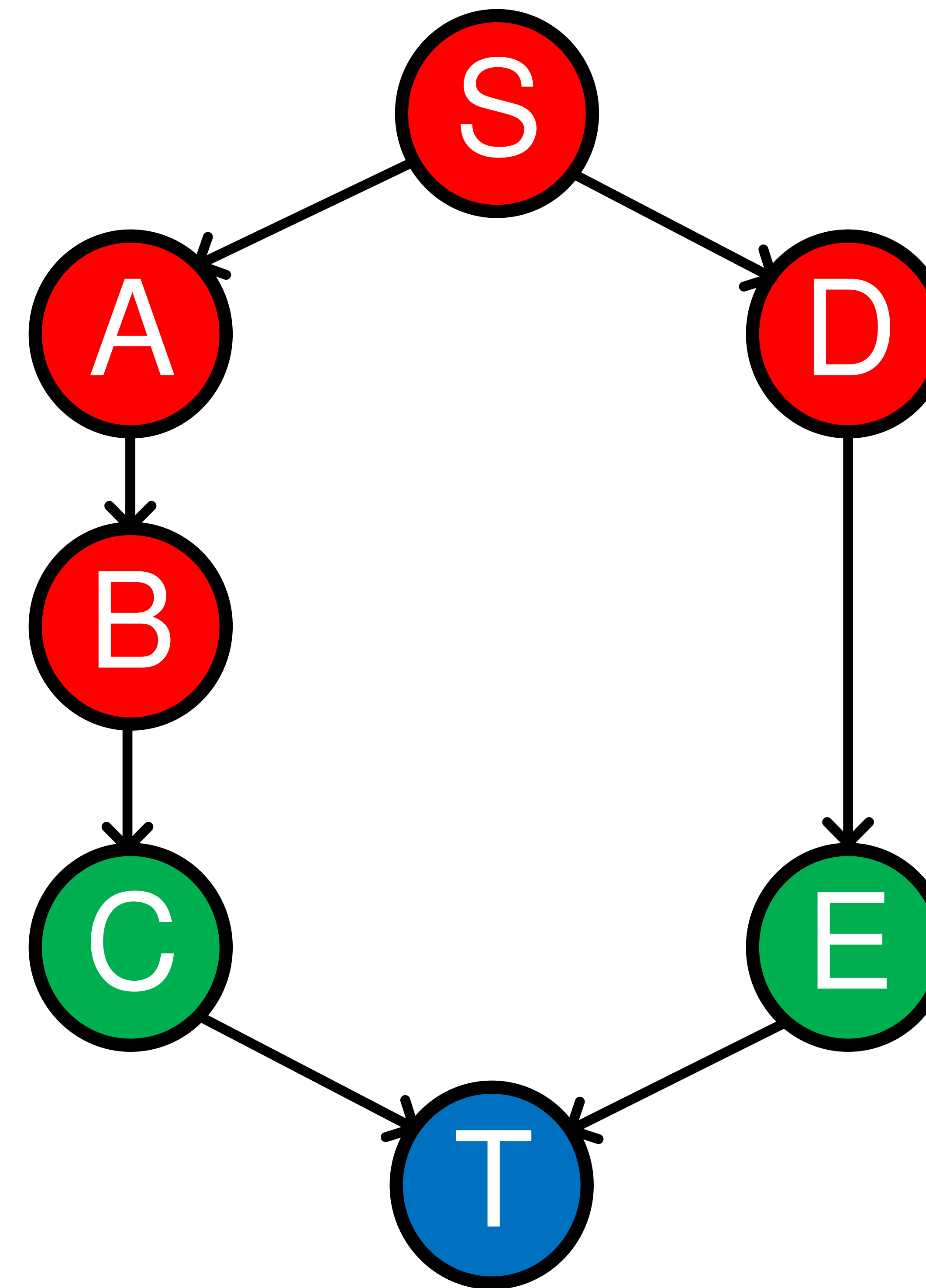
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S,A,D}
- Queue = {B,E}



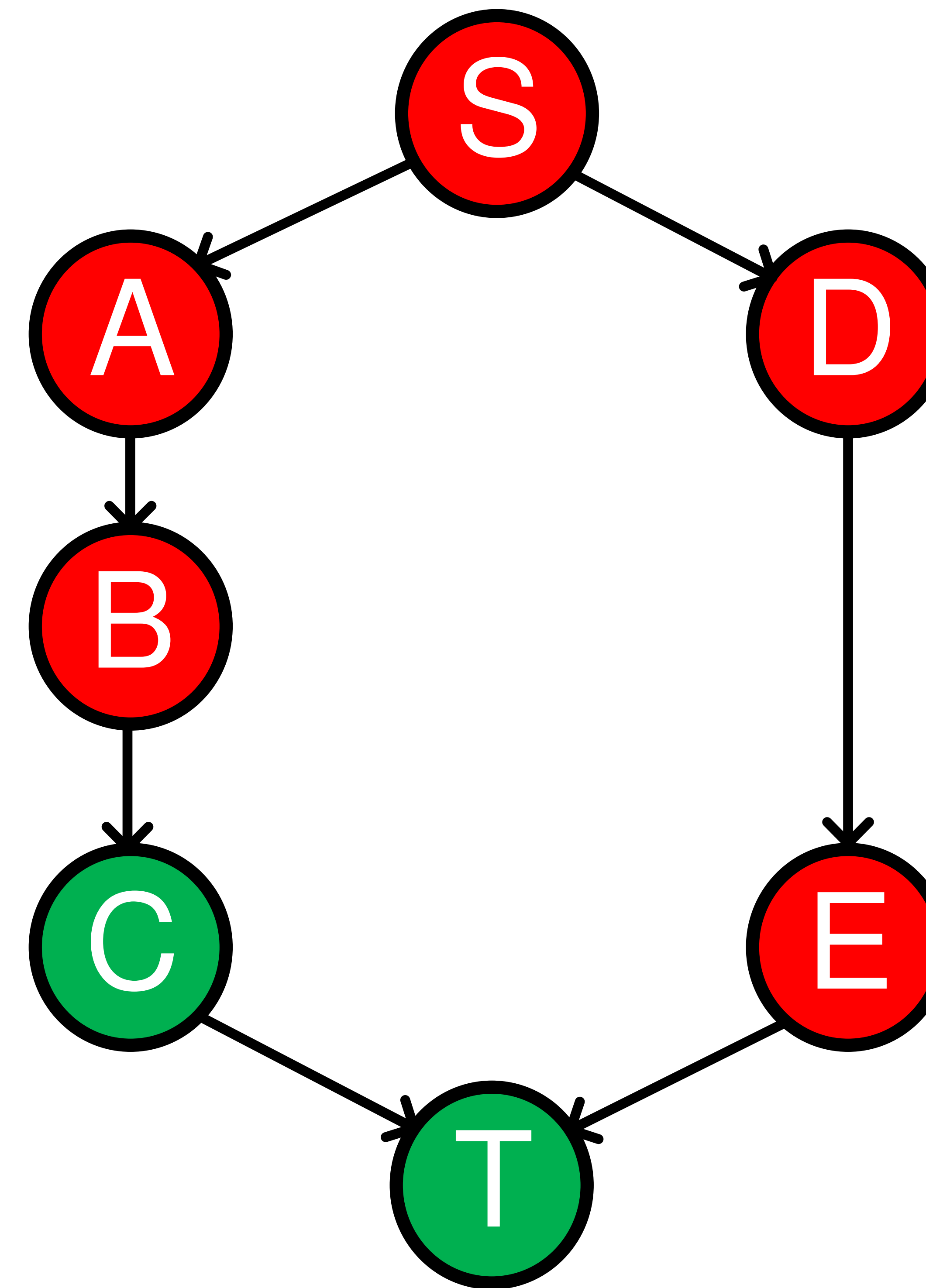
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S,A,D,B}
- Queue = {E,C}



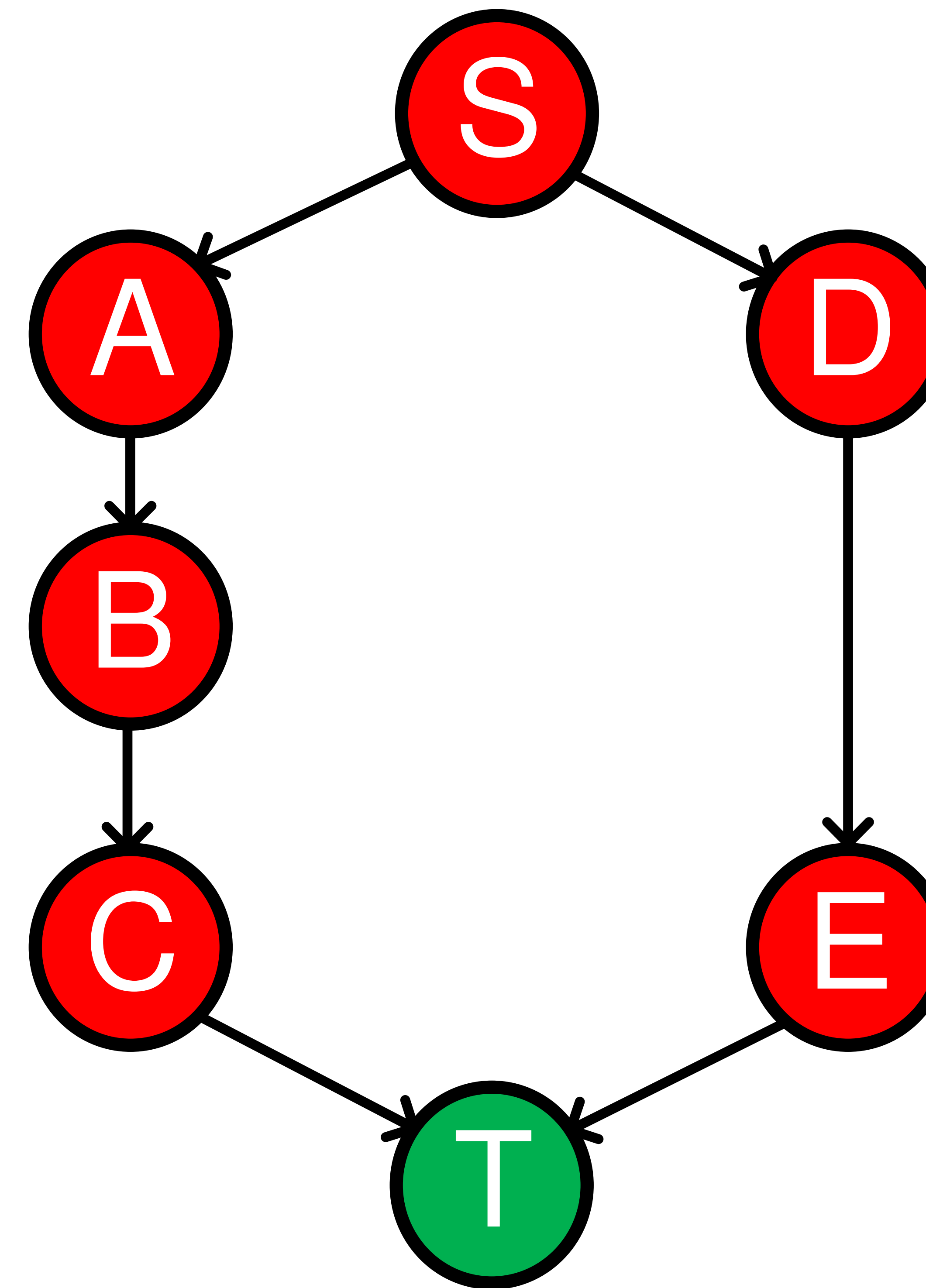
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S,A,D,B,E}
- Queue = {C,T}



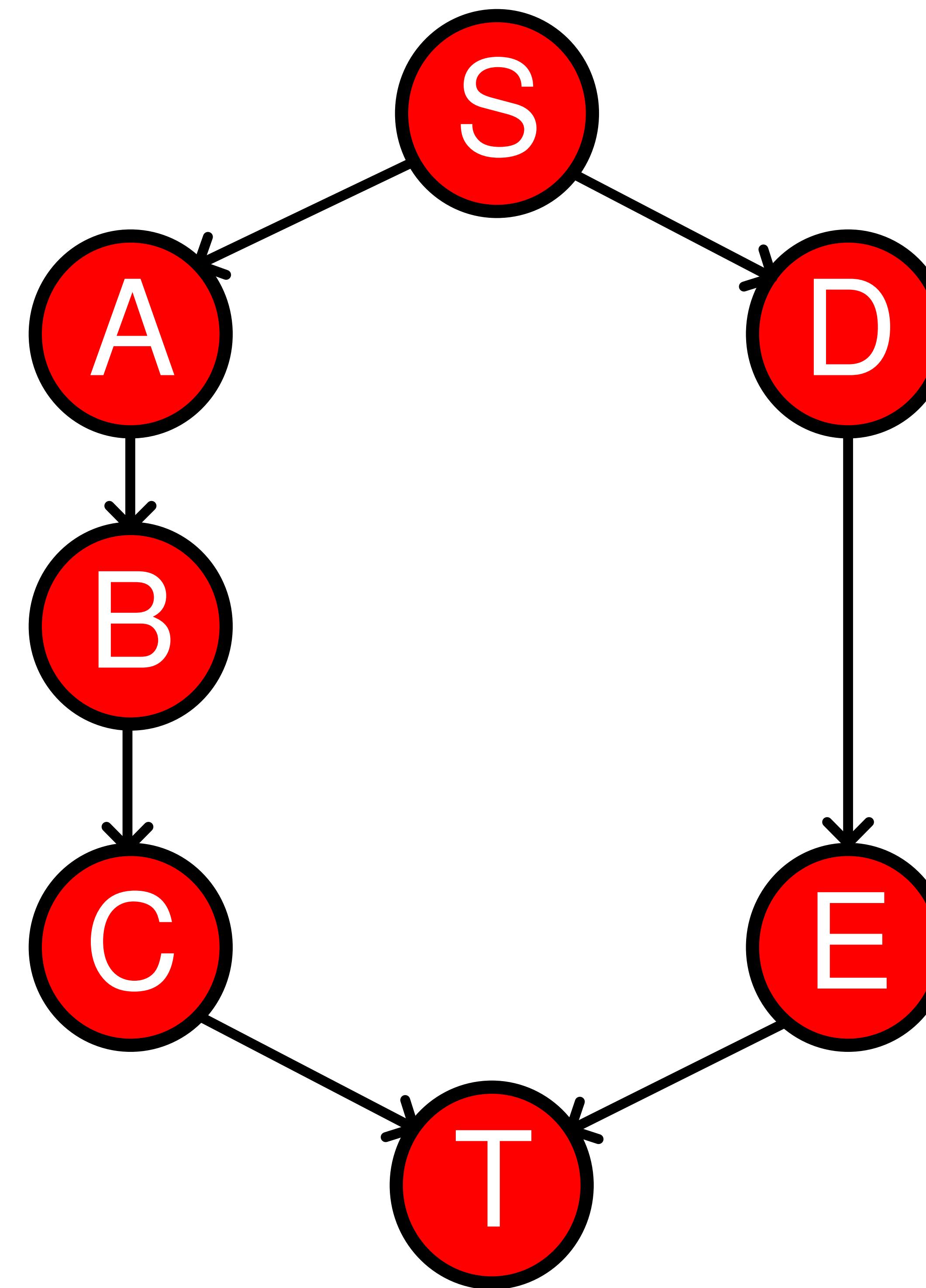
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S,A,D,B,E,C}
- Queue = {T}



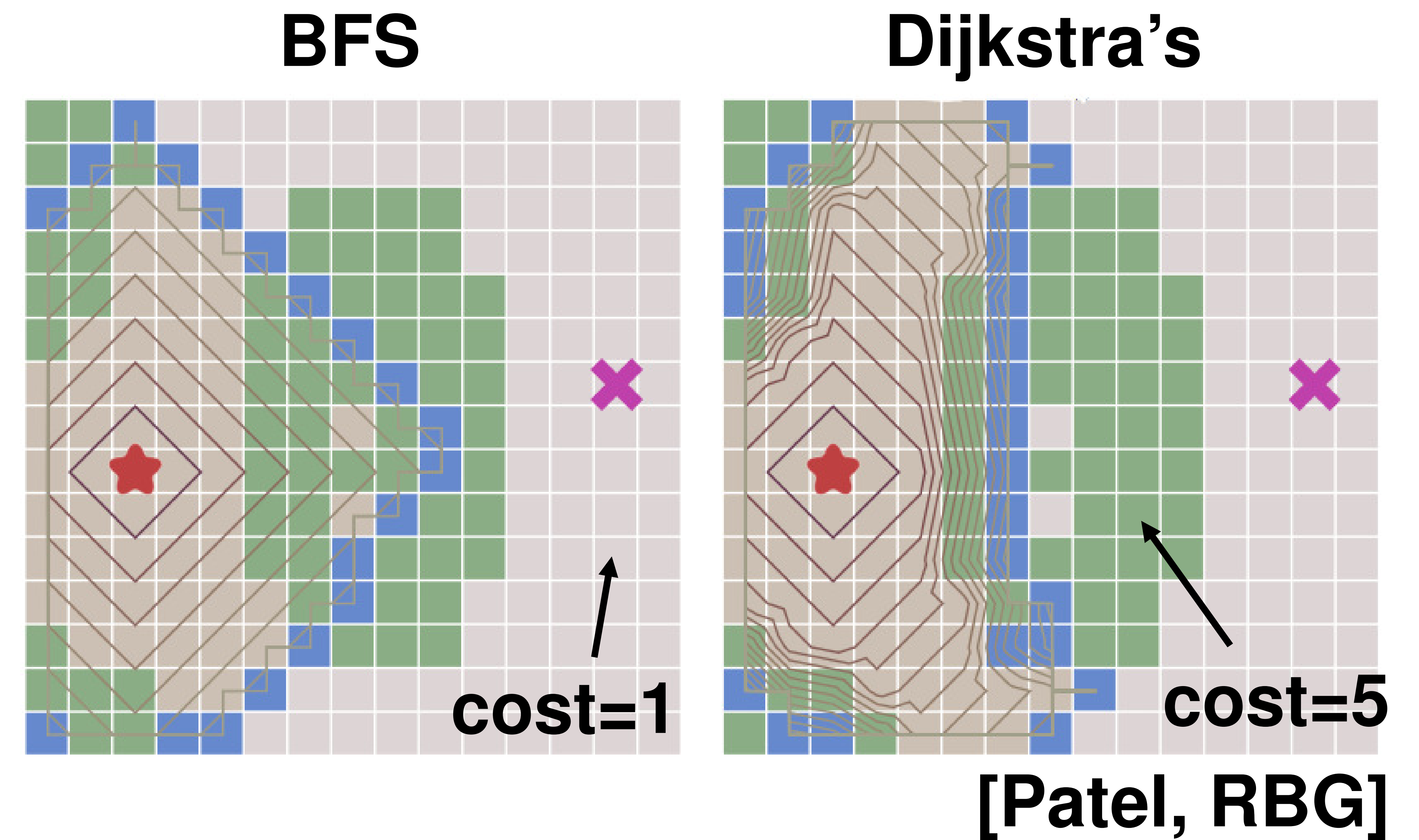
Graph Algorithms- Breadth First Search

- Explore entire graph from Source (S)
- Visited = {S,A,D,B,E,C,T}
- Queue = {}



Graph Algorithms- Dijkstra's

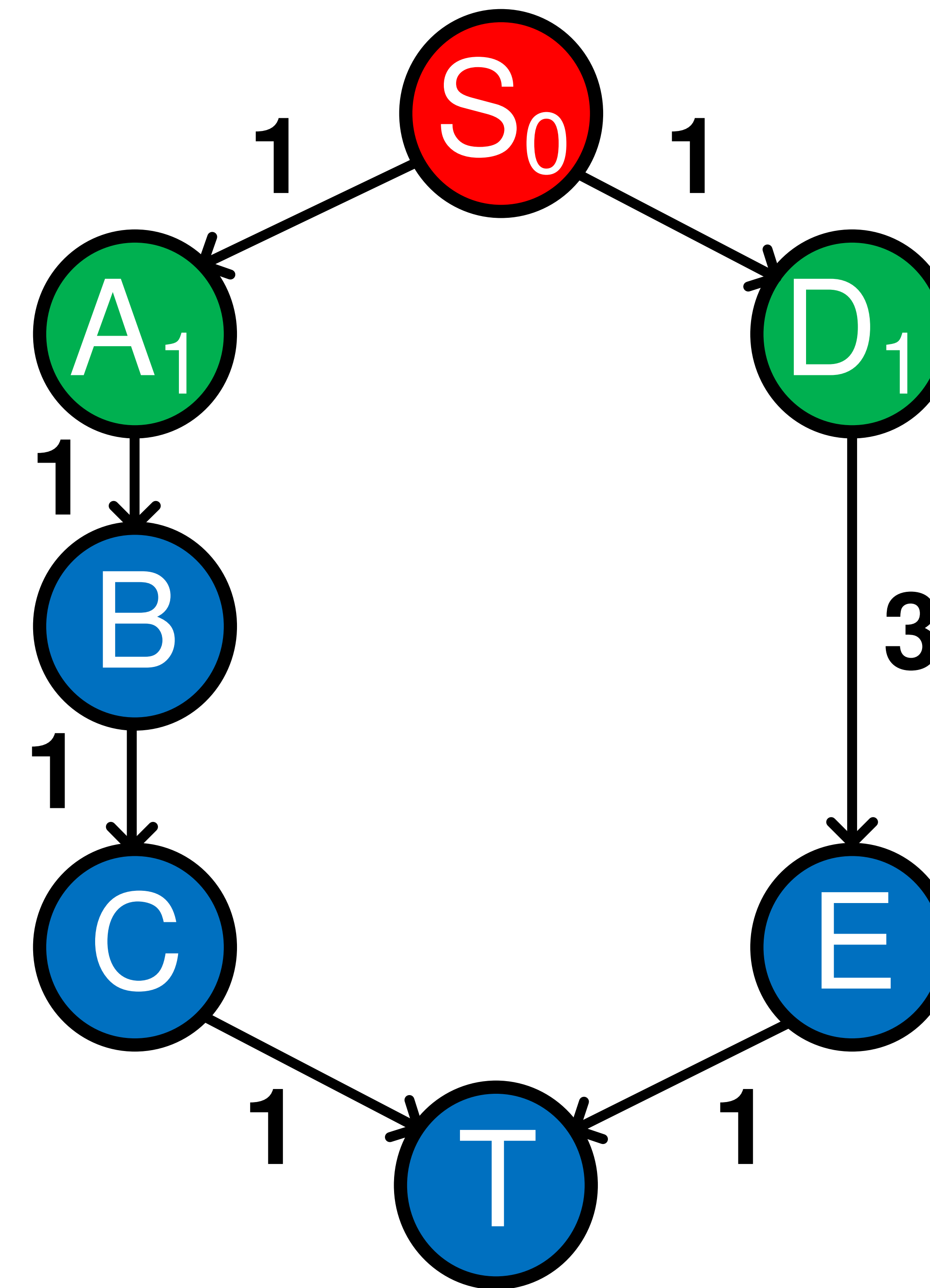
- **Similar to BFS**
- **Uniform cost search when edges have weights**
- **Search “cheapest” neighbor first**
- **Keeps track of distance FROM start**



Graph Algorithms- Dijkstra's

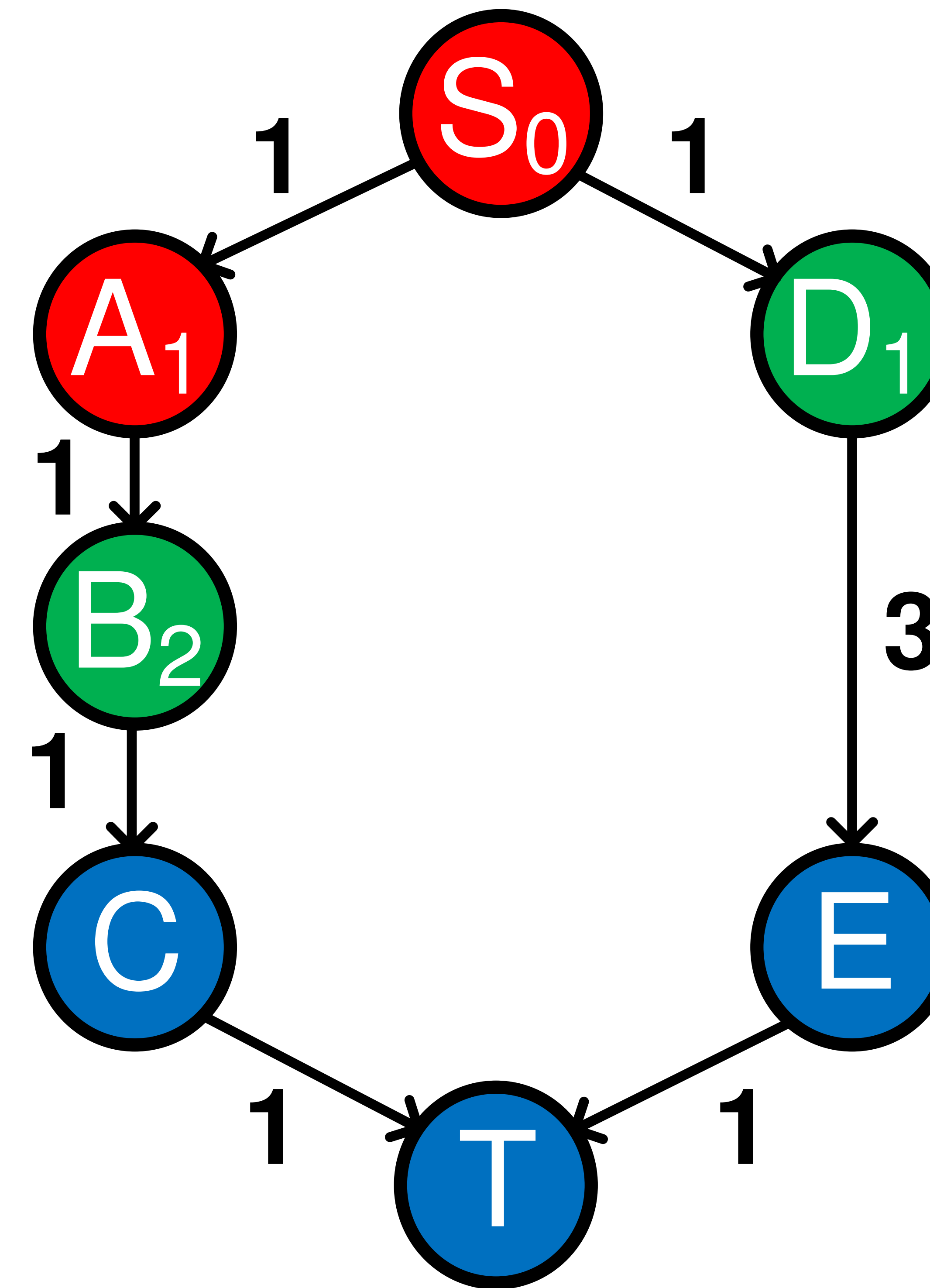
- Find shortest path from Source (S) to Target (T)
- Visited={S₀}
- Cost=0
- PriorityQueue={A_{s1},D_{s1}}

Store Parent Node for
traceback



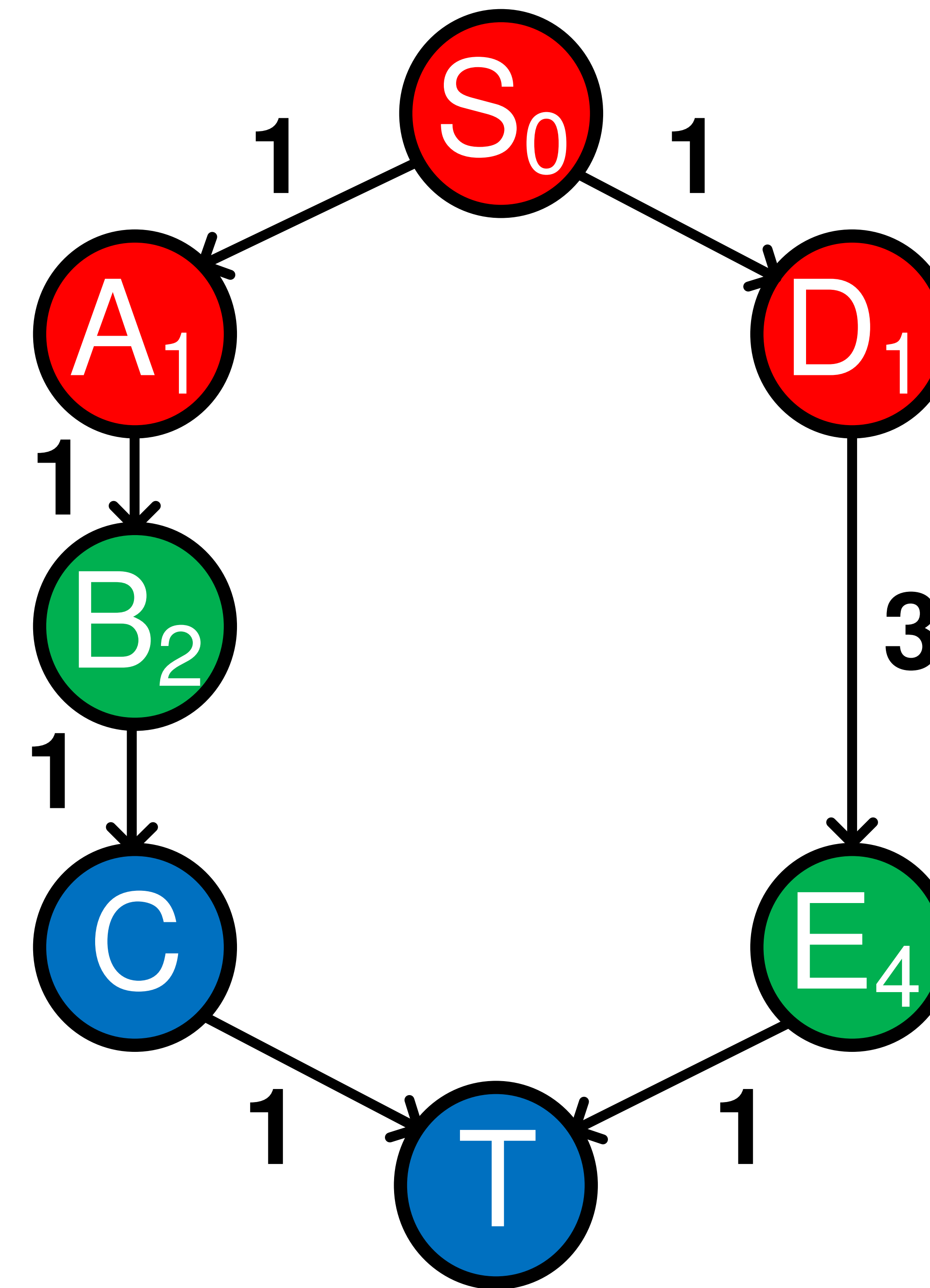
Graph Algorithms- Dijkstra's

- Find shortest path from Source (S) to Target (T)
- Visited={S₀, A_{S1}}
- Cost=1
- PriorityQueue={D_{S1}, B_{A2}}



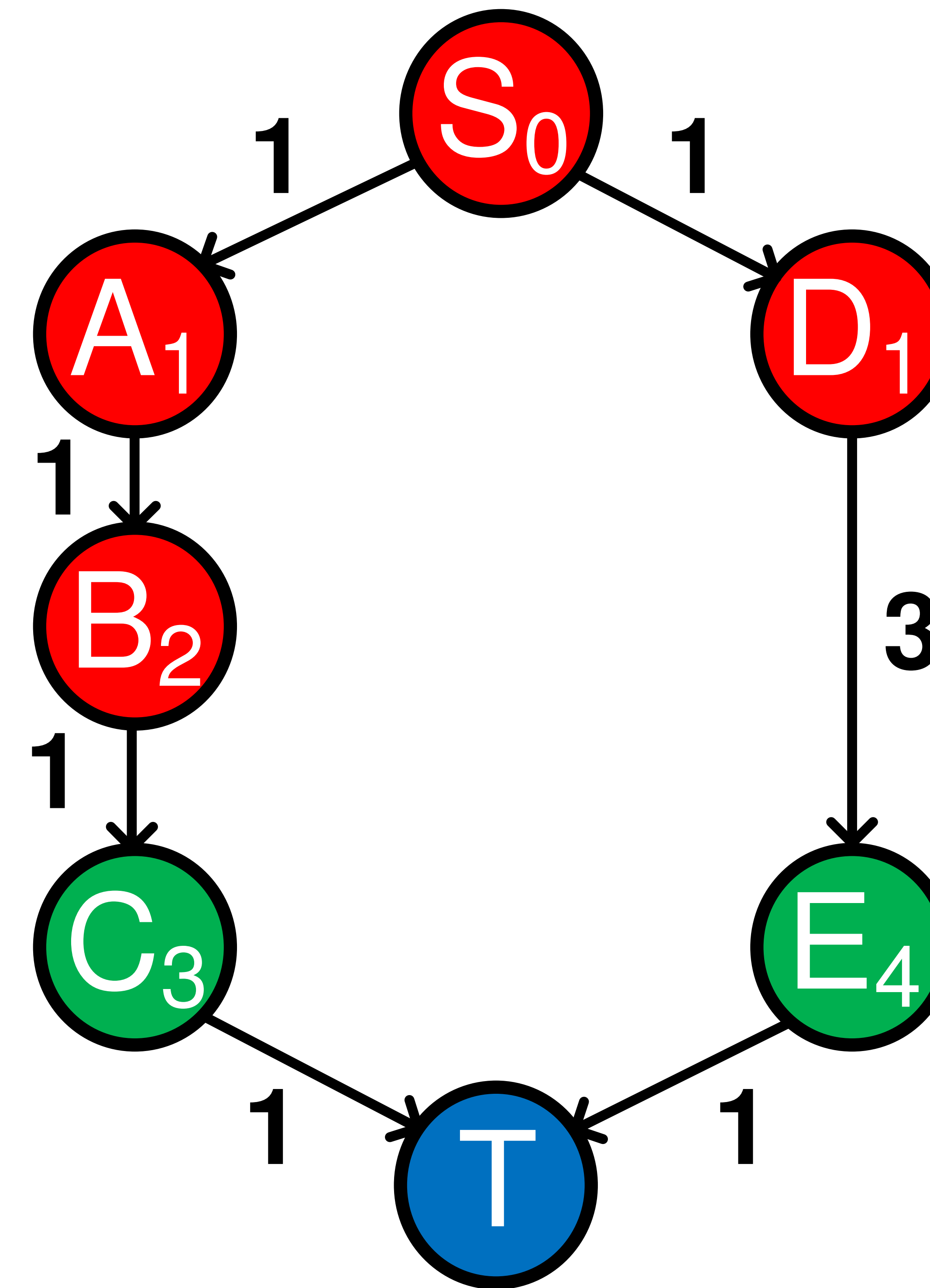
Graph Algorithms- Dijkstra's

- Find shortest path from Source (S) to Target (T)
- Visited= $\{S_0, A_{S1}, D_{S1}\}$
- Cost=1
- PriorityQueue= $\{B_{A2}, E_{D4}\}$



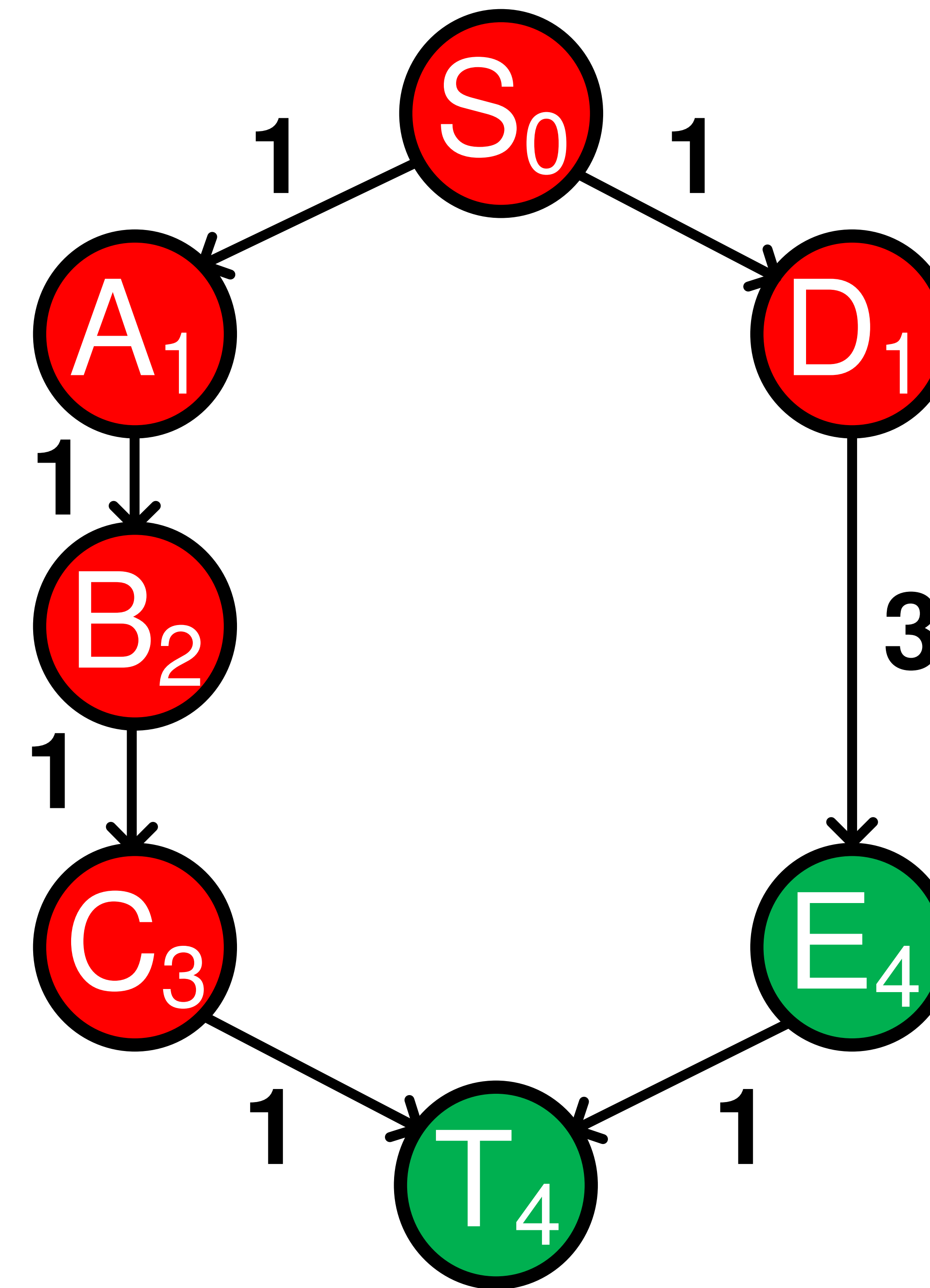
Graph Algorithms- Dijkstra's

- Find shortest path from Source (S) to Target (T)
- Visited={ $S_0, A_{S1}, D_{S1}, B_{A2}$ }
- Cost=2
- PriorityQueue={ C_{B3}, E_{D4} }



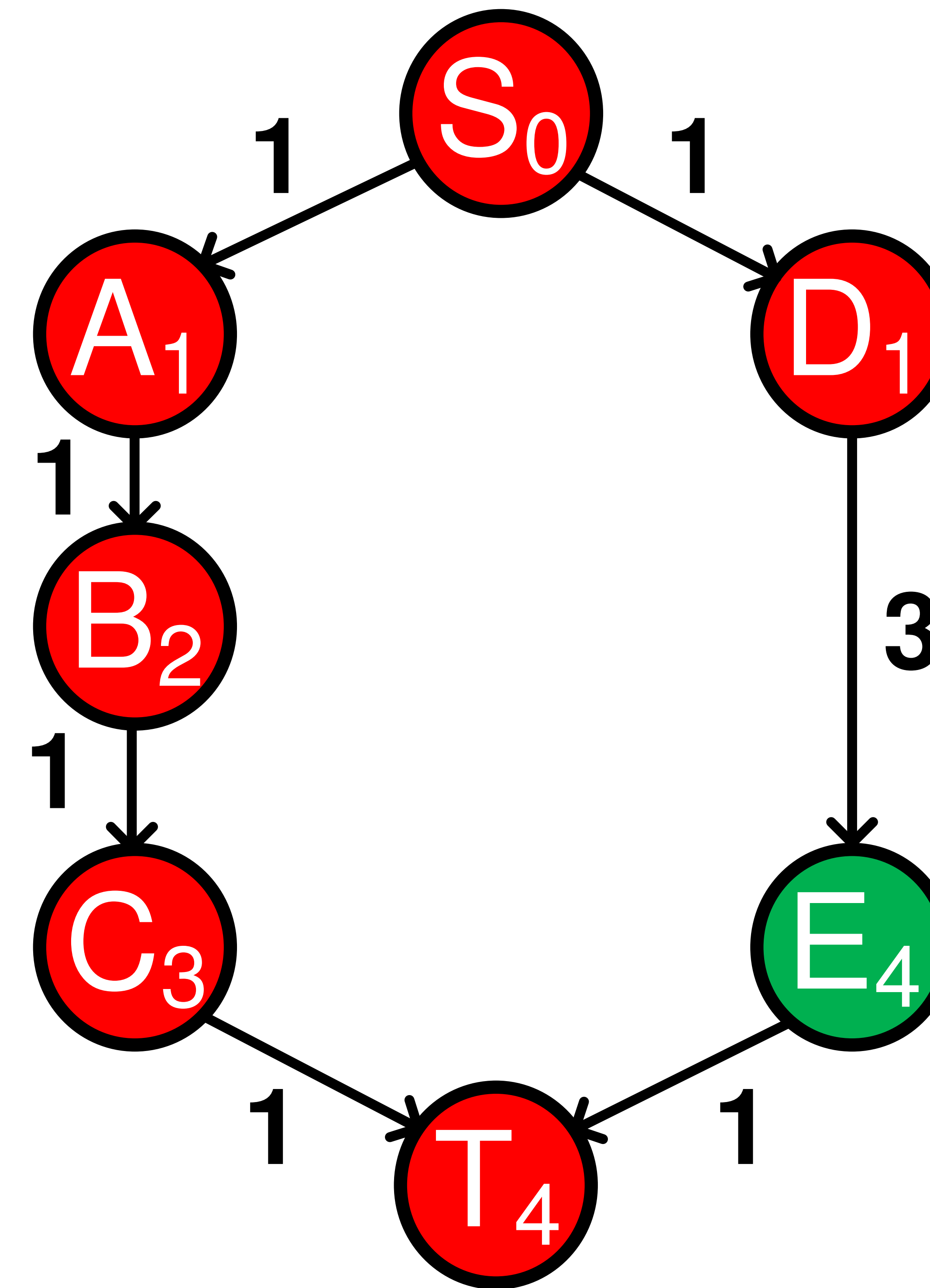
Graph Algorithms- Dijkstra's

- Find shortest path from Source (S) to Target (T)
- Visited= $\{S_0, A_{S1}, D_{S1}, B_{A2}, C_{B3}\}$
- Cost=3
- PriorityQueue= $\{T_{C4}, E_{D4}\}$



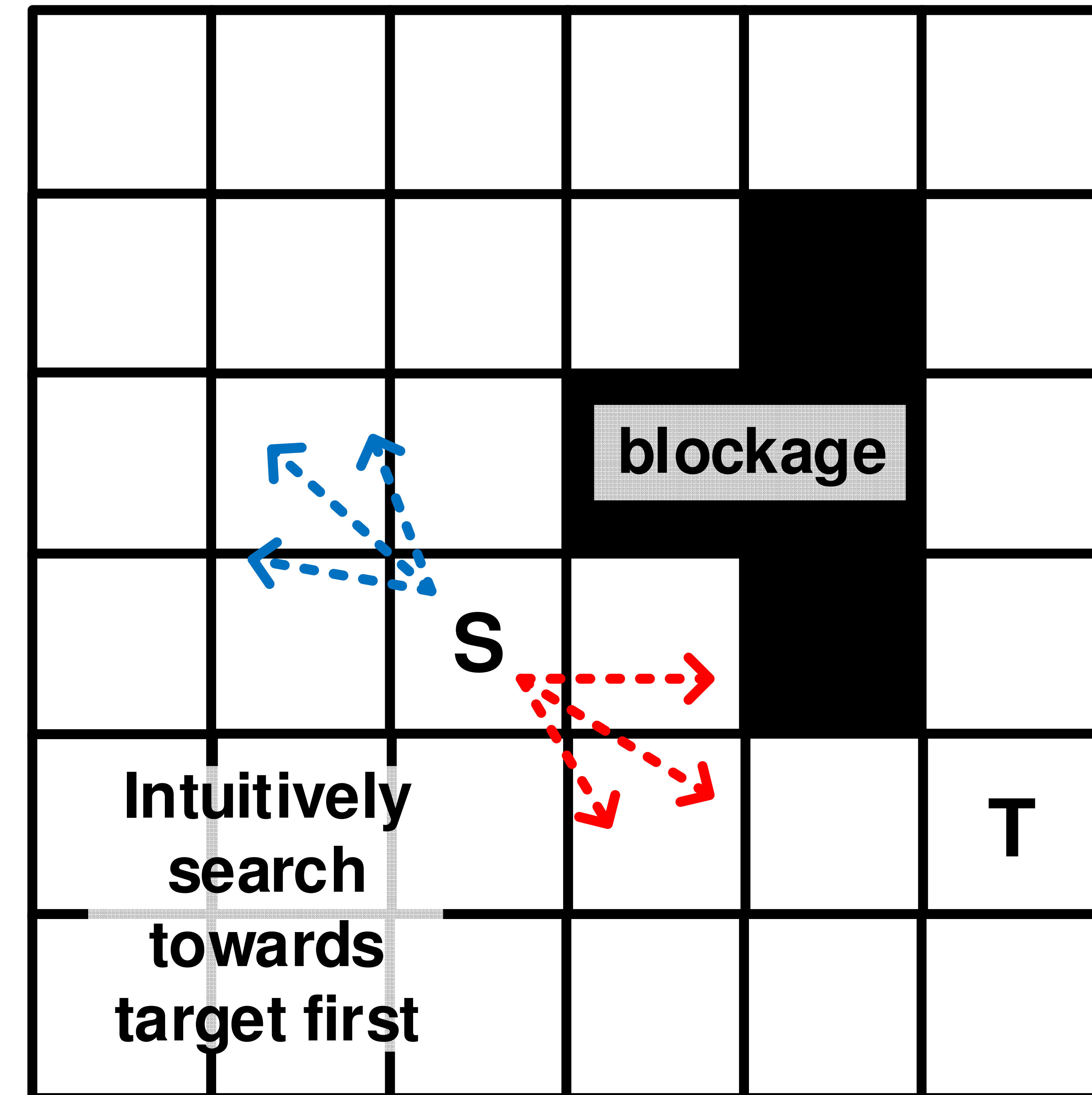
Graph Algorithms- Dijkstra's

- Find shortest path from Source (S) to Target (T)
- Visited= $\{S_0, A_{S1}, D_{S1}, B_{A2}, C_{B3}, T_{C4}\}$
- Cost=3
- PriorityQueue= $\{E_{D4}\}$

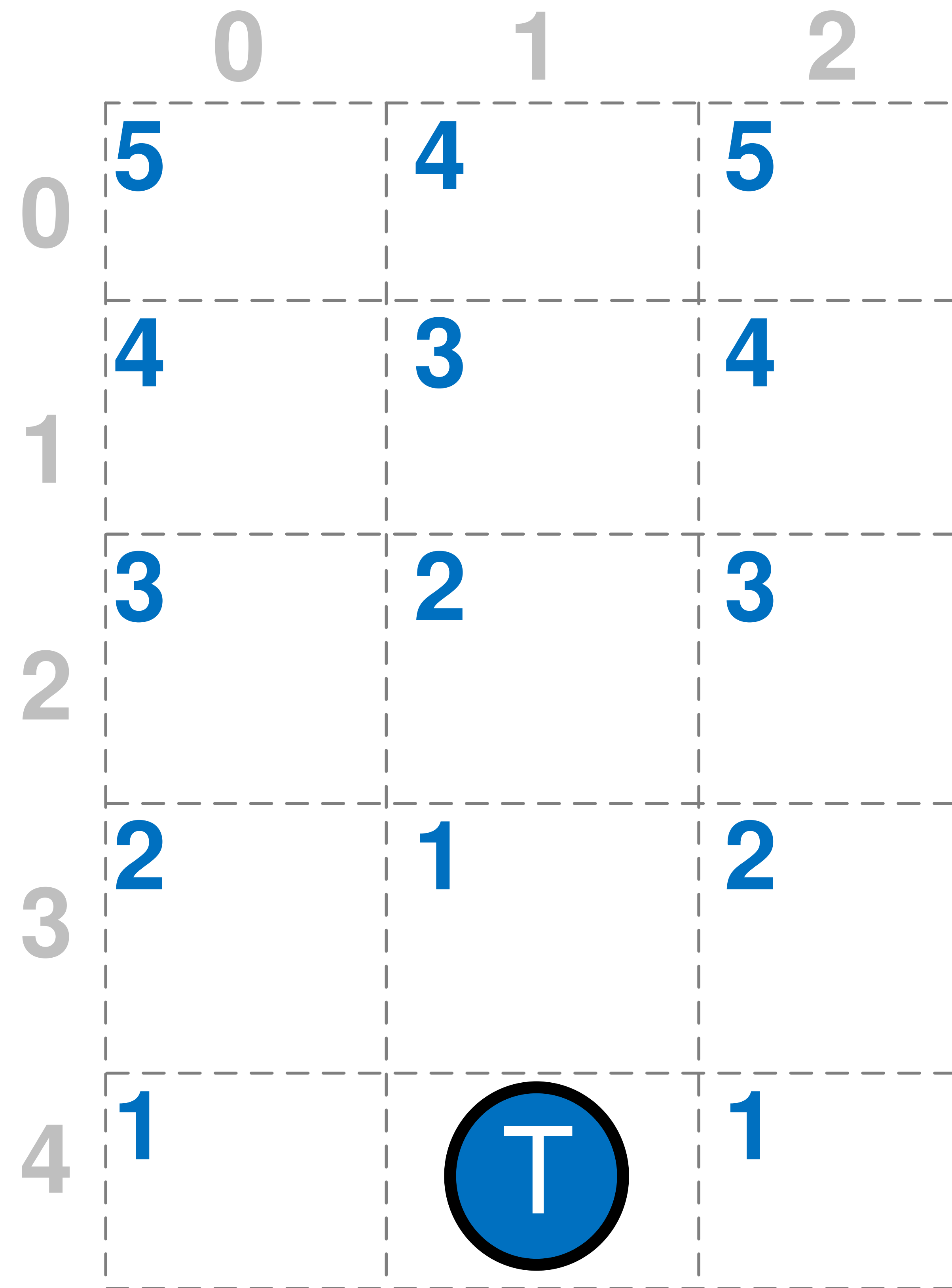


Graph Algorithms- A*

- **A*** guides the search towards the destination
- **Cost(n) = F(n) + H(n)**
 - F(n) = actual distance from source
 - H(n) = heuristic that *predicts* the distance to target
 - H(n) = 0 for Dijkstra's
- Provides optimal path if $H(n) \leq$ Actual Cost



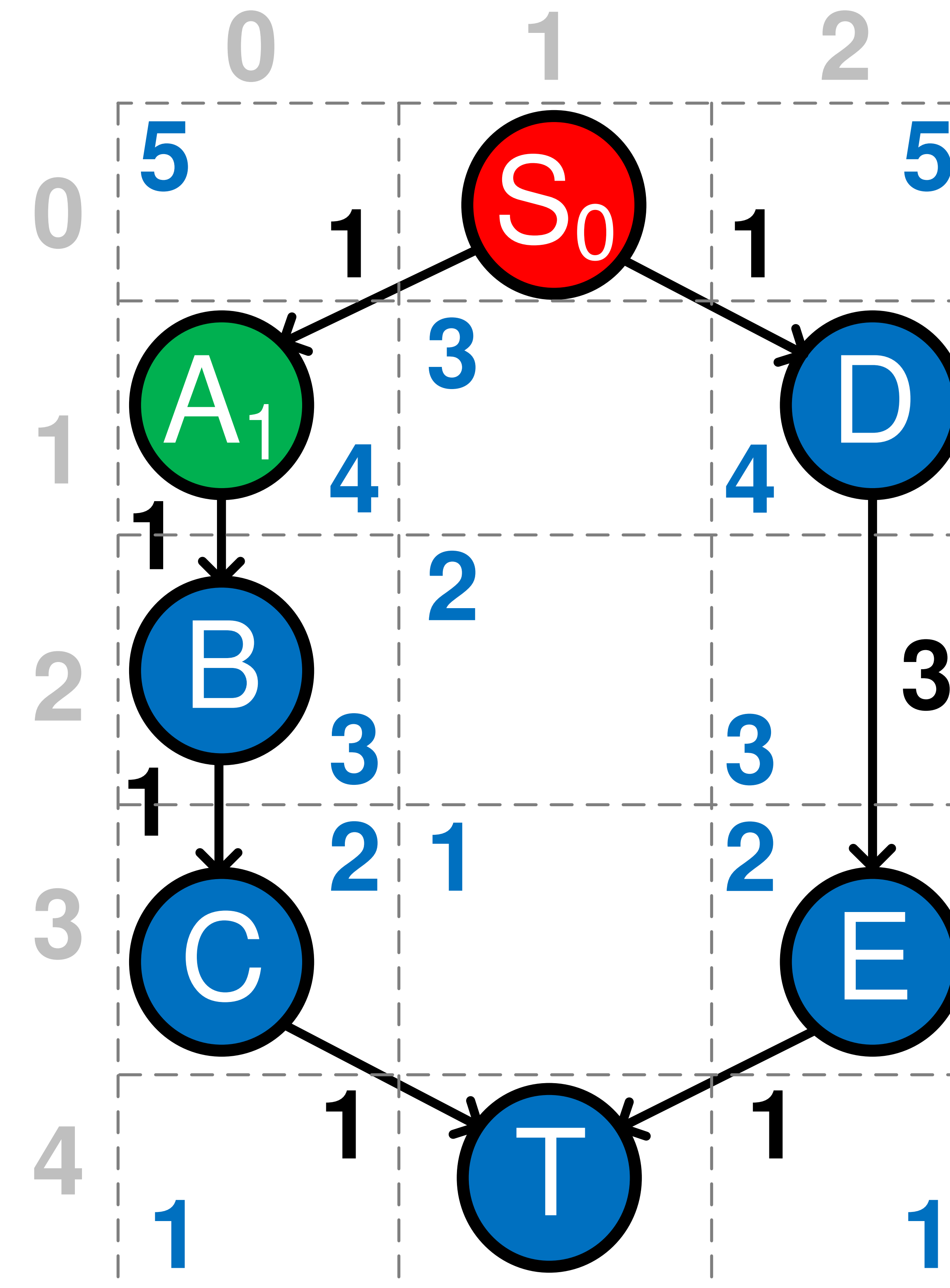
Graph Algorithms- A*



- Let $H(n) = \text{Manhattan Distance}$
- $H(n) = \text{abs}(x_i - x_T) + \text{abs}(y_i - y_T)$

Graph Algorithms- A*

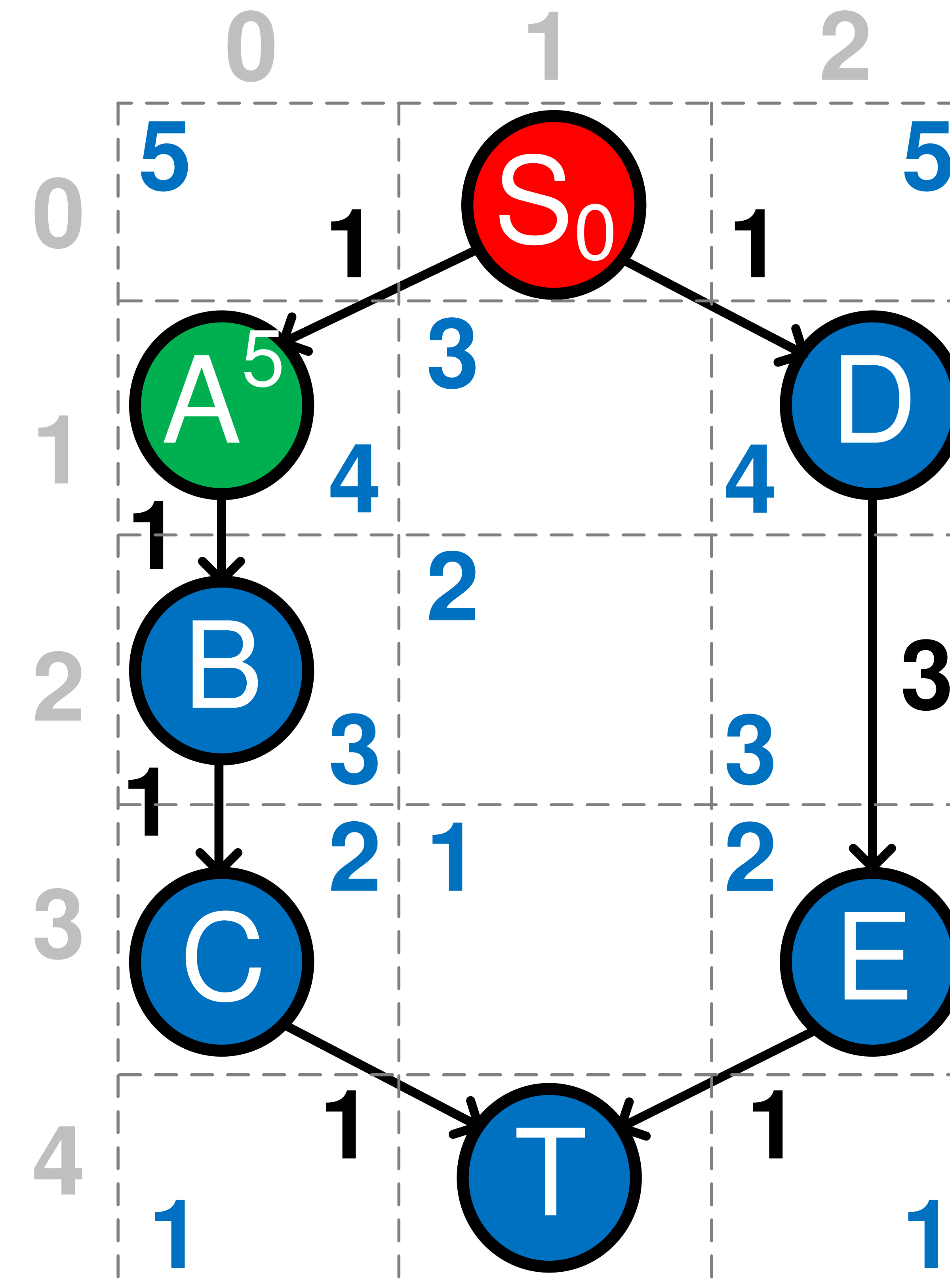
- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited= $\{S_0\}$
- Cost/ $F(n)=0$
- PriorityQueue= $\{A_{S_1}\}$



Graph Algorithms- A*

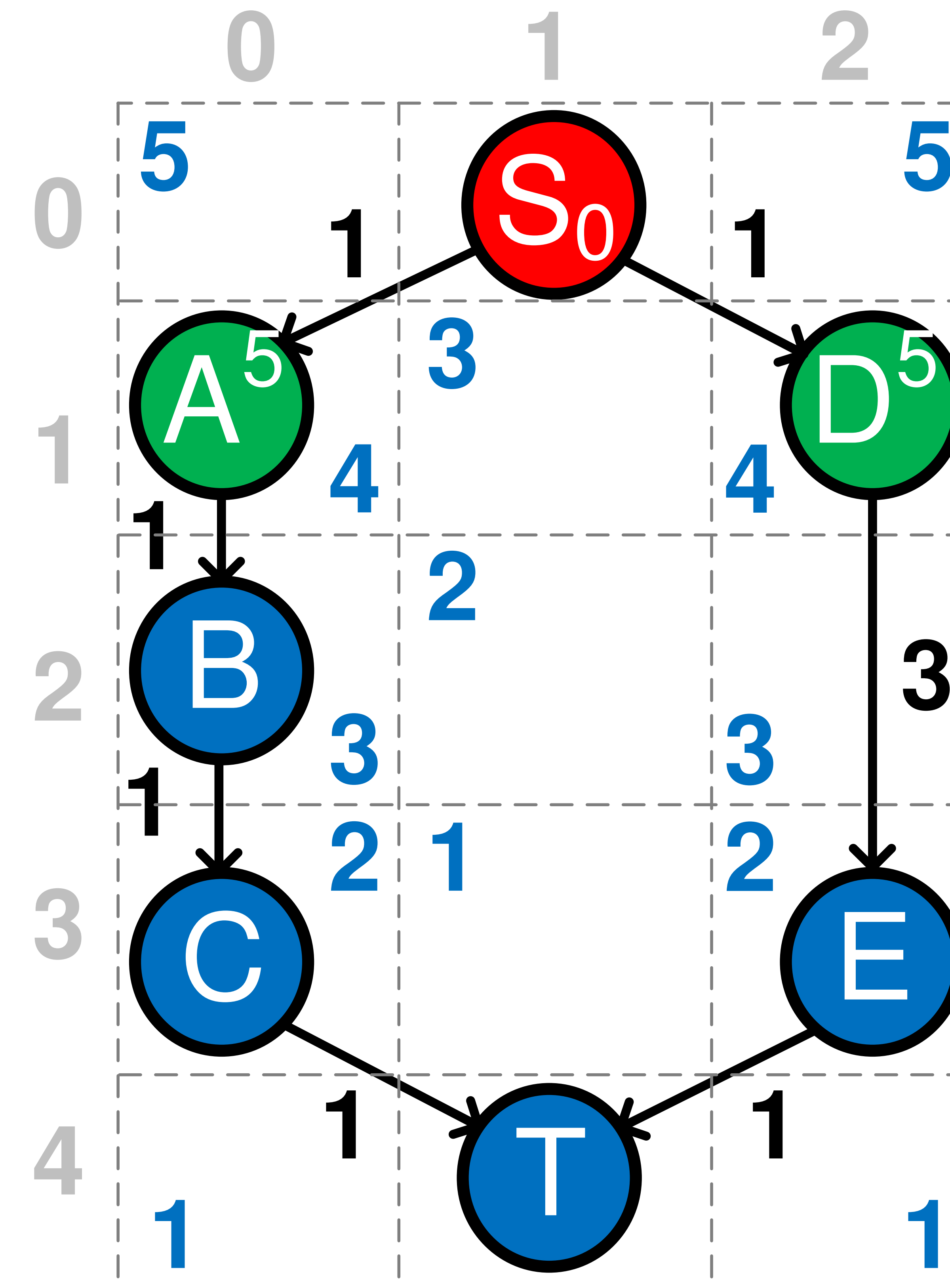
- Find path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited= $\{S_0\}$
- Cost/ $F(n)=0$
- PriorityQueue= $\{A_{S_1}^5\}$

$$C(A) = F(A) + H(A) = 1 + 4$$



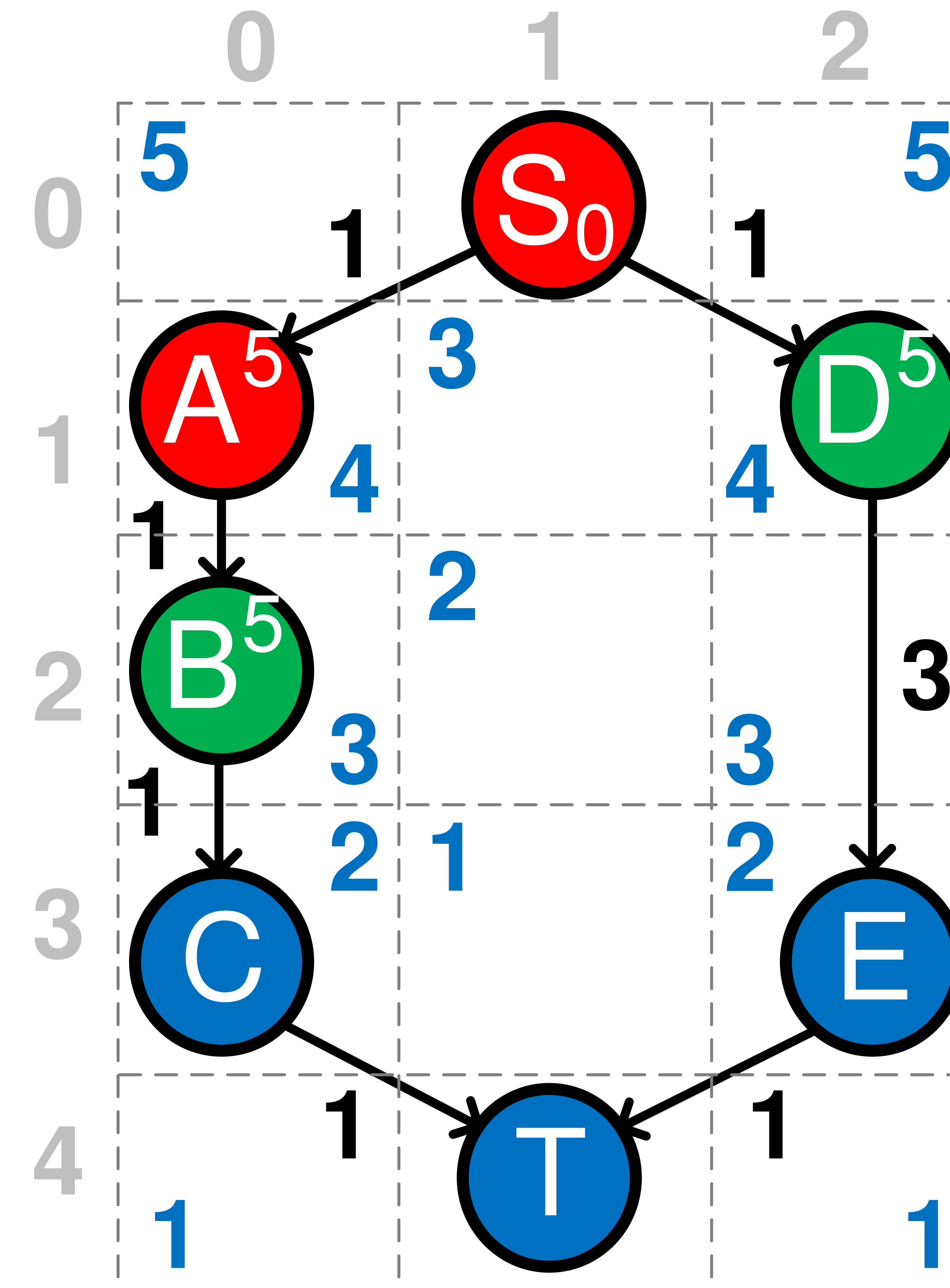
Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- **Visited**={S₀}
- **Cost**/F(n)=0
- **PriorityQueue**={A_{S1}⁵, D_{S1}⁵}



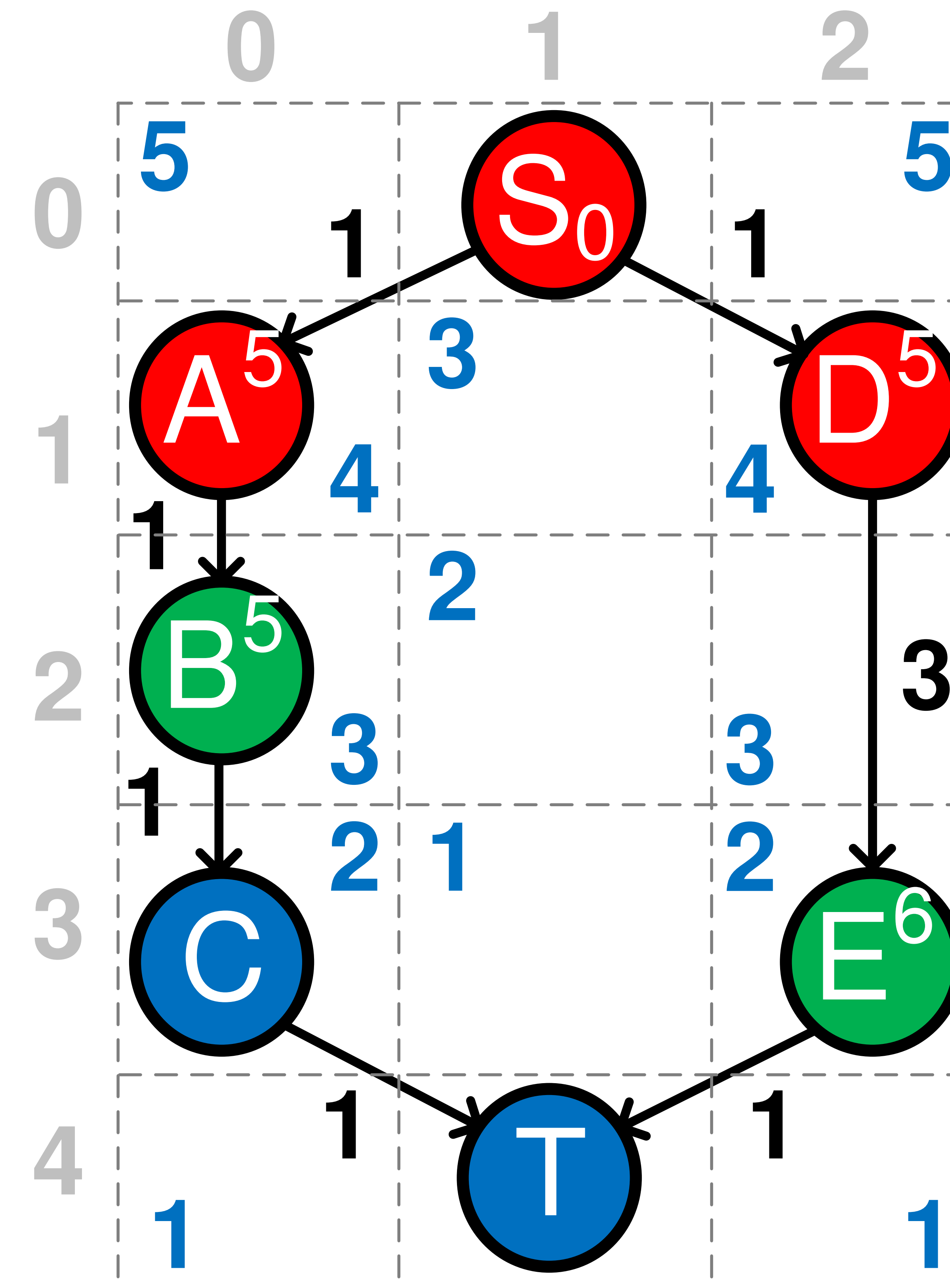
Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited = $\{S_0, A_{S_1}^5\}$
- Cost/ $F(n) = 1$
- Priority Queue = $\{B_{A_2}^5, D_{S_1}^5\}$



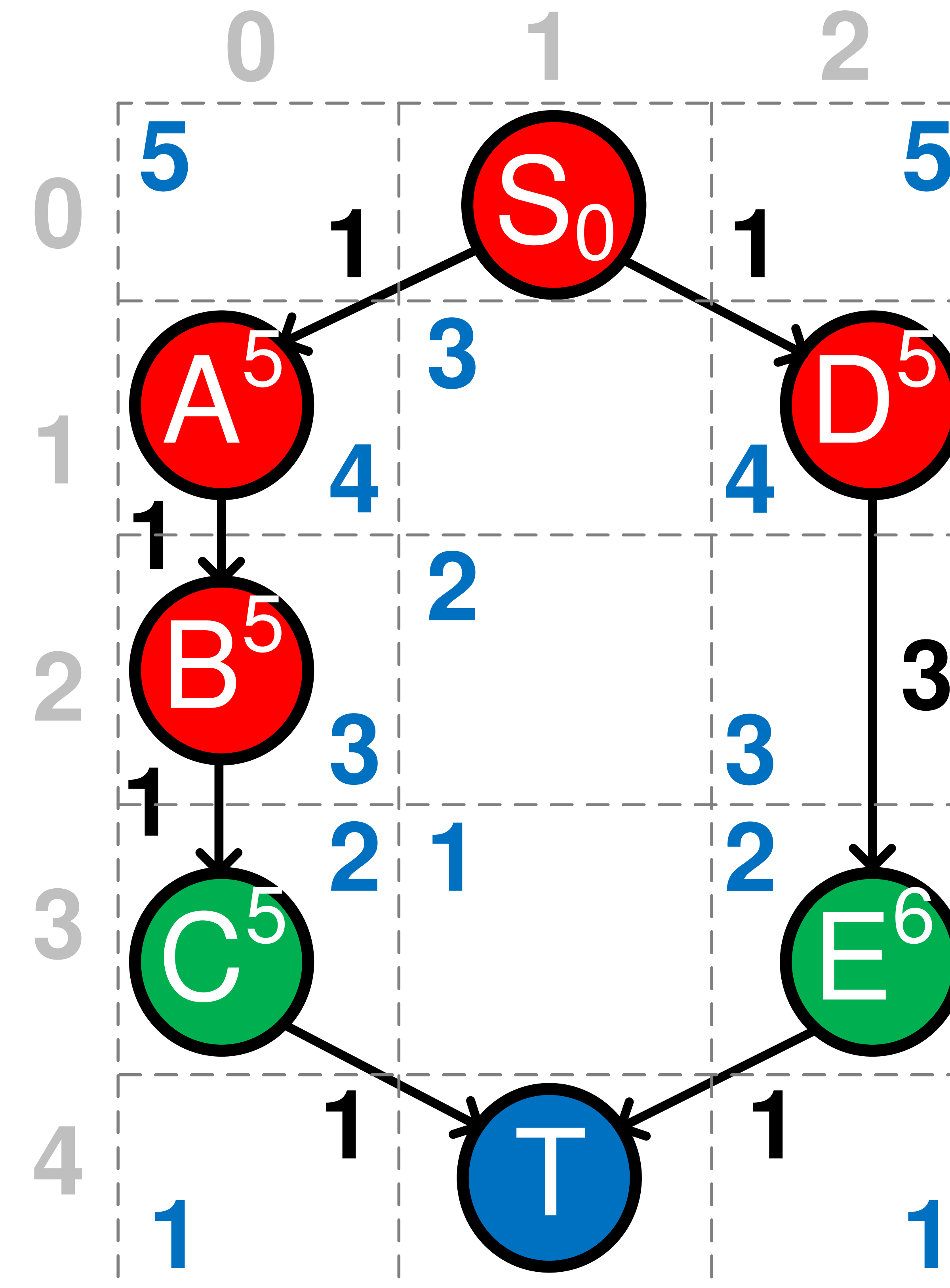
Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited = $\{S_0, A_{S_1}^5, D_{S_1}^5\}$
- Cost/F(n) = 1
- Priority Queue = $\{B_{A_2}^5, E_{D_4}^6\}$



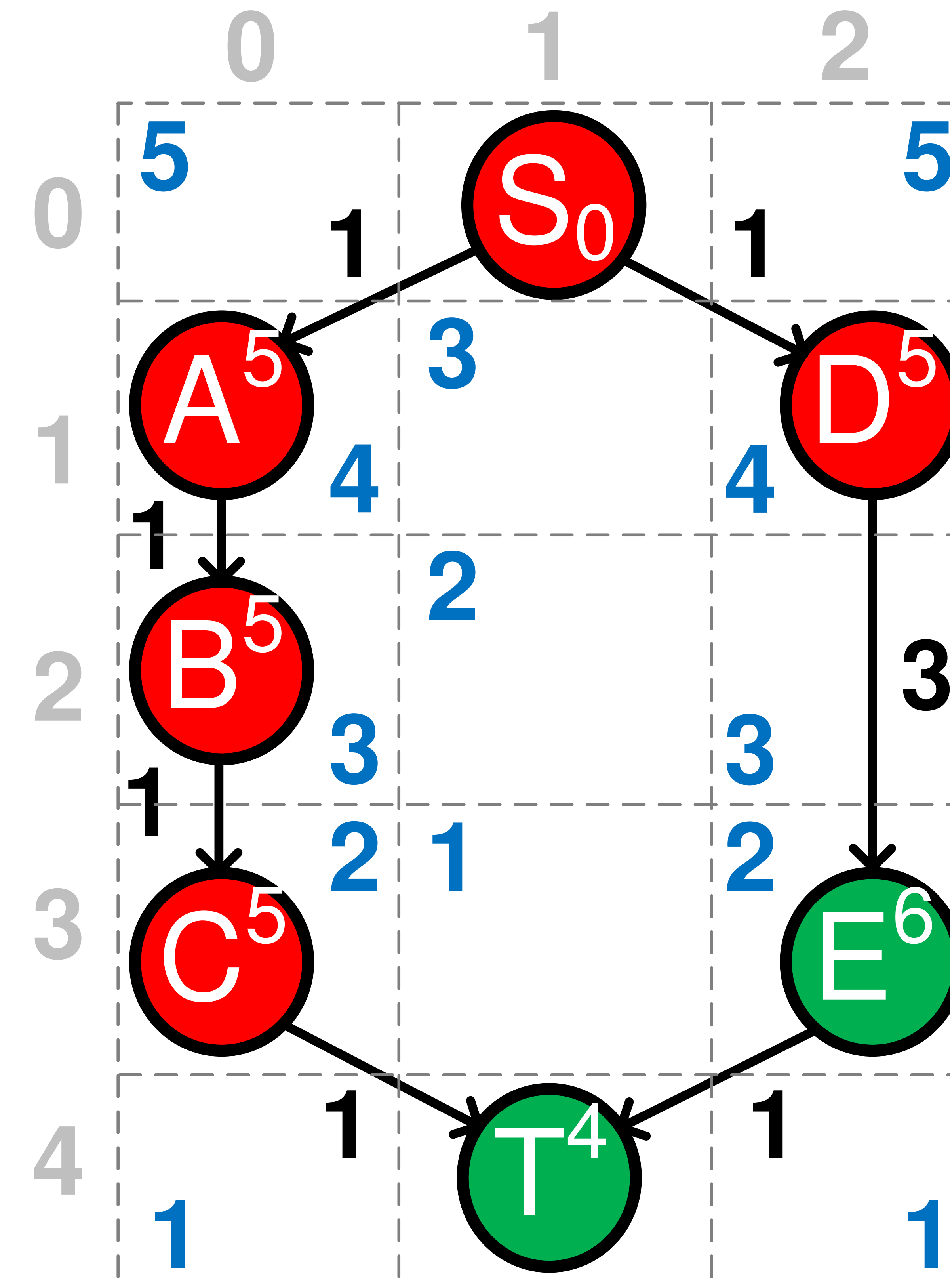
Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited = $\{S_0, A_{S_1}^5, D_{S_1}^5, B_{A_2}^5\}$
- Cost/ $F(n) = 2$
- Priority Queue = $\{C_{B_3}^5, E_{D_4}^6\}$



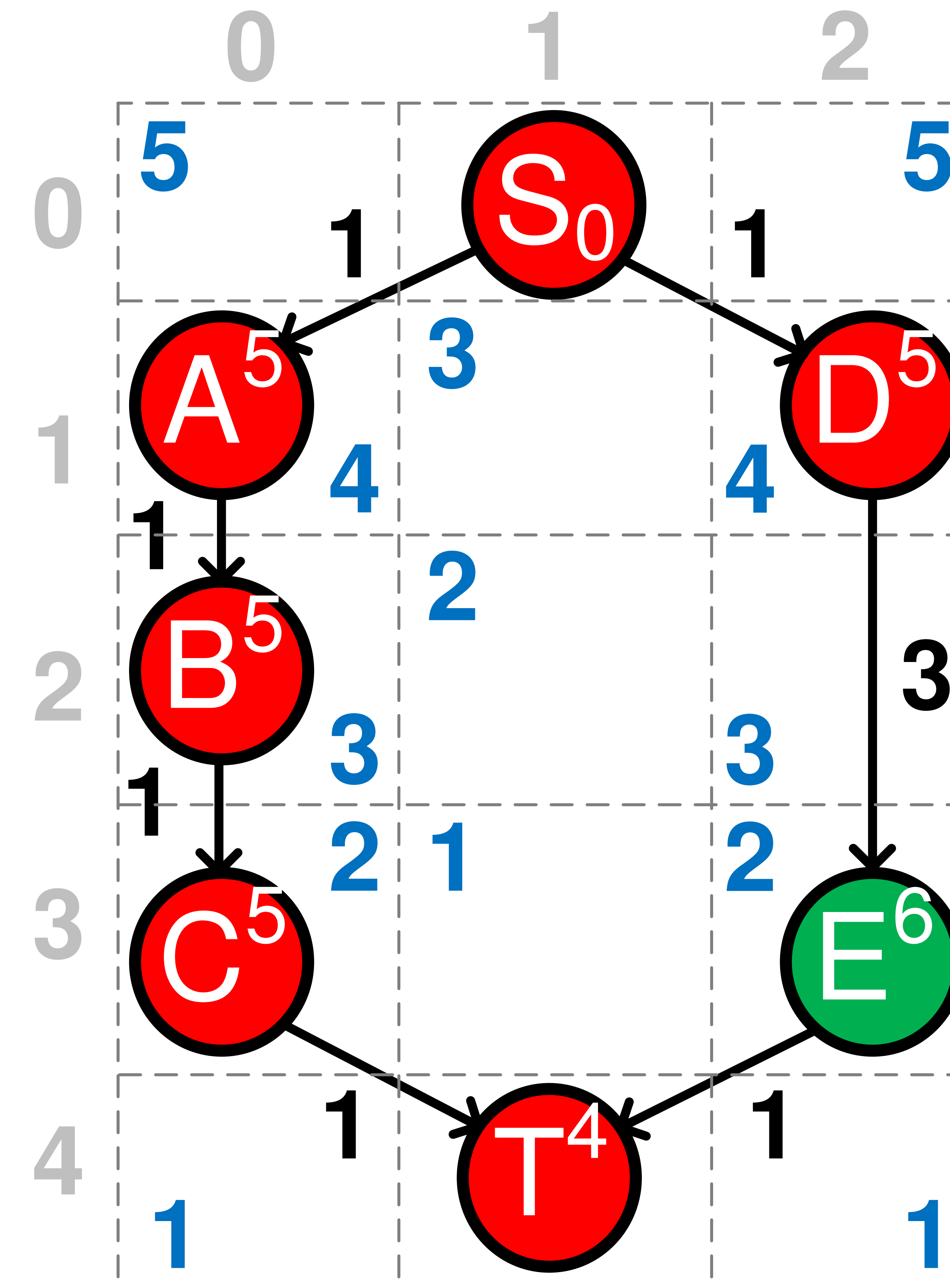
Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited = $\{S_0, A_{S_1}^5, D_{S_1}^5, B_{A_2}^5, C_{B_3}^5\}$
- Cost/F(n) = 3
- PriorityQueue = $\{T_{C_4}^4, E_{D_4}^6\}$

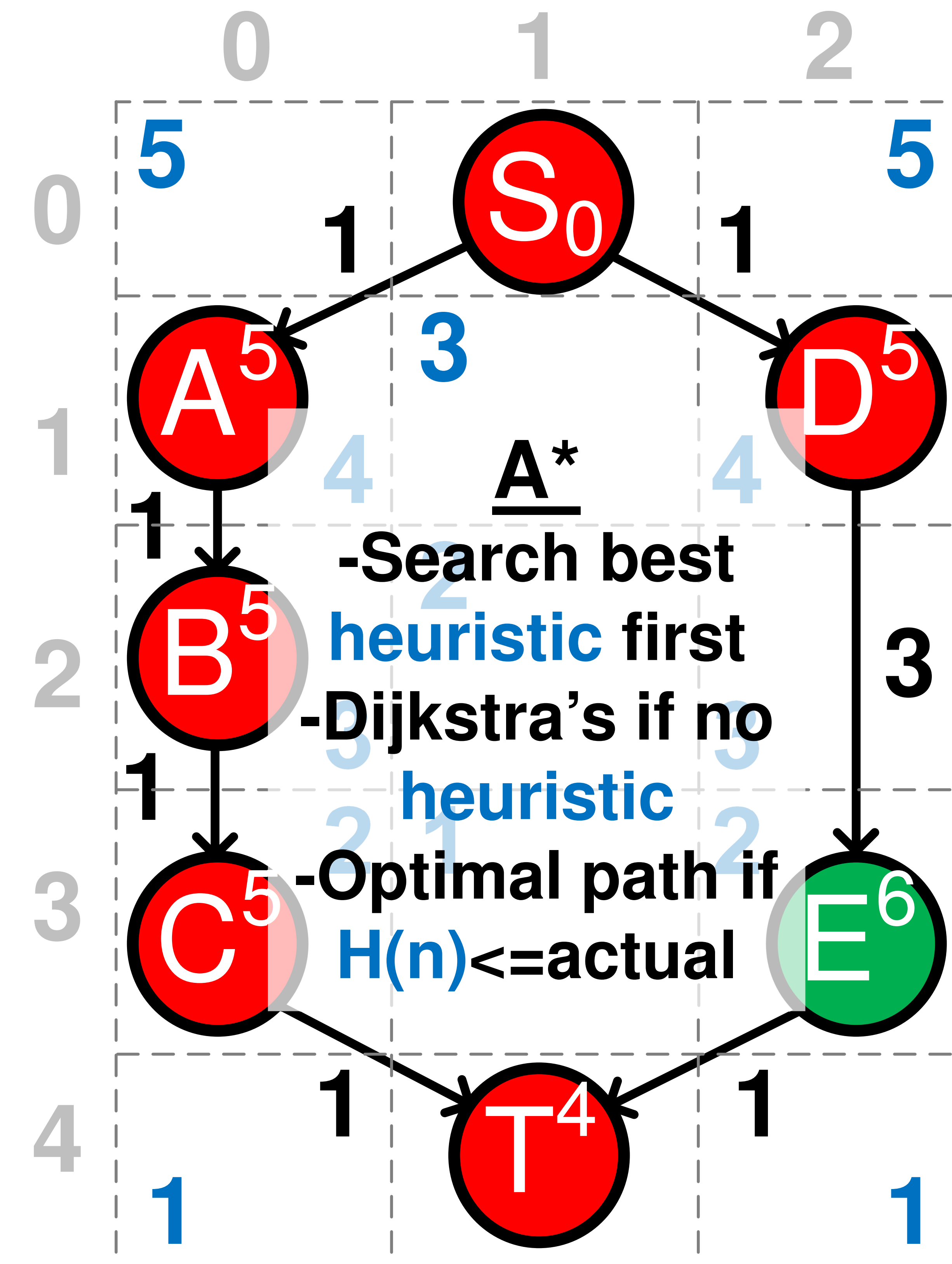
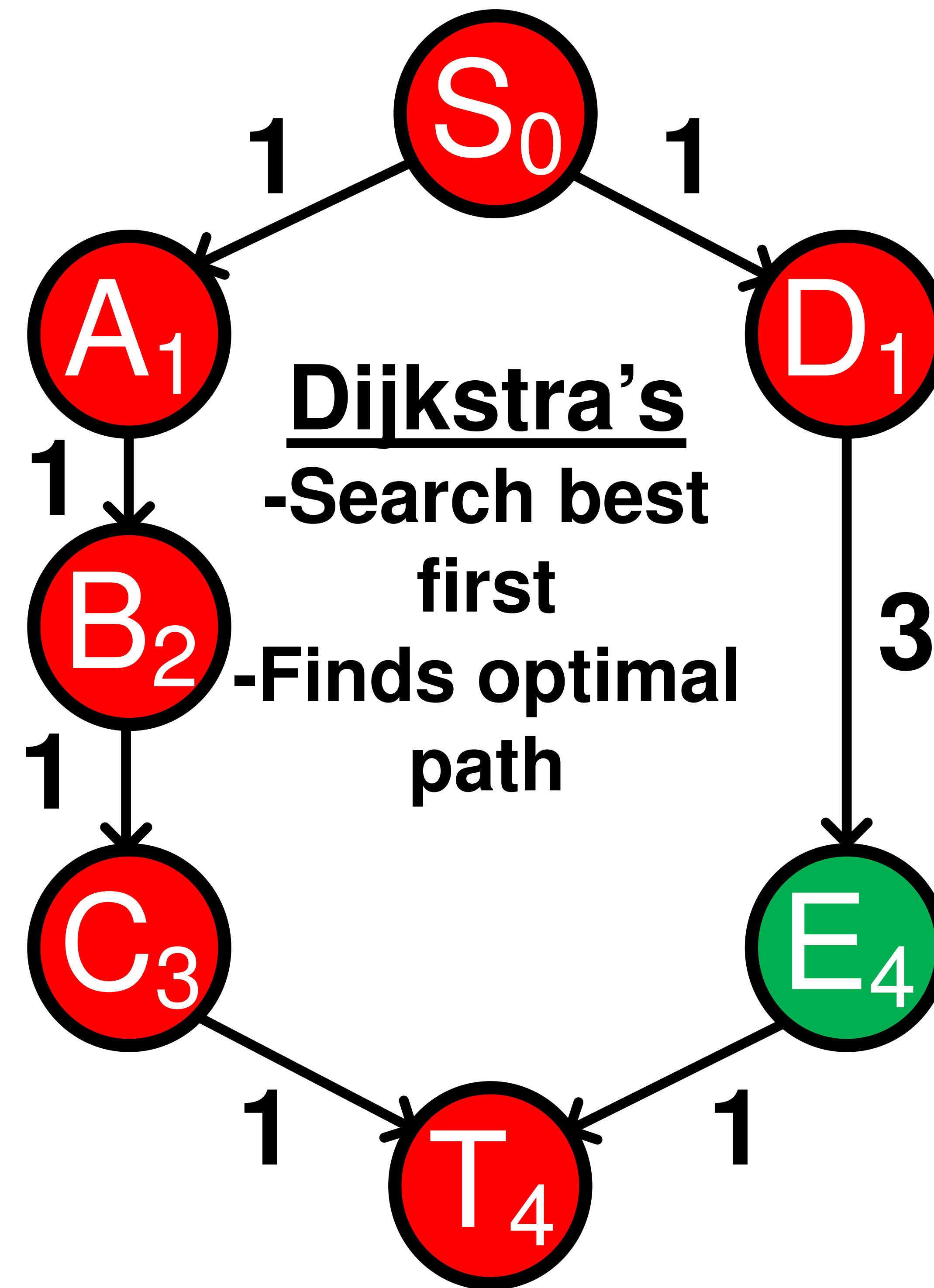
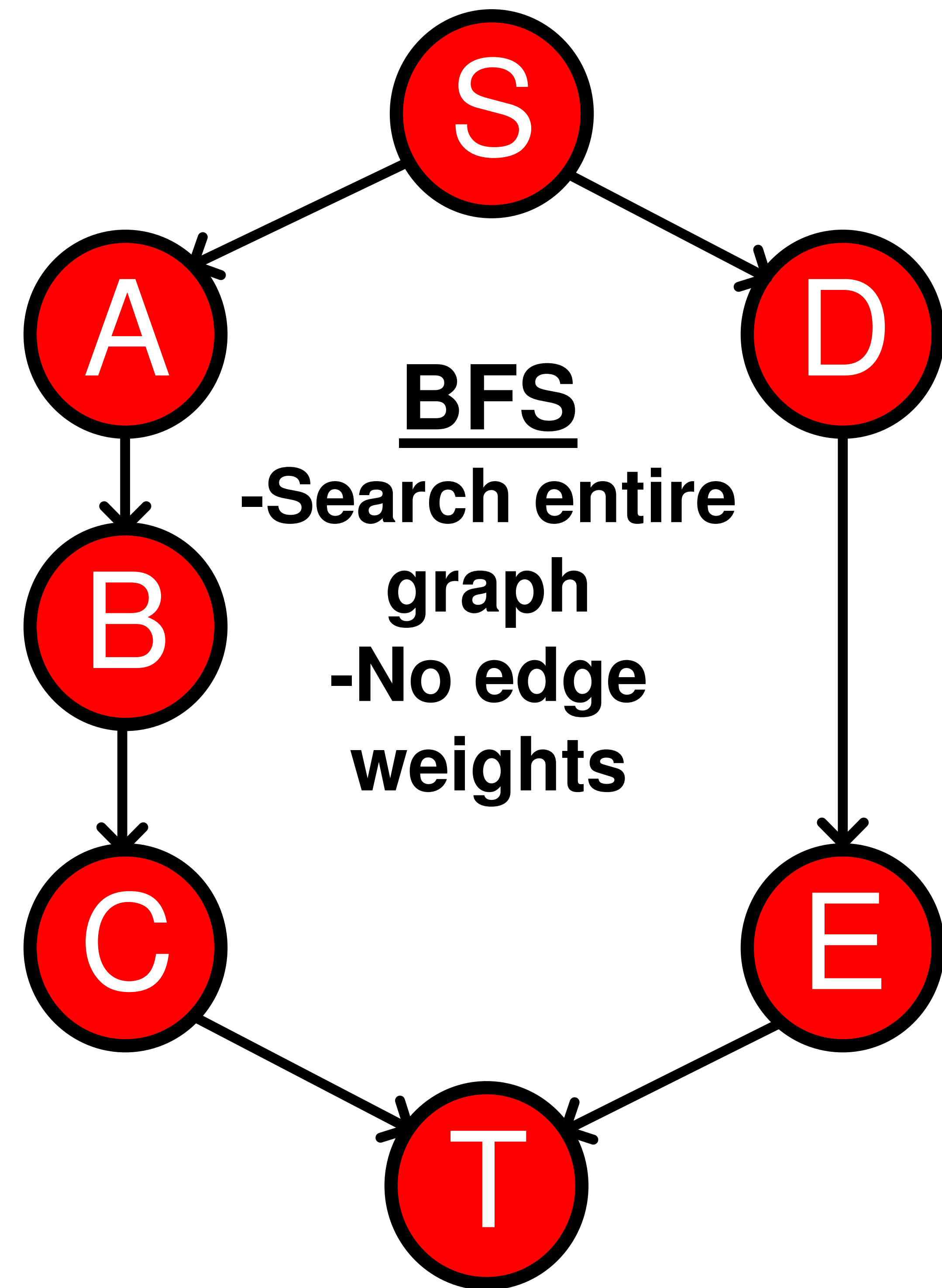


Graph Algorithms- A*

- Find shortest path from Source (S) to Target (T)
- Let $H(n) = \text{Manhattan Distance}$
- Visited = $\{S_0, A_{S1}^5, D_{S1}^5, B_{A2}^5, C_{B3}^5, T_{C4}^4\}$
- Cost/ $F(n) = 3$
- Priority Queue = $\{E_{D4}^6\}$

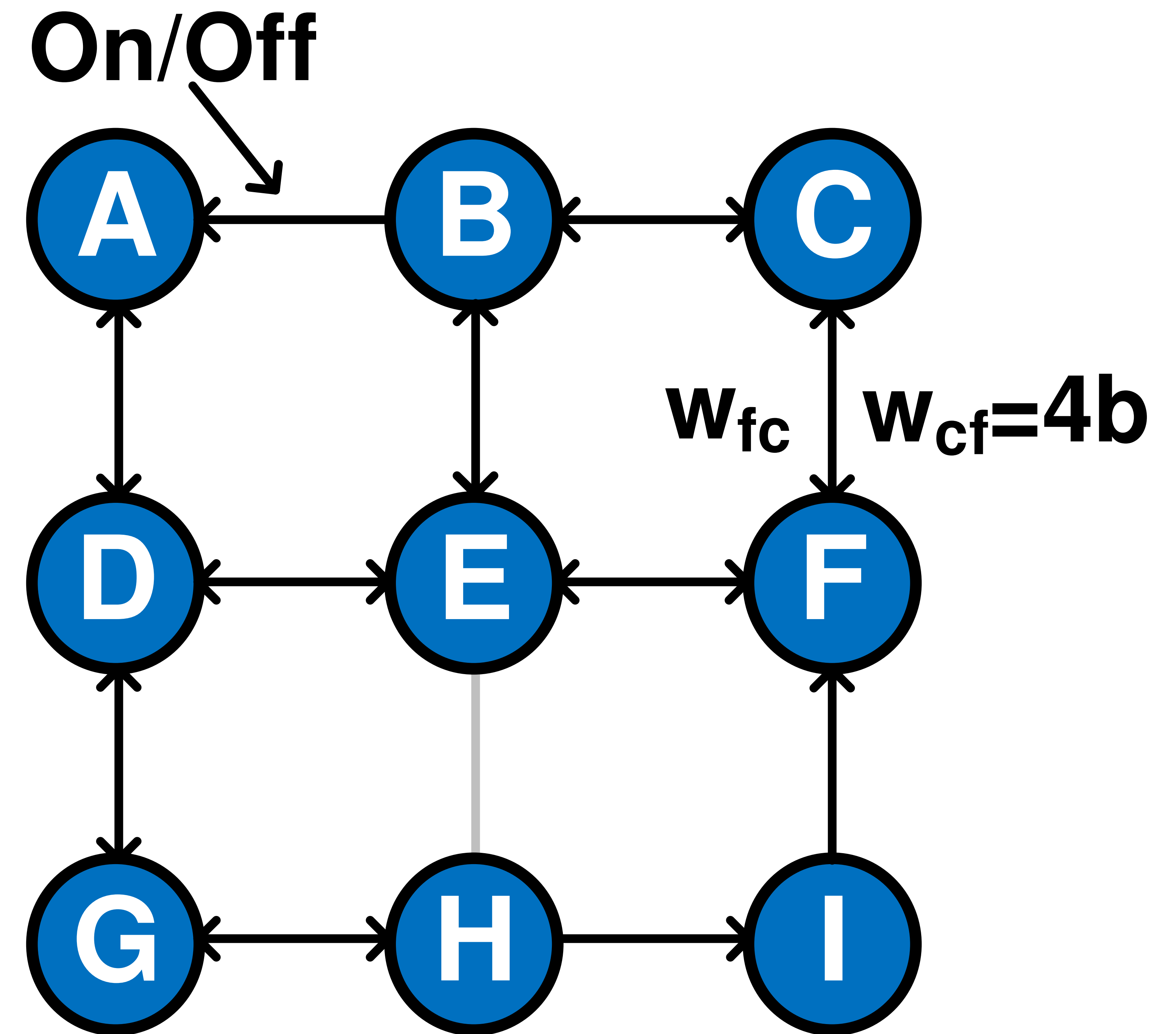


Algorithm Review



Graph Structure- This Work

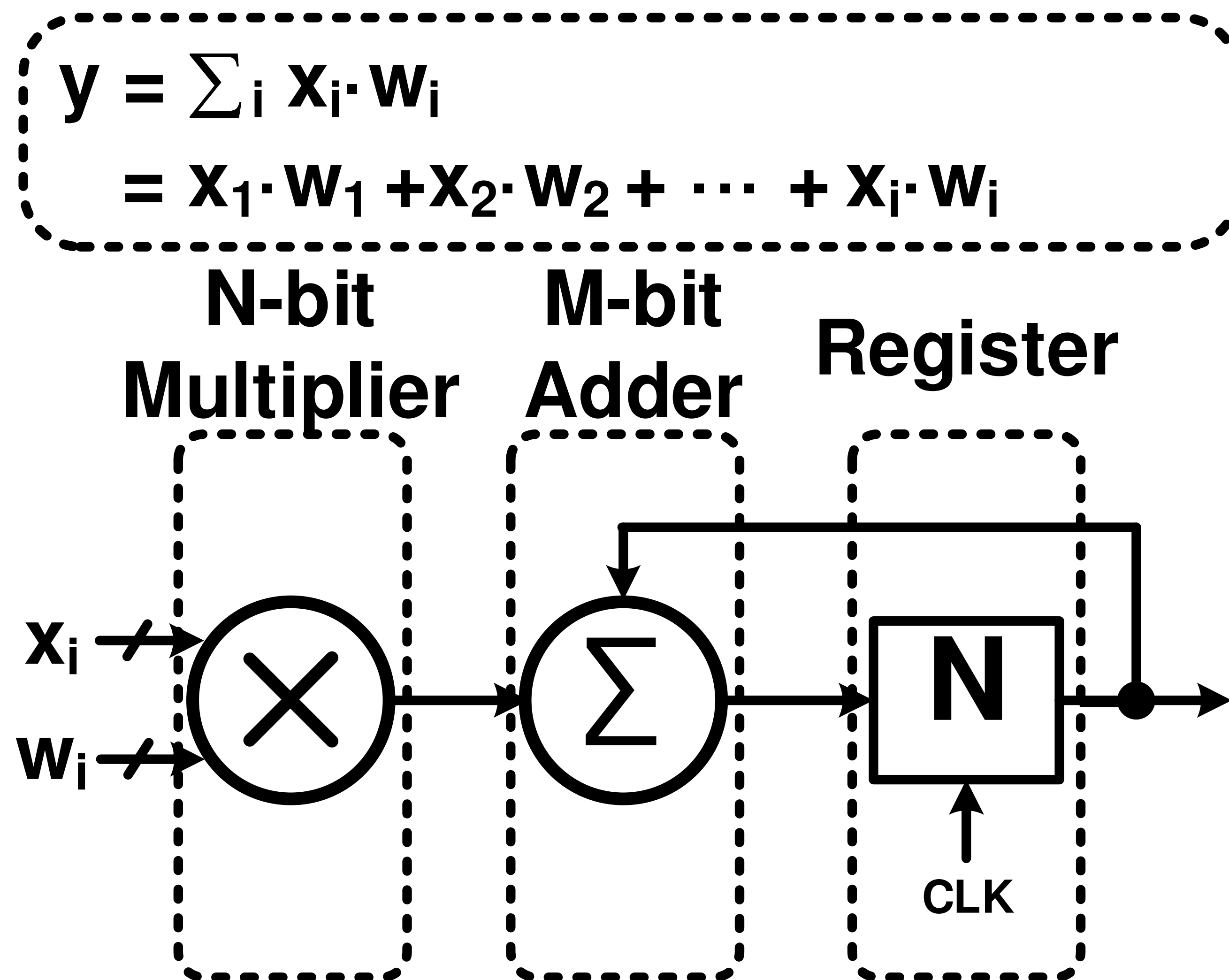
- 4-neighbor *grid*
- Directed edges
- Each direction binary on/off control
- 4 bit digital control edge weight
- Programmable at runtime



Outline

- Background: Path Planning Algorithms
- **Time-Based A* ASIC**
 - Time-Based Primer
 - Top Level Design
 - Vertex Details
 - Edge Details
- 65nm Test Chip Results
- Applications
- Conclusion

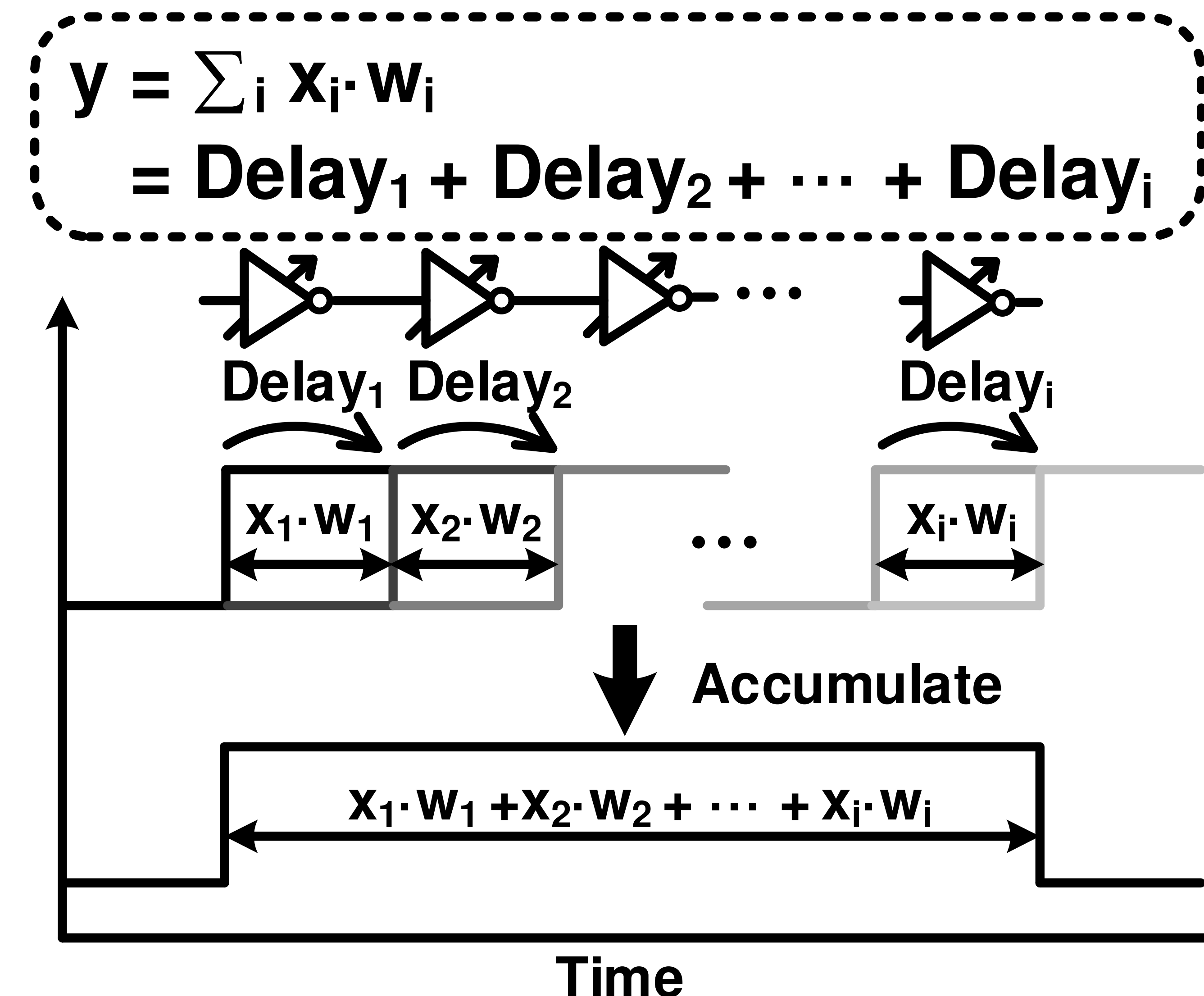
Digital Computing



Advantages of digital arithmetic:

- Binary representation
- Less “buy-in” required
- Existing IP for rapid SoC development
- No calibration

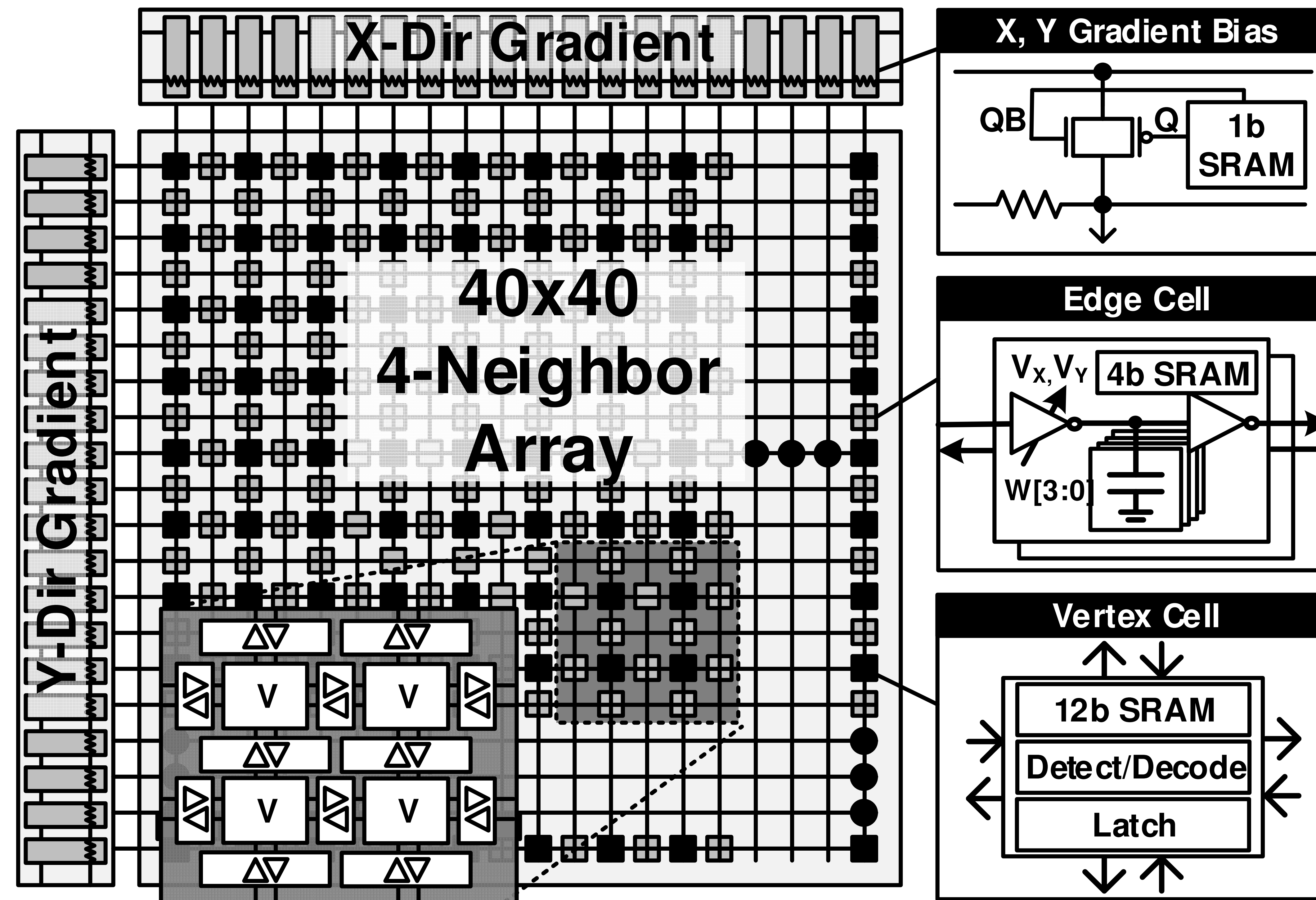
Time-based Computing



Advantages of time-based circuits:

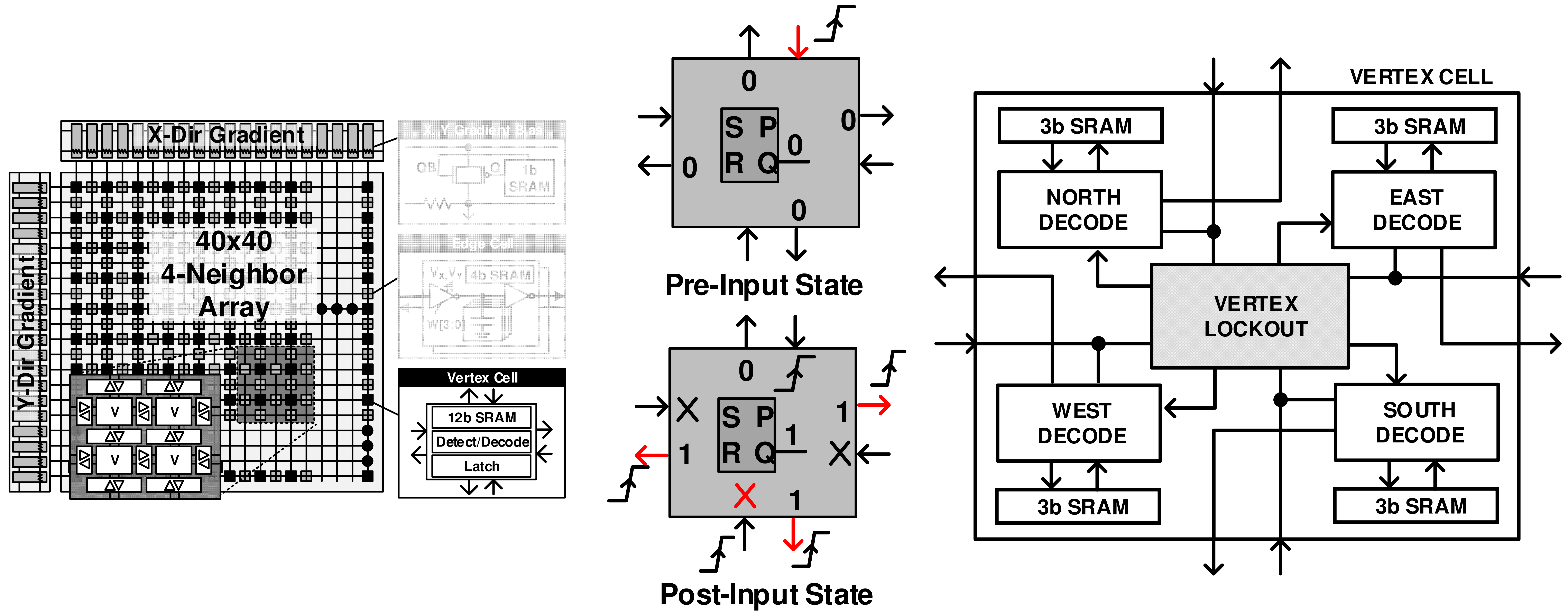
- Compact area
- Low power consumption
- High precision tunability

40×40 A* ASIC

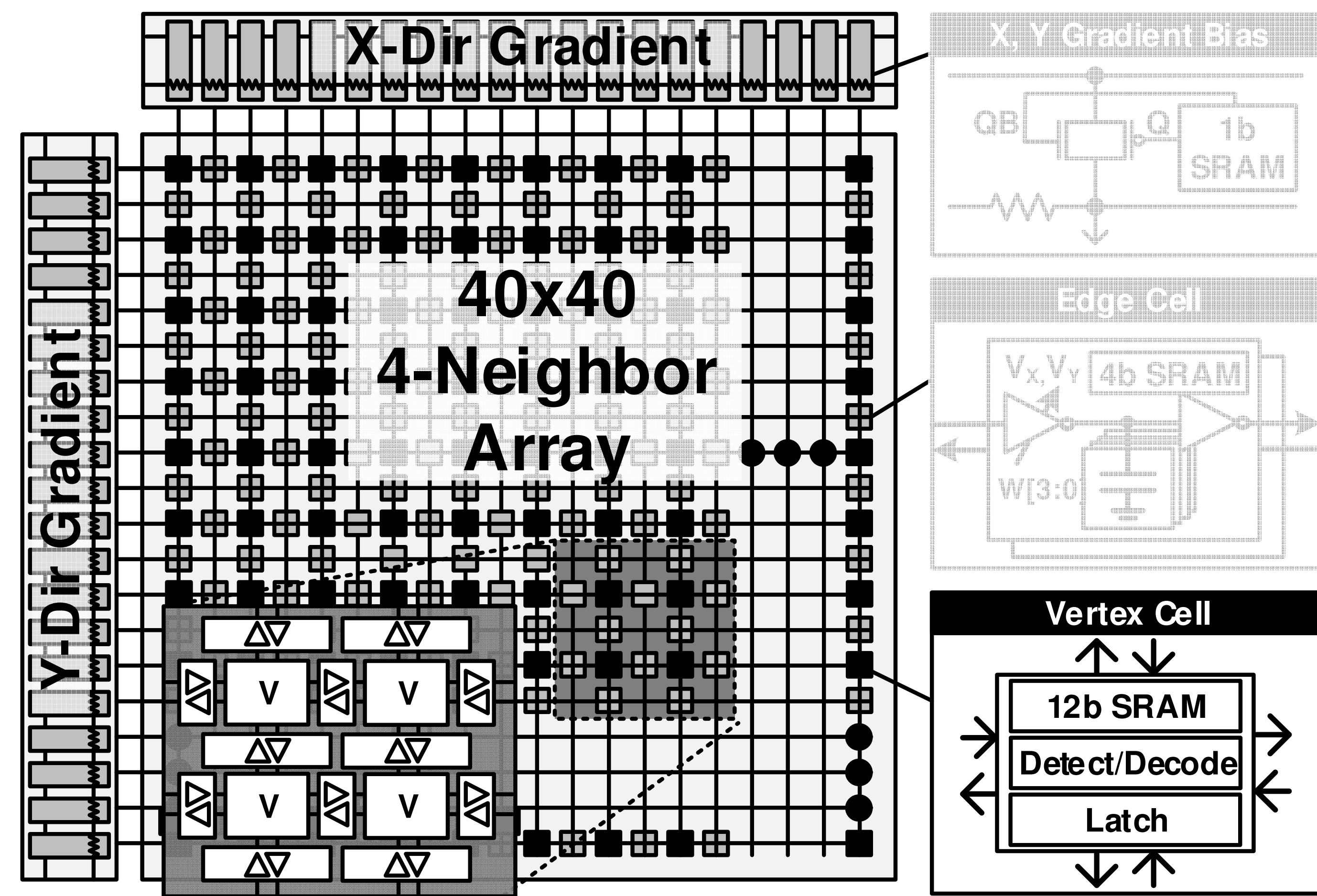


- **Time \propto Distance**
- **Each vertex stores first input**
- **Directly readout shortest path**
- **Multiple start points**

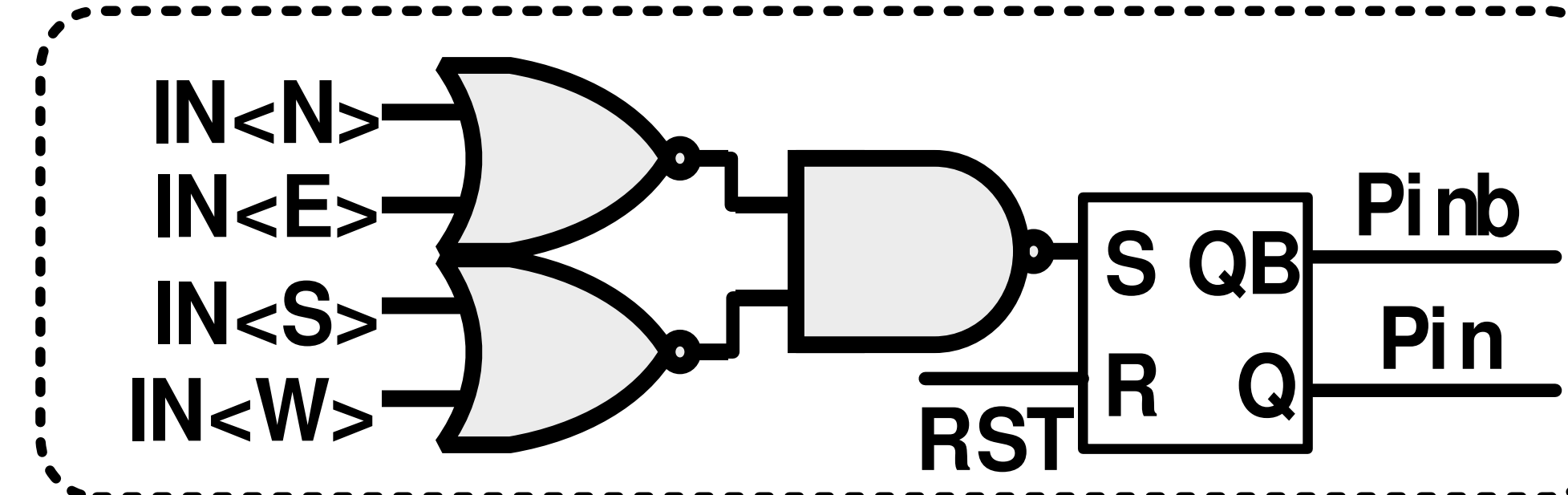
40x40 A* ASIC- Vertex



40x40 A* ASIC- Vertex

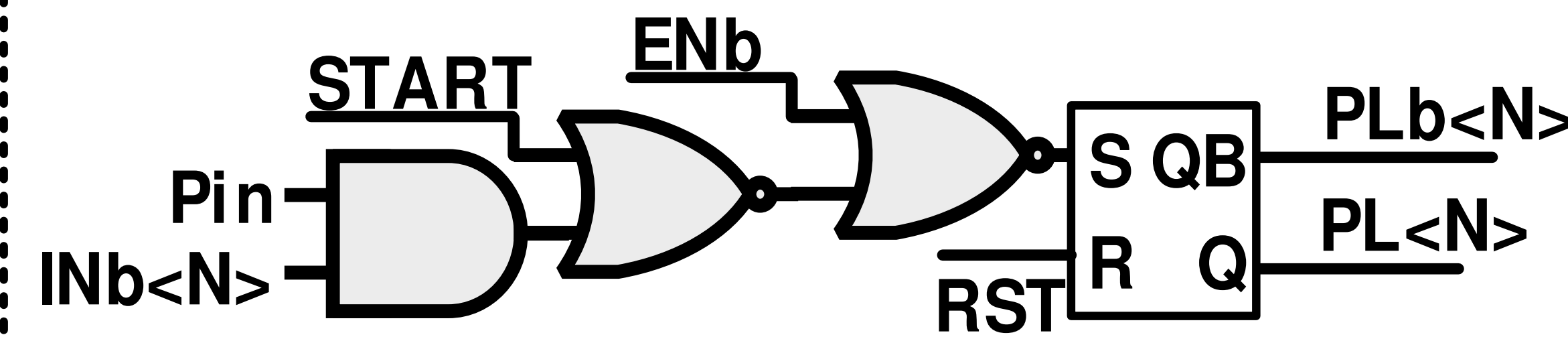


VERTEX LOCKOUT

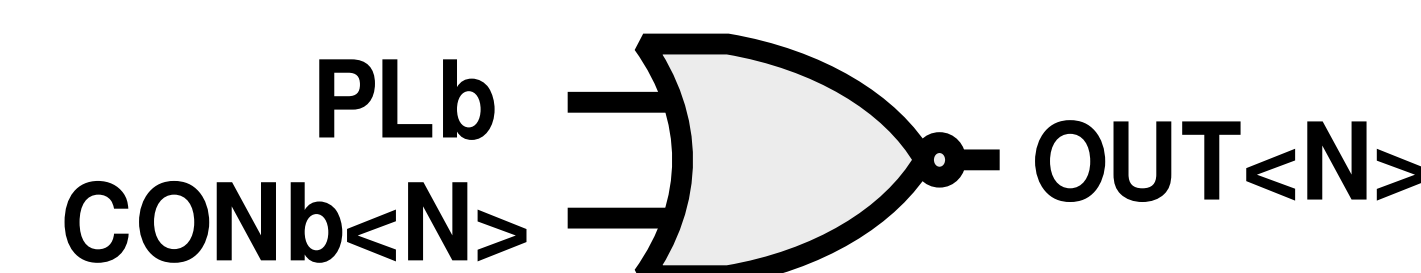


N,W,S,E DECODE

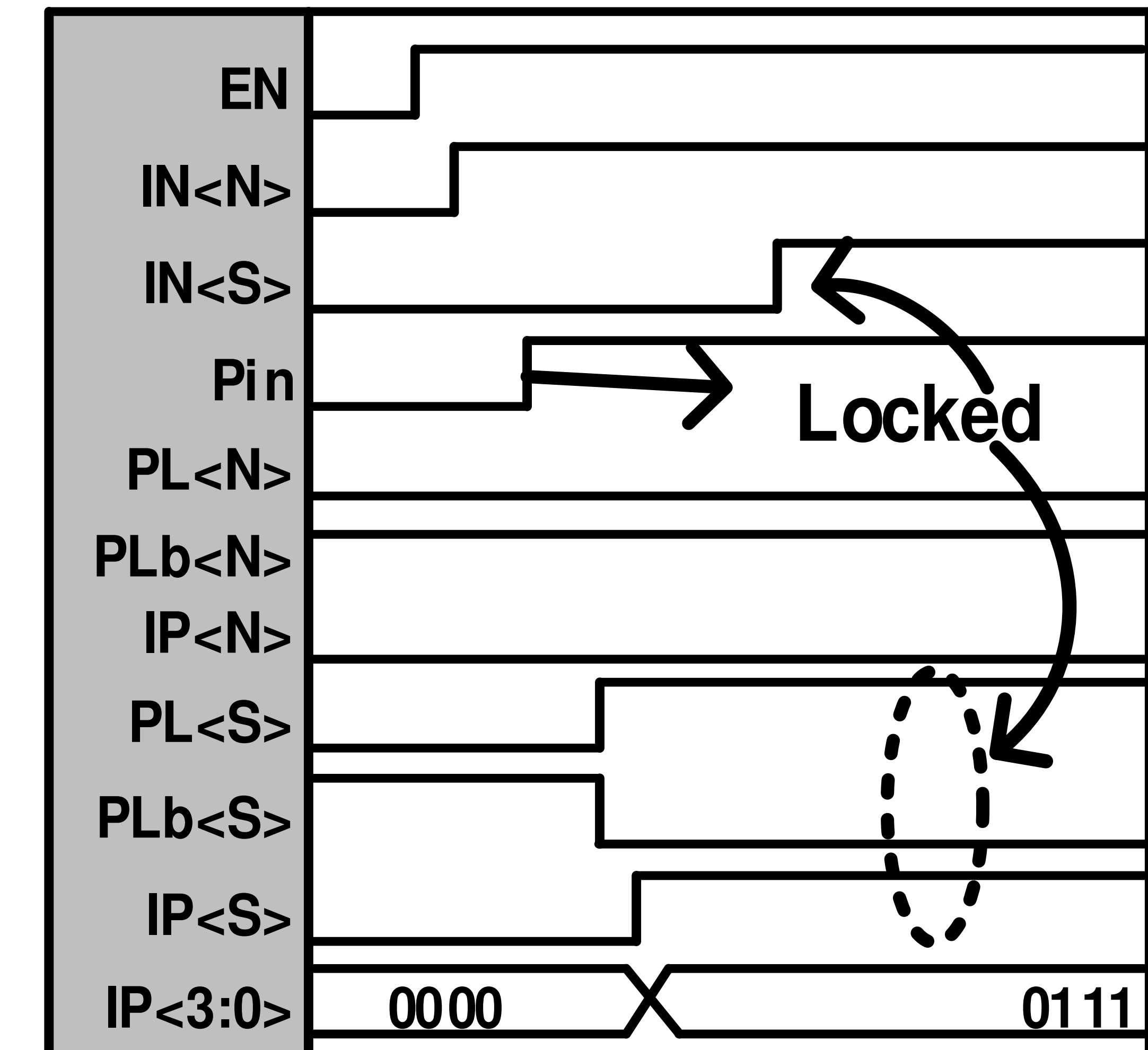
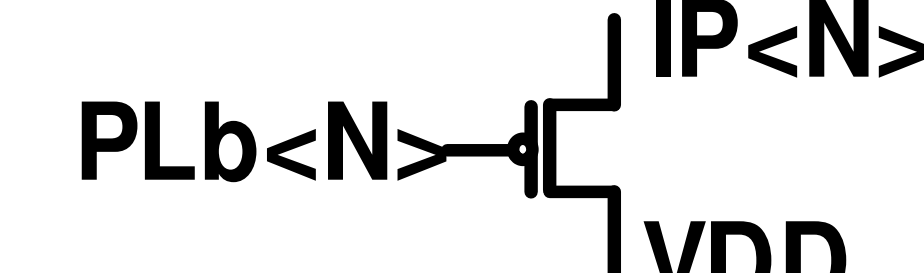
PULSE DIRECTION DETECTOR



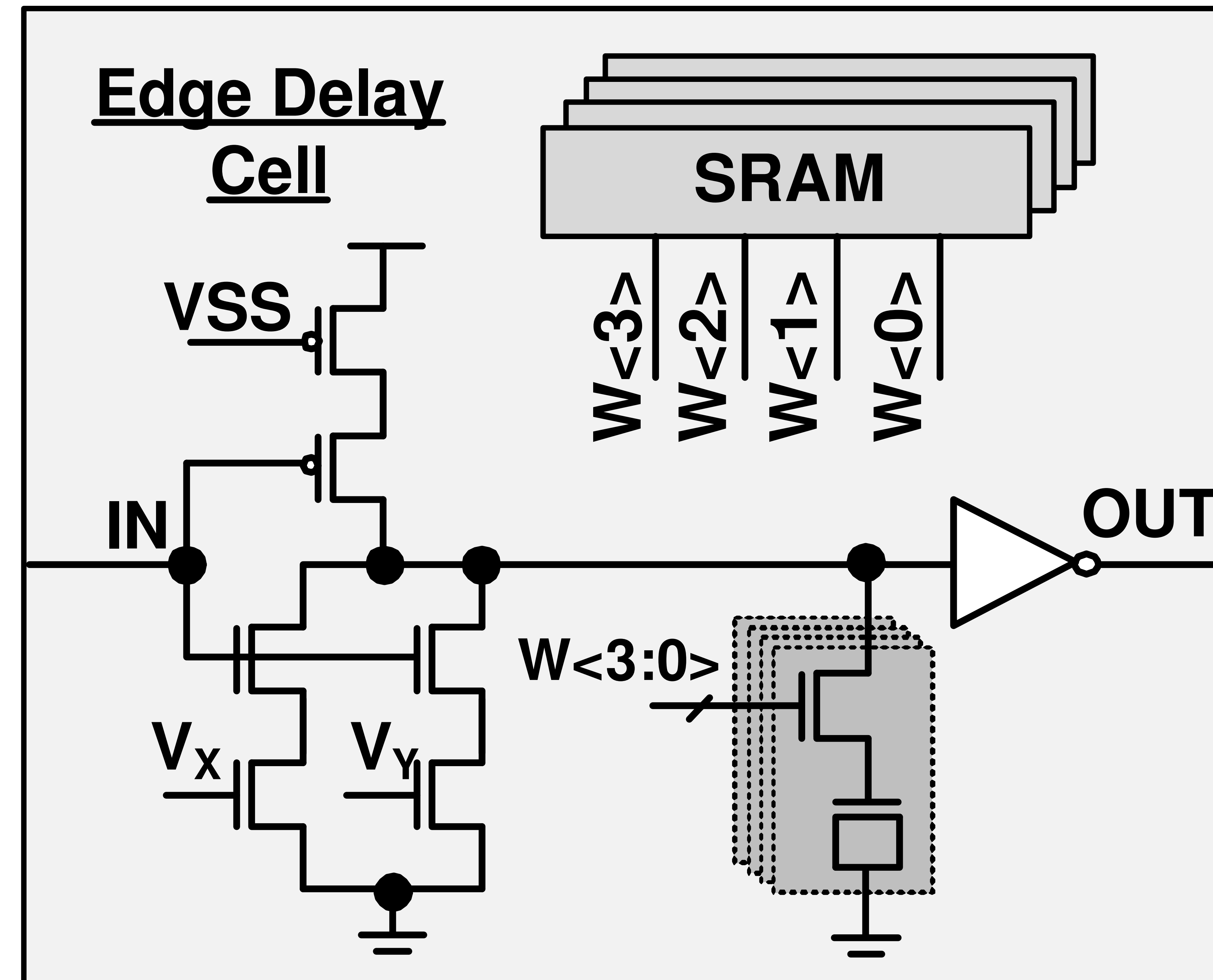
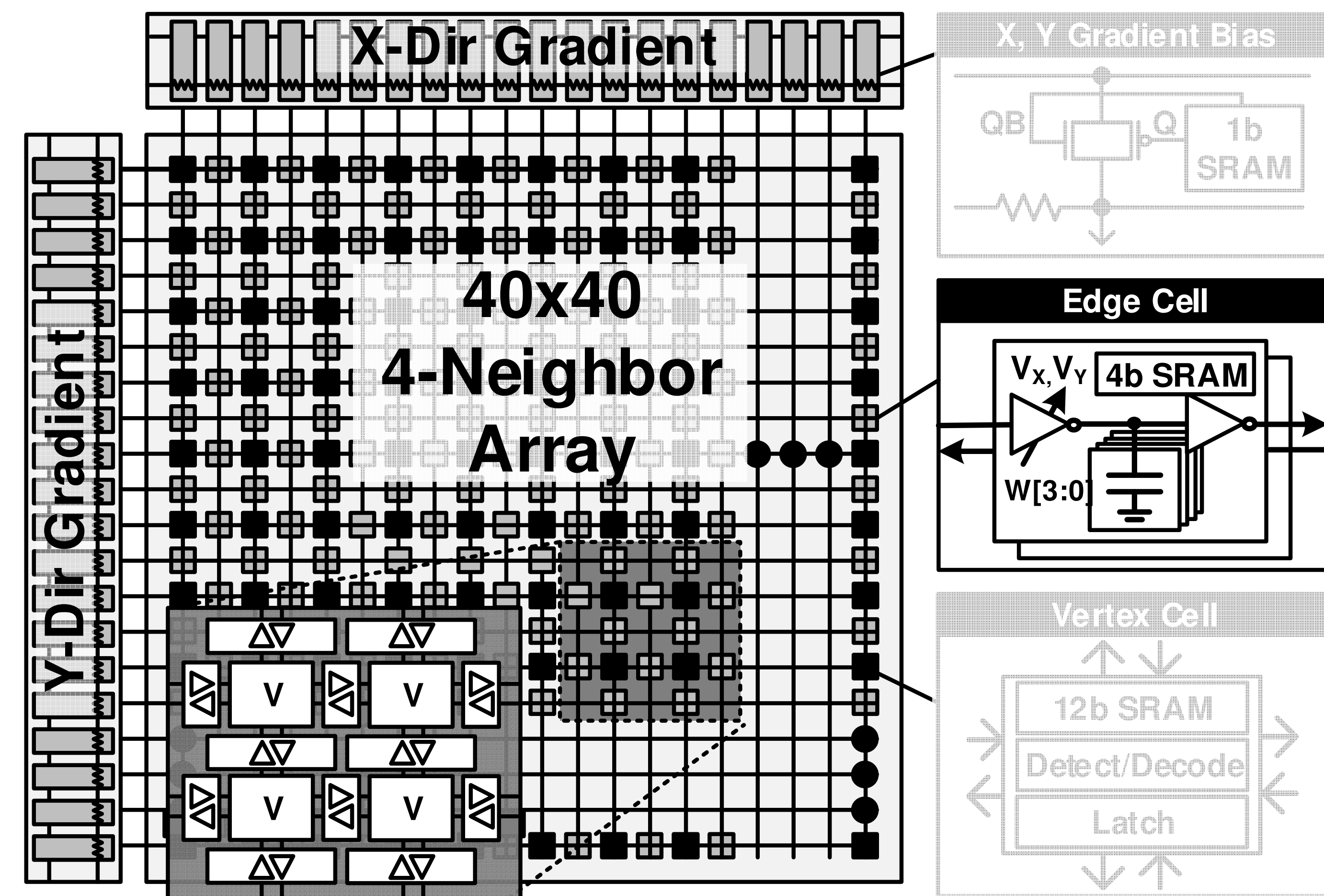
PULSE REPEATER



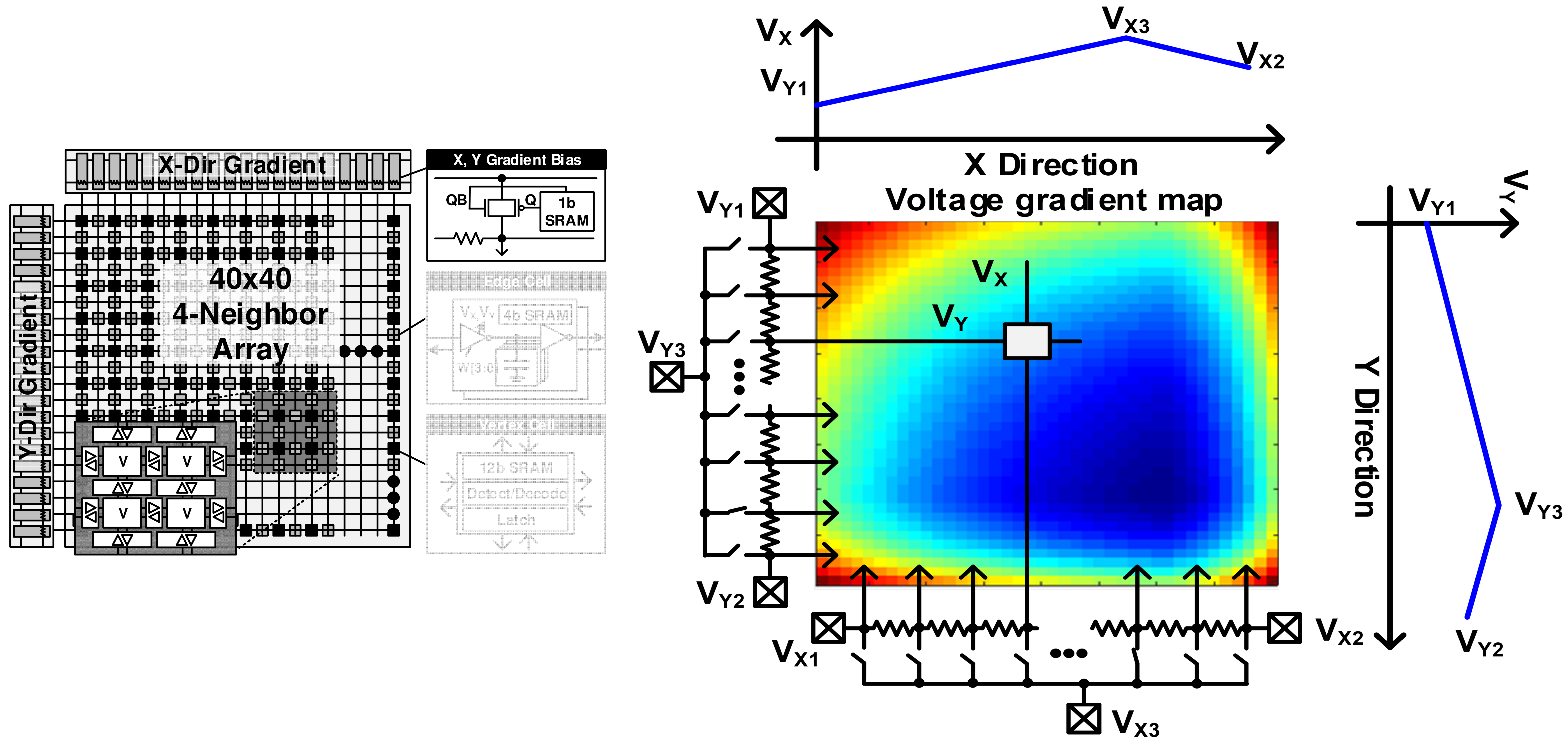
SRAM DMA



40×40 A* ASIC - Edge



40×40 A* ASIC- Gradient



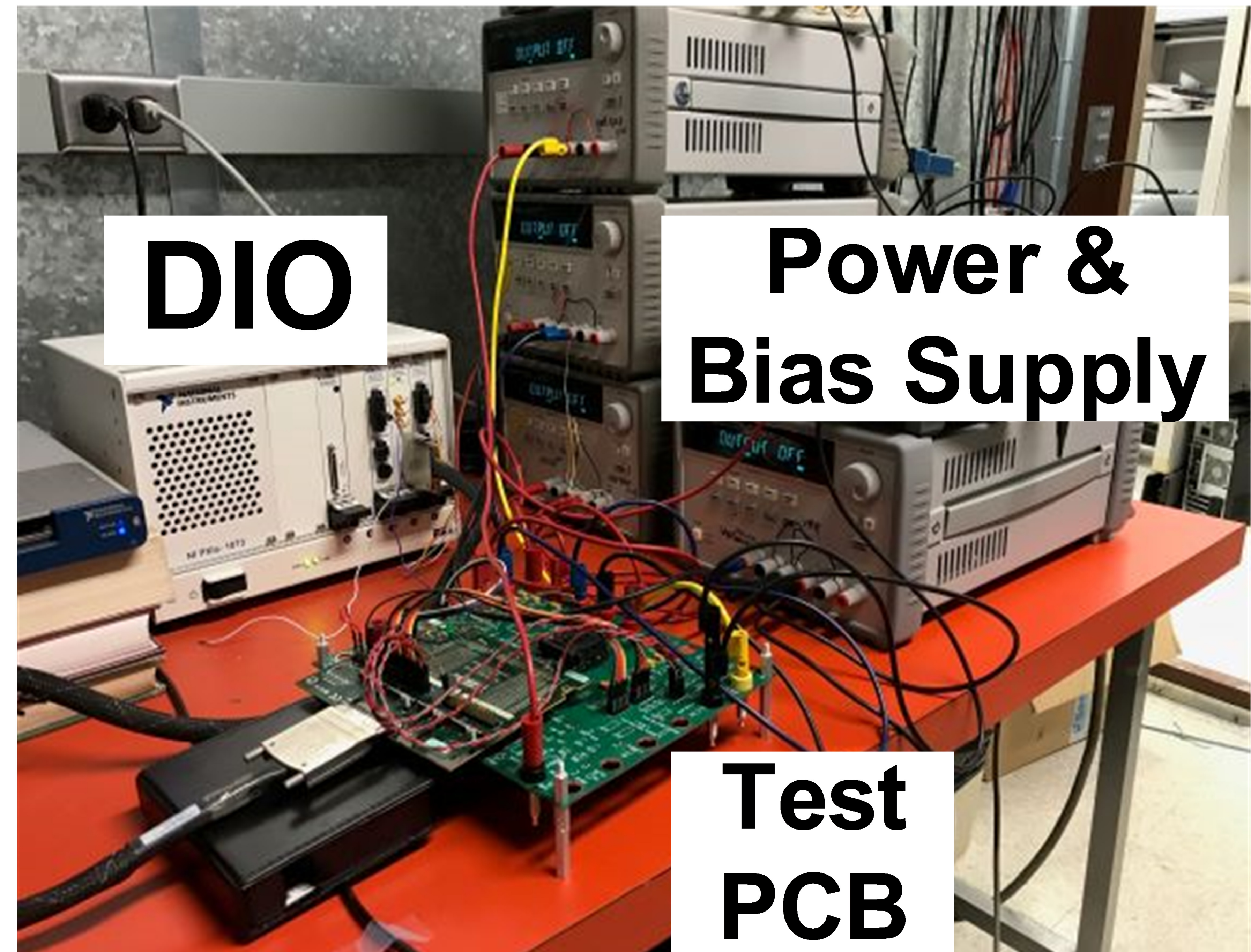
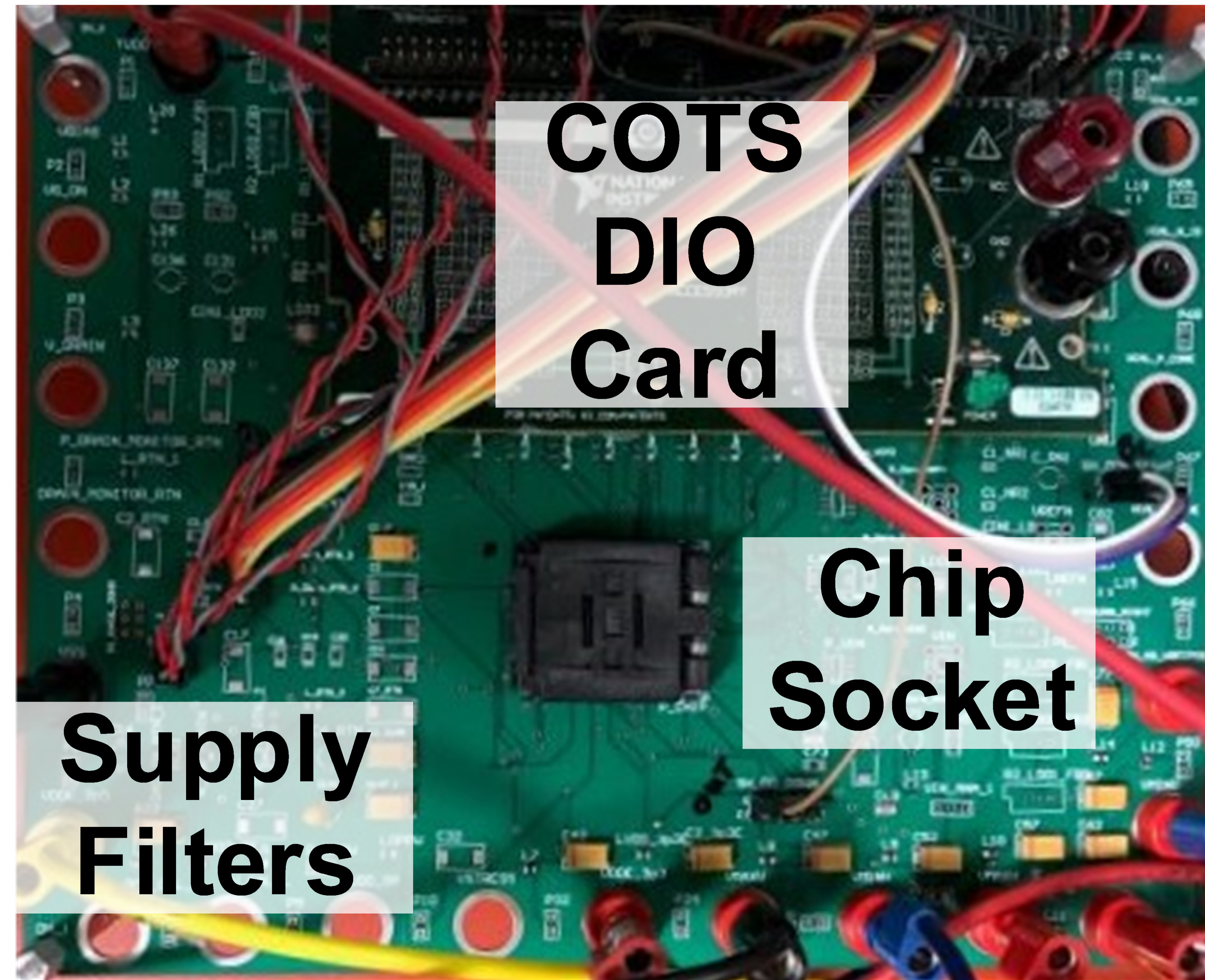
Outline

- Background: Path Planning Algorithms
- Time-Based A* ASIC
- **65nm Test Chip Results**
- Applications
- Conclusion

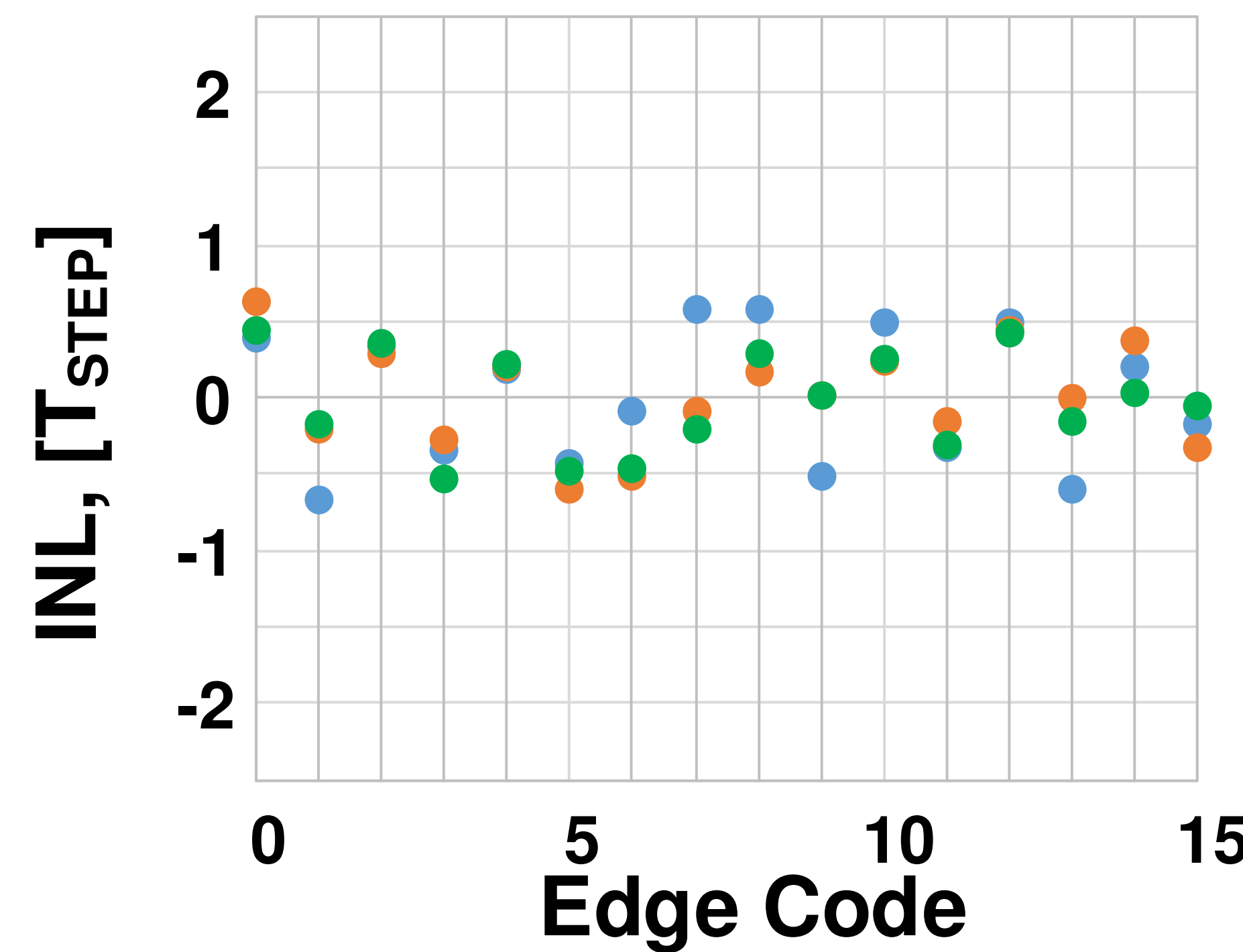
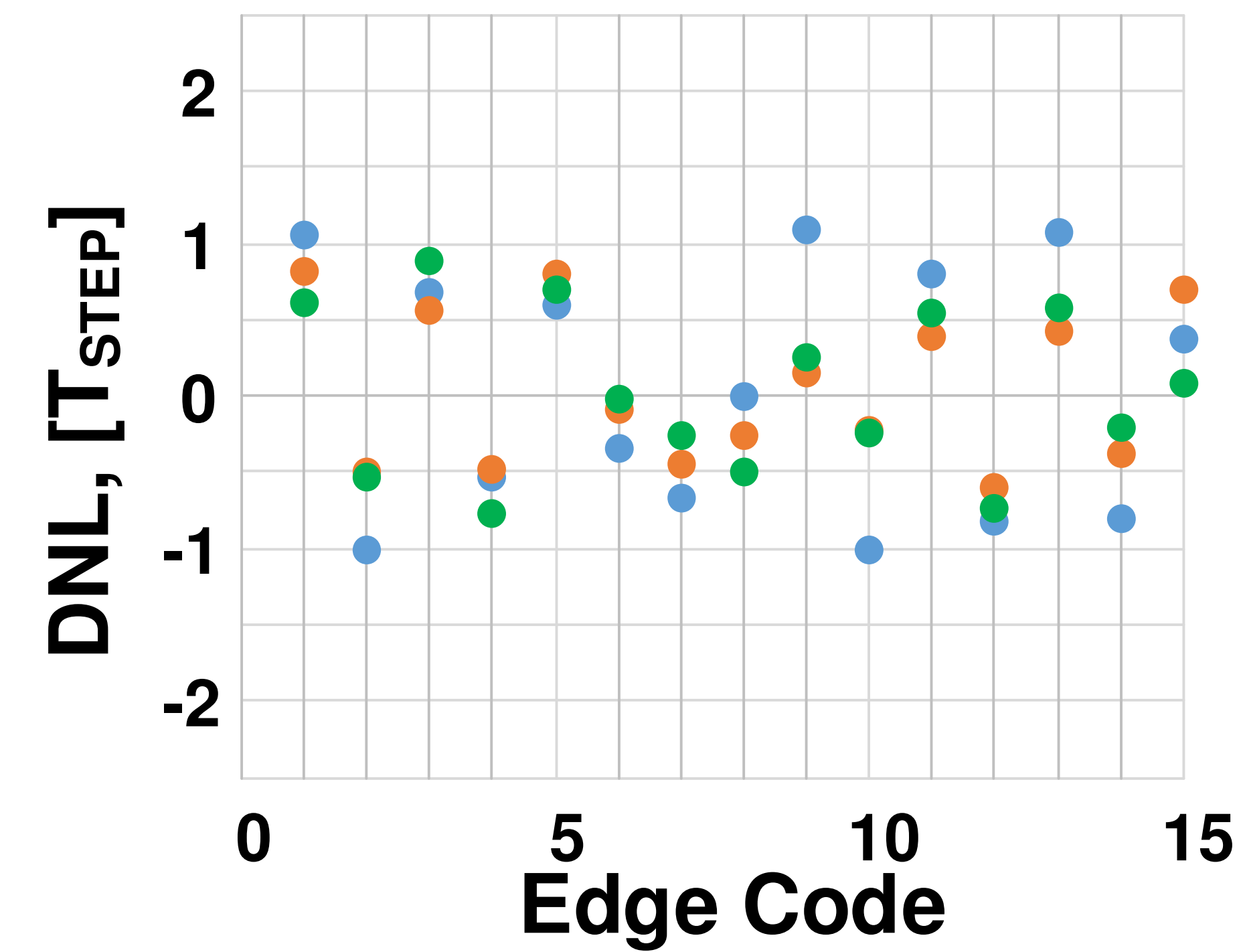
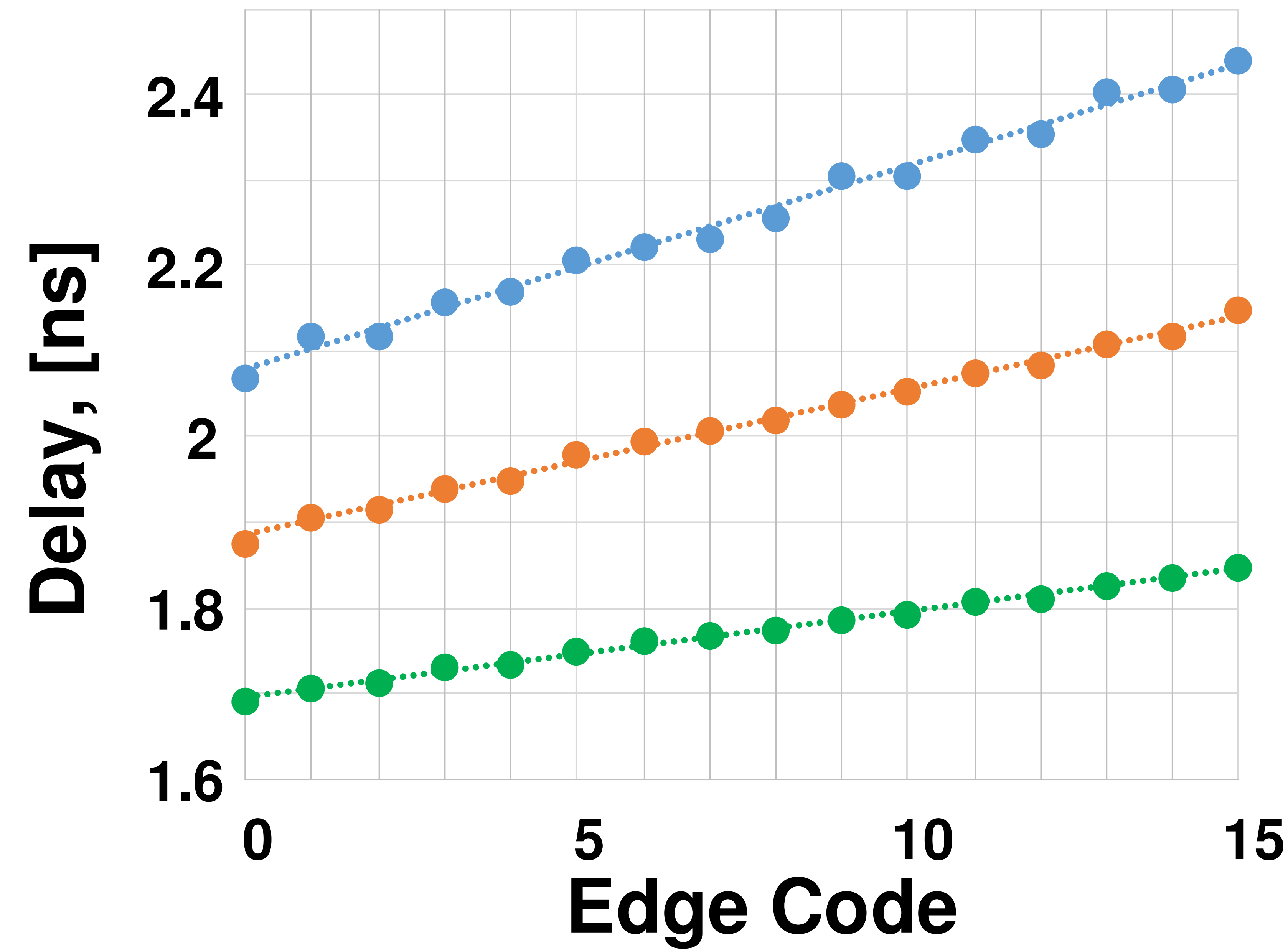
Test Setup

Custom Test PCB

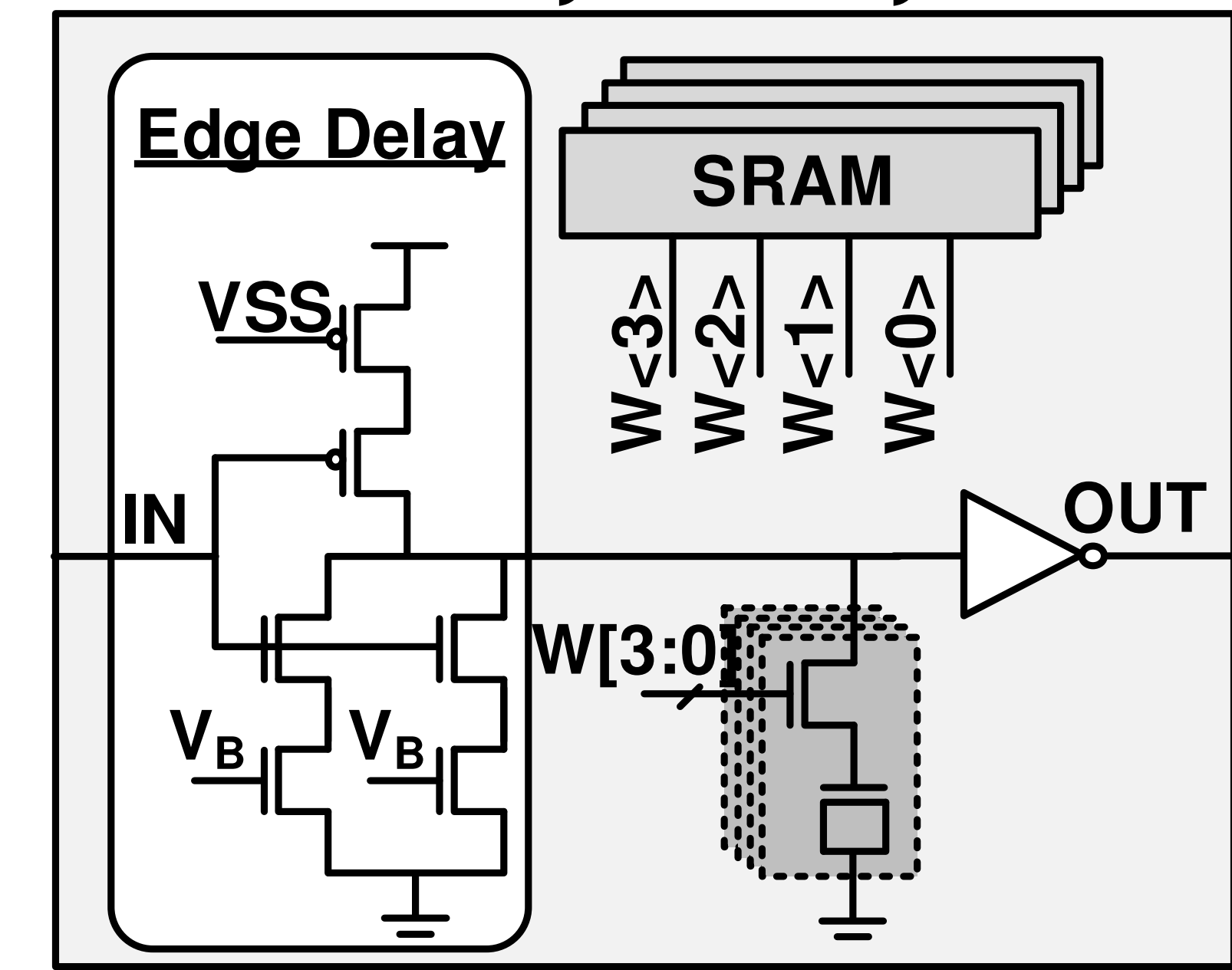
Test Environment



Edge Delay Linearity



65nmLP, 1.2v, 25°C



- $V_B = 0.75$
- $V_B = 0.80$
- $V_B = 0.90$

Comparison Table

Architecture	This Work	FPGA [4]	μ Processor	CPU [5]	GPU [5]
Product	ASIC	Xilinx Virtex	ARM Cortex-M0	Intel Xeon E5630	NVIDIA Tesla K20c
Technology	65nm	20nm	40nm	32nm	28nm
Voltage	1.2V	-	1.1V	0.7-1.35V	-
Peak Power	26.4mW	24.22W	127 μ W	20W/core	225W
Throughput [MTEPS]	559	731	5.34*10 ⁻⁴	0.83	9.0
Energy per Node	0.328pJ*	33nJ	89.1nJ	24.1 μ J	25 μ J
Normalized Energy	1x	10 ⁵	2.7x10 ⁵	1.19x10 ⁶	2.3x10 ⁷

*55% from SRAM Program (does not include cache access energy)

Energy/Node=Unit Delay*Unit Power

MTEPS = Million Traversed Edges Per Second

[4]S. Zhou, *et al*, "High-Throughput and Energy-Efficient Graph Processing on FPGA," *IEEE International Symp. on Field-Programmable Custom Computing Machines*, pp. 103-110, 2016.

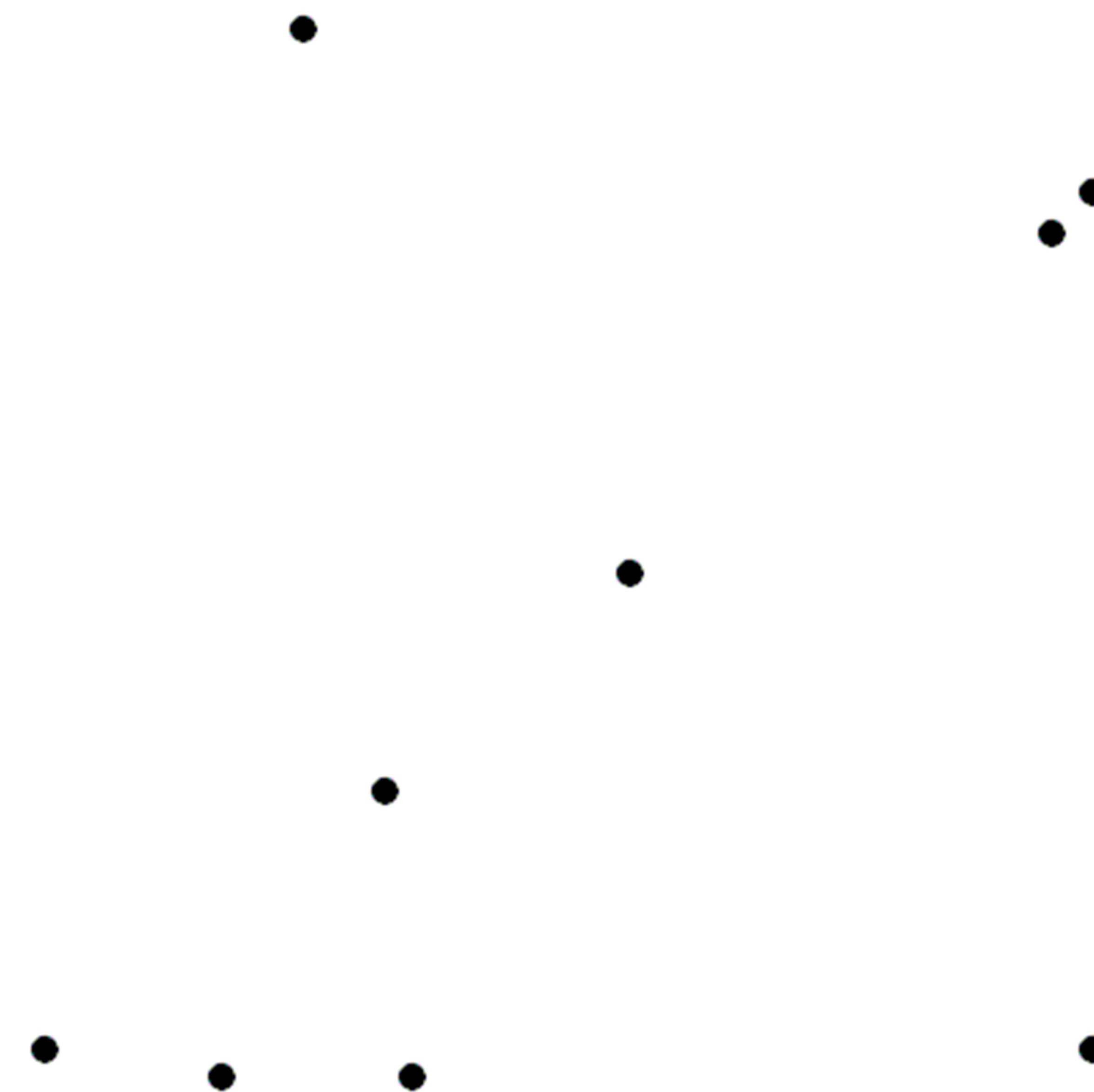
[5] Y. Zhou and J. Zeng, "Massively Parallel A* Search on a GPU," *Conference on Artificial Intelligence (AAAI)*, pp. 1248-1254, 2015.

Outline

- Background: Path Planning Algorithms
- Time-Based A* ASIC
- 65nm Test Chip Results
- **Applications**
- Conclusion

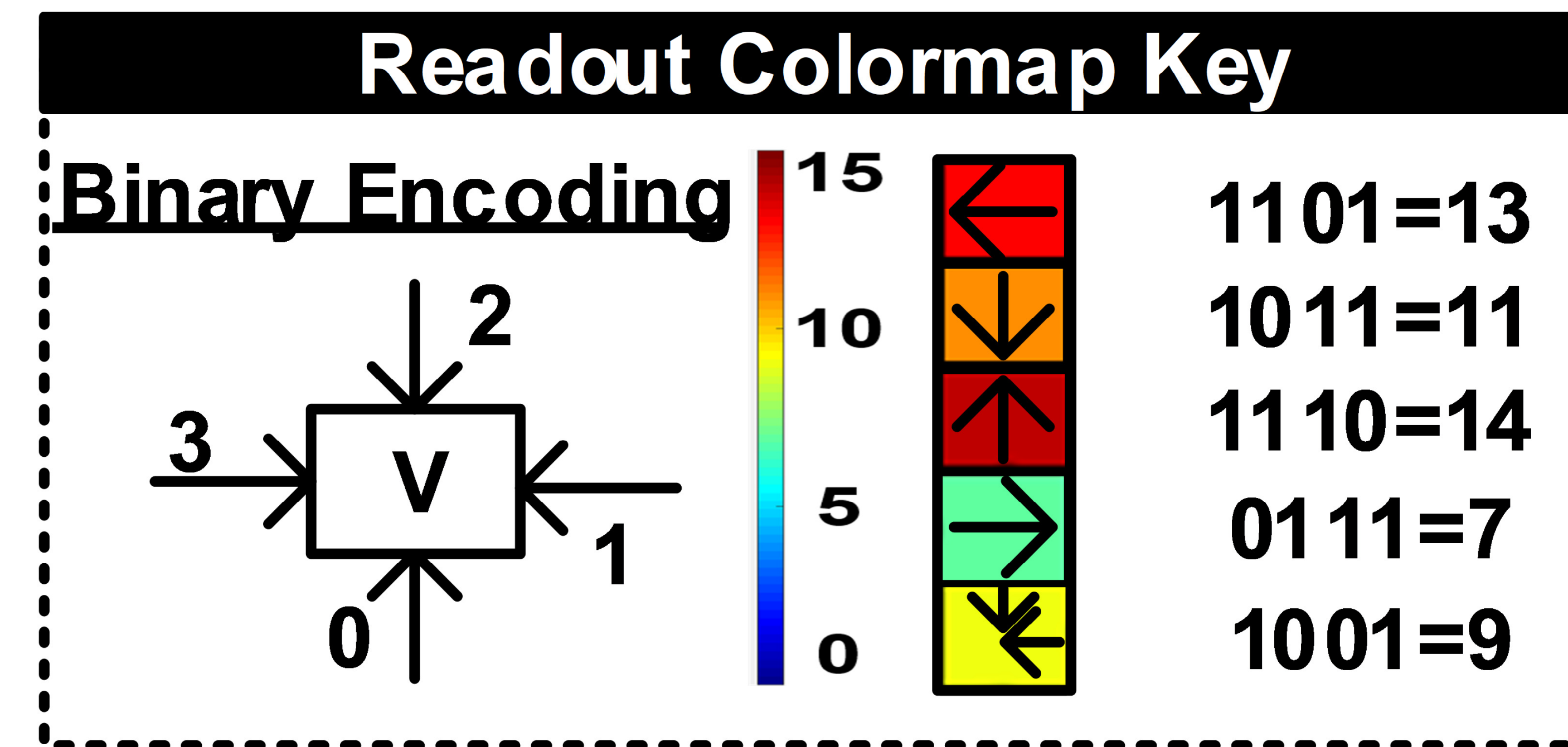
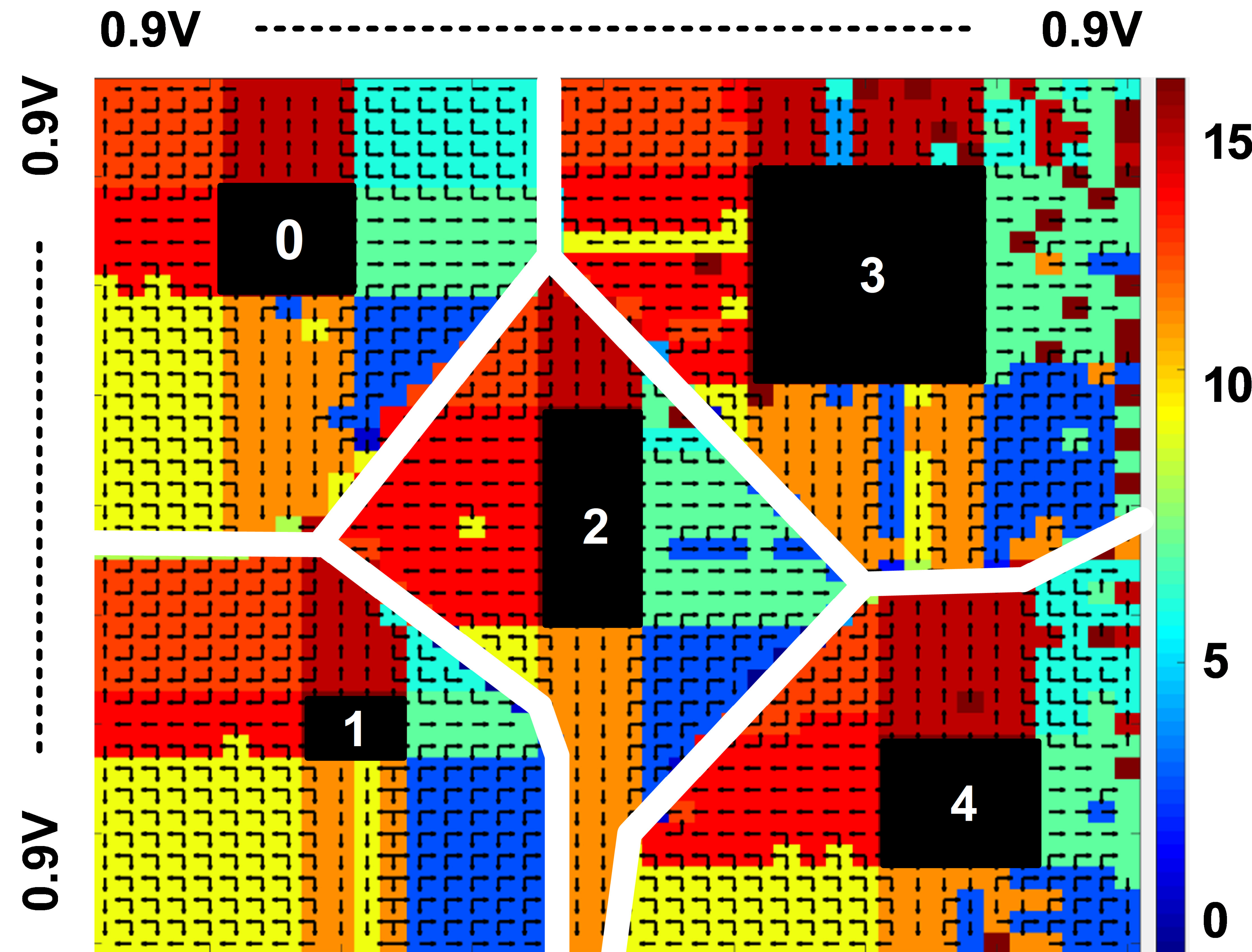
Voronoi Diagrams

- **Segmentation of plane such that distance to seeds is maximized**
- **Applications:**
 - **kNN classification**
 - **Biological structures (bone and cells)**
 - **Computational fluid dynamics meshes**
 - **Autonomous vehicles**

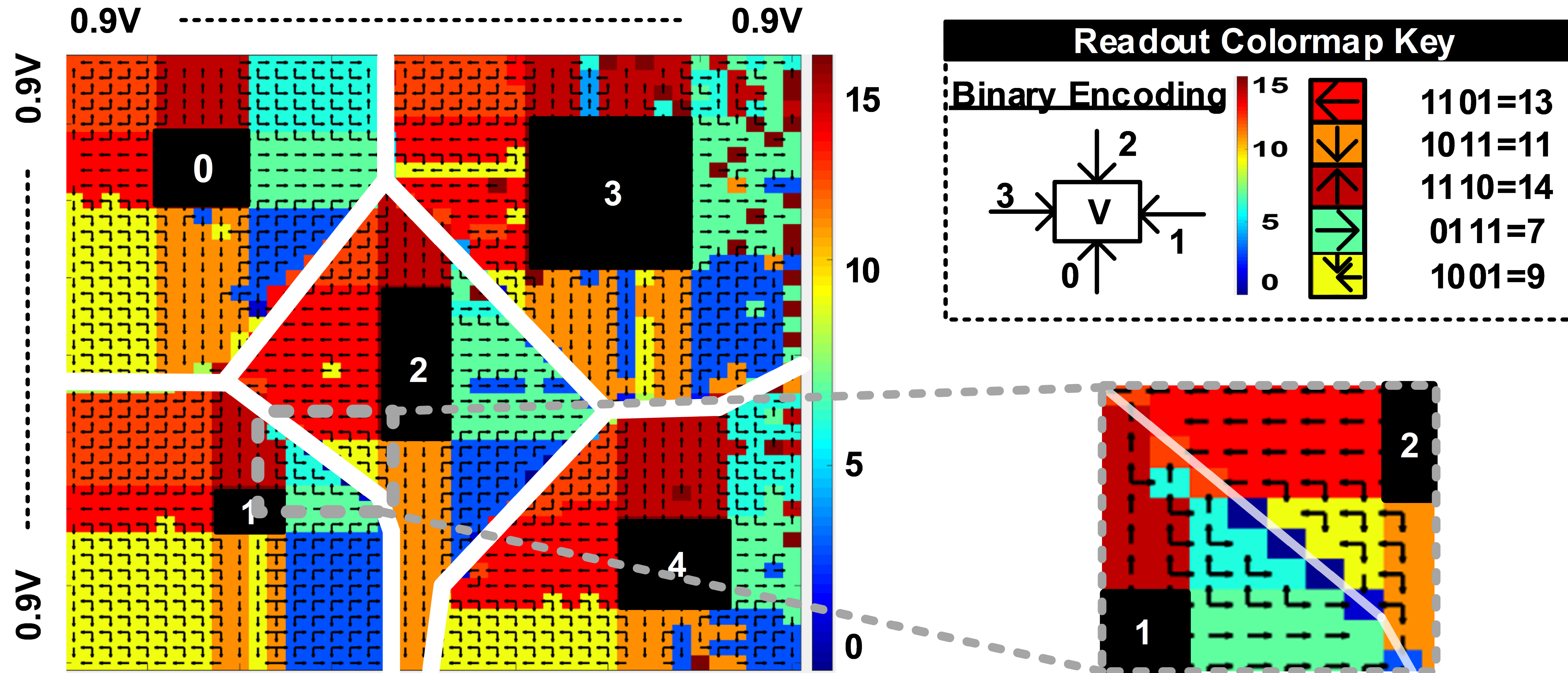


[Wikipedia]

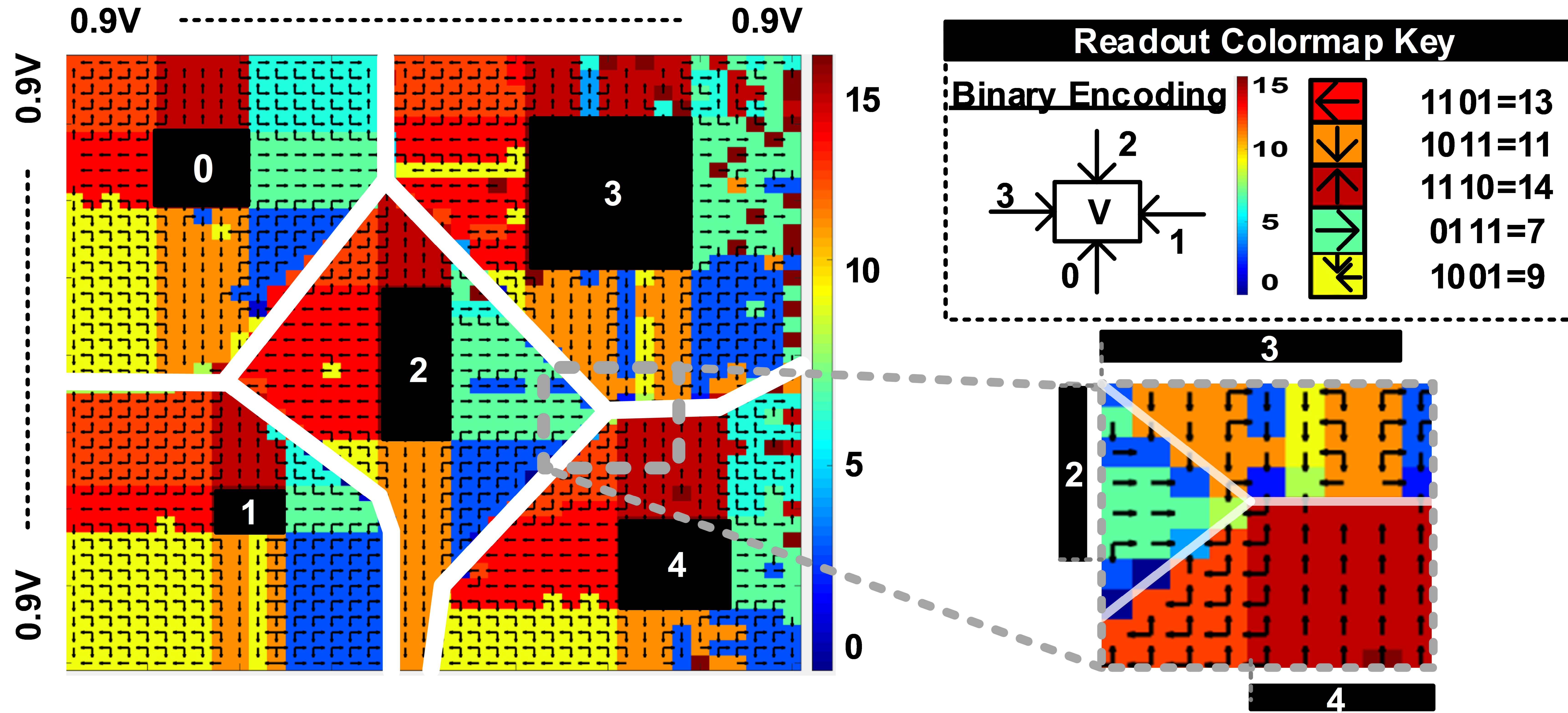
Collision Avoidance



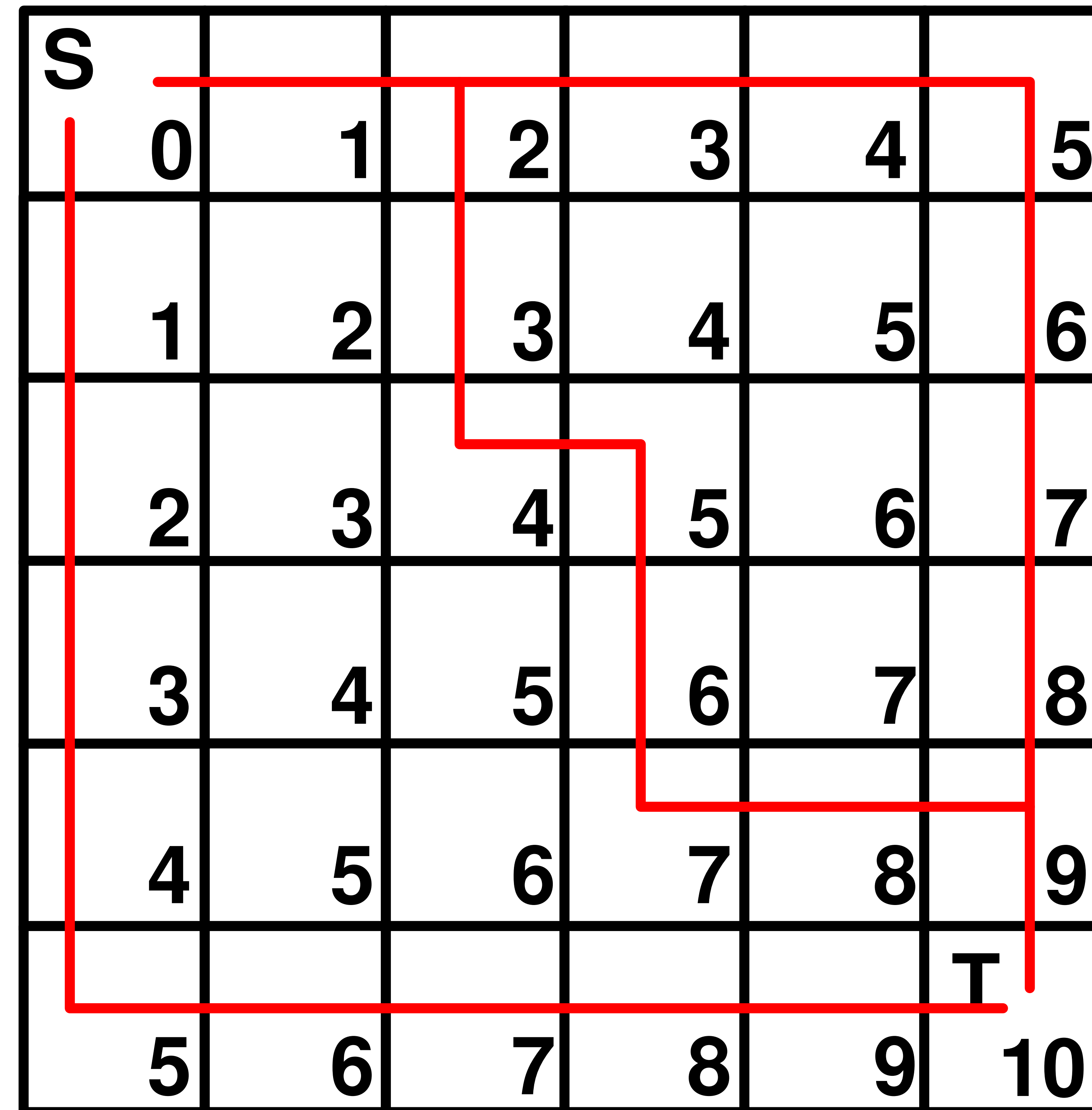
Collision Avoidance



Collision Avoidance

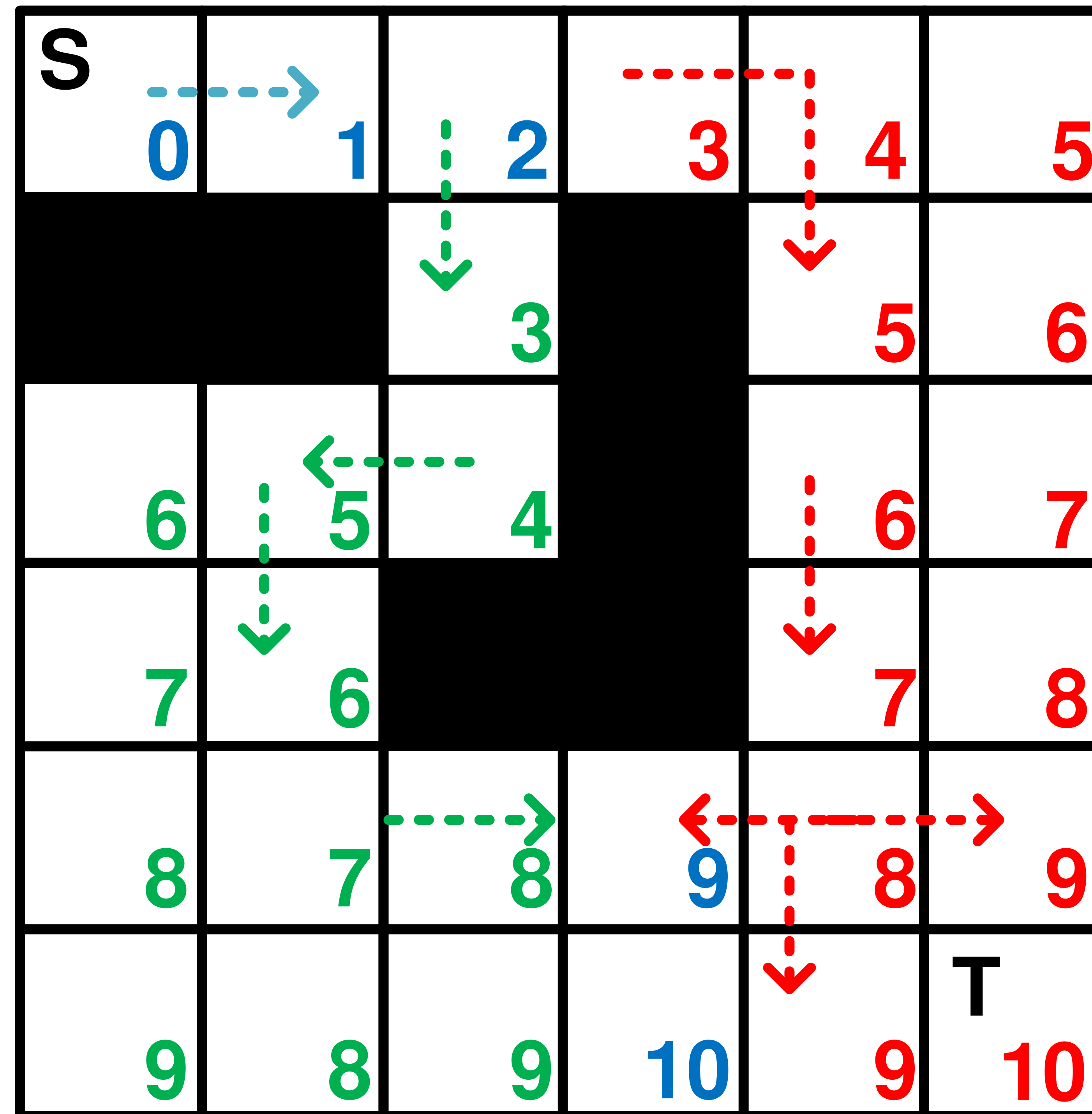


Path Planning - Dijkstra's



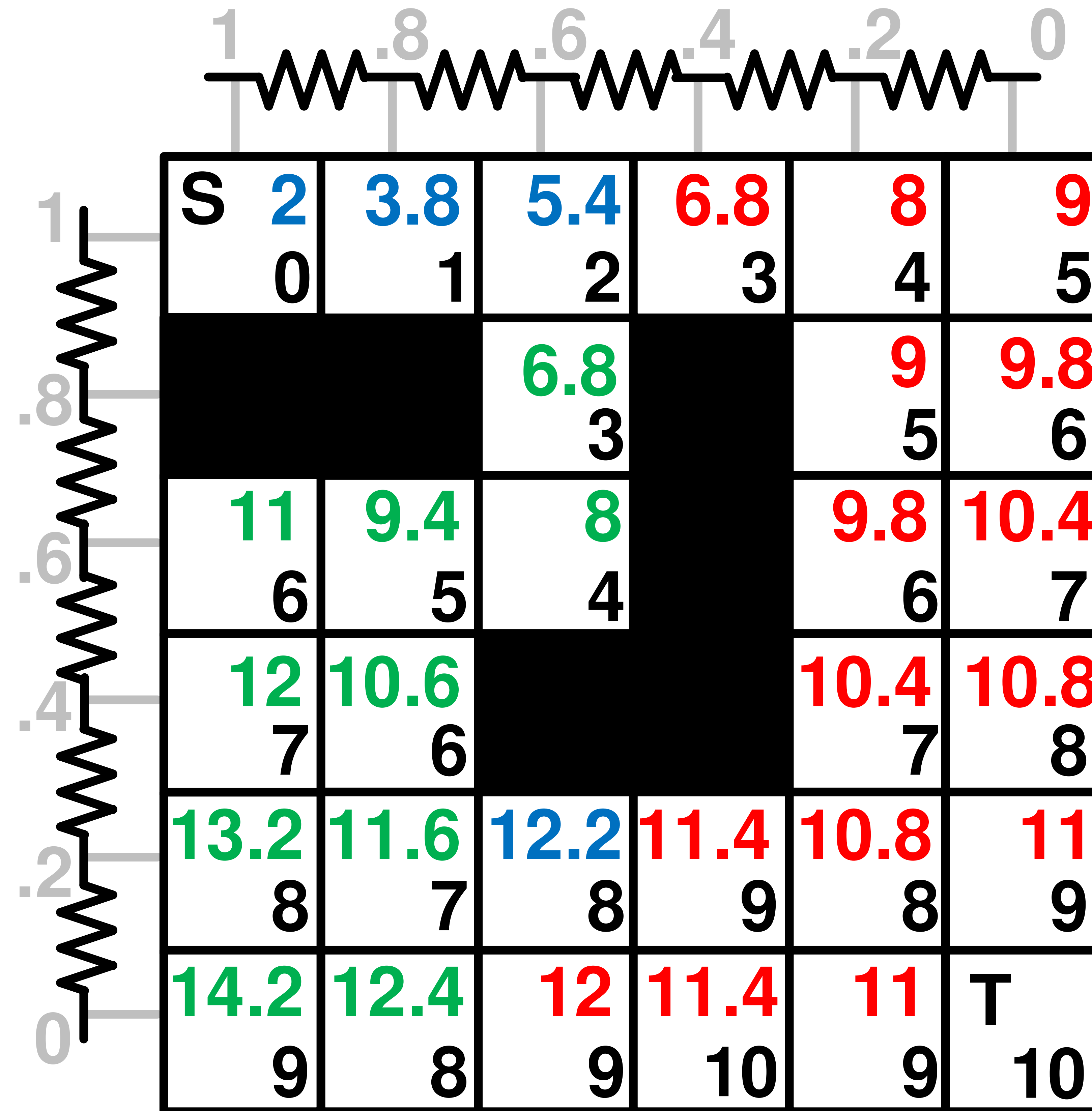
**All diagonal
paths are
equal**

Path Planning - Dijkstra's



**Paths above
blockage
dominate
paths below**

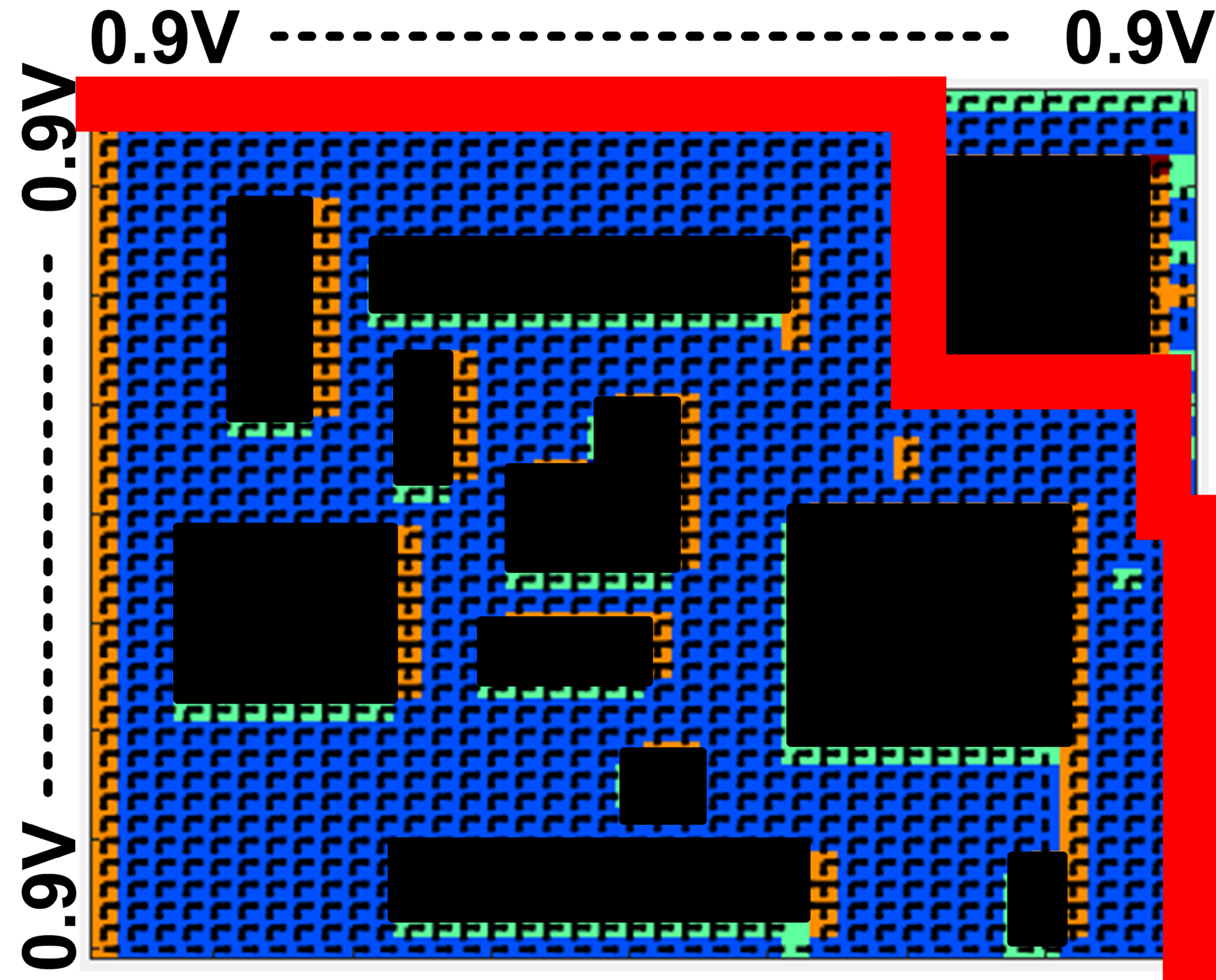
Path Planning – Gradient A*



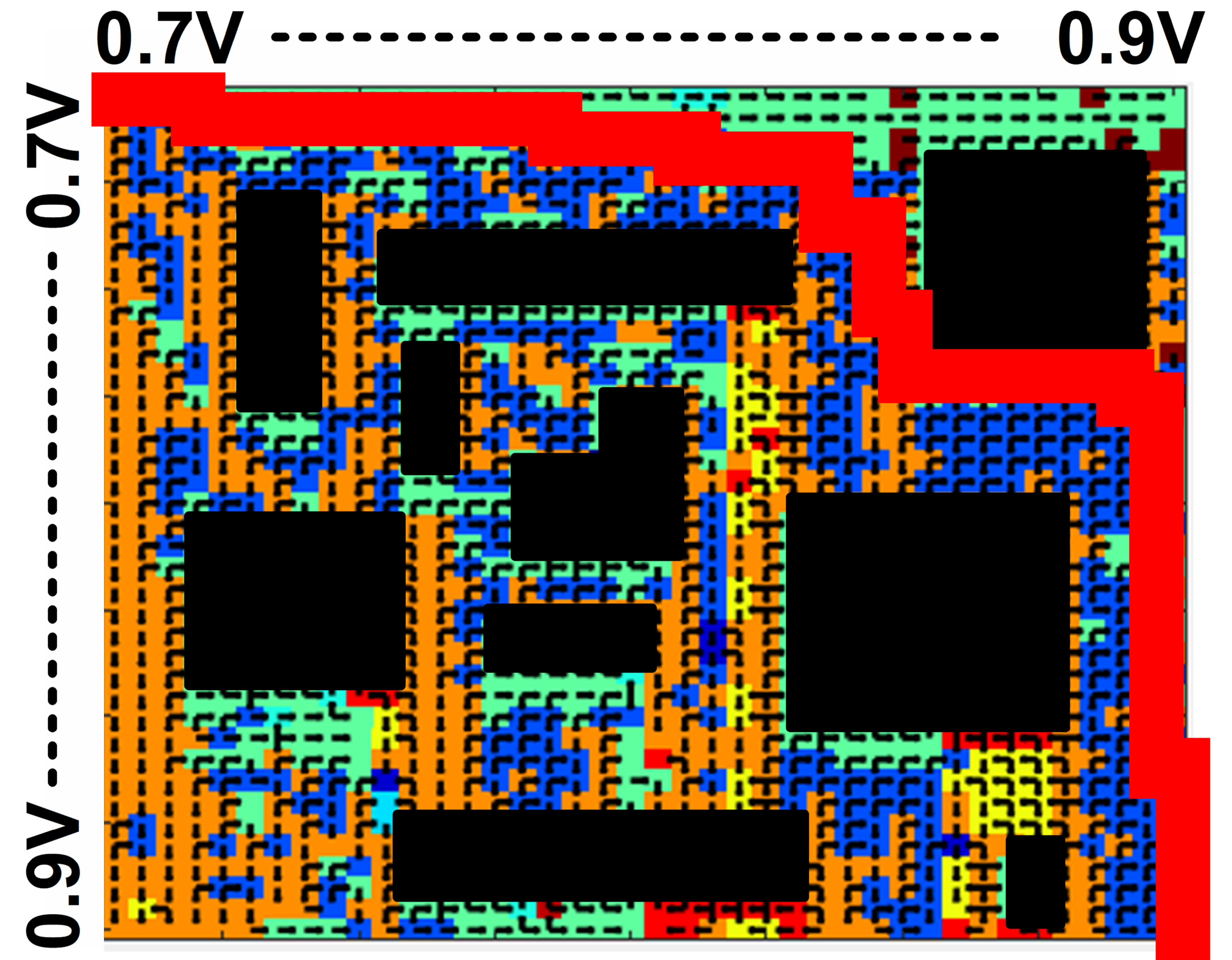
Gradient
improves
dominant
 path

Path Planning

Finding shortest path w/o gradient



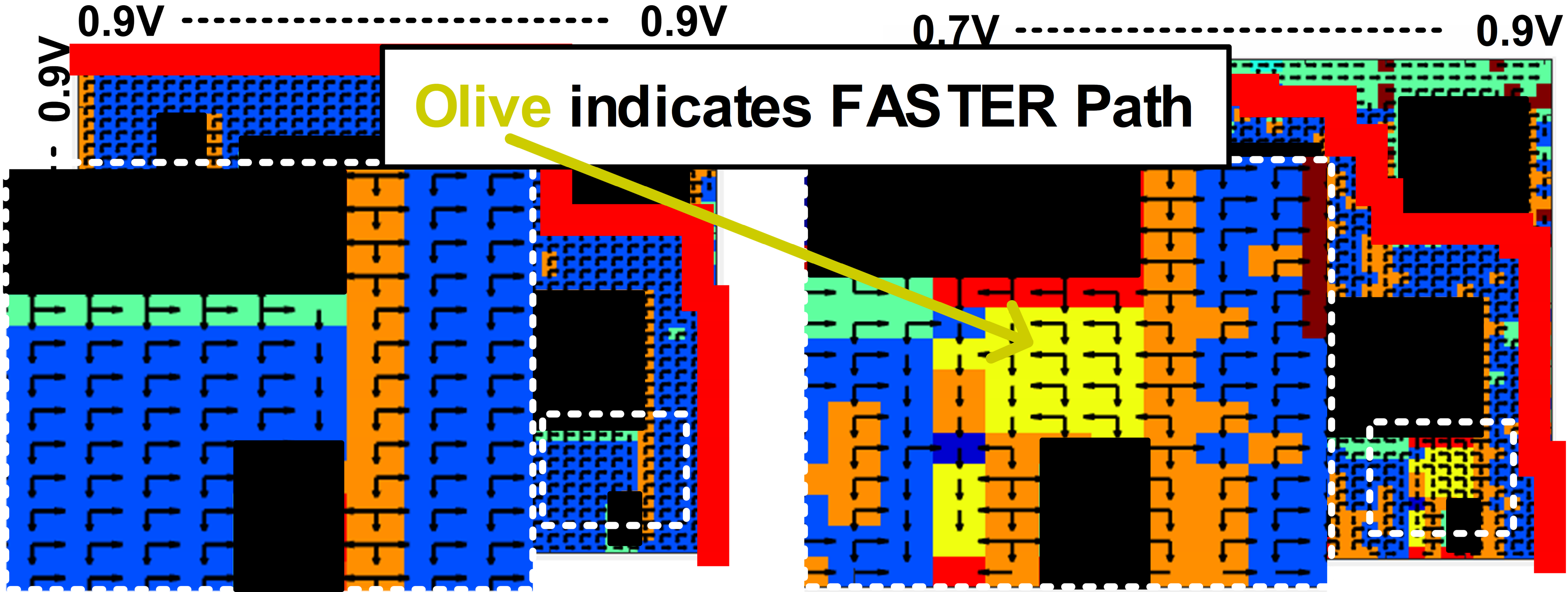
Finding shortest path w/ gradient



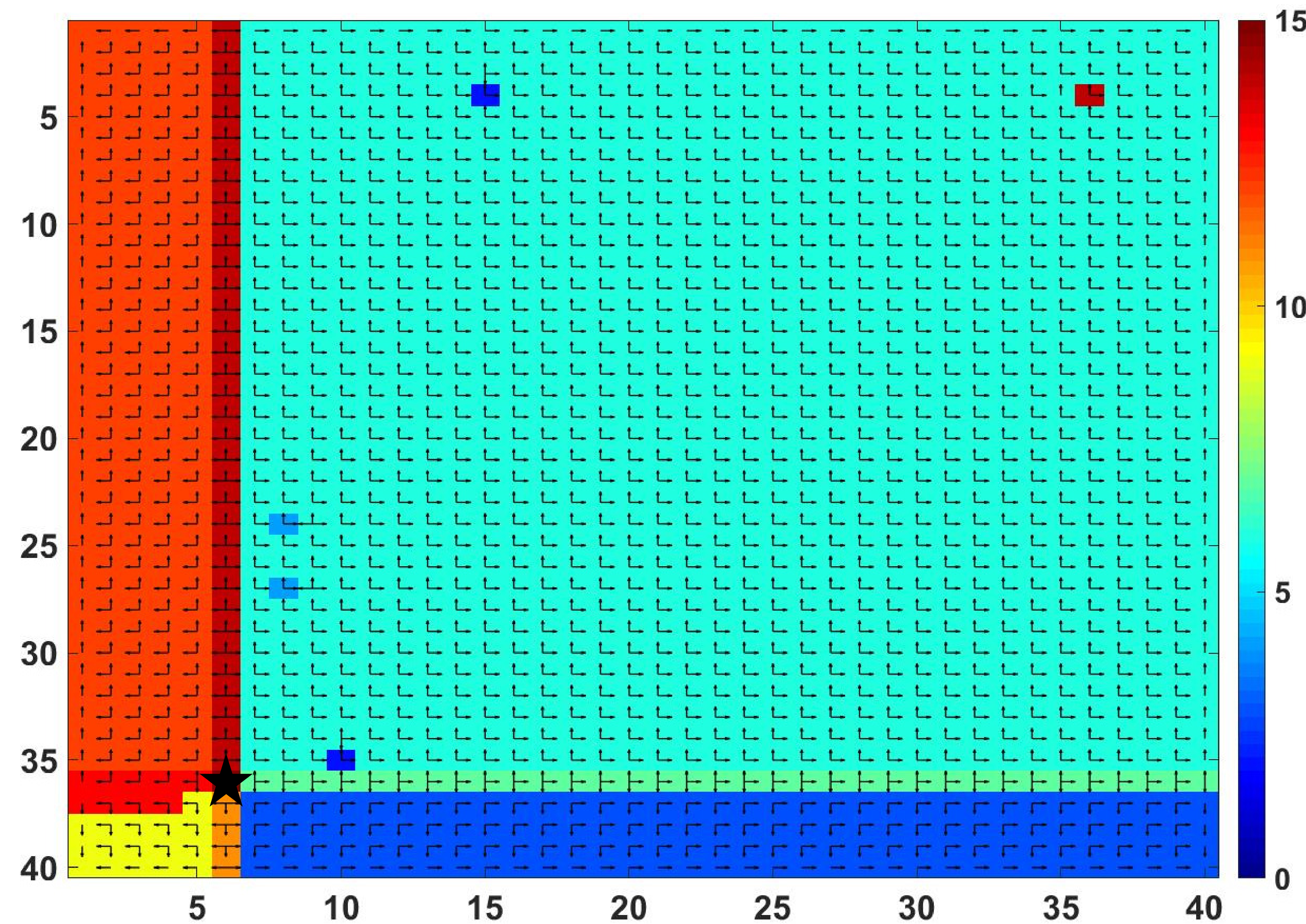
Path Planning

Finding shortest path w/o gradient

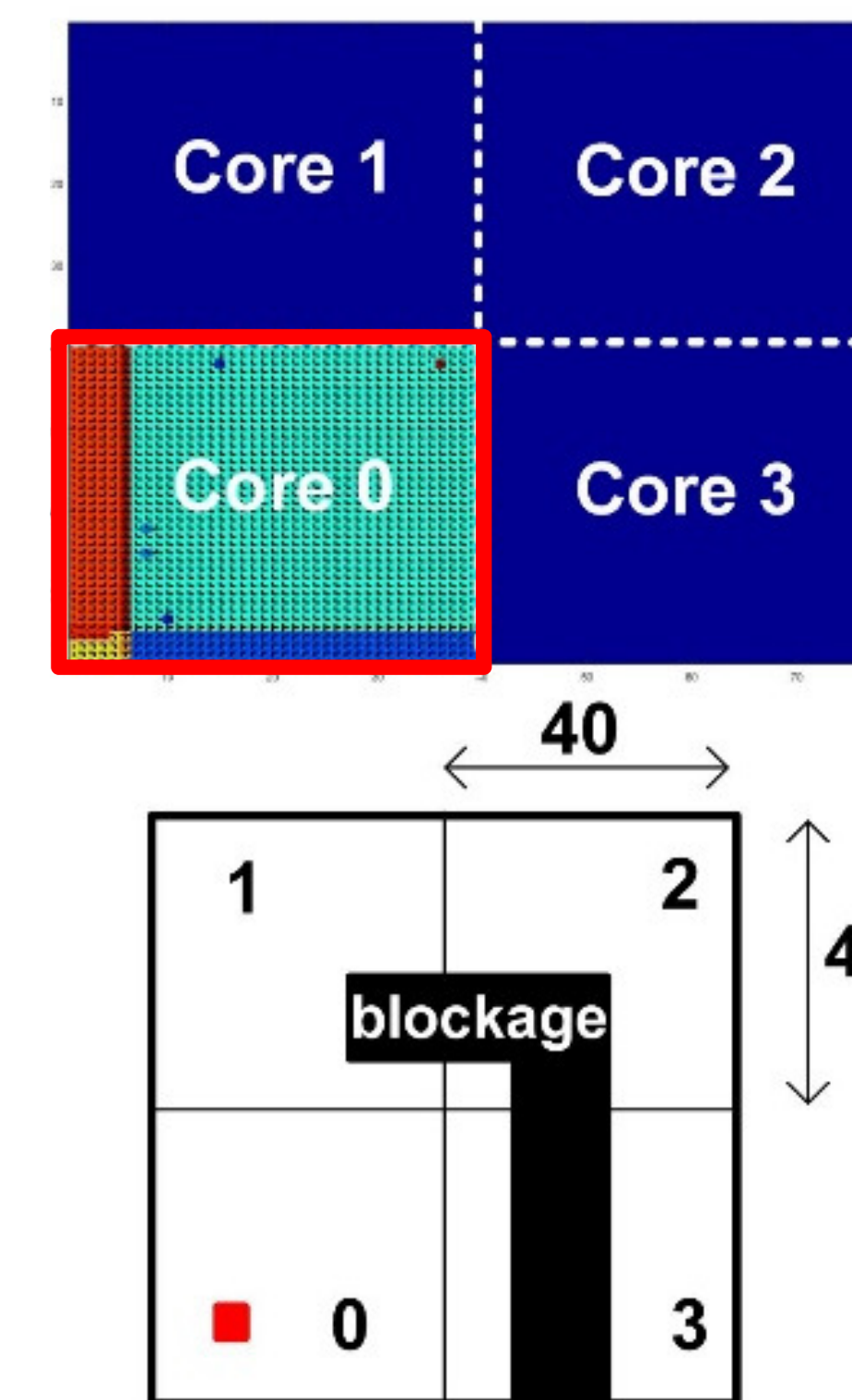
Finding shortest path w/ gradient



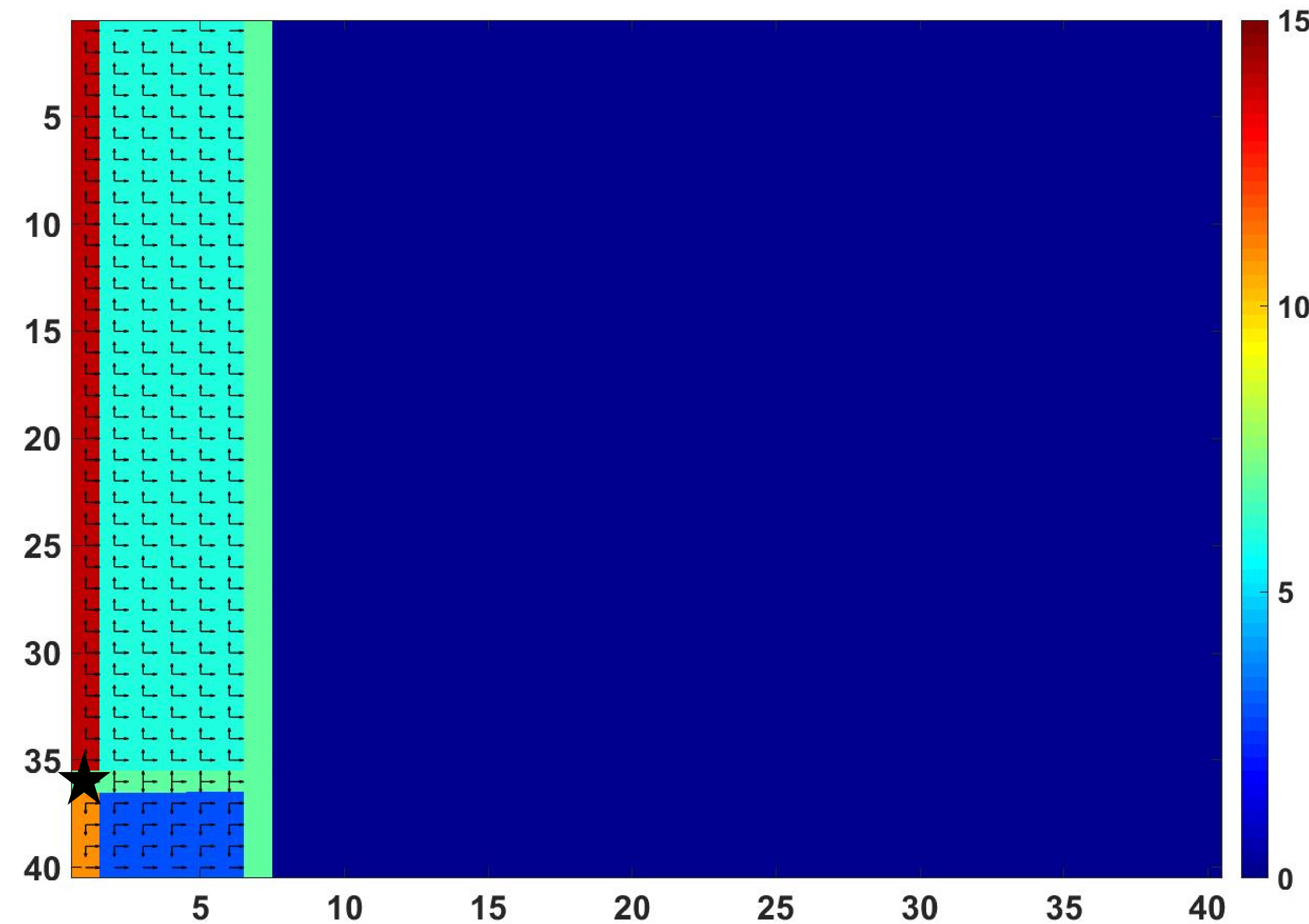
Multicore Evaluation (1/5)



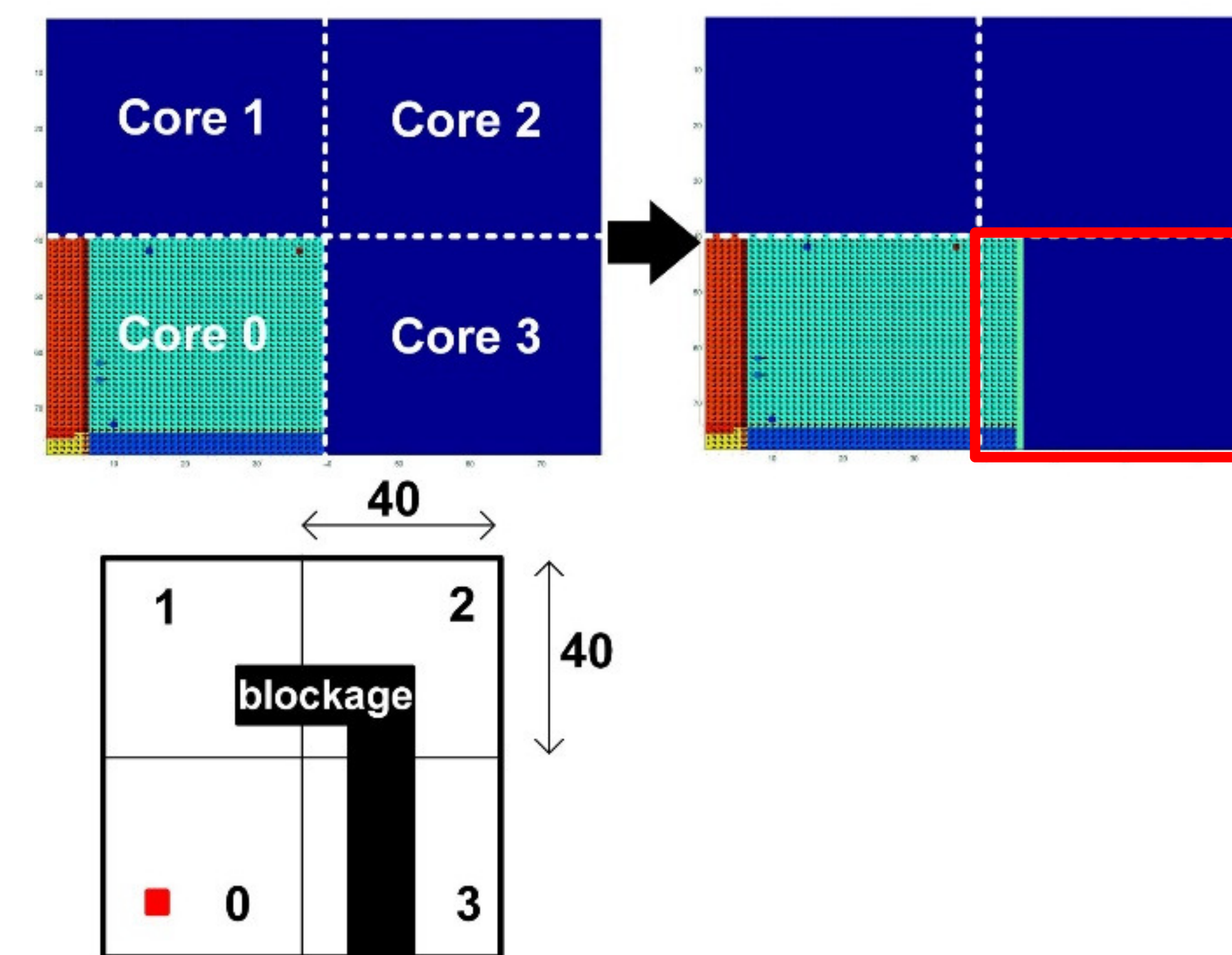
Global Map



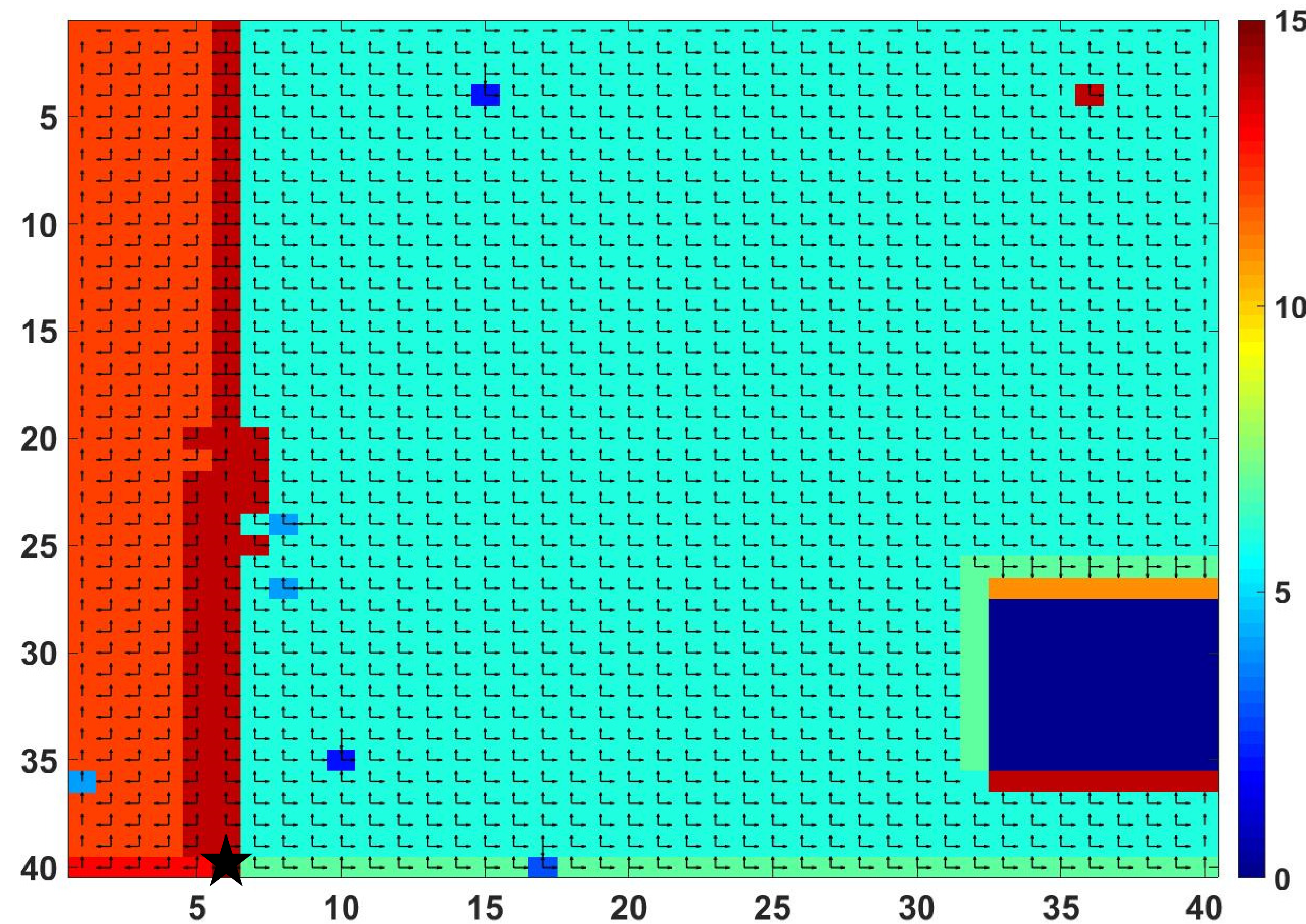
Multicore Evaluation (2/5)



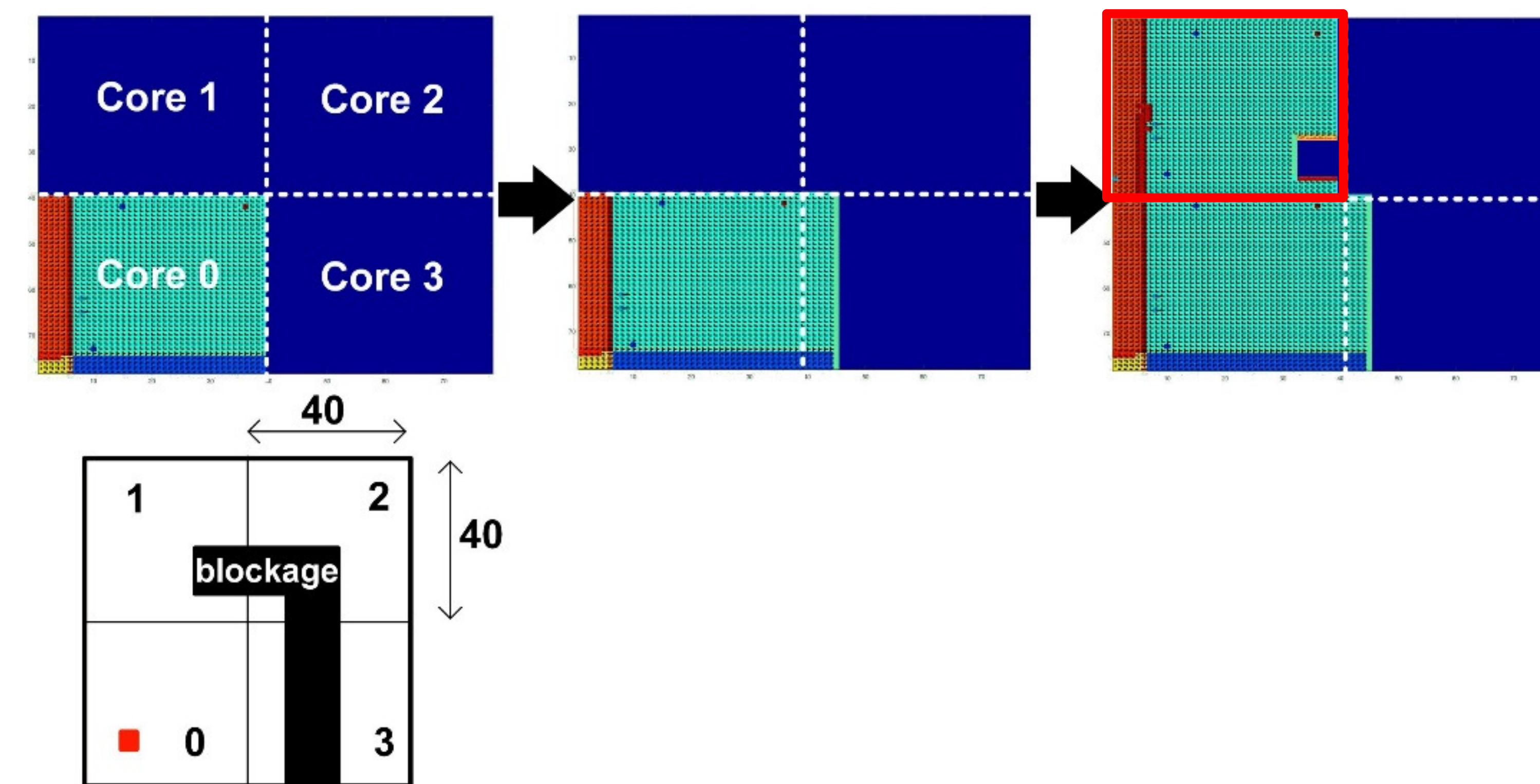
Global Map



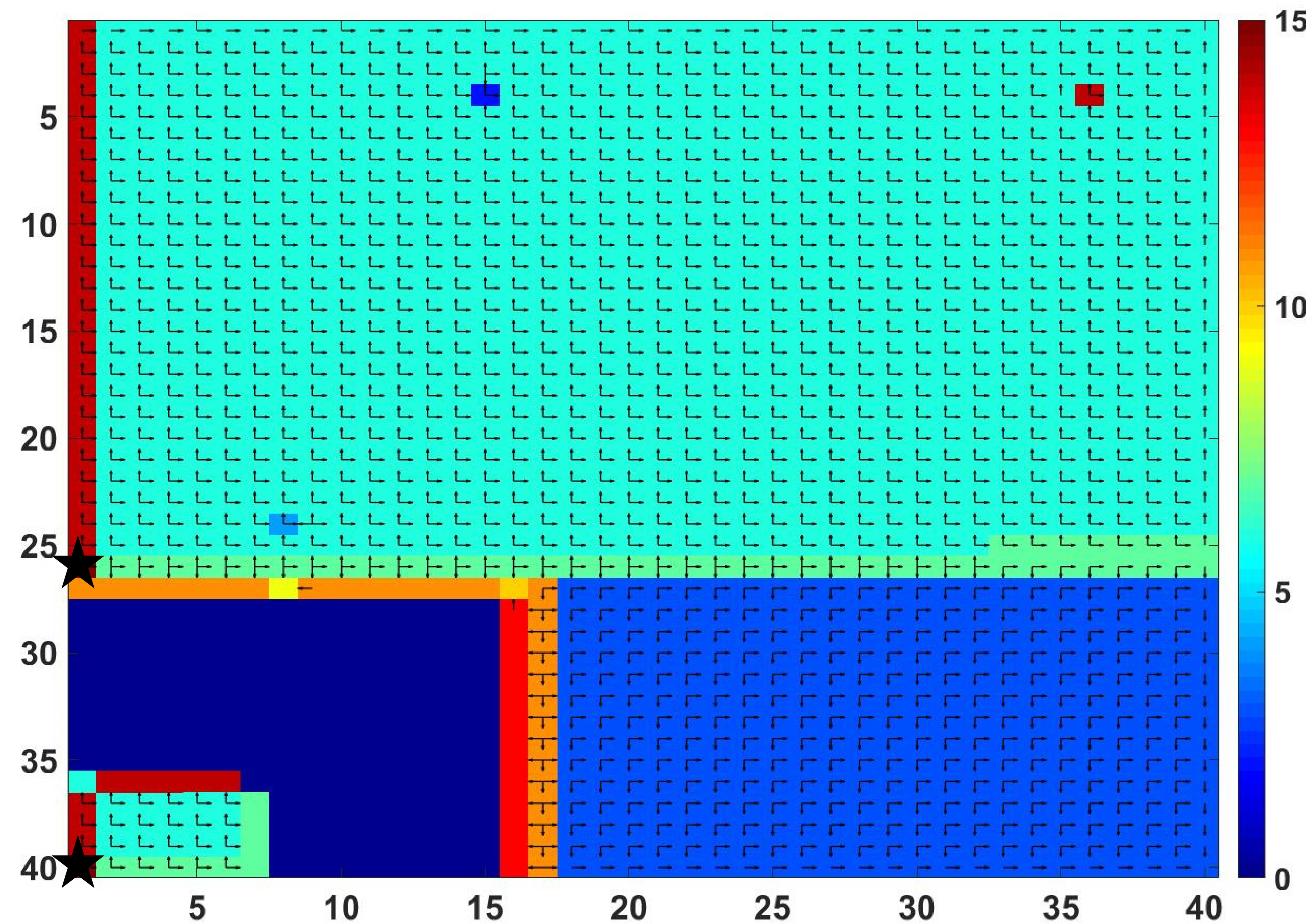
Multicore Evaluation (3/5)



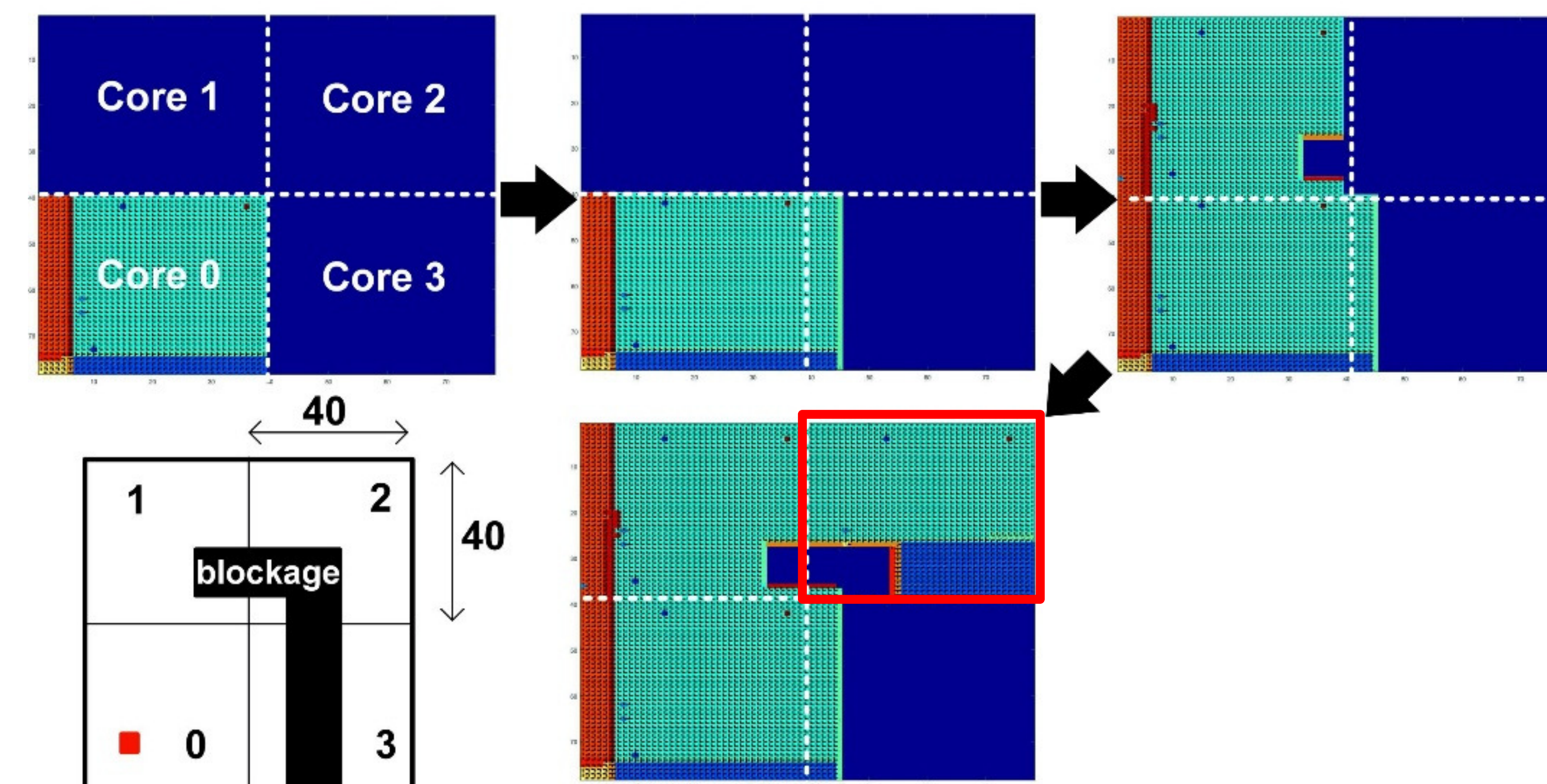
Global Map



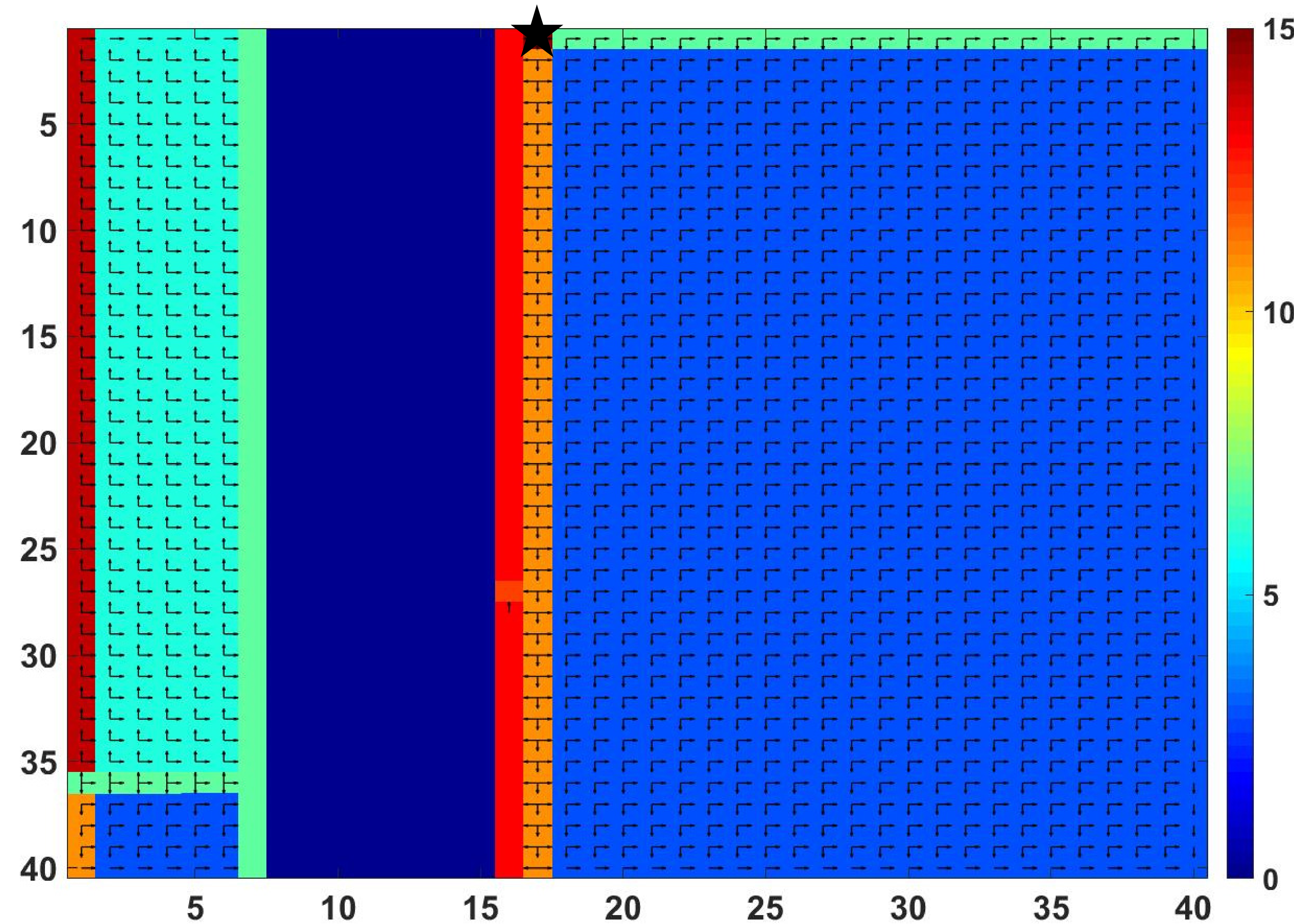
Multicore Evaluation (4/5)



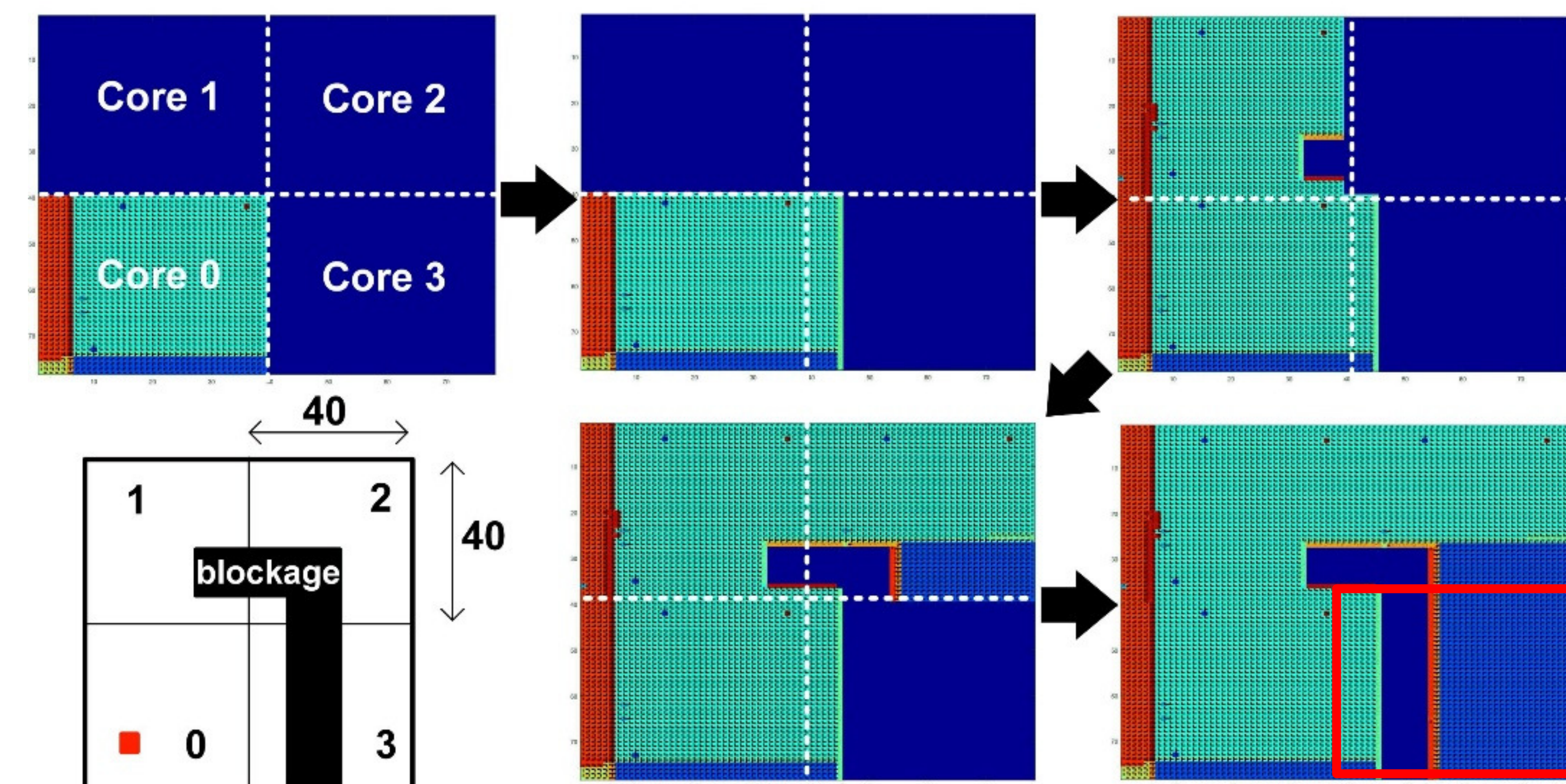
Global Map



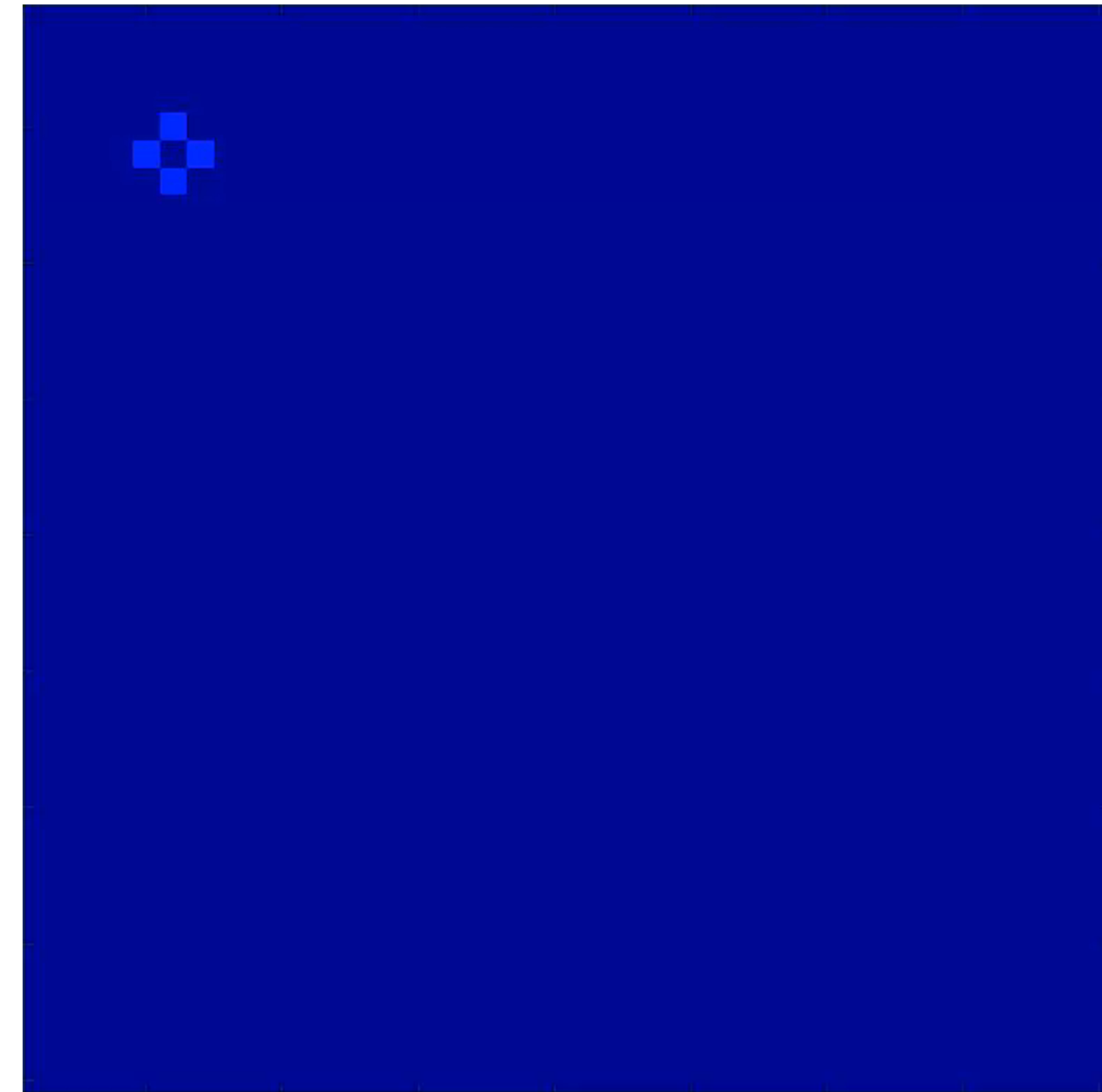
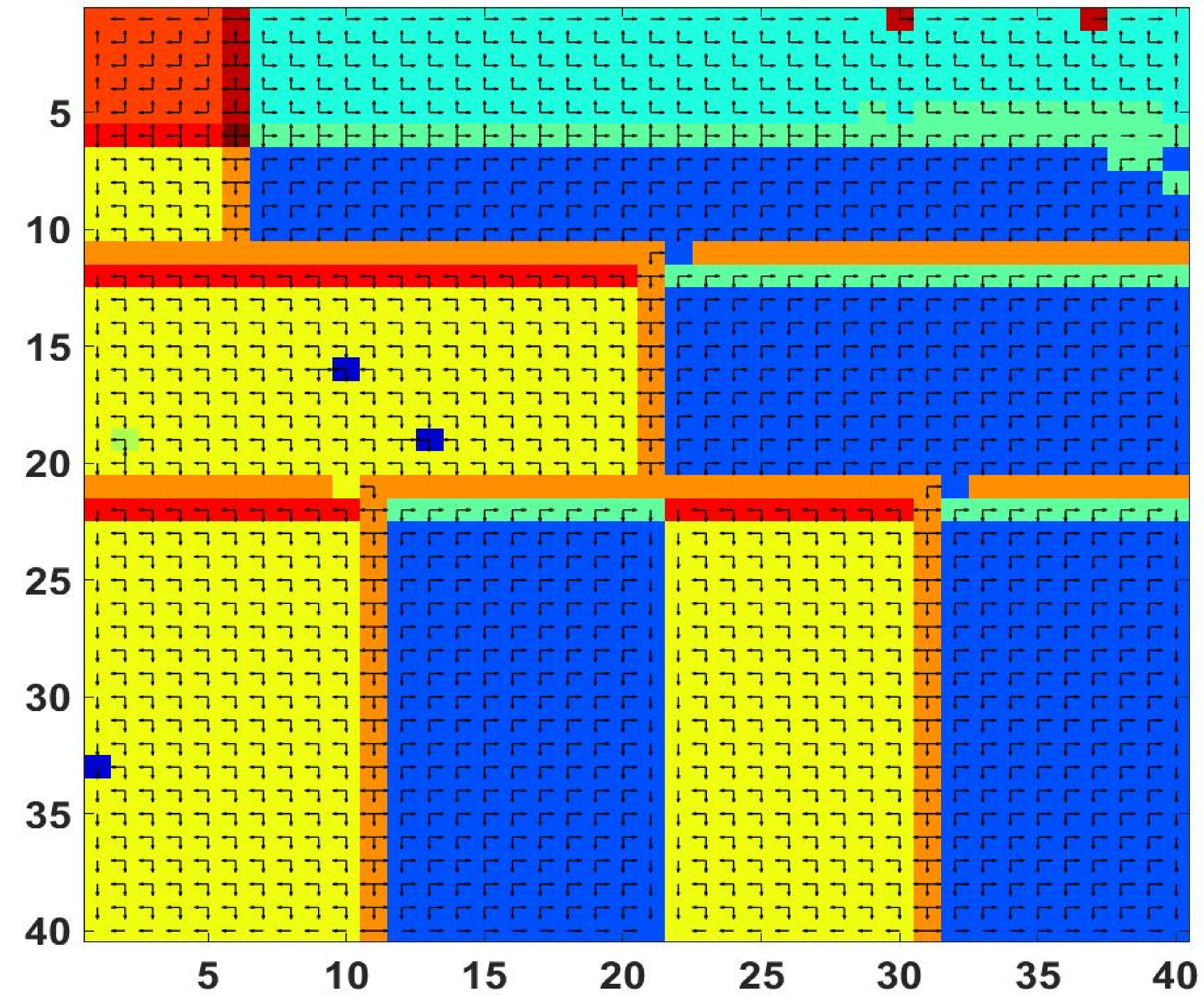
Multicore Evaluation (5/5)



Global Map



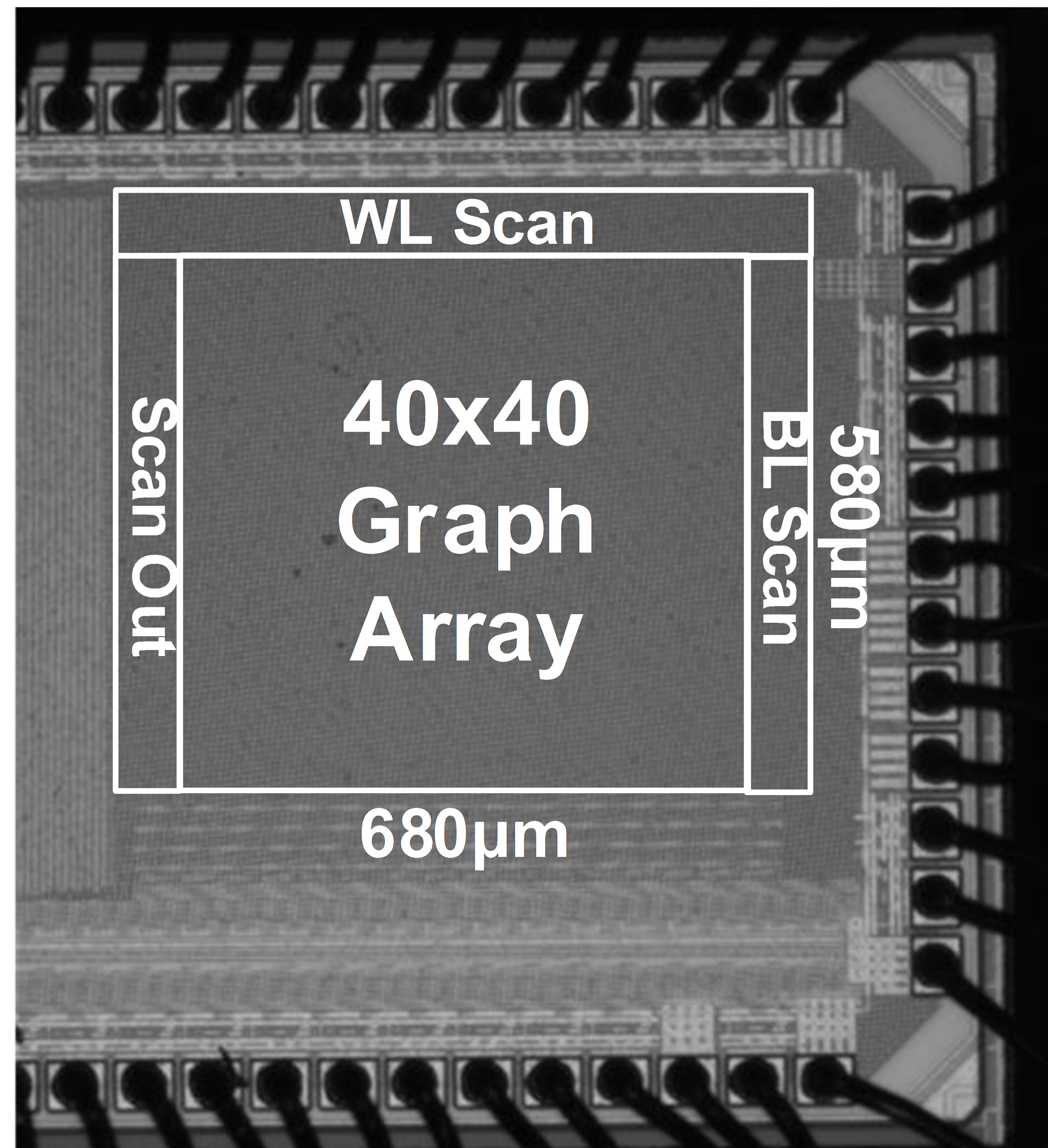
Optics Experiment



Outline

- Background: Path Planning Algorithms
- Time-Based A* ASIC
- 65nm Test Chip Results
- Applications
- **Conclusion**

Die Photo and Chip Summary



Applications	A* shortest path, obstacle avoidance, scientific computation (optics)
Technology	65nm LP CMOS
Architecture	Time-based
# of Vertices	1600
Unit Area	249 μm^2
# of Edges	6400
Edge Resolution	4b + Analog Gradient
Voltage	1.2V
Peak Power	26.4mW
Delay per Node	1.79ns @ [$V_B=0.9\text{V}$, $V_{DD}=1.2\text{V}$]
Power per Node	183.1 μW
Energy per Node	0.238pJ

Conclusion

- **40×40 A* ASIC in 65nmLP**
- **A* heuristic implemented with analog bias gradient**
- **Vertex cells asynchronously evaluate and lockout**
- **Diverse range of applications**
- **Scalable to multicore for larger maps**
- **Orders of magnitude improvement in energy efficiency**

This research was supported in part by the National Science Foundation under award number CCF-1763761.

