CS221 Section 3: Search

DP, UCS and A*

Contents

Uniform Cost Search

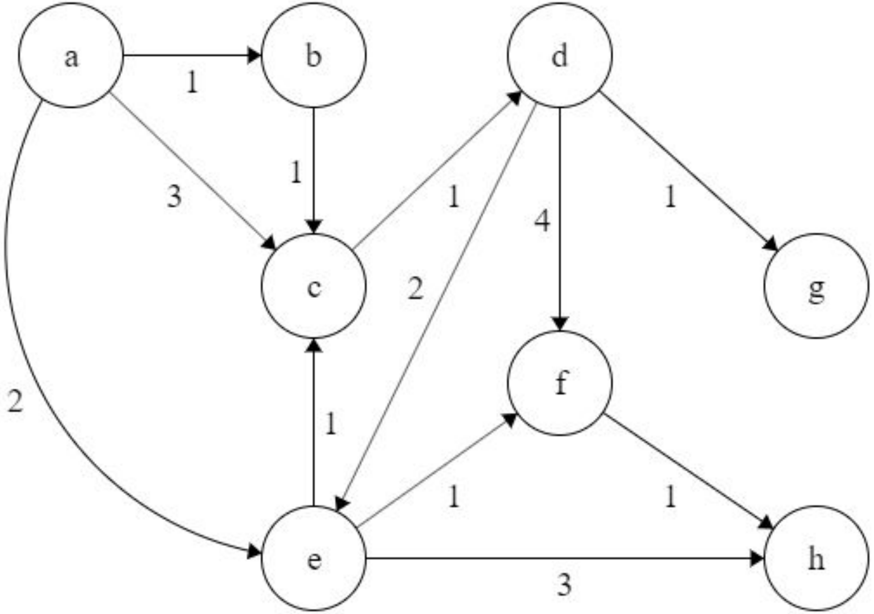
Idea: In UCS, we find the shortest cost to a node by using the fact we already know the shortest path to a set of nodes.

Recall: We have the following three sets

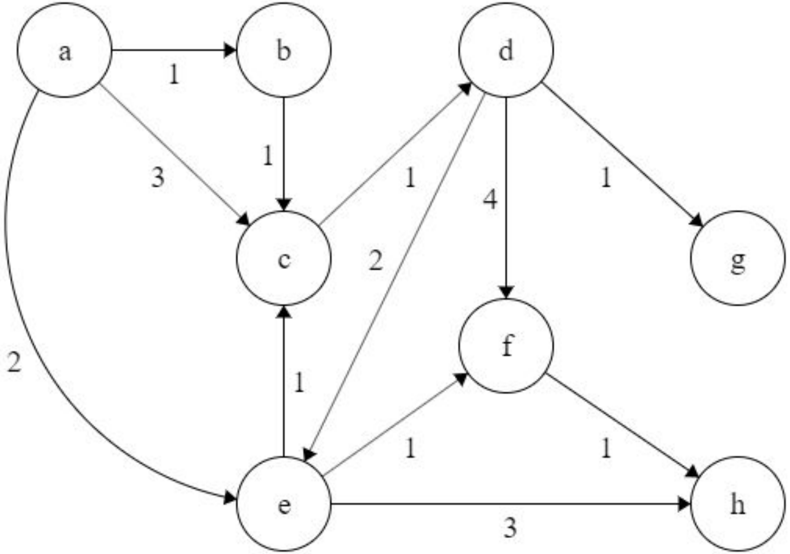
- Explored Set: contains nodes we know the path length to
- Frontier Set: contains nodes that are neighbors of those in the explored set, but we don't know their costs yet
- Unexplored Set: Nodes in the graph we haven't encountered

Problem - UCS

In the following graph, find the costs to reach each node given that we start on node **a**.



Problem - UCS



Explored

[a : 0]

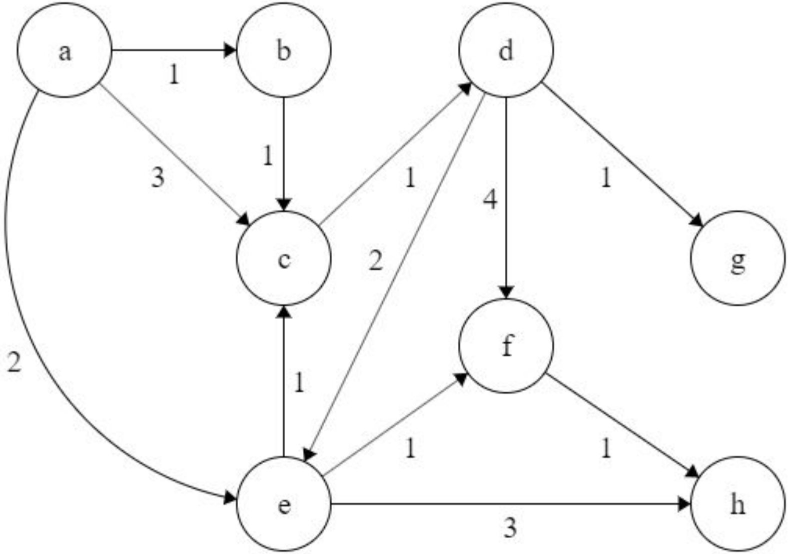
Frontier

[b : 0 + 1, e : 0 + 2, c : 0 + 3]

Unexplored

We start with node **a**. We add all neighbors of **a** to the frontier. Note: [a : 0] means it takes 0 cost to get to node a.

Problem - UCS



Explored

[a : 0, **b : 1**]

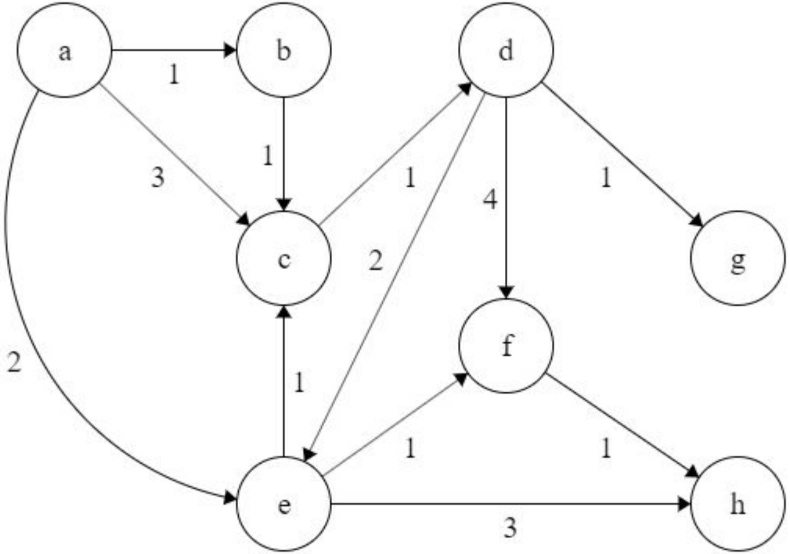
Frontier

[**c : 1 + 1**, e : 0 + 2]

Unexplored

In the frontier, **b** has the lowest cost. Thus, we can add it to the explored set. We add all neighbors of **b** to the frontier, updating costs to reach some nodes if necessary (we updated **c**).

Problem - UCS



Explored

[a : 0, b : 1, **c : 2**]

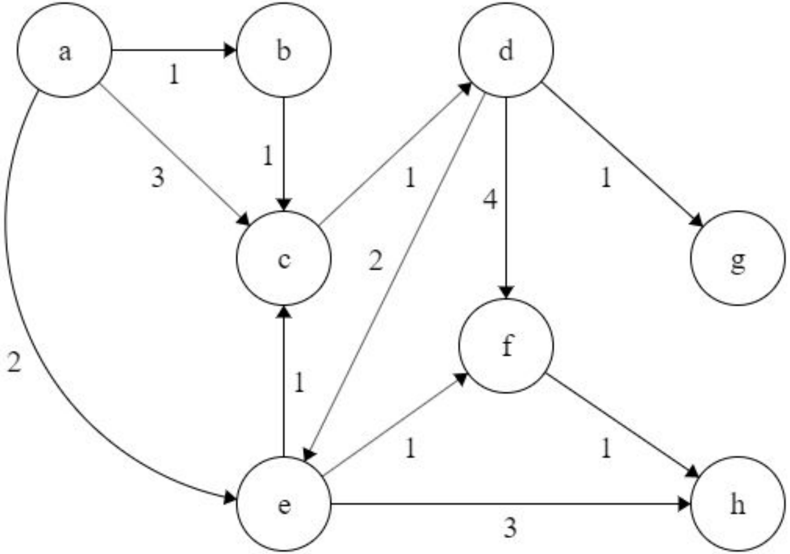
Frontier

[e : 0 + 2, **d : 2 + 1**]

Unexplored

In the frontier, **c** has the lowest cost (ties broken alphabetically here). Thus, we can add it to the explored set. We add all neighbors of **c** to the frontier, updating as necessary.

Problem - UCS



Explored

[a : 0, b : 1, c : 2, **e : 2**]

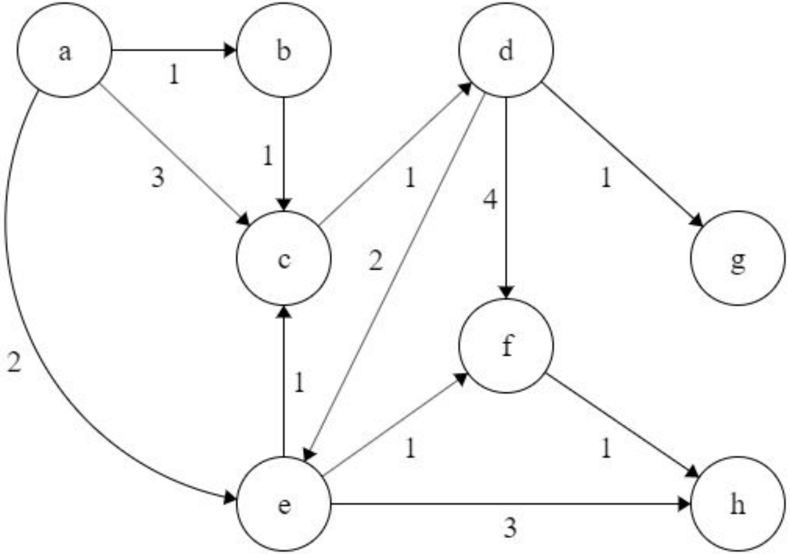
Frontier

[d : 2 + 1, **f : 2 + 1, h : 2 + 3**]

Unexplored

In the frontier, **e** has the lowest cost. Thus, we can add it to the explored set. We add all neighbors of **e** to the frontier, updating as necessary.

Problem - UCS



Explored

[a : 0, b : 1, c : 2, e : 2, **d : 3**]

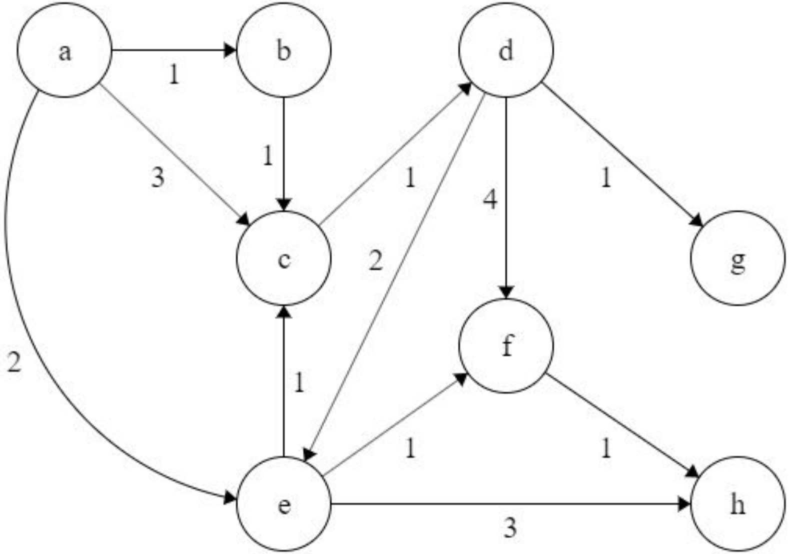
Frontier

[f : 2 + 1, **g : 3 + 1,** h : 2 + 3]

Unexplored

In the frontier, **d** has the lowest cost. Thus, we can add it to the explored set. We add all neighbors of **d** to the frontier, updating as necessary.

Problem - UCS



Explored

[a : 0, b : 1, c : 2, e : 2, d : 3, **f : 3**]

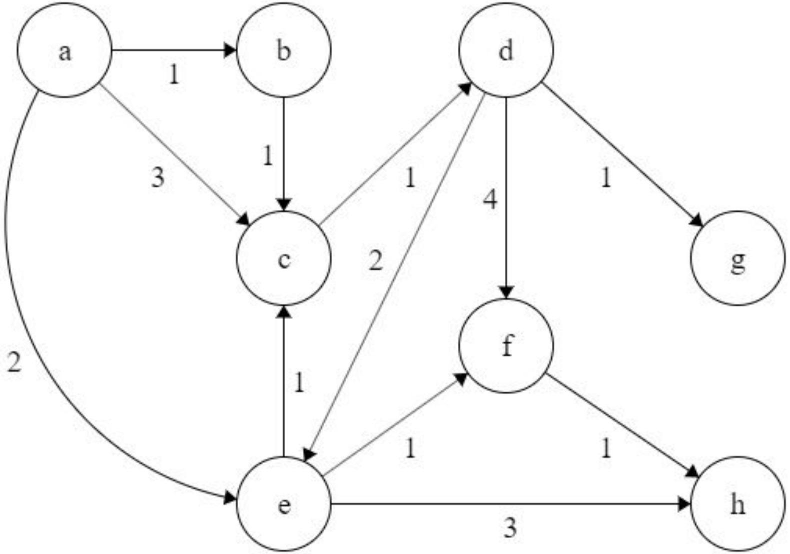
Frontier

[g : 3 + 1, **h : 3 + 1**]

Unexplored

In the frontier, **f** has the lowest cost. Thus, we can add it to the explored set. We add all neighbors of **f** to the frontier, updating as necessary.

Problem - UCS



Explored

[a : 0, b : 1, c : 2, e : 2, d : 3, f : 3, **g : 4**]

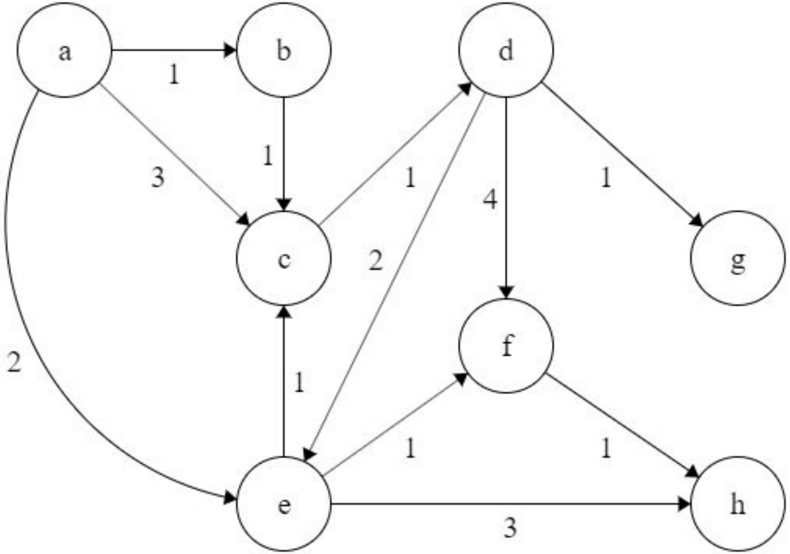
Frontier

[h : 3 + 1]

Unexplored

In the frontier, **g** has the lowest cost. Thus, we can add it to the explored set. We add all neighbors of **f** to the frontier, updating as necessary.

Problem - UCS



Explored

[a : 0, b : 1, c : 2, e : 2, d : 3, f : 3, g : 4, **h : 4**]

Frontier

Unexplored

In the frontier, **h** has the lowest cost. Thus, we can add it to the explored set. There are no more nodes in the frontier, so we are done.

Uniform Cost Search

Algorithm: uniform cost search [Dijkstra, 1956]

Add s_{start} to **frontier** (priority queue)
Repeat until frontier is empty:
    Remove s with smallest priority p from frontier
    If $\text{IsEnd}(s)$: return solution
    Add s to **explored**
    For each action $a \in \text{Actions}(s)$:
        Get successor $s' \leftarrow \text{Succ}(s, a)$
        If s' already in explored: continue
        Update **frontier** with s' and priority $p + \text{Cost}(s, a)$

Contents

1. Uniform Cost Search
2. **Defining States**
3. Dynamic Programming
4. A* Search

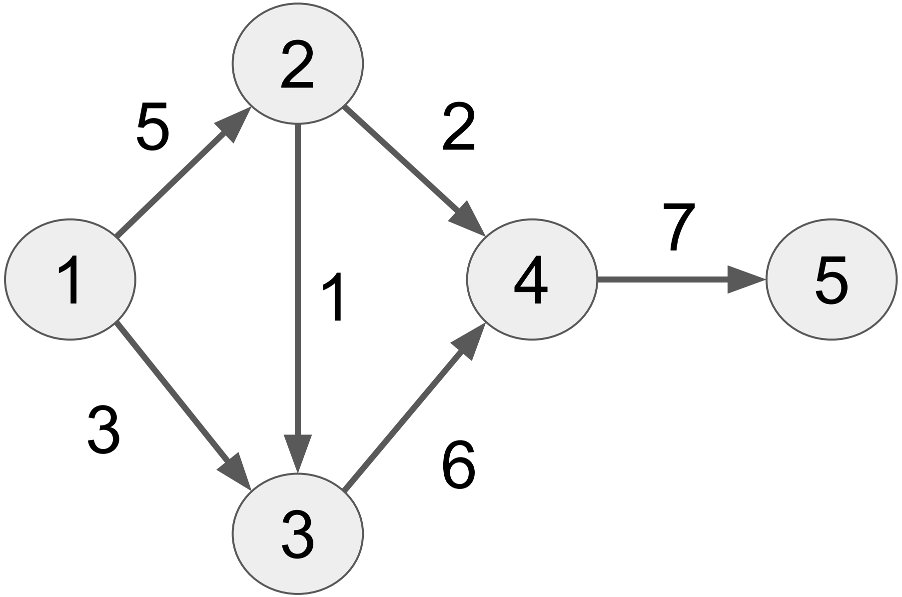
Problem

There exists $\mathbf{N}$ cities, conveniently labelled from $\mathbf{1}$ to $\mathbf{N}$.

There are roads connecting some pairs of cities. The road connecting city $\mathbf{i}$ and city $\mathbf{j}$ takes $\mathbf{c(i,j)}$ time to traverse. However, one can only travel from a city with smaller label to a city with larger label (i.e. each road is one-directional).

From city $\mathbf{1}$, we want to travel to city $\mathbf{N}$. What is the shortest time required to make this trip, given the additional constraint that we should visit more odd-labeled cities than even labeled cities?

Example



Best path is [1, 3, 4, 5] with cost 16.

[1, 2, 4, 5] has cost 14 but visits equal number of odd and even cities.

State Representation

Key idea: state

A **state** is a summary of all the past actions sufficient to choose future actions **optimally**.

State Representation

We need to know where we are currently at: **current_city**

We need to know how many odd and even cities we have visited thus far: **#odd, #even**

State Representation: **(current_city, #odd, #even)**

Total number of states: $\mathbf{O(N^3)}$

Can We Do Better?

Check if all the information is really required

We store **#odd** and **#even** so that we can check whether **#odd** - **#even** > 0 at (N, **#odd**, **#even**)

Why not store **#odd - #even** directly instead?

(current_city, #odd - #even) -- **O(N^2)** states

Solving the Problem

Since we are computing shortest path, which is some form of optimization, we consider DP and UCS.

Recall:

- DP can handle negative edges but works only on DAGs
- UCS works on general graphs, but cannot handle negative edges

Since we have a DAG and all edges are positive, both work! We already went through UCS, so we solve this with DP.

Contents

Solving the Problem: Dynamic Programming

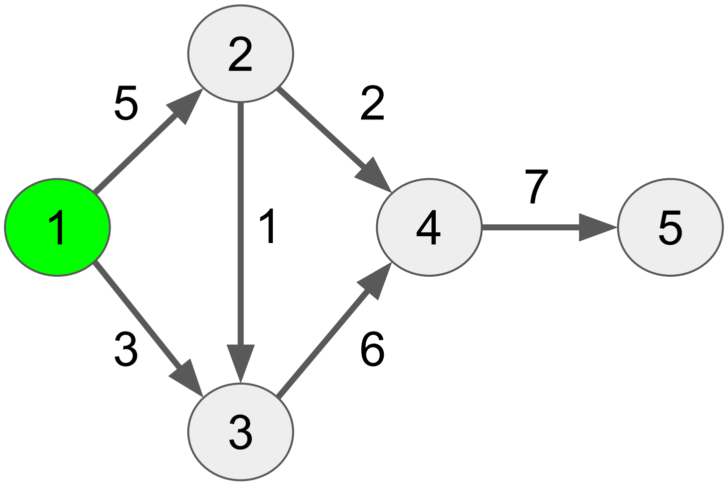
$$\mathsf{FutureCost}(s) = \begin{cases} 0 & \text{if } \mathsf{IsGoal}(s) \\ \min_{a \in \mathsf{Actions}(s)} [\mathsf{Cost}(s,a) + \mathsf{FutureCost}(\mathsf{Succ}(s,a))] & \text{otherwise} \end{cases}$$

If s has no successors, we set it as undefined

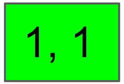
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



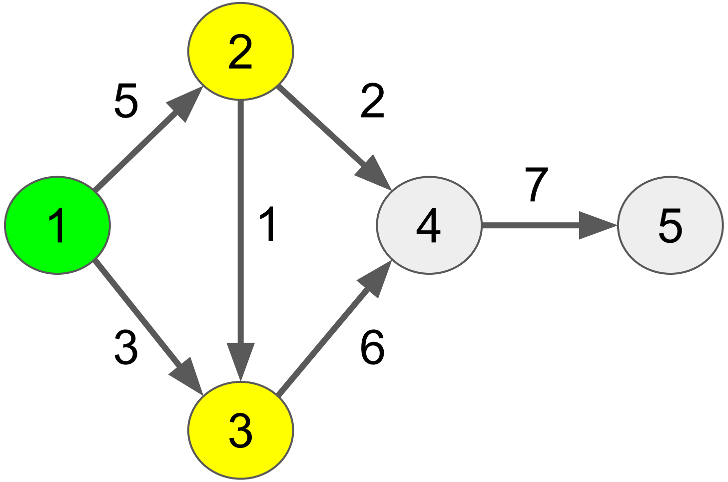
Cache	
Key	Value

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

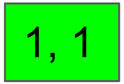
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



Cache

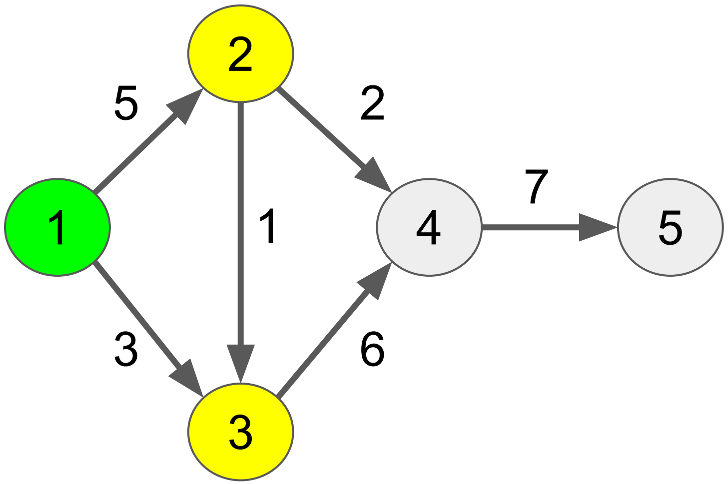
Key	Value

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if } \text{IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s, a) + \text{FutureCost}(\text{Succ}(s, a))] & \text{otherwise} \end{cases}$$

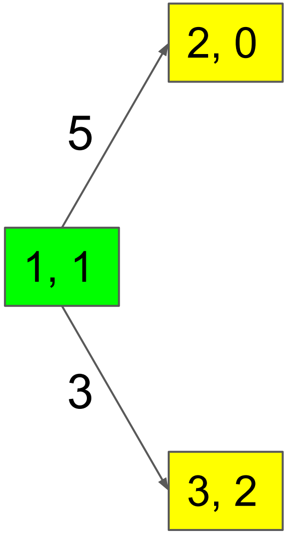
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



Cache

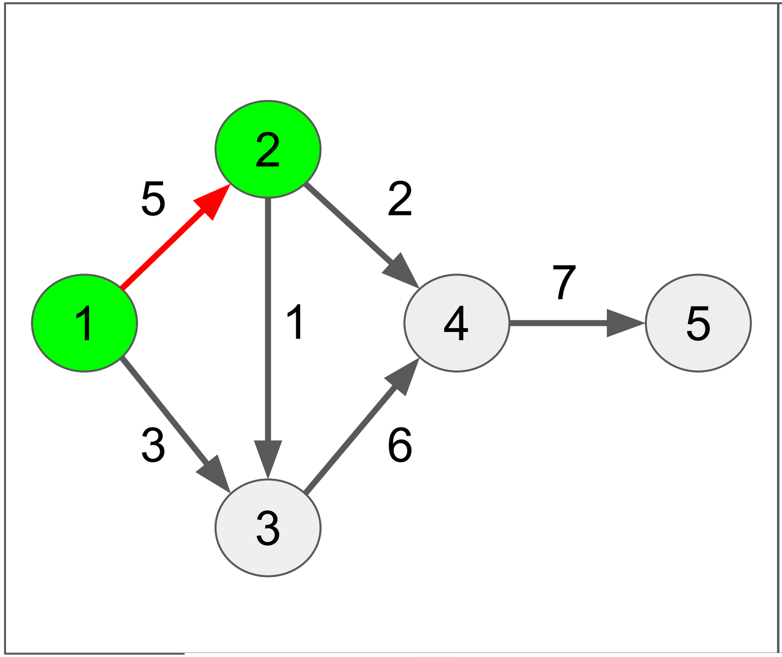
Key	Value

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

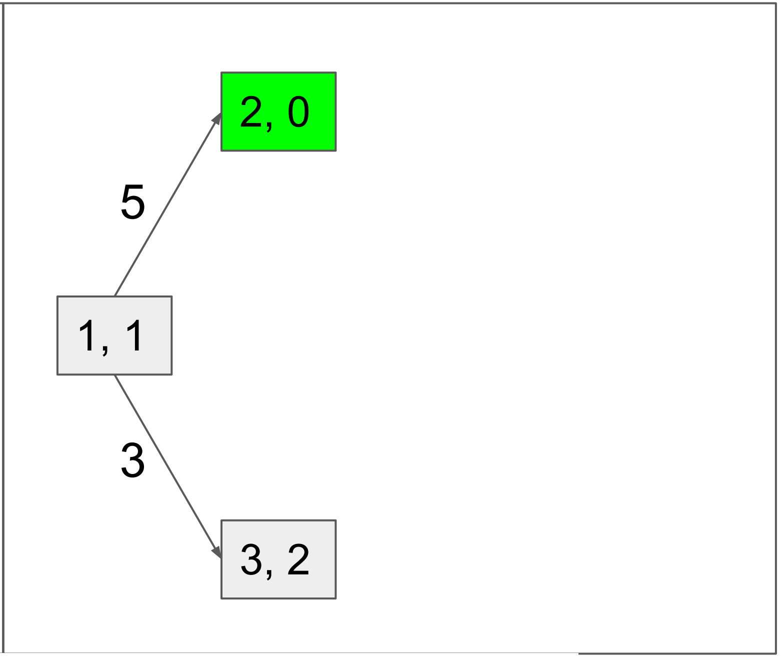
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s, a) + \text{FutureCost}(\text{Succ}(s, a))] & \text{otherwise} \end{cases}$$

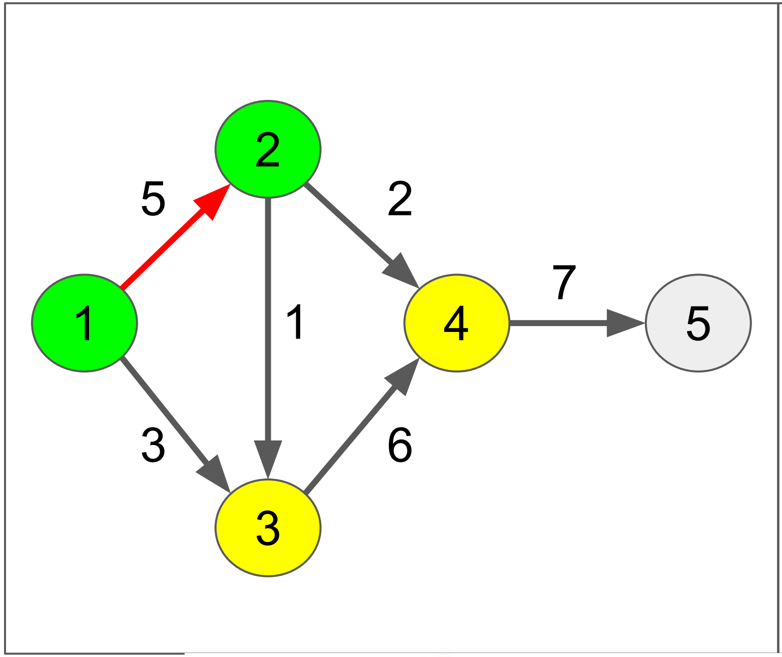
Cache

Key	Value

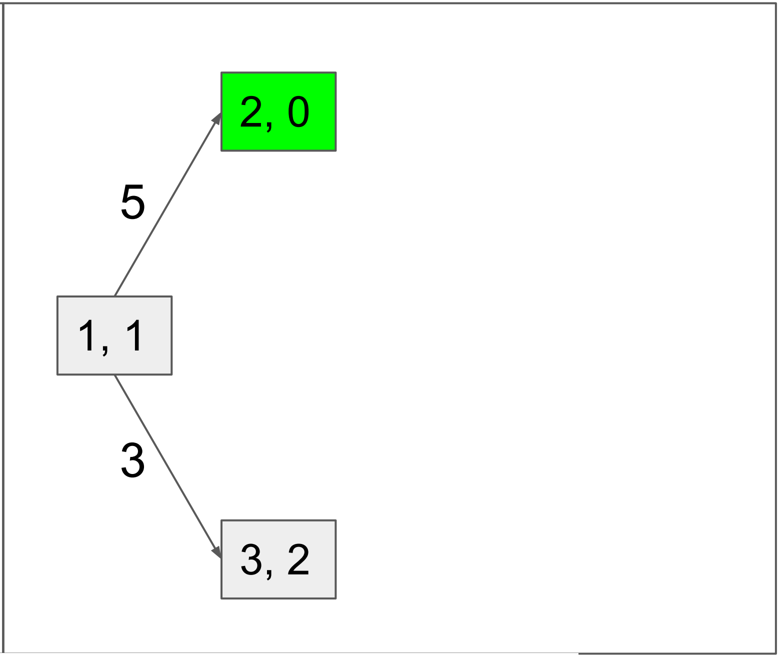
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



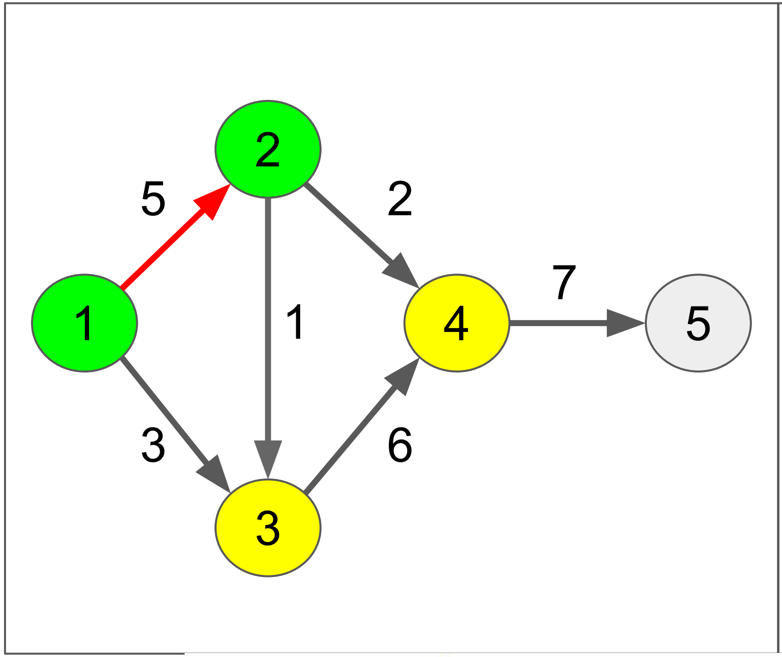
Cache	
Key	**Value**

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s, a) + \text{FutureCost}(\text{Succ}(s, a))] & \text{otherwise} \end{cases}$$

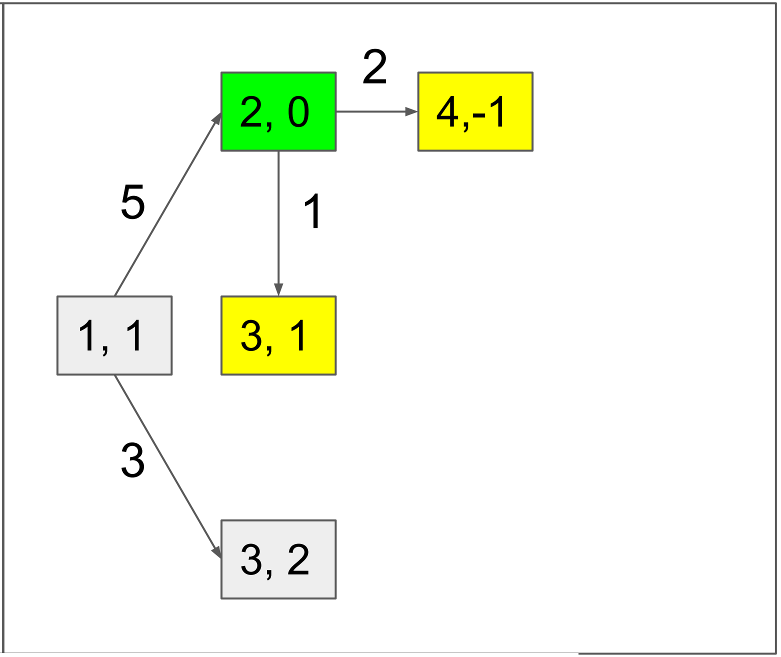
Simulation of DP

Visiting | Successors | Completed

Regular Graph



State Graph



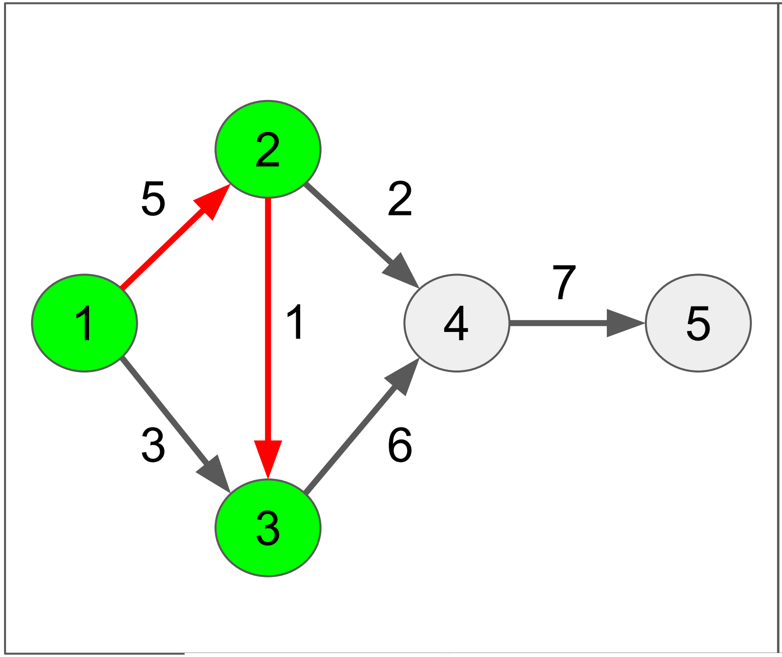
Cache	
Key	**Value**

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)}[\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

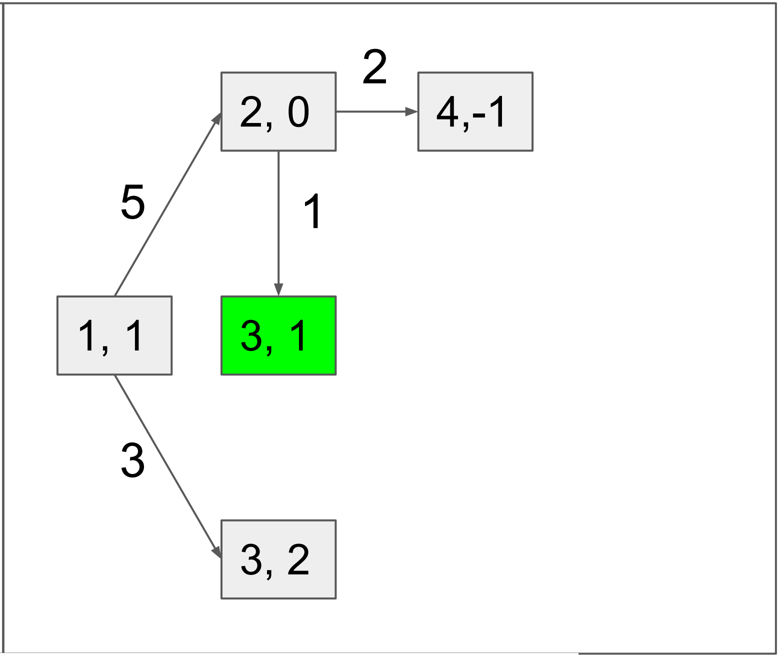
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



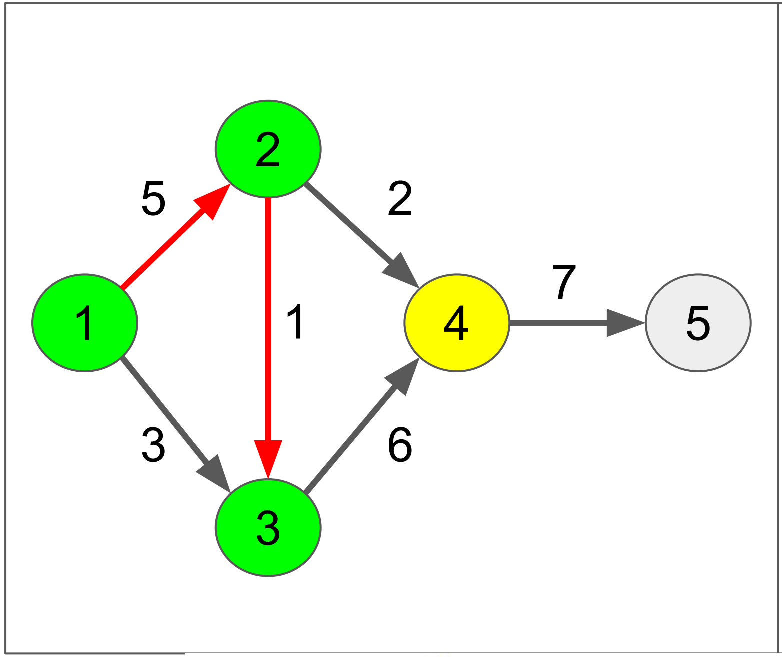
Cache	
Key	Value

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

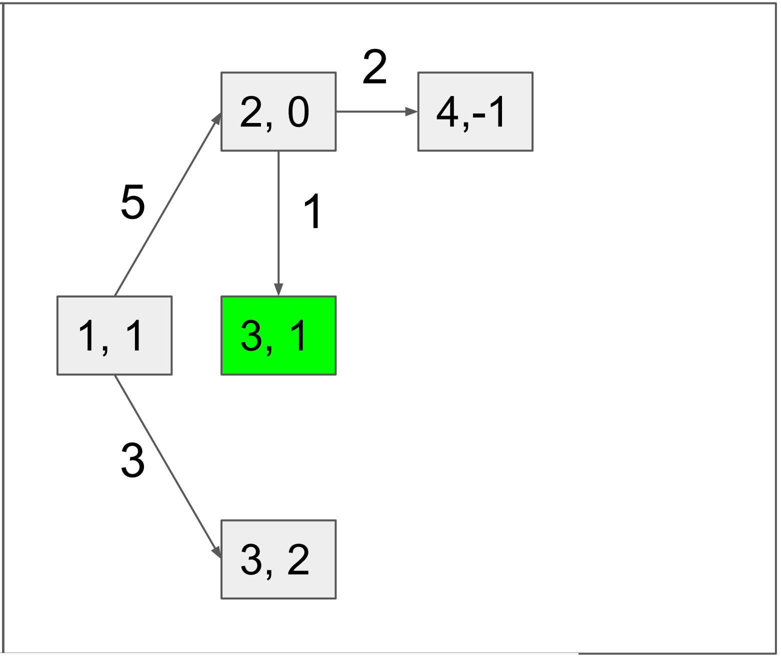
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



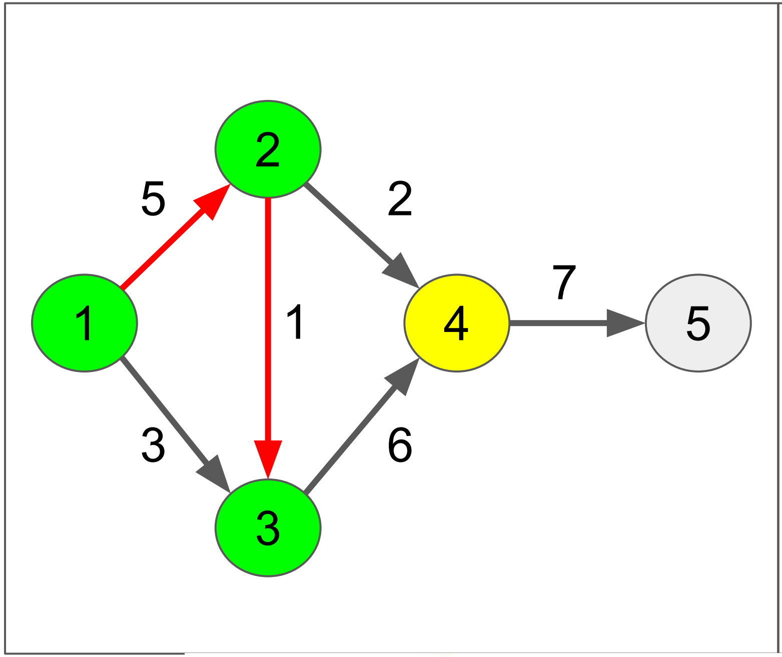
Cache	
Key	Value

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

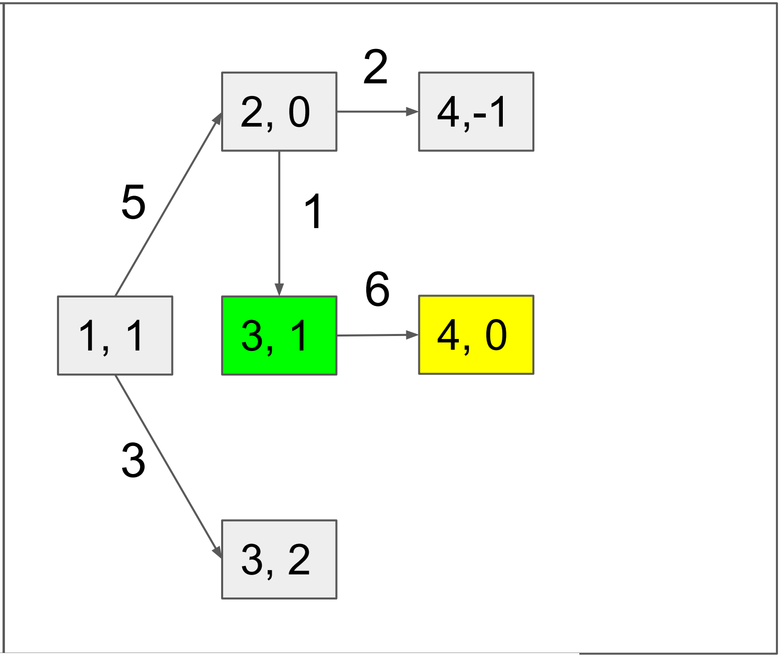
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

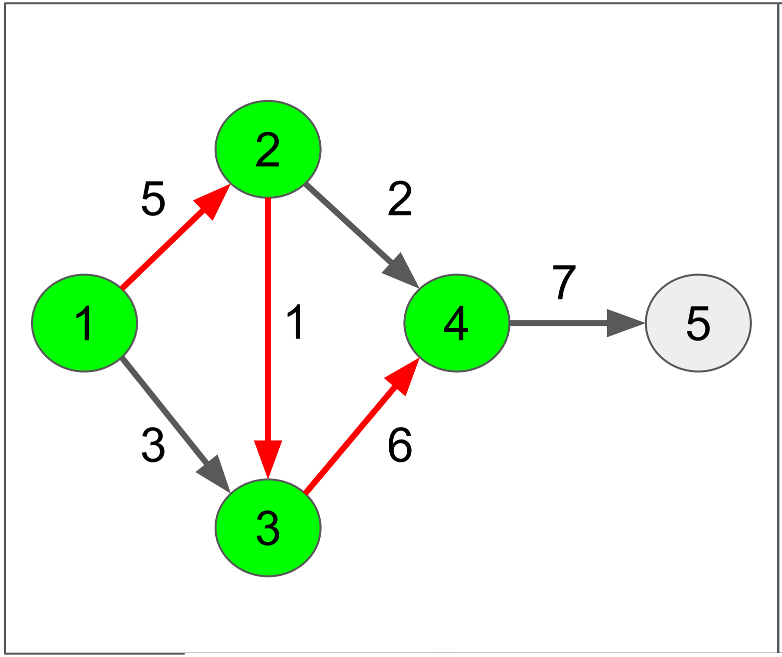
Cache

Key	Value

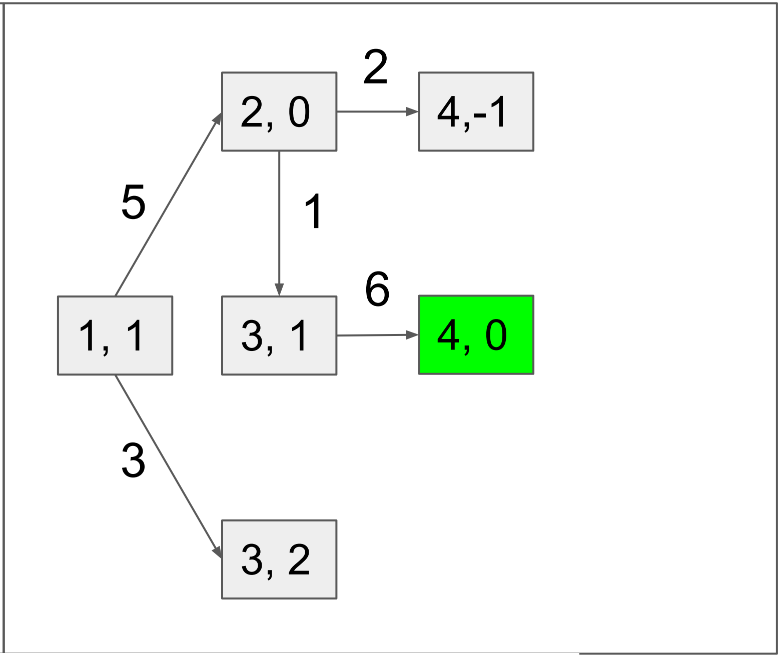
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



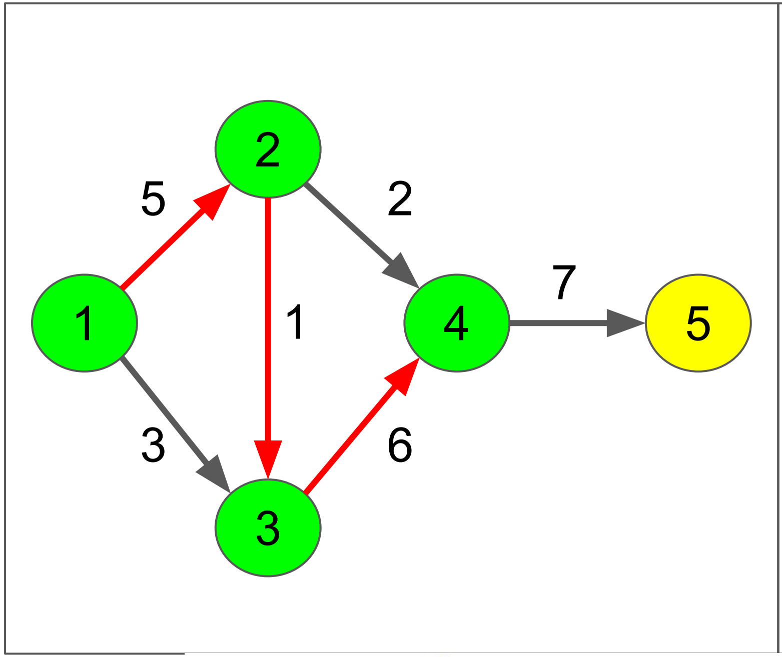
Cache	
Key	**Value**

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

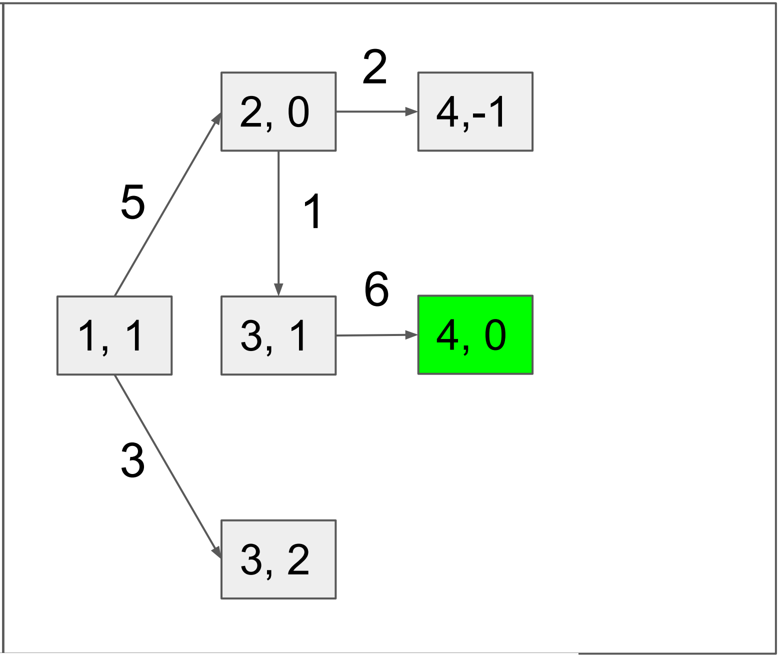
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



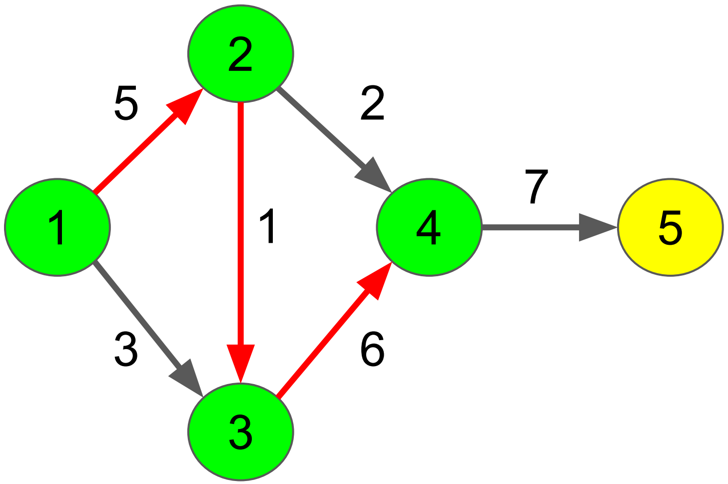
$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Cache	
Key	**Value**

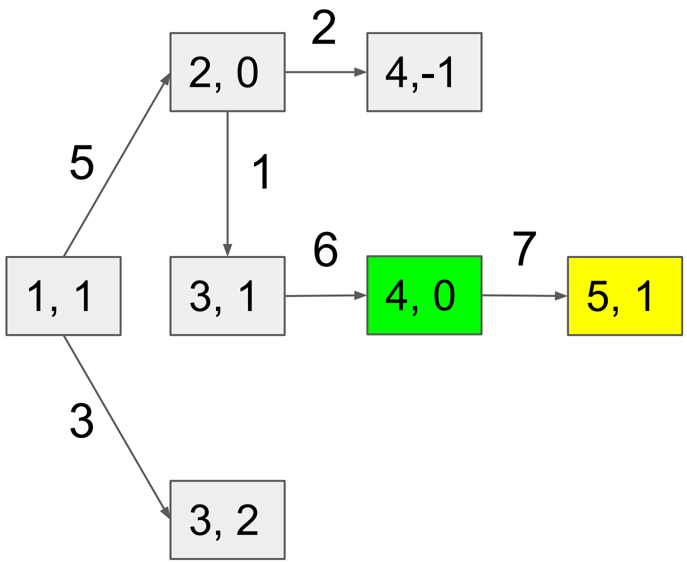
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



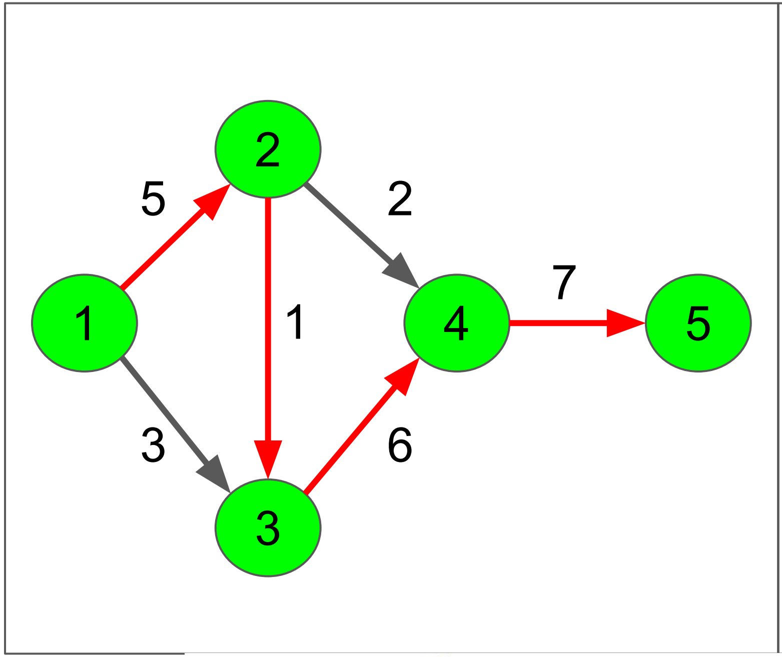
Cache	
Key	Value

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s, a) + \text{FutureCost}(\text{Succ}(s, a))] & \text{otherwise} \end{cases}$$

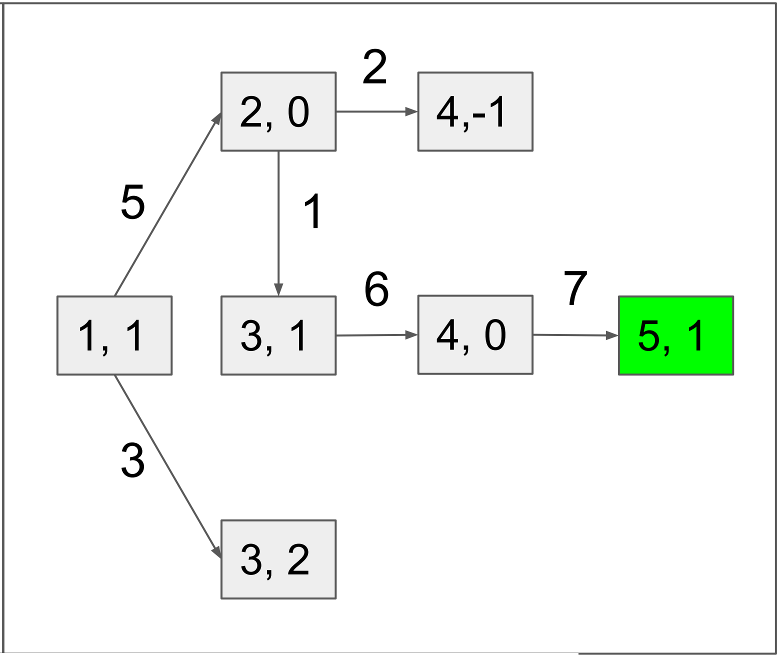
Simulation of DP

Visiting | **Successors** | **Completed**

Regular Graph



State Graph



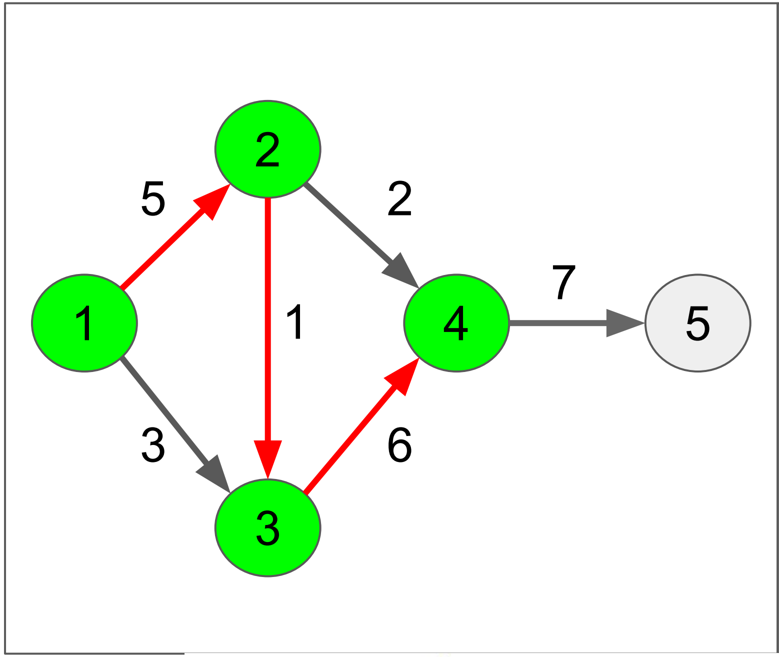
Cache	
Key	**Value**
(5, 1)	0

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

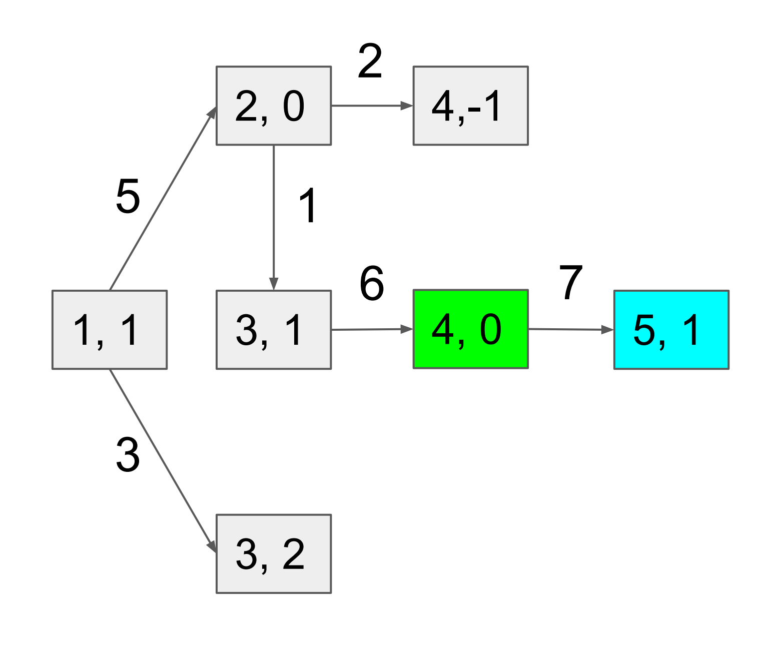
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



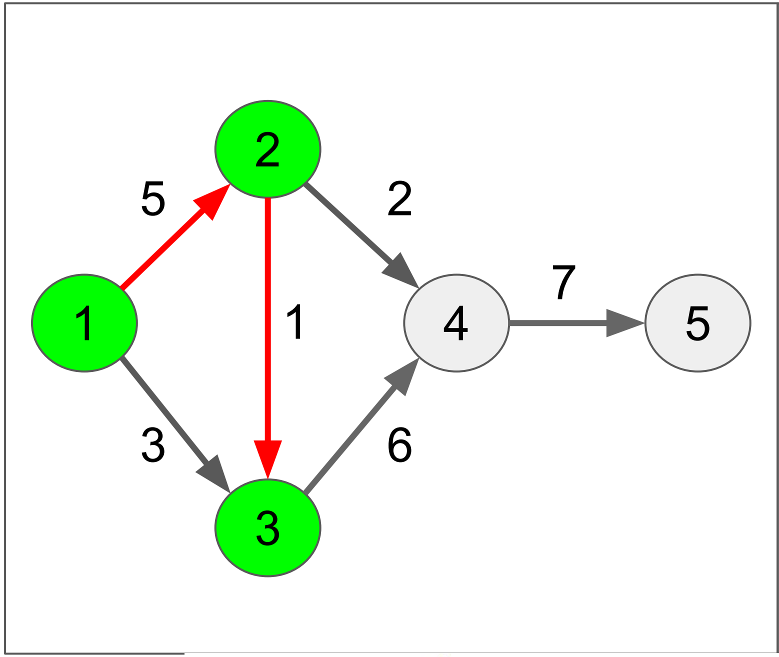
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	0 + 7

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

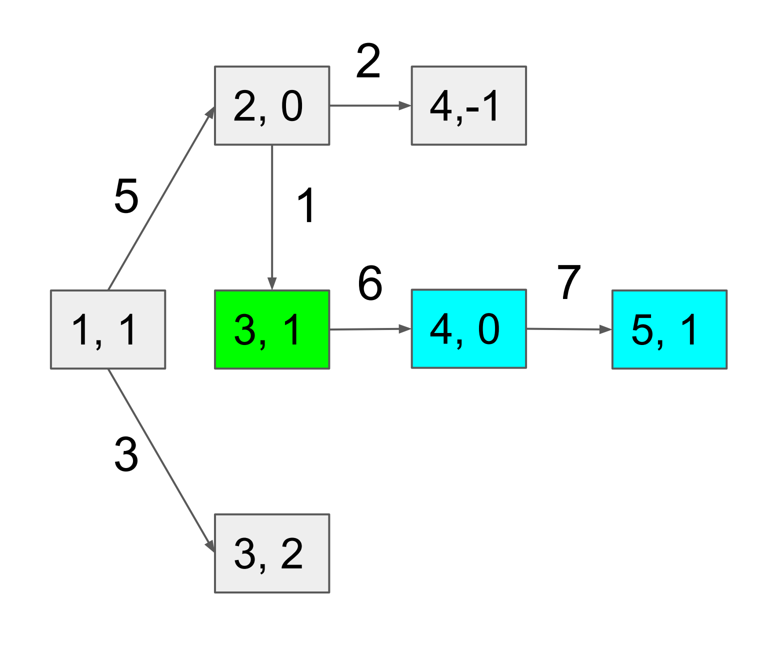
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



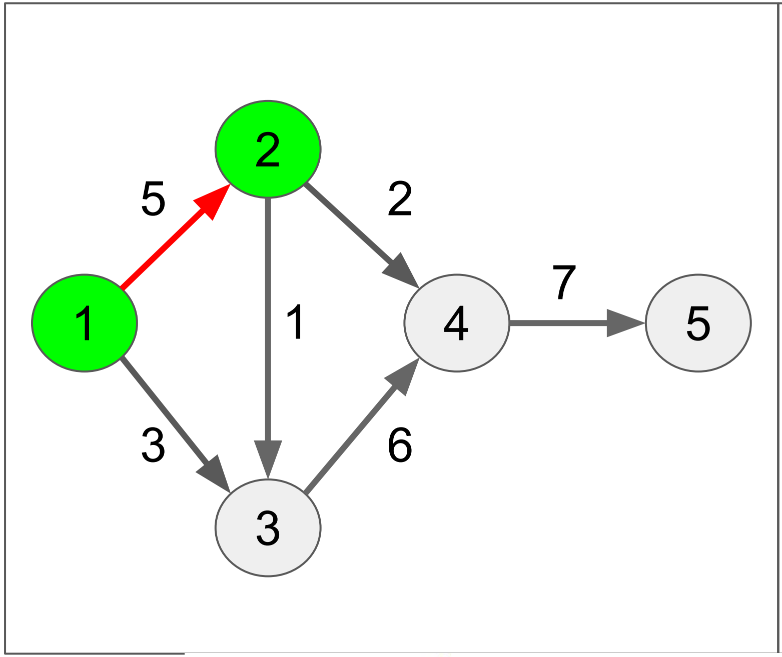
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	7 + 6 = 13

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

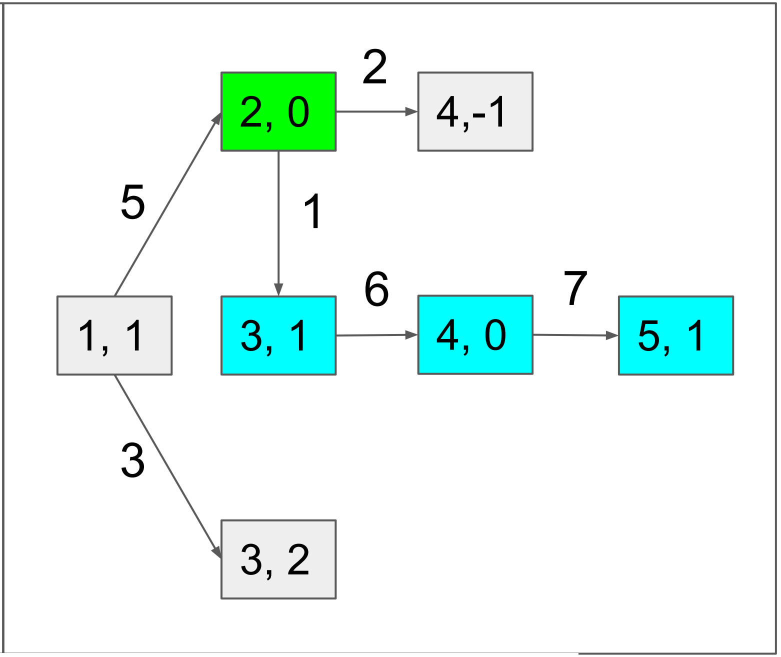
Simulation of DP

Visiting | Successors | Completed

Regular Graph



State Graph



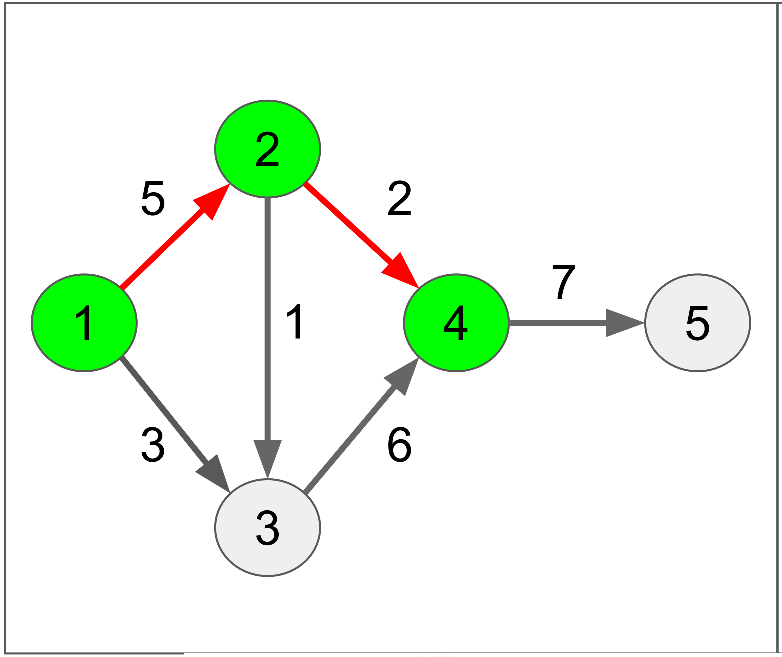
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

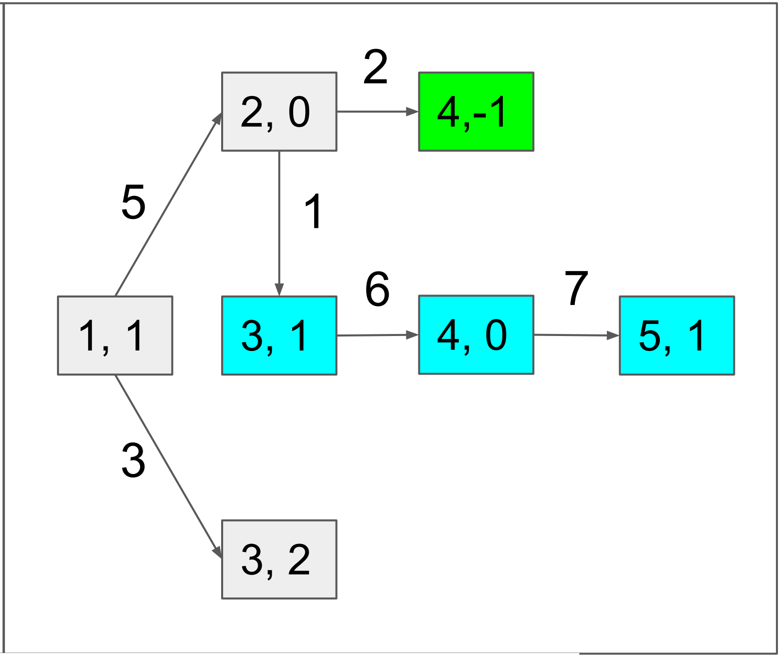
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



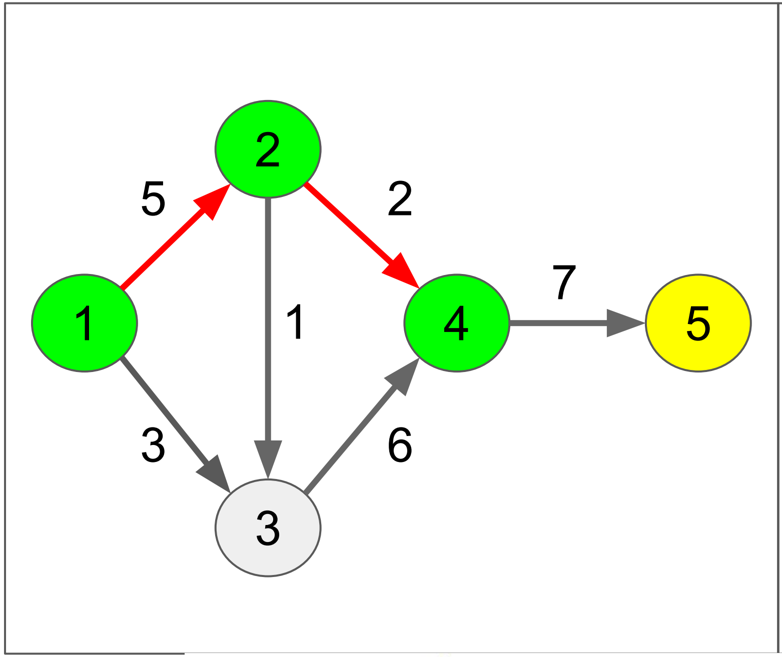
Cache

Key	Value
(5, 1)	0
(4, 0)	7
(3, 1)	13

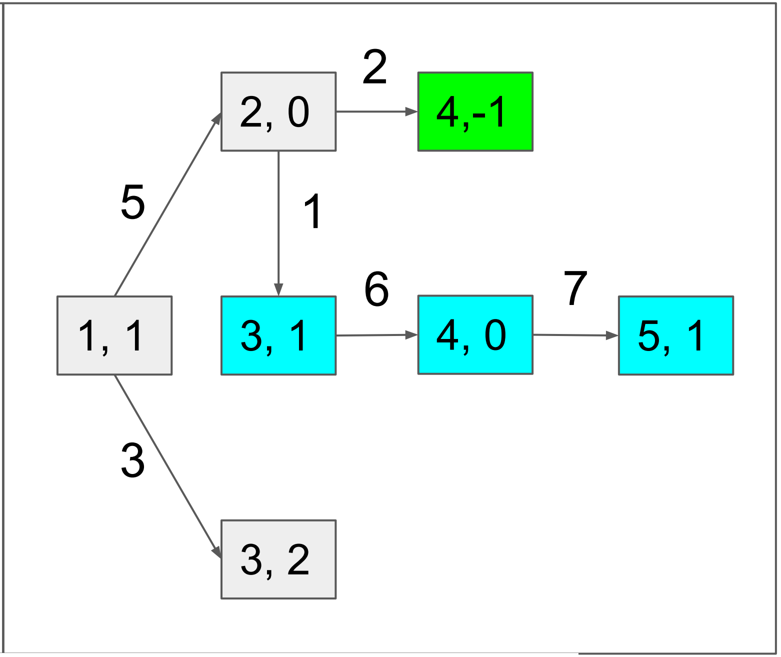
$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

Regular Graph



State Graph

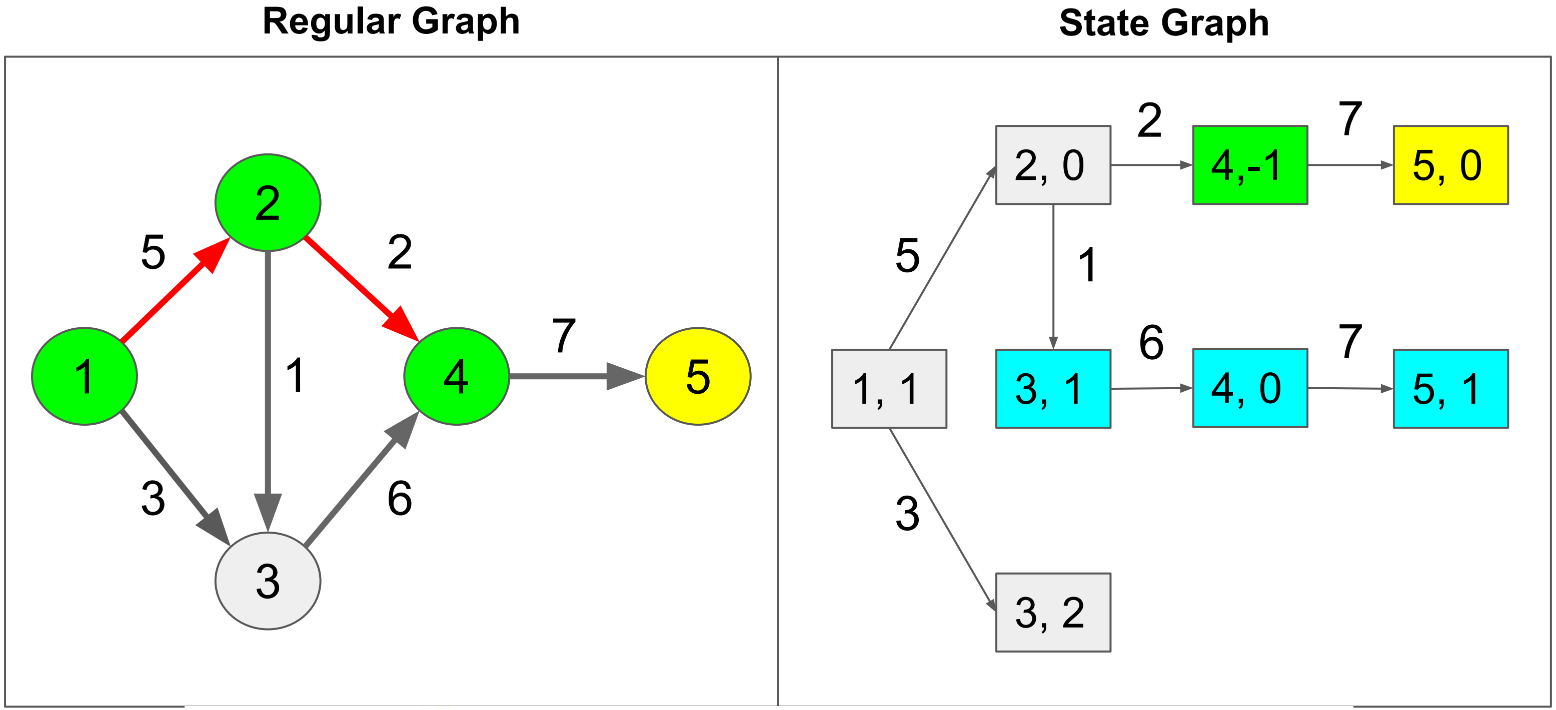


Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

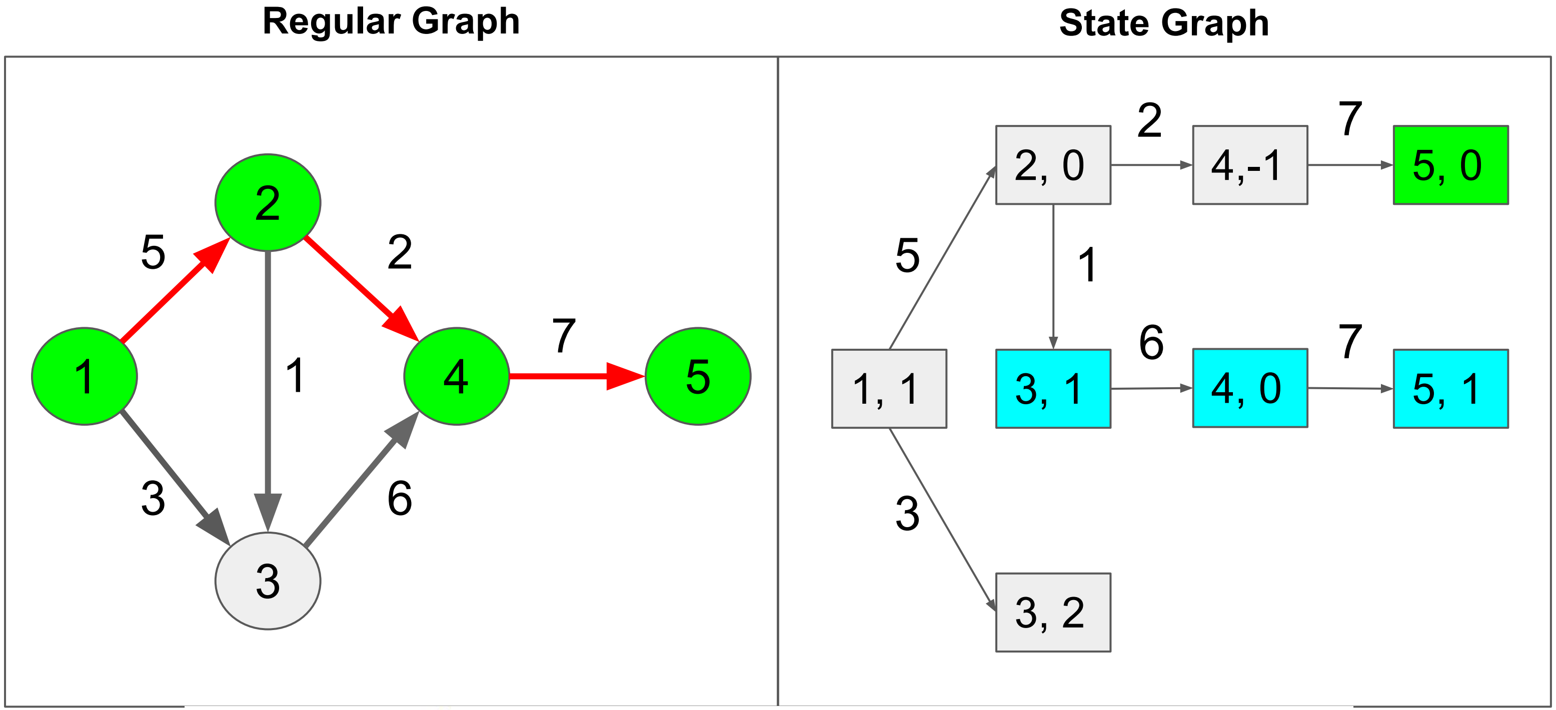
Visiting | Successors | Completed



Cache	
Key	Value
(5, 1)	0
(4, 0)	7
(3, 1)	13

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP



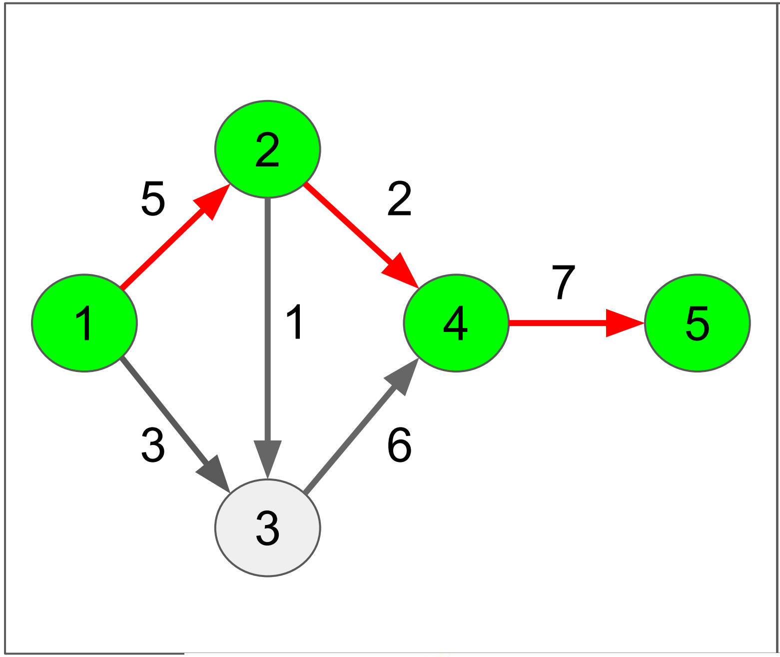
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

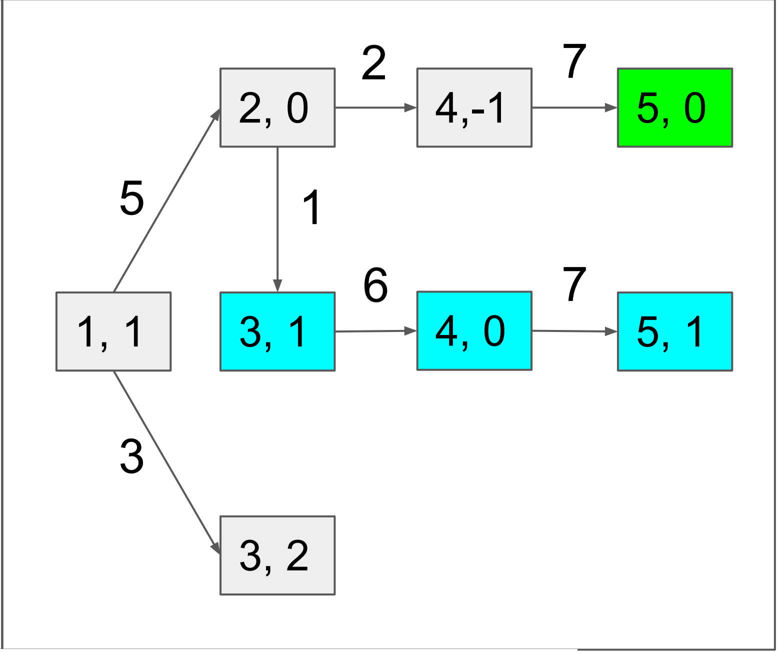
Simulation of DP

Visiting | Successors | Completed

Regular Graph



State Graph



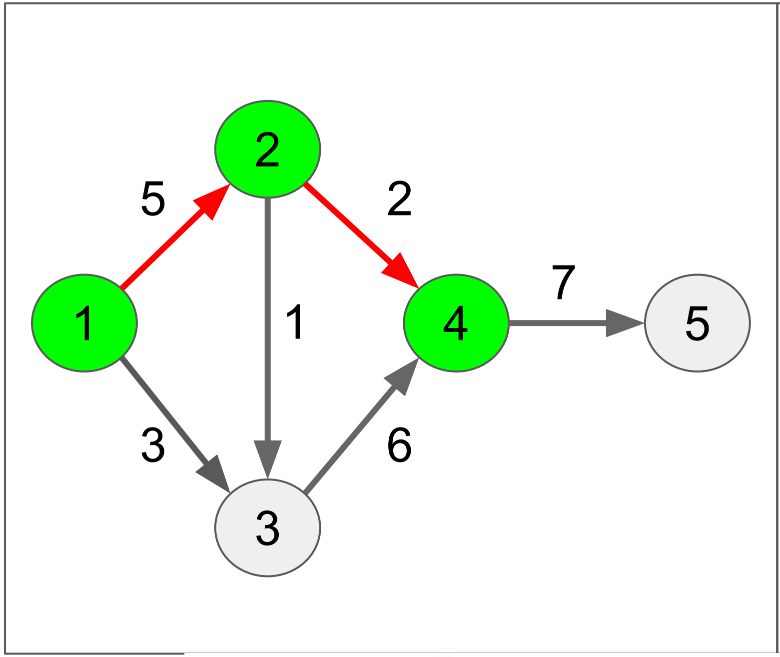
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s, a) + \text{FutureCost}(\text{Succ}(s, a))] & \text{otherwise} \end{cases}$$

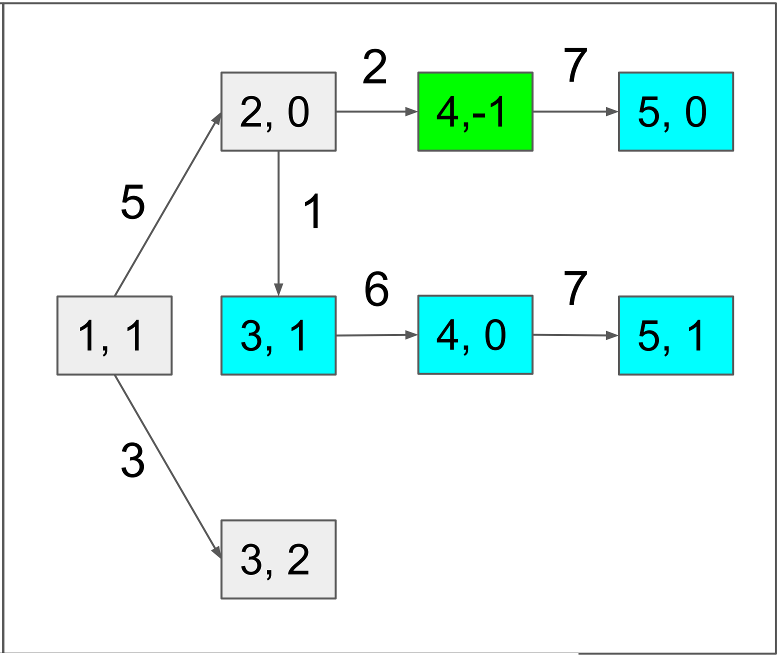
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



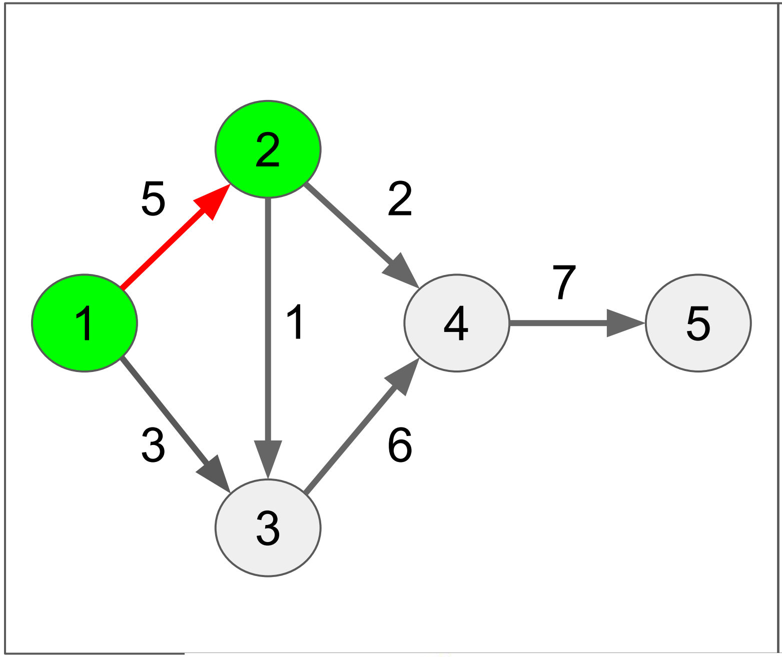
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

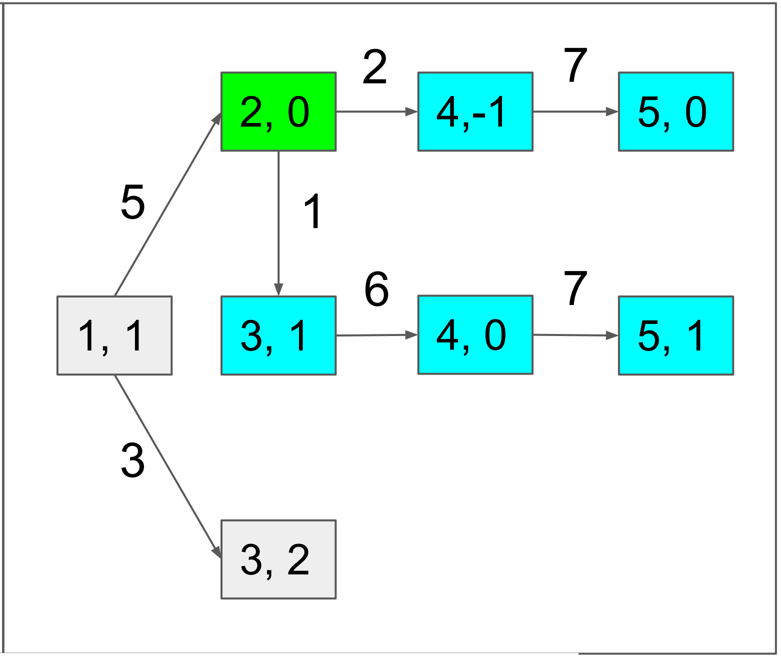
Simulation of DP

Visiting	Successors	Completed

Regular Graph



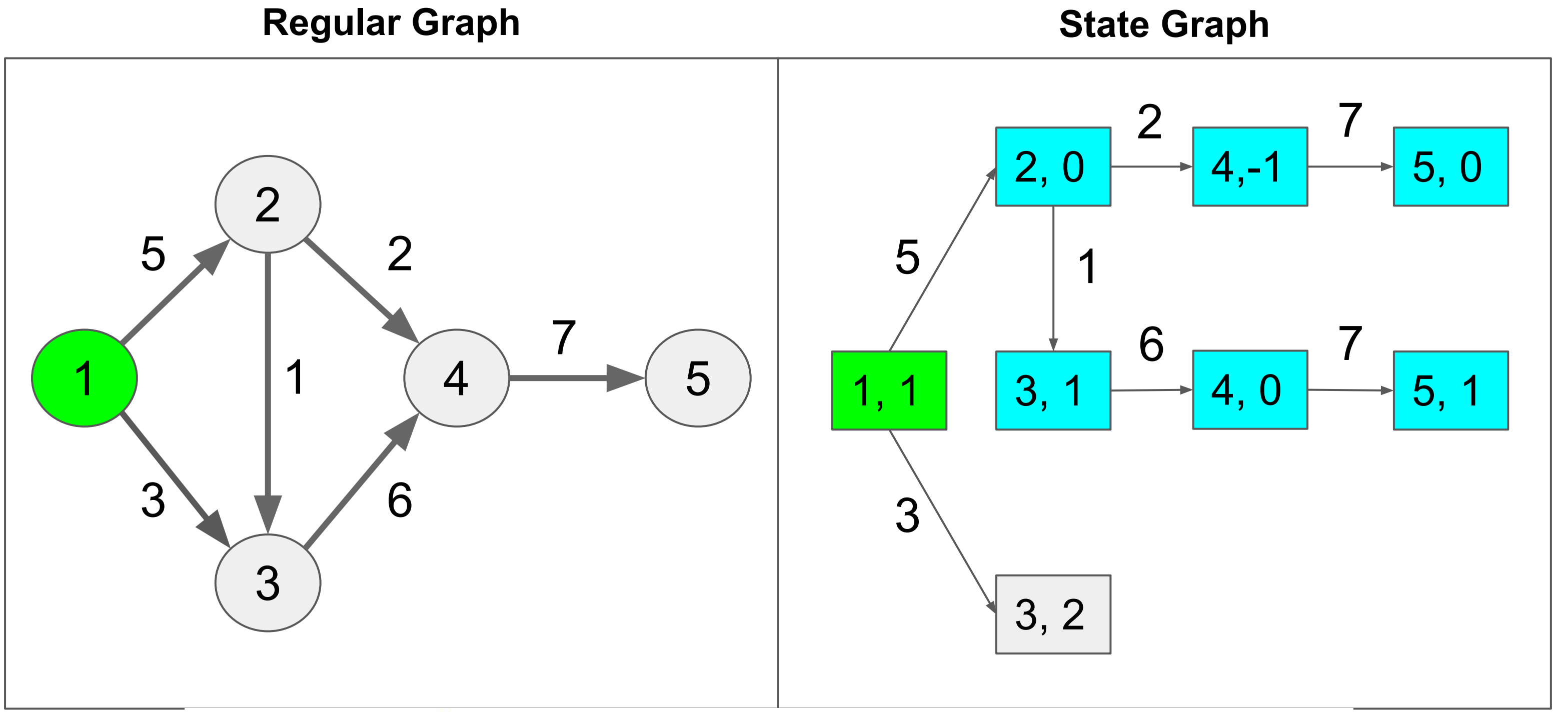
State Graph



Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	13+1 = 14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s, a) + \text{FutureCost}(\text{Succ}(s, a))] & \text{otherwise} \end{cases}$$

Simulation of DP

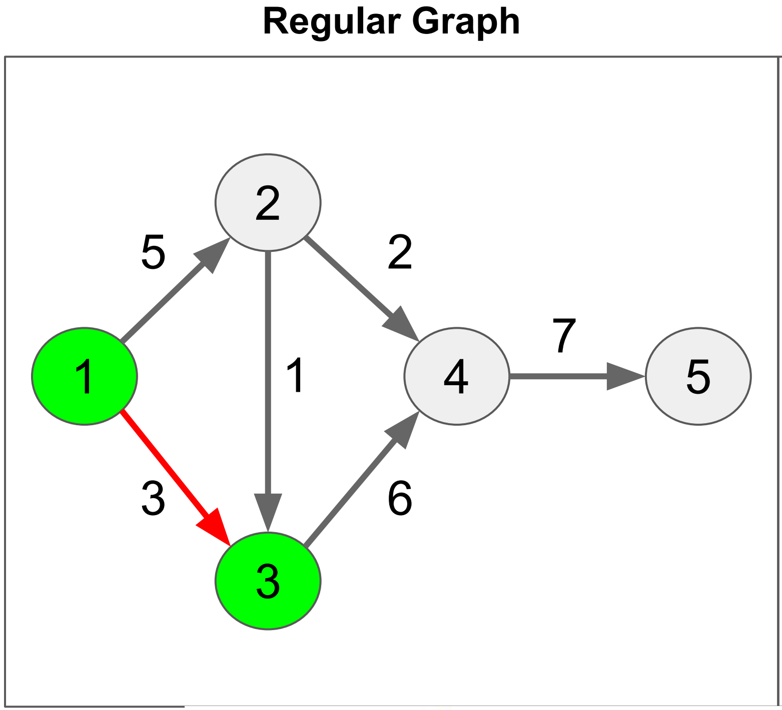


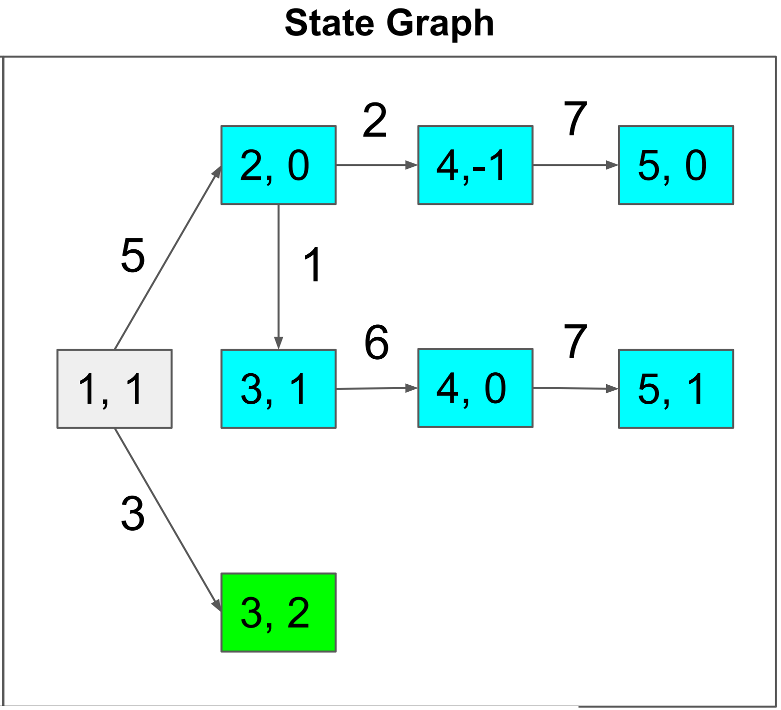
Cache	
Key	Value
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

Visiting	Successors	Completed





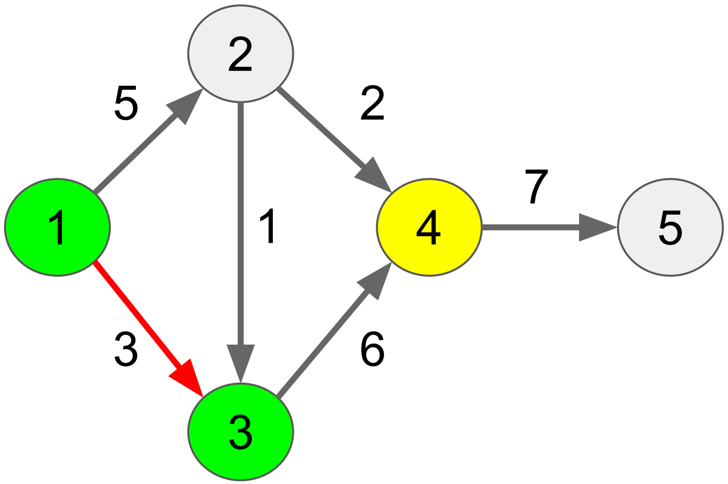
Cache	
Key	Value
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

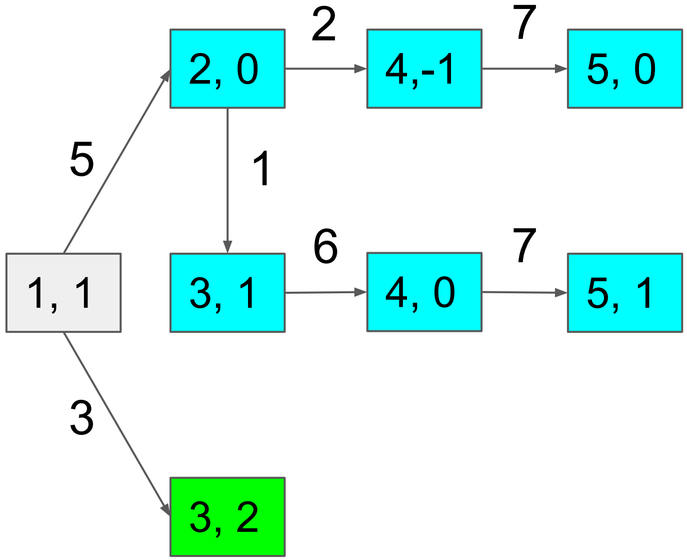
Simulation of DP

Visiting | **Successors** | **Completed**

Regular Graph



State Graph



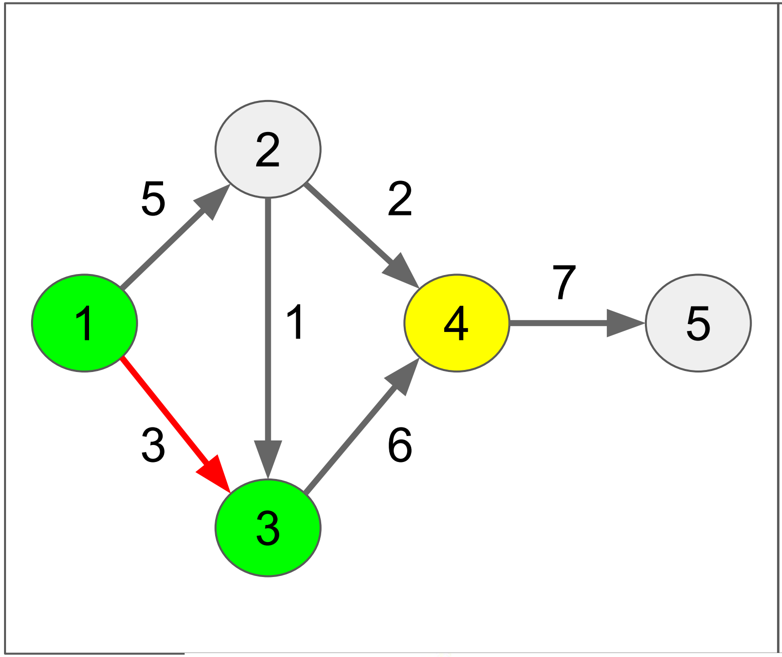
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

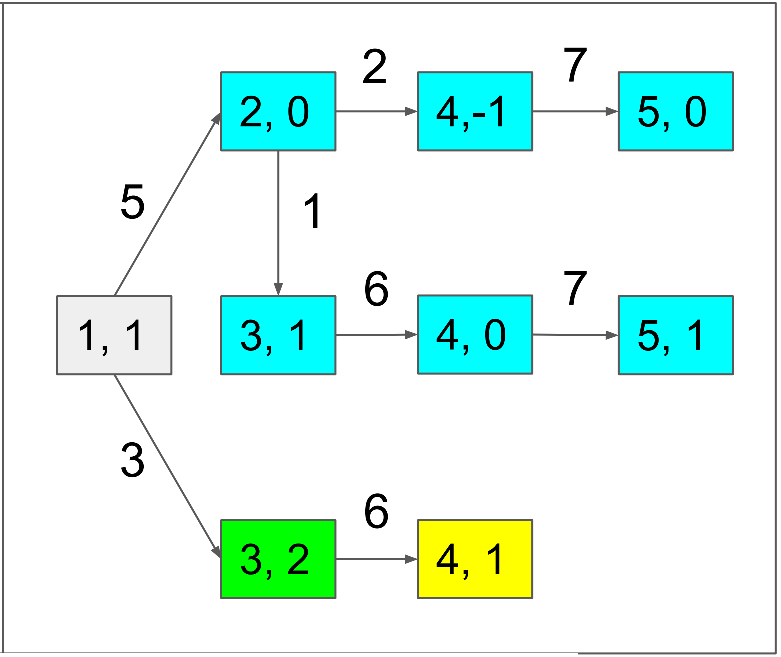
Simulation of DP

Visiting | **Successors** | **Completed**

Regular Graph



State Graph

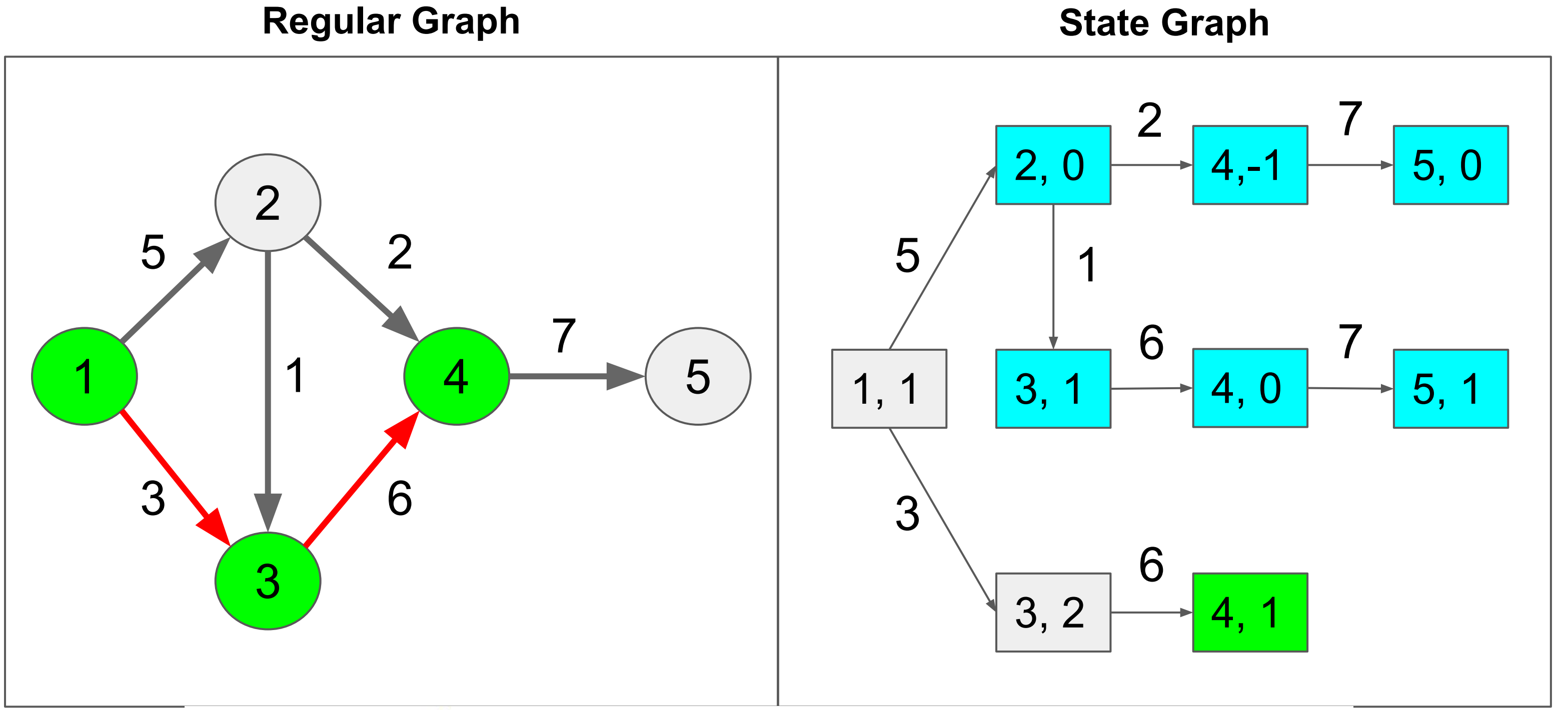


Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

Visiting	Successors	Completed

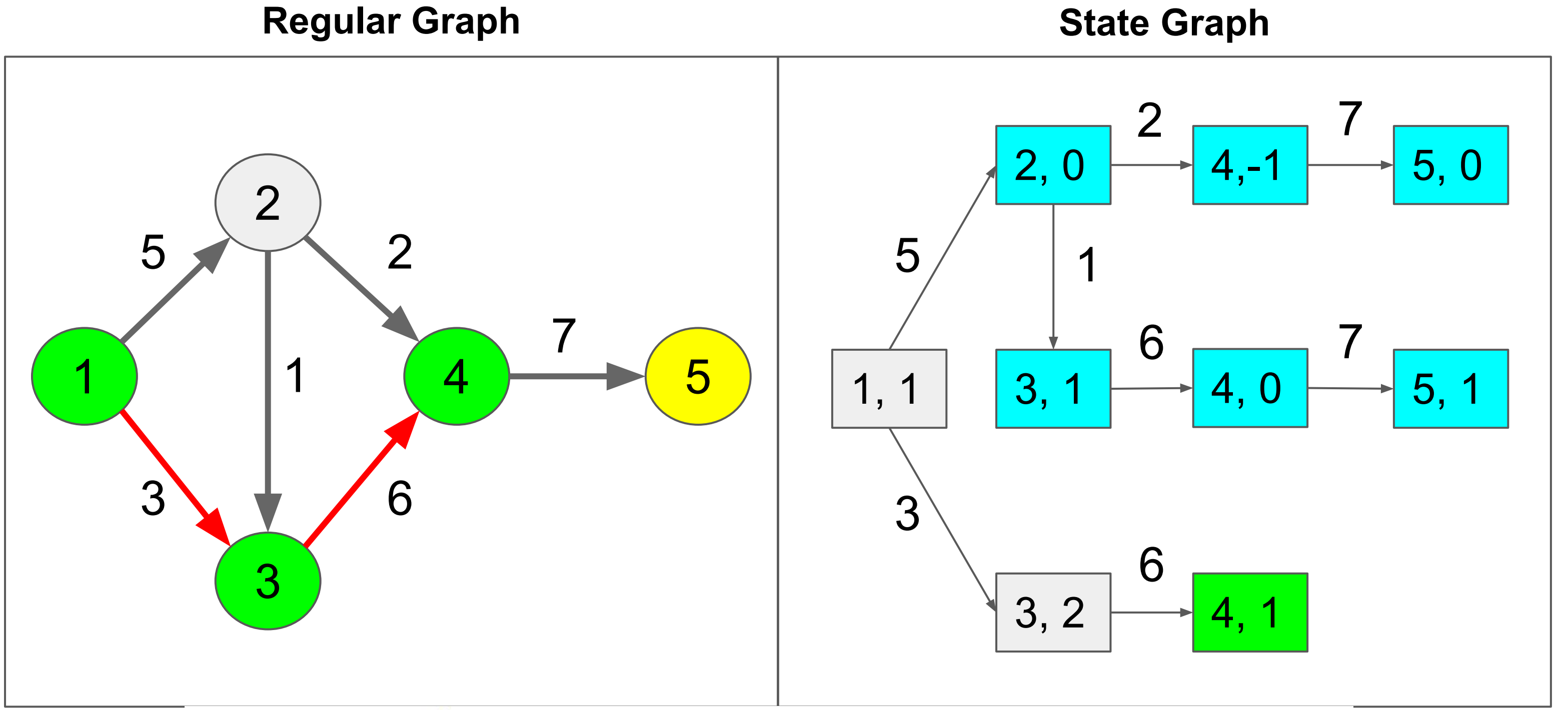


Cache	
Key	Value
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

Visiting	Successors	Completed

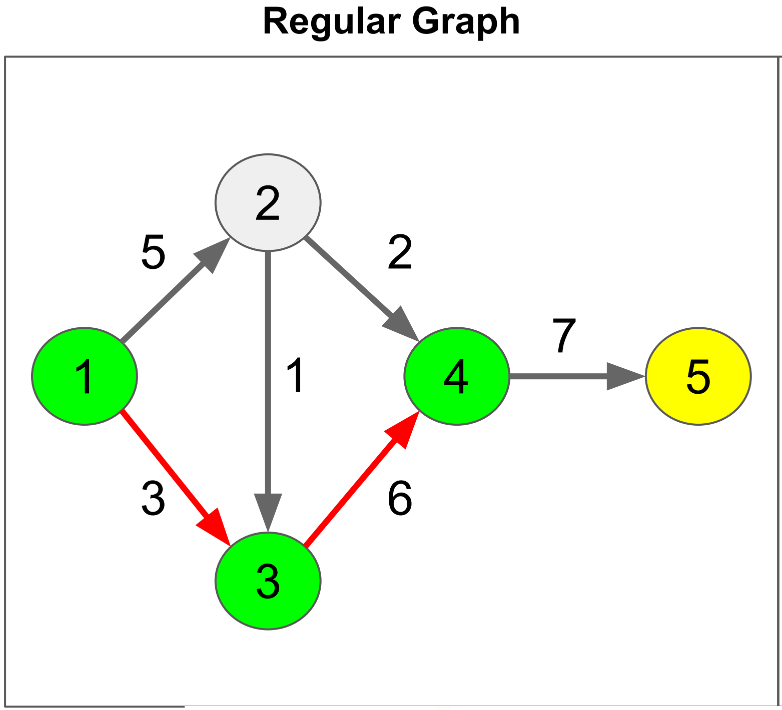


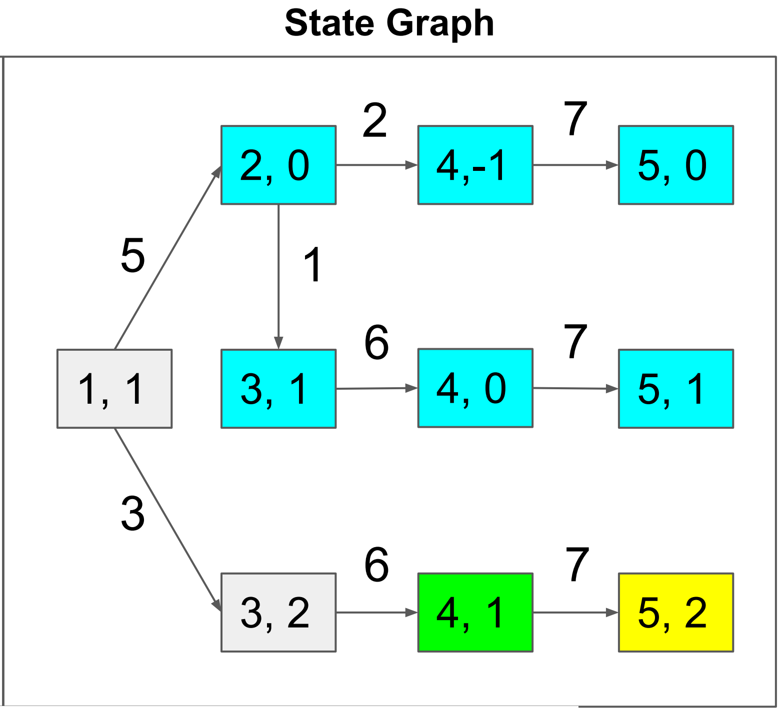
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

Visiting	Successors	Completed





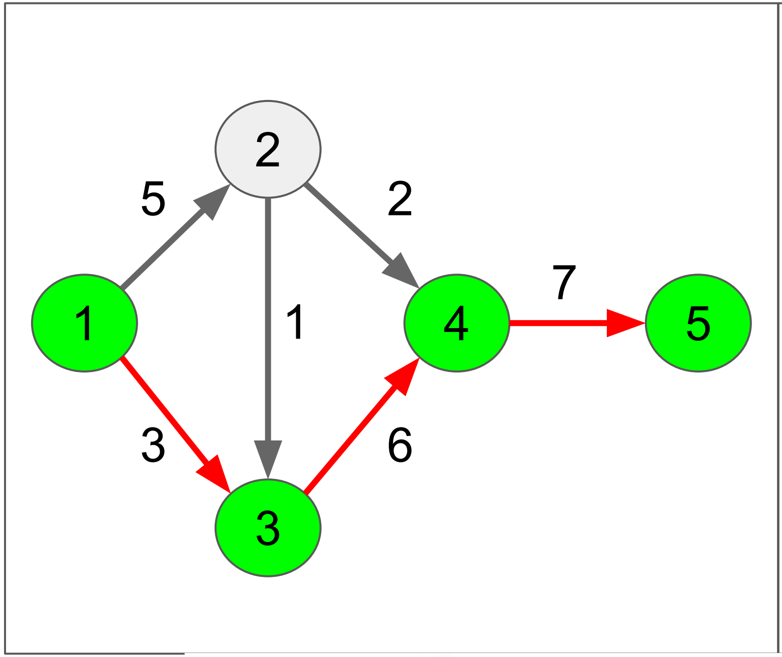
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

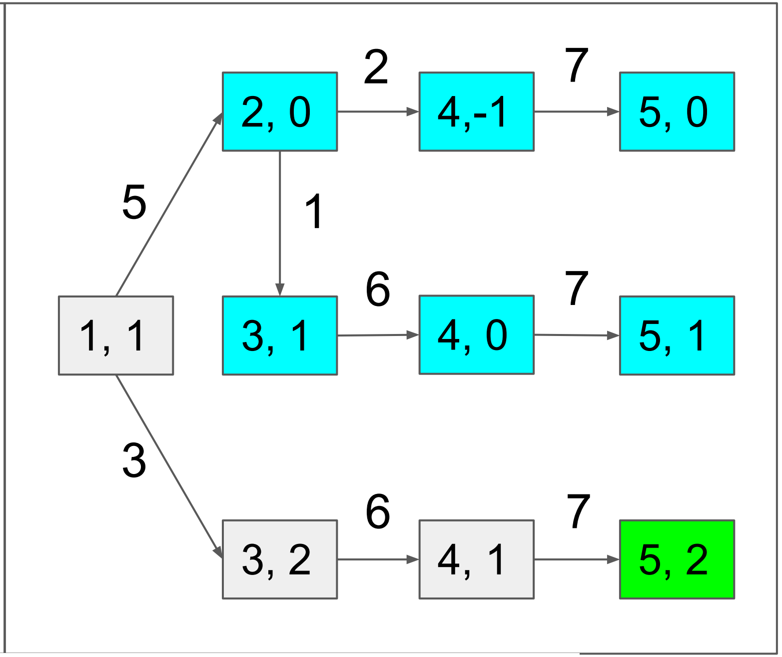
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph



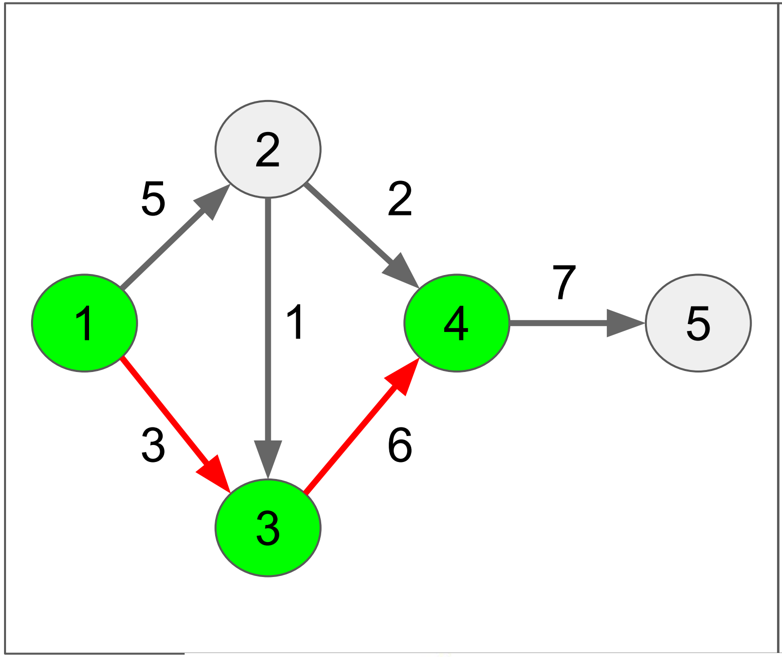
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14
(5, 2)	0

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

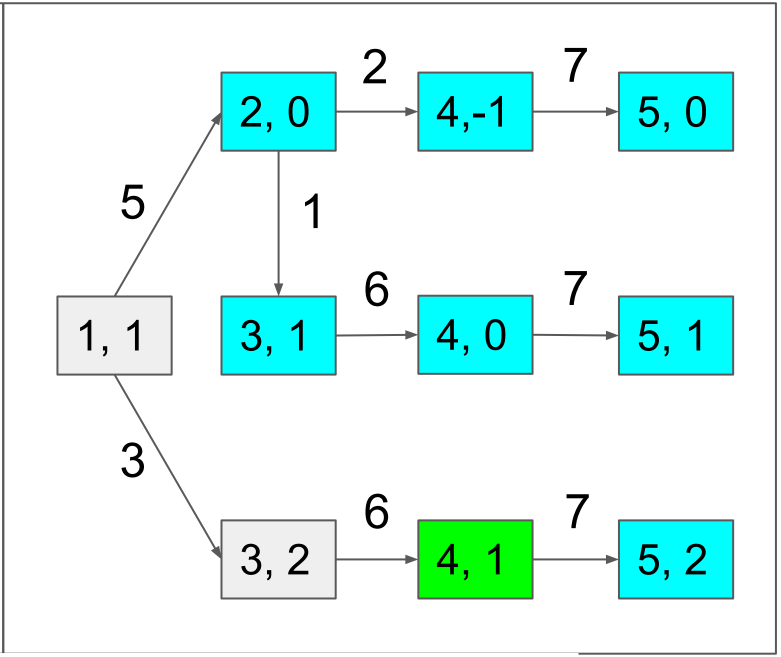
Simulation of DP

Visiting | Successors | Completed

Regular Graph



State Graph



Cache

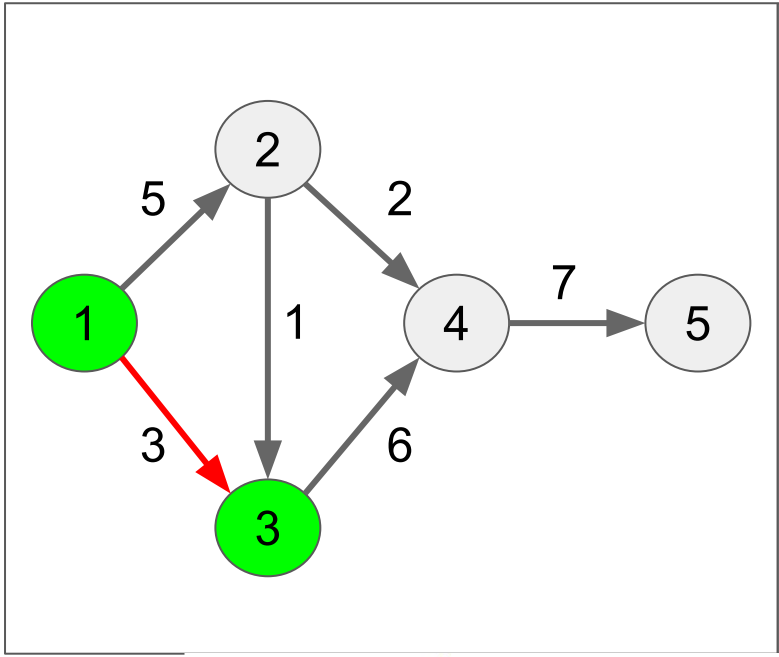
Key	Value
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14
(5, 2)	0
(4, 1)	7

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

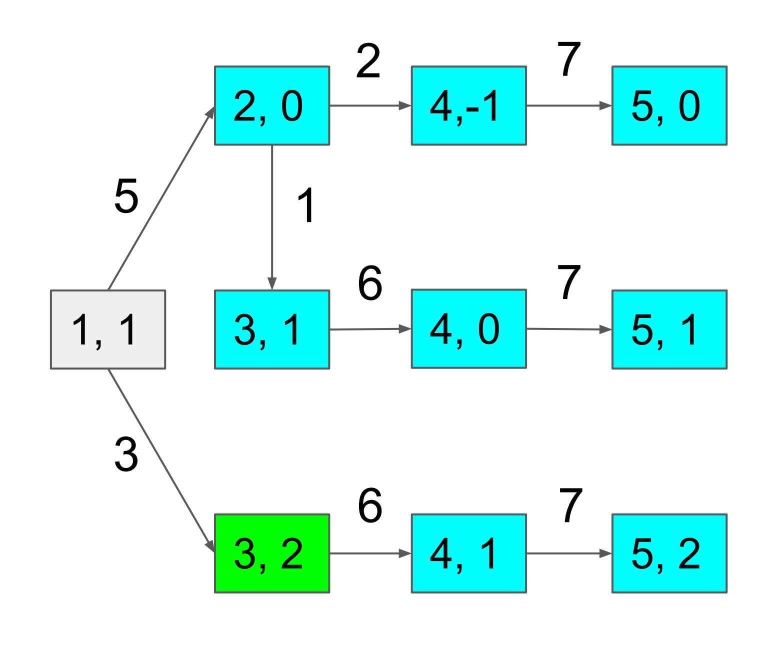
Simulation of DP

Visiting	Successors	Completed

Regular Graph



State Graph

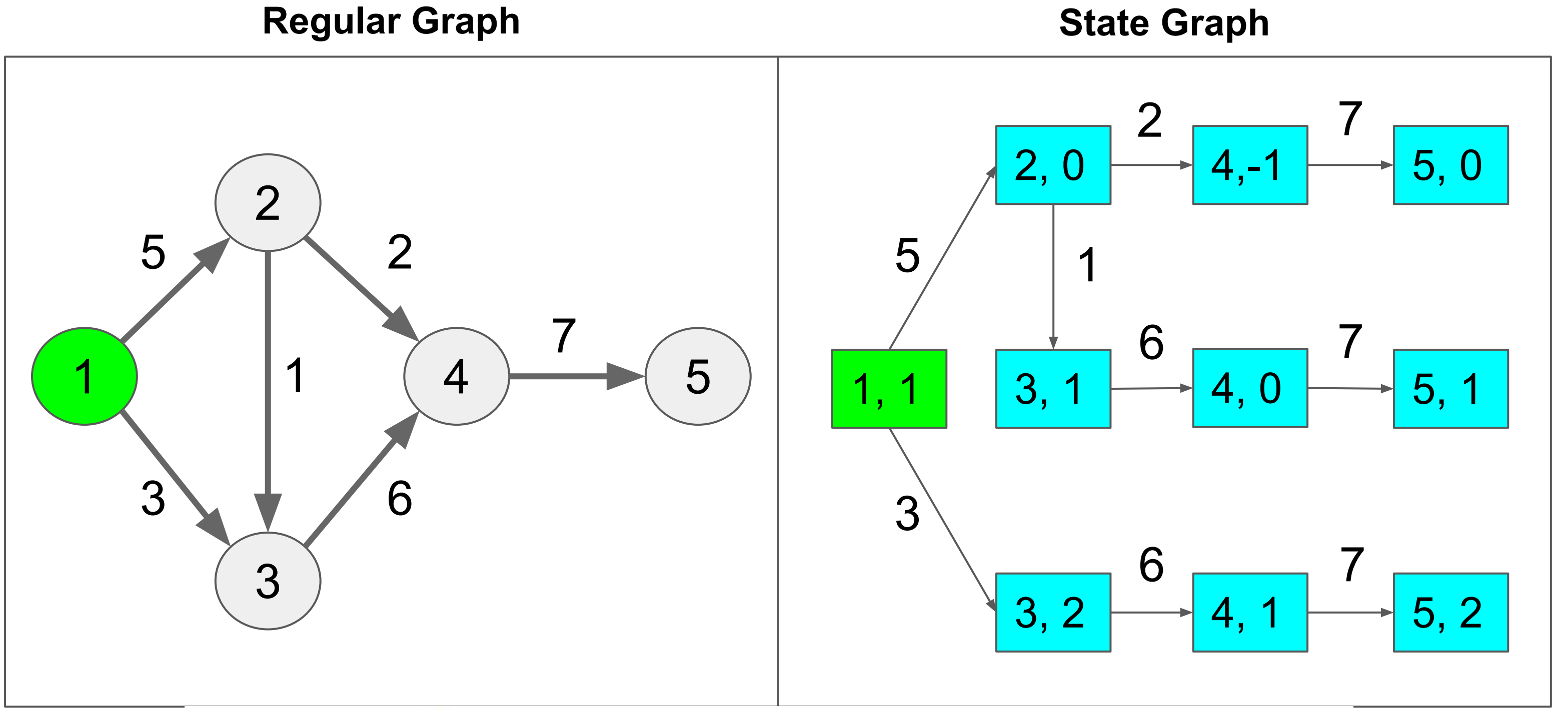


Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14
(5, 2)	0
(4, 1)	7
(3, 2)	13

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Simulation of DP

Visiting	Successors	Completed



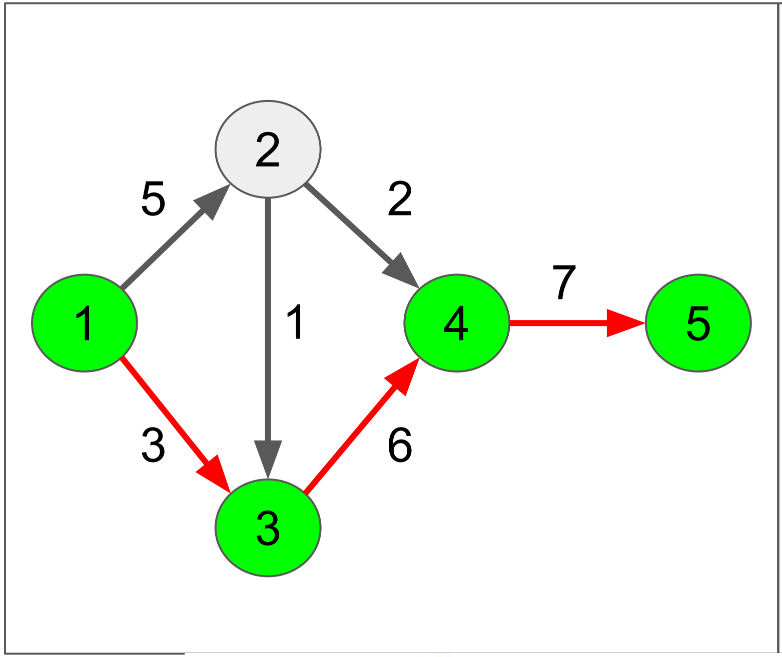
Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14
(5, 2)	0
(4, 1)	7
(3, 2)	13
(1, 1)	16

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

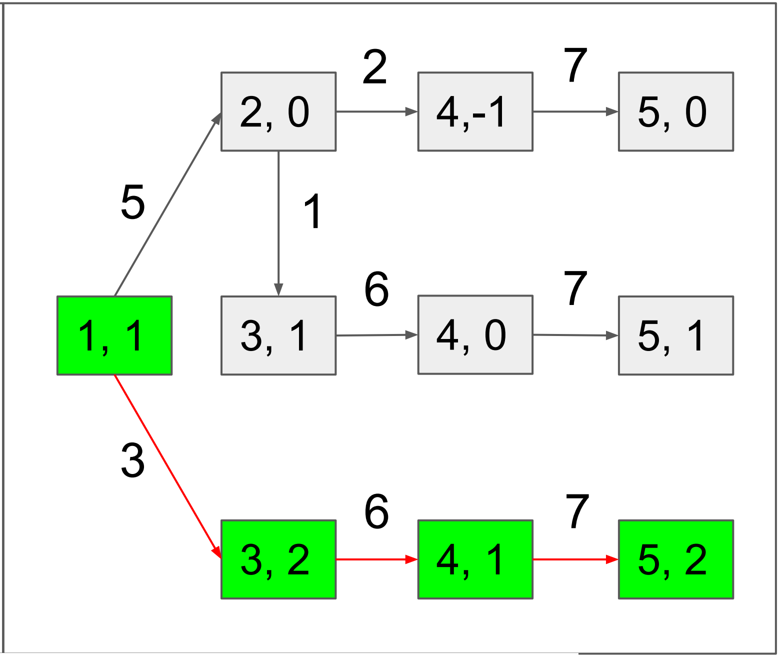
Simulation of DP

Visiting | **Successors** | **Completed**

Regular Graph



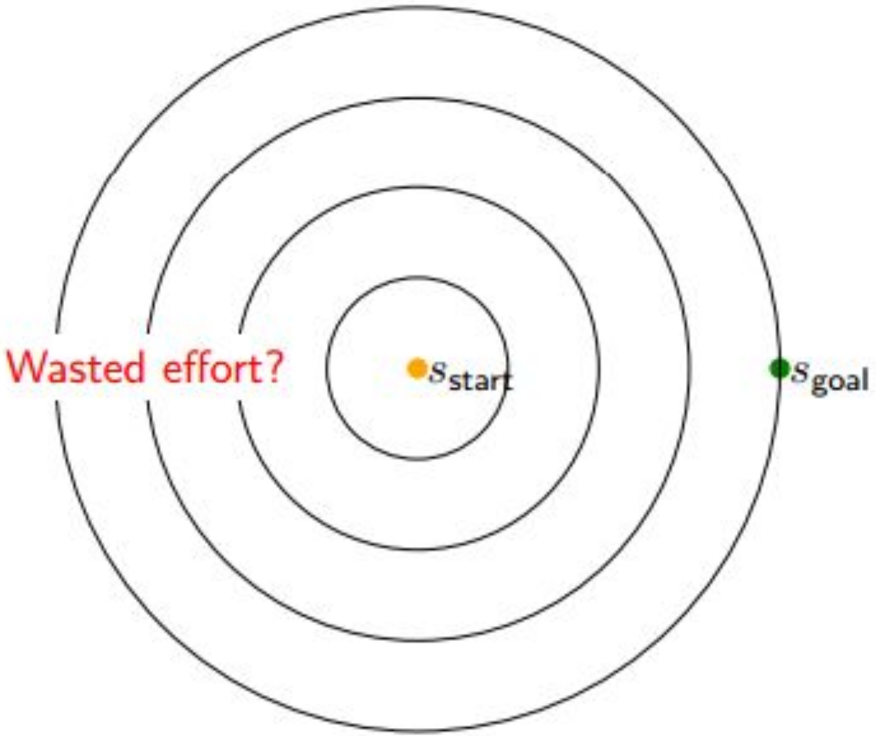
State Graph



Cache	
Key	**Value**
(5, 1)	0
(4, 0)	7
(3, 1)	13
(5, 0)	∞
(4, -1)	∞
(2, 0)	14
(5, 2)	0
(4, 1)	7
(3, 2)	13
(1, 1)	16

$$\text{FutureCost}(s) = \begin{cases} 0 & \text{if IsGoal}(s) \\ \min_{a \in \text{Actions}(s)} [\text{Cost}(s,a) + \text{FutureCost}(\text{Succ}(s,a))] & \text{otherwise} \end{cases}$$

Improve UCS: A* Search



Contents

1. Uniform Cost Search
2. Defining States
3. Dynamic Programming
4. **A* Search**

Recap of A* Search

- We want to avoid wasted effort (to go from SF to LA, we probably don't want to end up looking at roads to Seattle, for example).
- To do this, we can use a heuristic to estimate how far is left until we reach our goal.
- The heuristic **must be optimistic**. It must underestimate the true cost. Why?

Recap of A* Search

- Modify the cost of edges and run UCS on the new graph
  - New cost = Current cost + future cost
  - Cost'(s, a) = Cost(s, a) + h(Succ(s, a)) - h(s)
- You can find a good consistent h by performing relaxation.
- If c is min cost on original graph, c' is min cost on modified graph, then c' = c + h(s_goal) - h(s_start)

Relaxation

A good way to come up with a reasonable heuristic is to solve an easier (less constrained) version of the problem

For example, we can use geographic distance as a heuristic for distance if we have the positions of nodes.

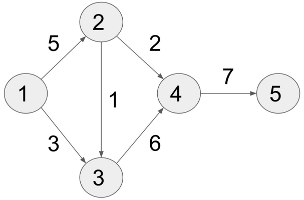
Note: The main point of relaxation is to attain a problem that **can be solved more efficiently**.

How to compute h for our example?

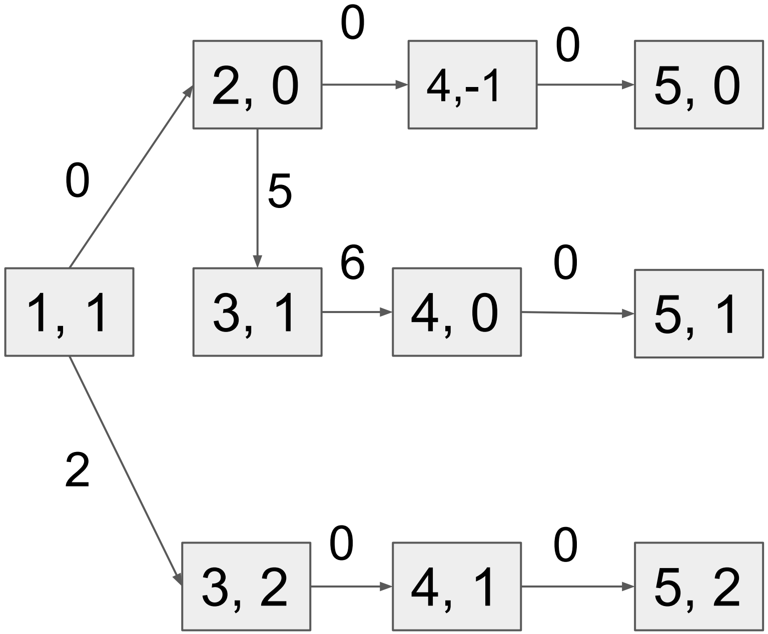
Consider again our example from before. Suppose we ignore the constraint that there must be more odd cities visited. This is a relaxation of the problem. The following is h for our graph:

city	1	2	3	4	5
h	14	9	13	7	0

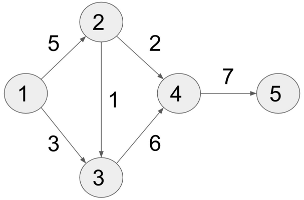
Modified State Graph



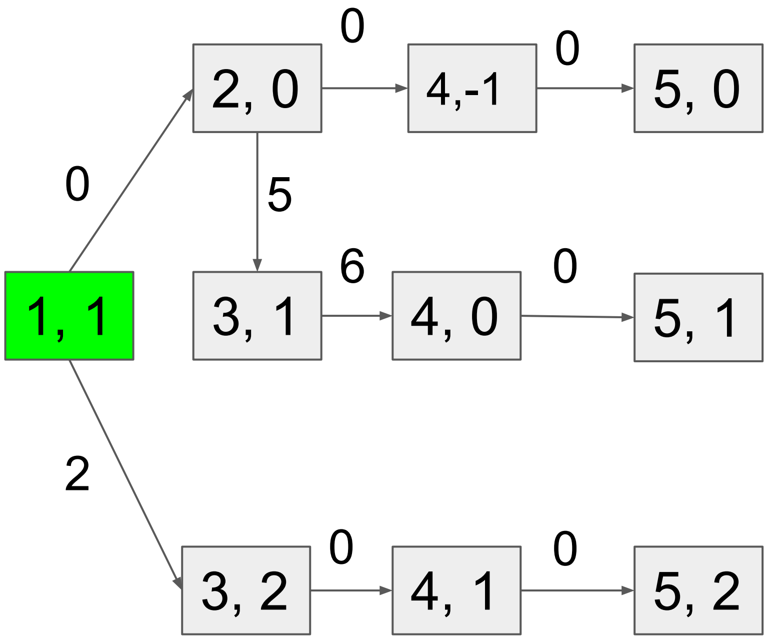
city	1	2	3	4	5
h	14	9	13	7	0



Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



Explored:

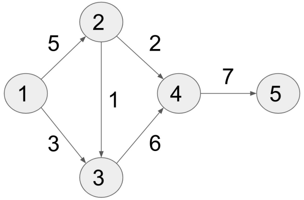
(1, 1) : 0

Frontier:

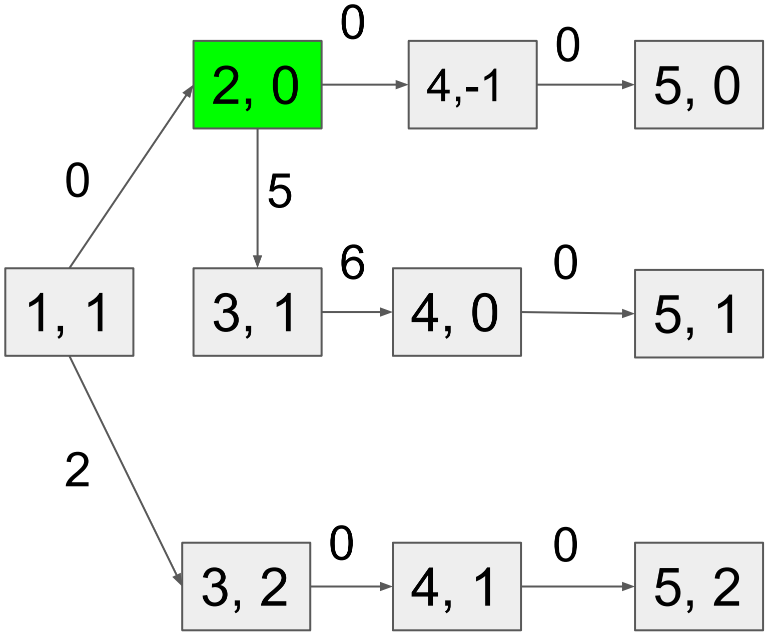
(2, 0) : 5 + 9 - 14 = 0

(3, 2) : 3 + 13 - 14 = 2

Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



Explored:

(1, 1) : 0

(2, 0): 0

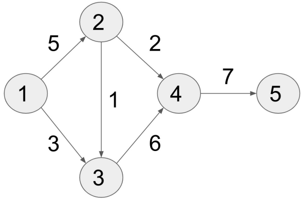
Frontier:

(3, 2) : 3 + 13 - 14 = 2

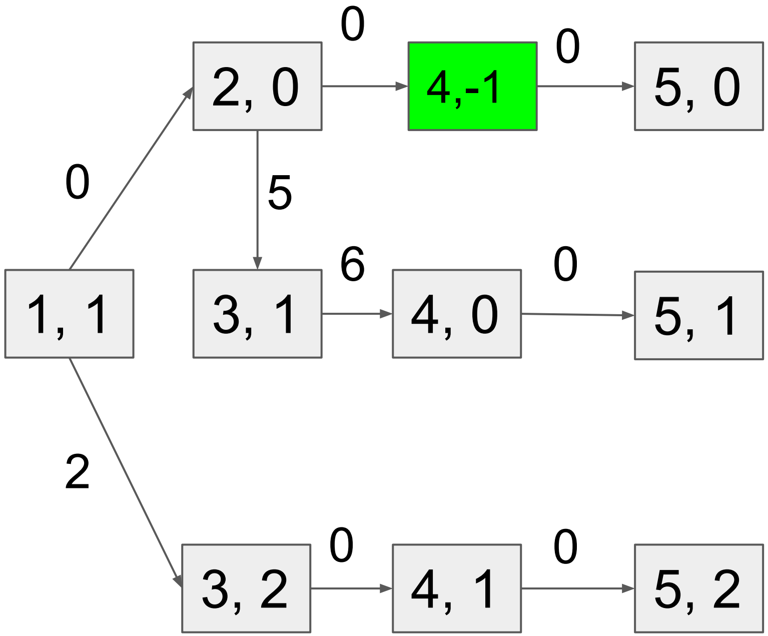
(3, 1): 1 + 13 - 9 = 5

(4, -1): 2 + 7 - 9 = 0

Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



Explored:

(1, 1) : 0

(2, 0): 0

(4, -1): 0

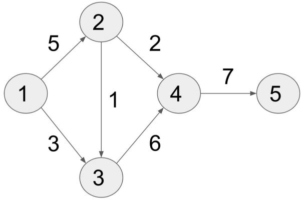
Frontier:

(3, 2) : 3 + 13 - 14 = 2

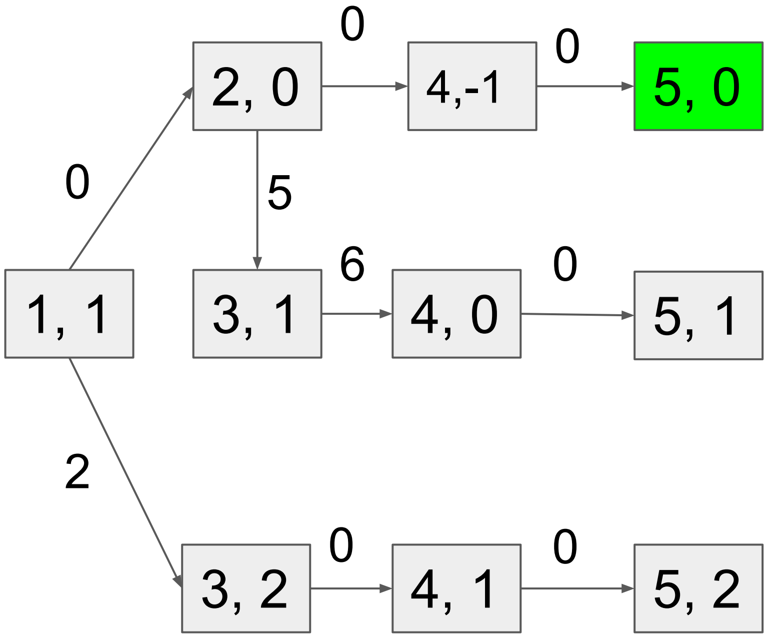
(3, 1): 1 + 13 - 9 = 5

(5, 0): 7 + 0 - 7 = 0

Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



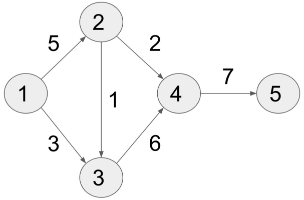
Explored:

(1, 1) : 0
(2, 0): 0
(4, -1): 0
(5, 0): 0

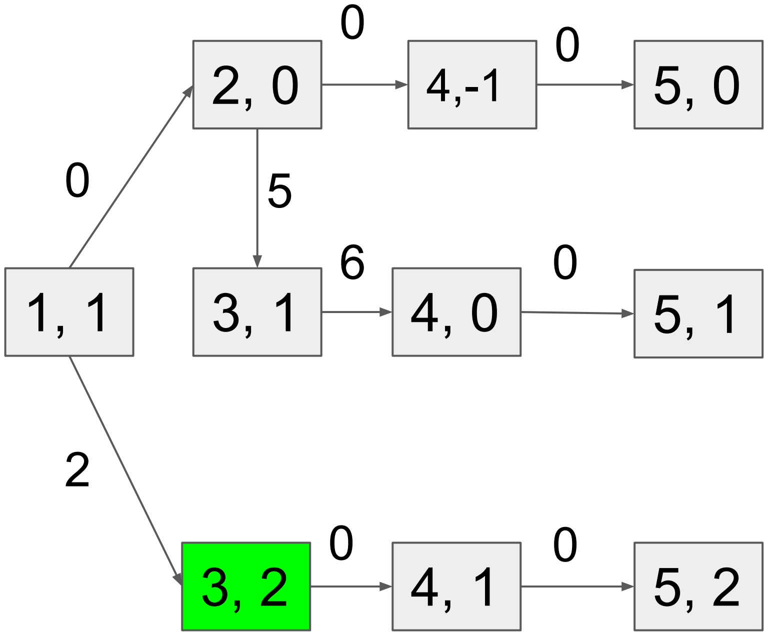
Frontier:

(3, 2) : 3 + 13 - 14 = 2
(3, 1): 1 + 13 - 9 = 5

Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



Explored:

(1, 1) : 0

(2, 0): 0

(4, -1): 0

(5, 0): 0

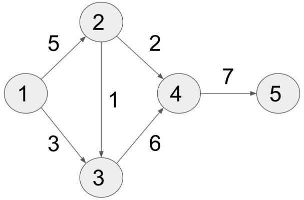
(3, 2): 2

Frontier:

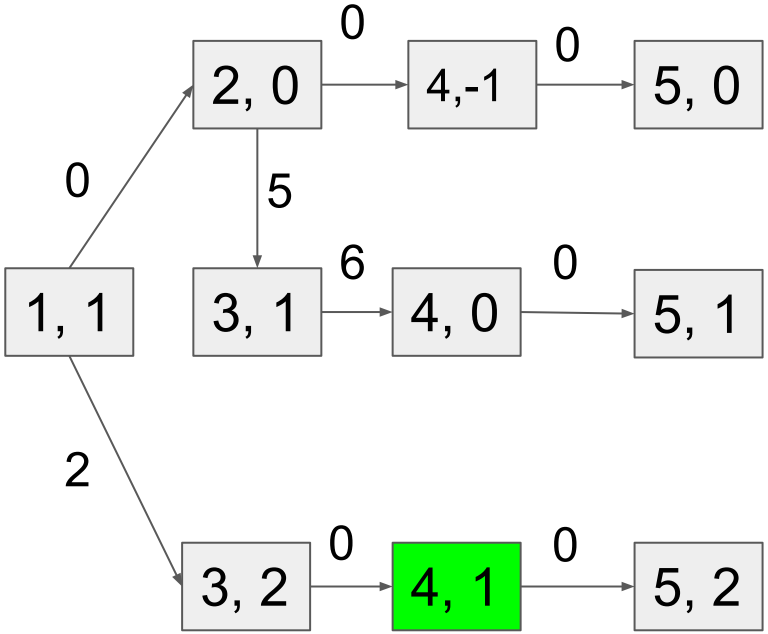
(3, 1): 1 + 13 - 9 = 5

(4, 1): 6 + 7 - 13 = 0

Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



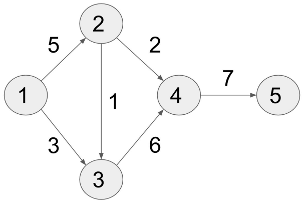
Explored:

(1, 1) : 0
(2, 0): 0
(4, -1): 0
(5, 0): 0
(3, 2): 2
(4, 1): 0

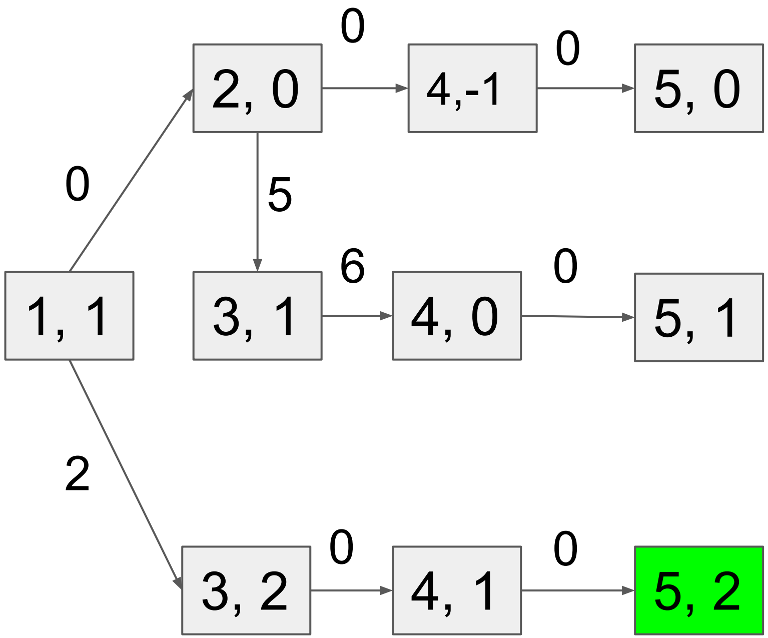
Frontier:

(3, 1): 1 + 13 - 9 = 5
(5, 2): 7 + 0 - 7 = 0

Simulation of UCS (A*)



city	1	2	3	4	5
h	14	9	13	7	0



Explored:

(1, 1) : 0
(2, 0): 0
(4, -1): 0
(5, 0): 0
(3, 2): 2
(4, 1): 0
(5, 2): 0

Frontier:

(3, 1): 1 + 13 - 9 = 5

STOP!

Comparison of States visited

UCS		UCS(A*)	
Explored:	Frontier:	Explored:	Frontier:
(1, 1) : 0	(5, 1) : 19	(1, 1) : 0	(3, 1): 5
(3, 2) : 3		(2, 0): 0	
(2, 0) : 5		(4, -1): 0	
(3, 1) : 6		(5, 0): 0	
(4, -1) : 7		(3, 2): 2	
(4, 1) : 9		(4, 1): 0	
(4, 0) : 12		(5, 2): 0	
(5, 0) : 14			
(5, 2) : 16			

Summary

- States Representation/Modelling
    - make state representation as compact as possible, remove unneccesary information
- DP
    - underlying graph cannot have cycles
    - visit all reachable states, but no log overhead
- UCS
    - actions cannot have negative cost
    - visit only a subset of states, log overhead
- A*
    - ensure that relaxed problem can be solved more efficiently