CS3243: Introduction to Artificial Intelligence

Semester 2, 2019/2020

Solving problems by Searching for Solutions

AIMA Chapter 3.1 – 3.4

Searching for Solutions

- Fully observable, deterministic, discrete environment
- Examples
  - Shortest route going through all cities
  - Winning sequence of moves in a **single player** game
  - Assembling a machine.
  - Find path through a maze
  - Route planning
  - Solution to Boolean satisfiability problem

Problem Formulation

Goal Test

- Is the state s equal the goal state?
- Explicit set of goal states, e.g., $\{In(Work)\}$
- Implicit function, e.g., $IsCheckmate(s)$

Path Cost

- Additive. e.g., sum of distances, number of actions executed
- $c(s, a, s')$: the step cost of taking action a in state s to reach state s'.
  Assumed to be ≥ 0

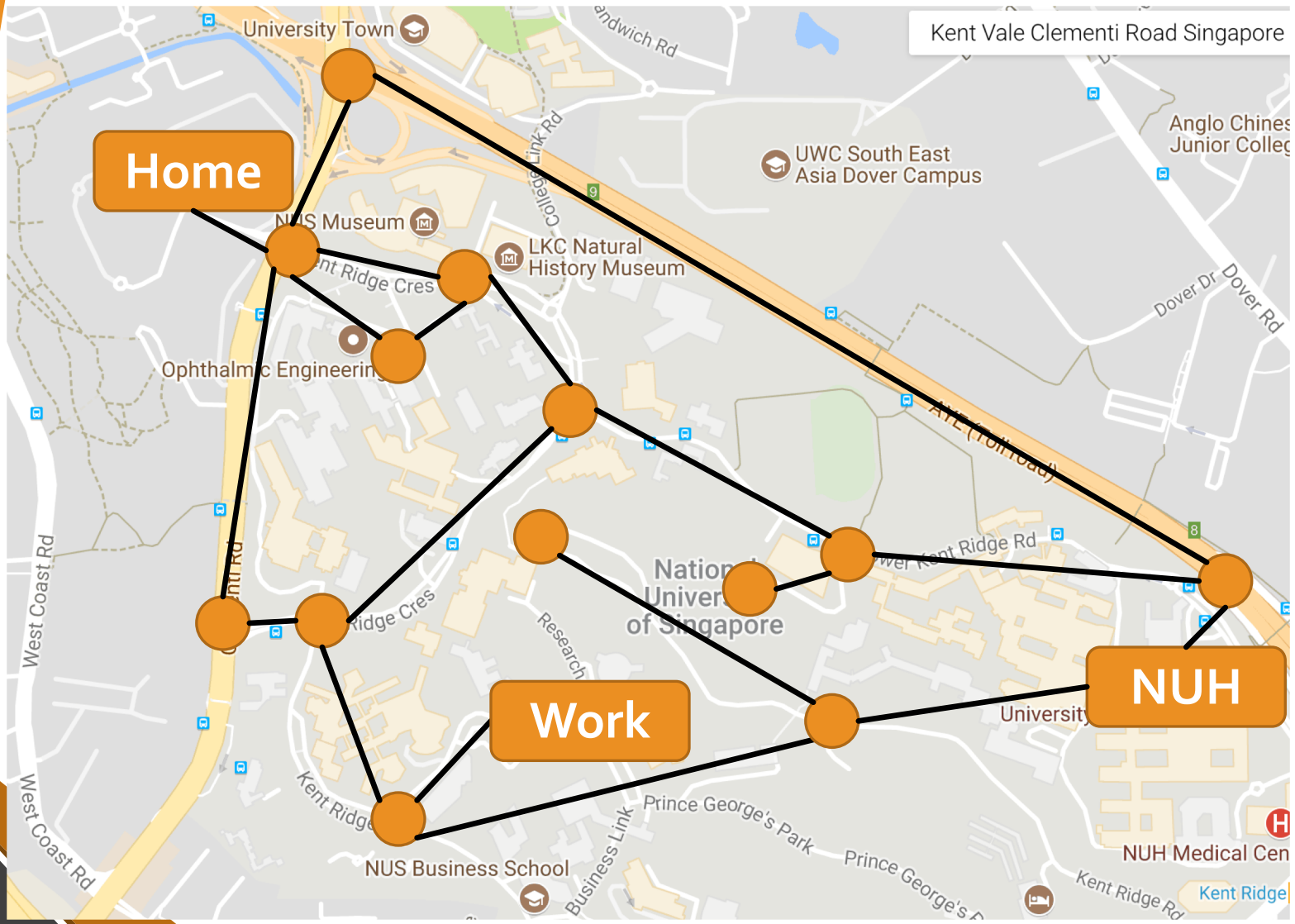
Solution: sequence of actions leading from initial to goal state

Is the solution unique?

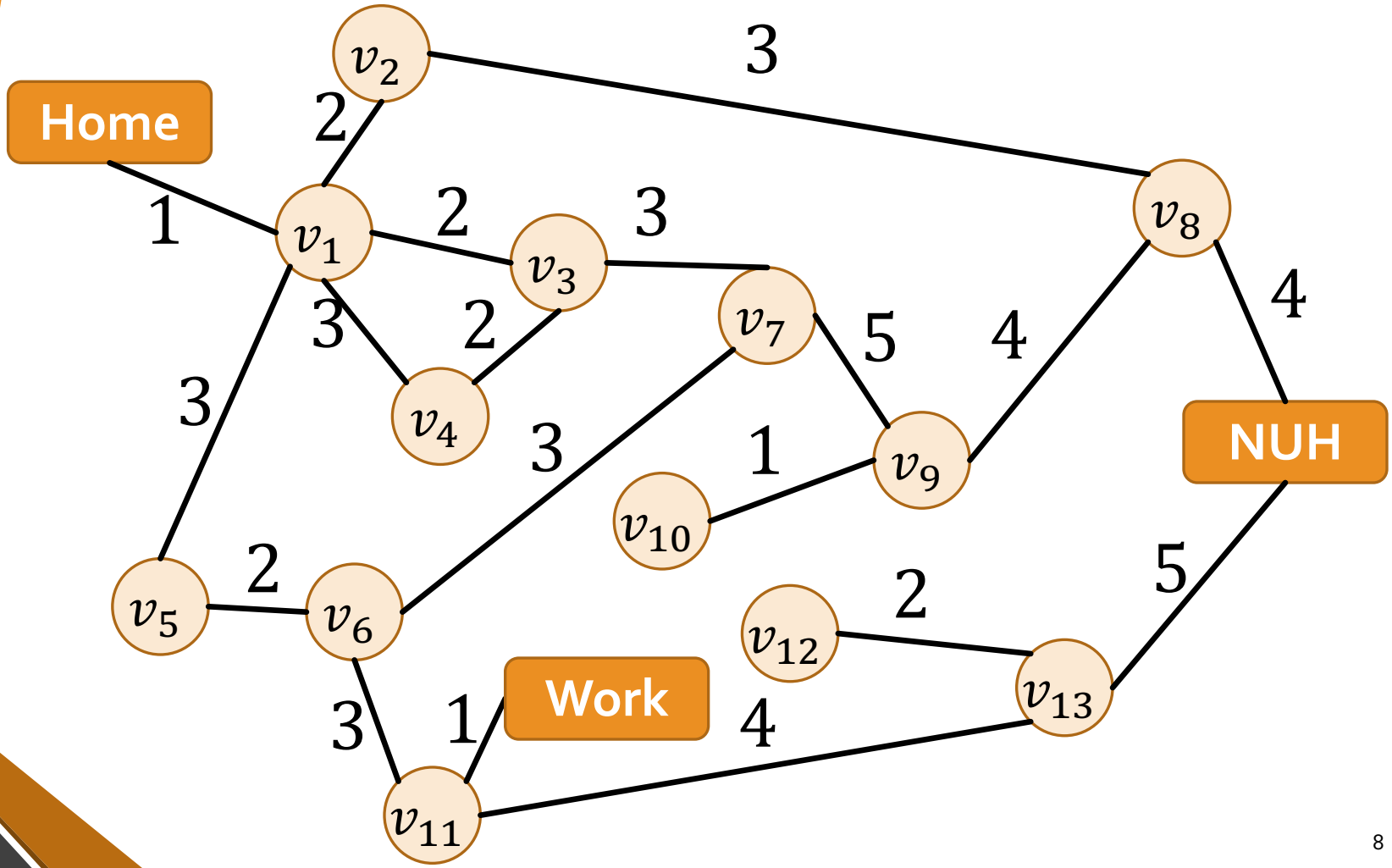
Is it optimal?

Can it be efficiently found?

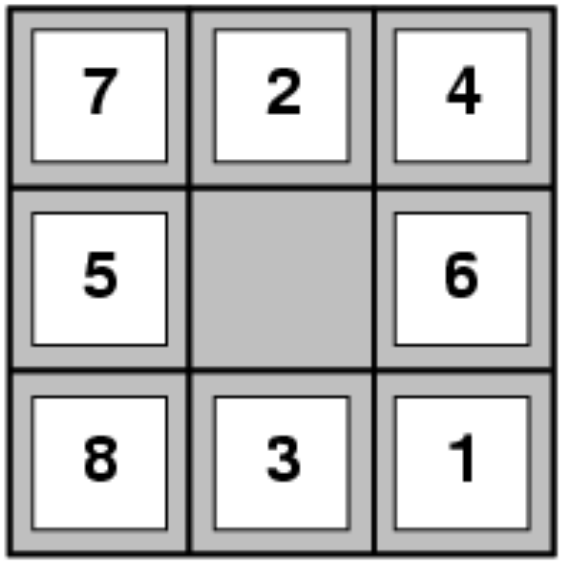
Example: Route Planning



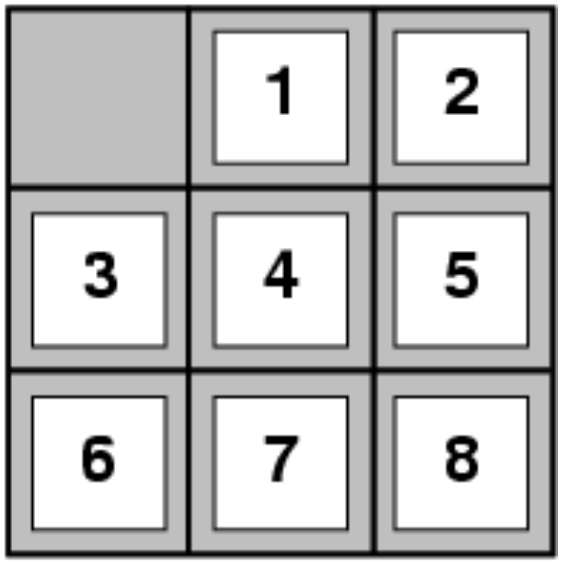
Example: Route Planning



Example: 8 Puzzle



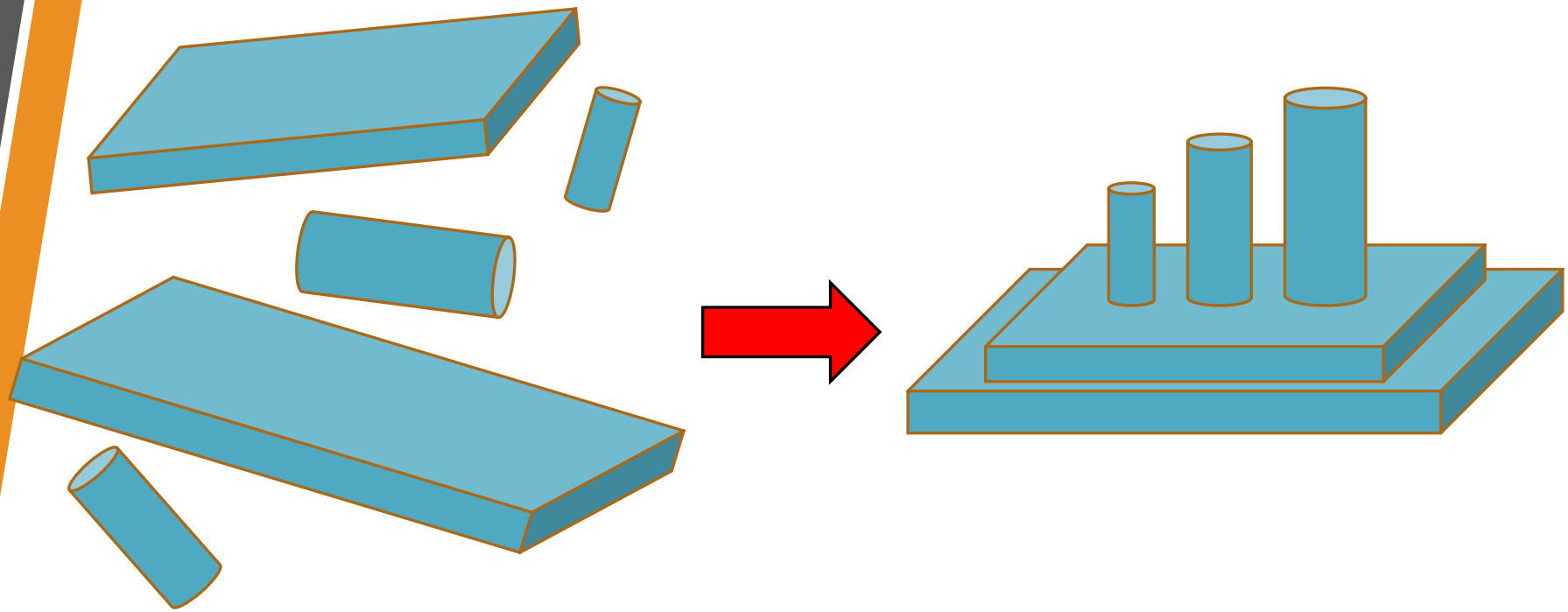
Start State



Goal State

1. States (Initial State)?
2. Actions?
3. Transition Model?
4. Goal Test?
5. Cost Function?

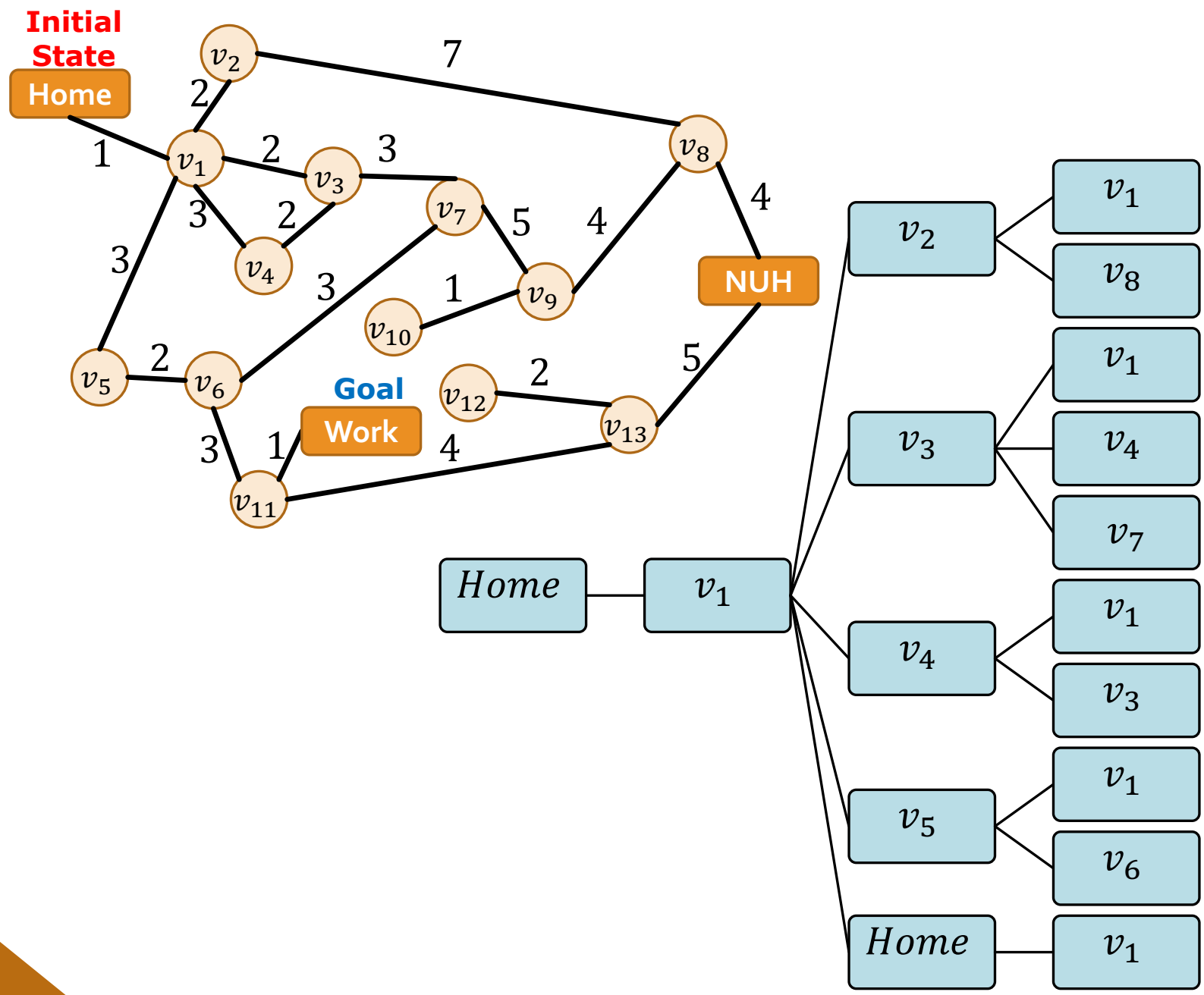
Example: Item Assembly



1. States (Initial State)?
2. Actions?
3. Transition Model?
4. Goal Test?
5. Cost Function?

Tree Search Algorithms

- Start at the source node, keep searching until we reach a goal state.
- Frontier: nodes that we have seen but haven't explored yet (at initialization: the frontier is just the source)
- At each iteration, choose a node from the frontier, explore it, and add its neighbors to the frontier.


Initial State
Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
Goal
Work
NUH
1
2
3
4
5
7
Home
v_1
v_2
v_3
v_4
v_5
Home
v_1
v_8
v_1
v_4
v_7
v_1
v_3
v_1
v_6
v_1

Traversal Methods

- In what order to we explore the frontier nodes? Search strategy
- Which neighbors do we add? All of them?

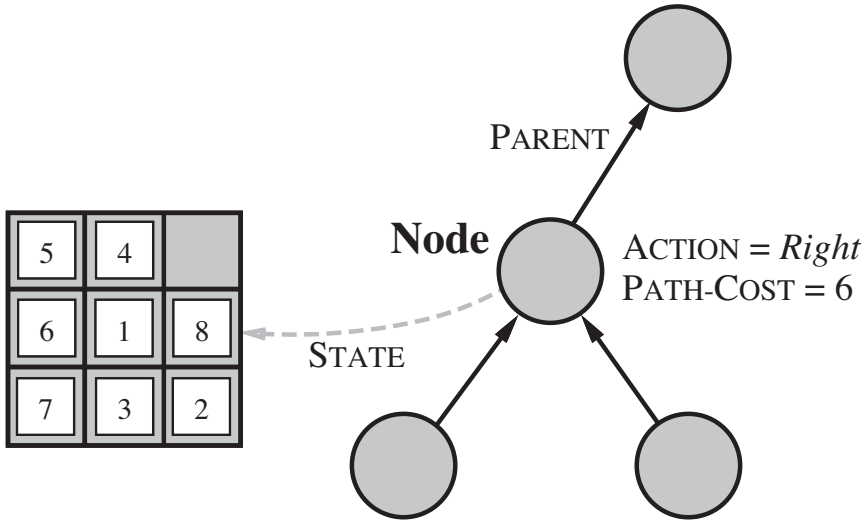
"Algorithms which do not remember their (search) history are doomed to repeat it"

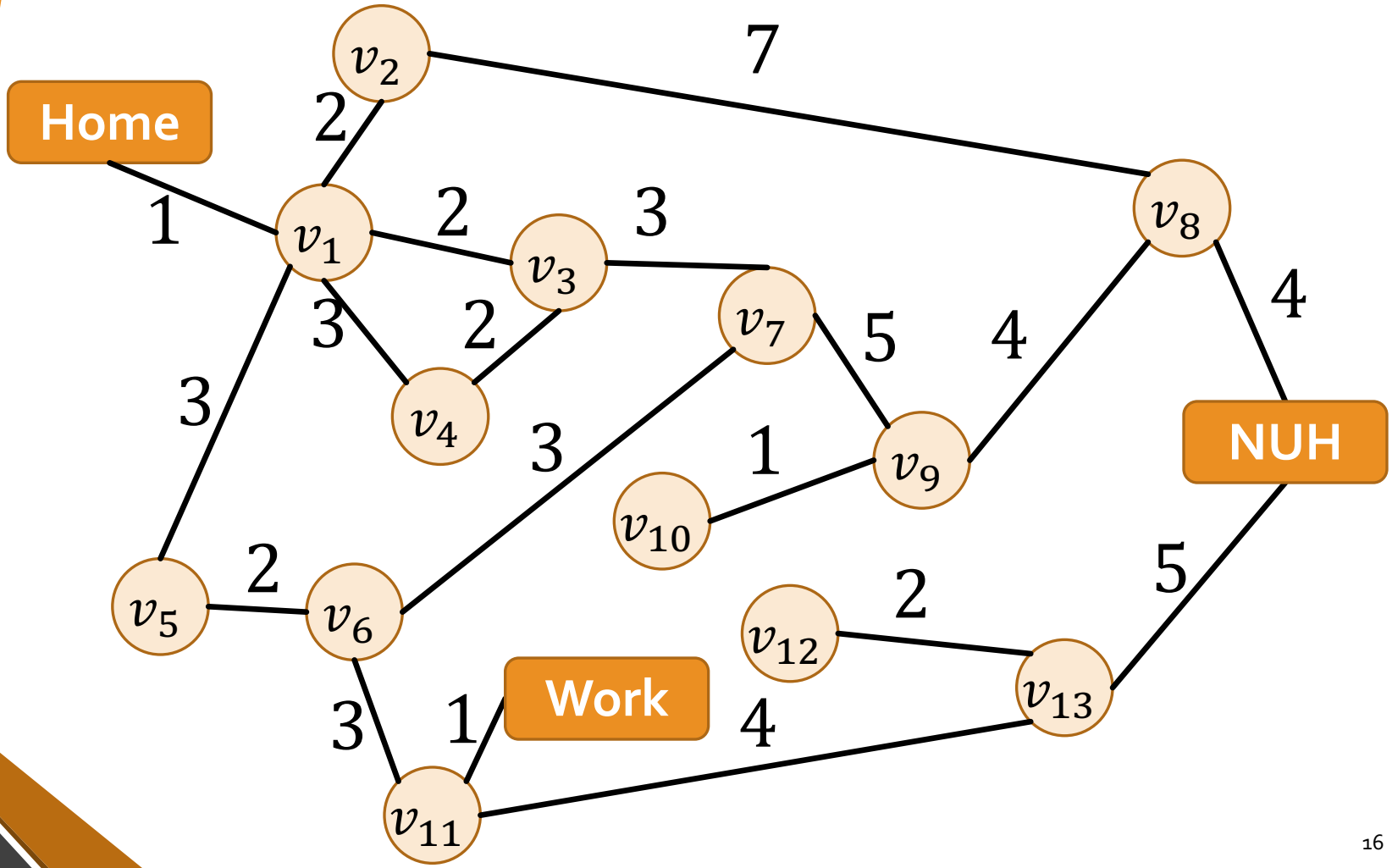
Graph Search

- A node that's been explored once will not be revisited.
- We will need to assume TREE-SEARCH is used for some theoretical results (next lecture)

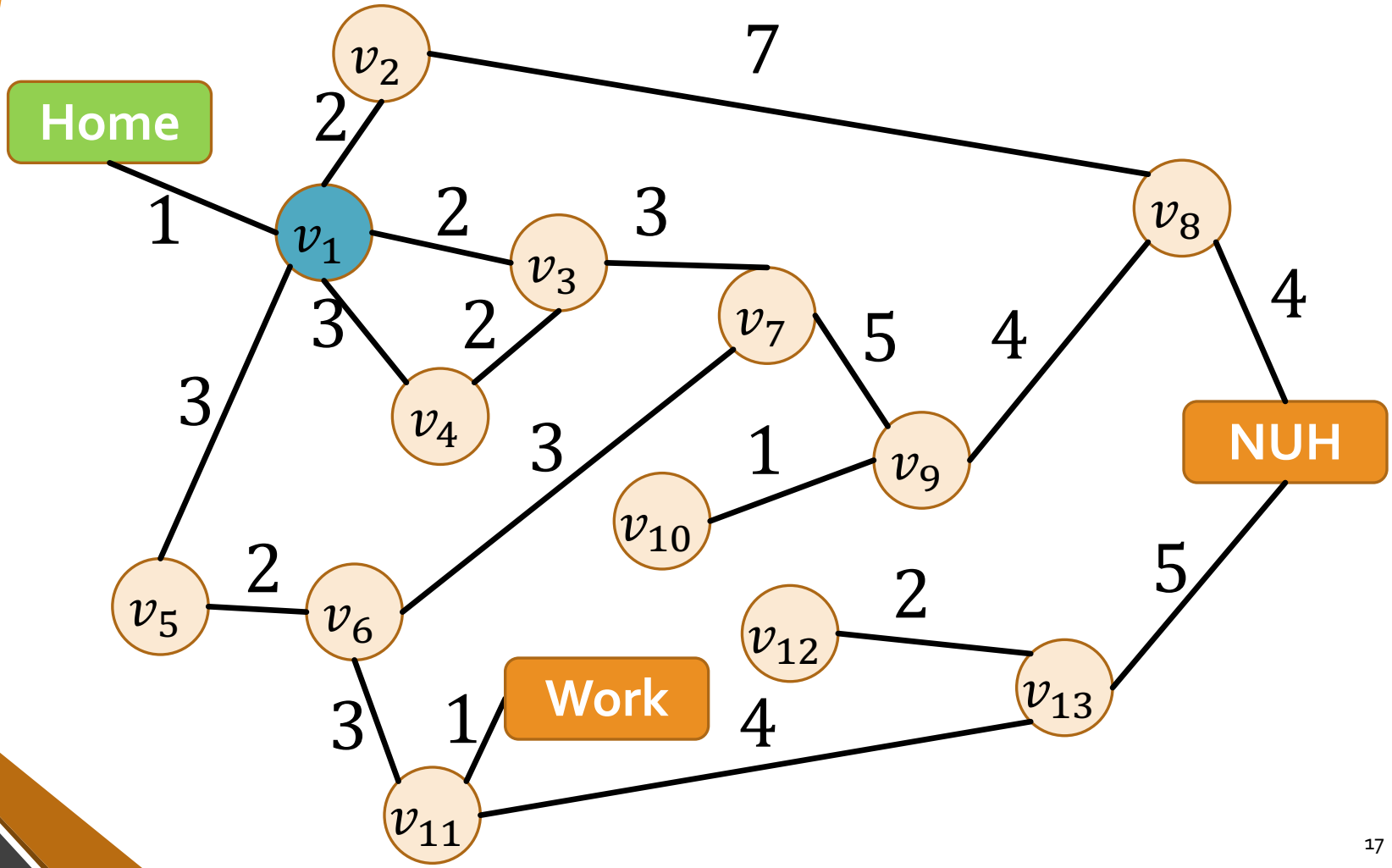
Implementation: States vs. Nodes

- A state represents a physical configuration
- A node is a data structure constituting part of search tree. It includes state, parent node, action, and path cost $g(n)$.
- Two different nodes are allowed to contain same world state



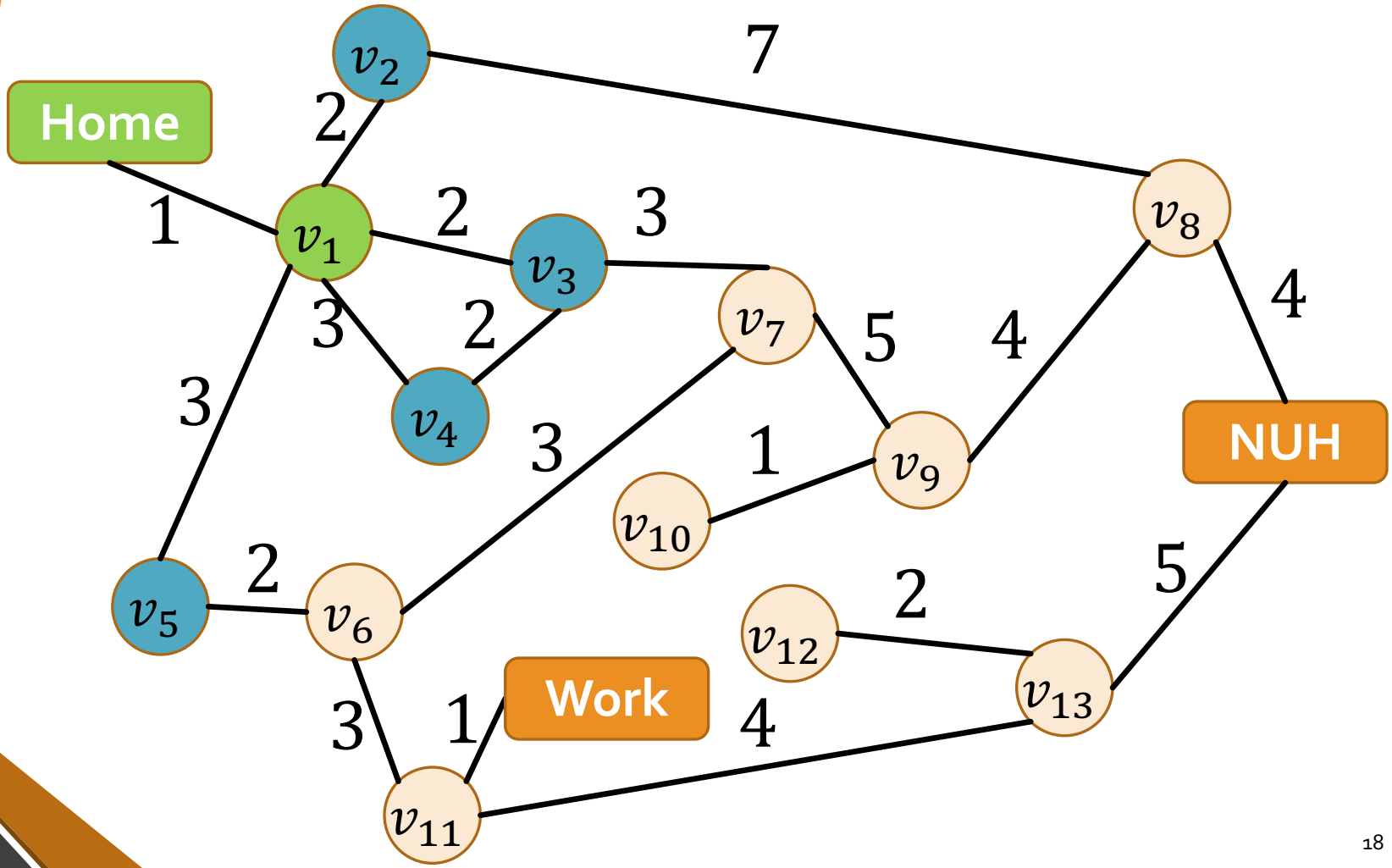

Home
NUH
Work
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
1
2
3
4
5
6
7

$\langle Home, cost = 0, parent = nil \rangle$

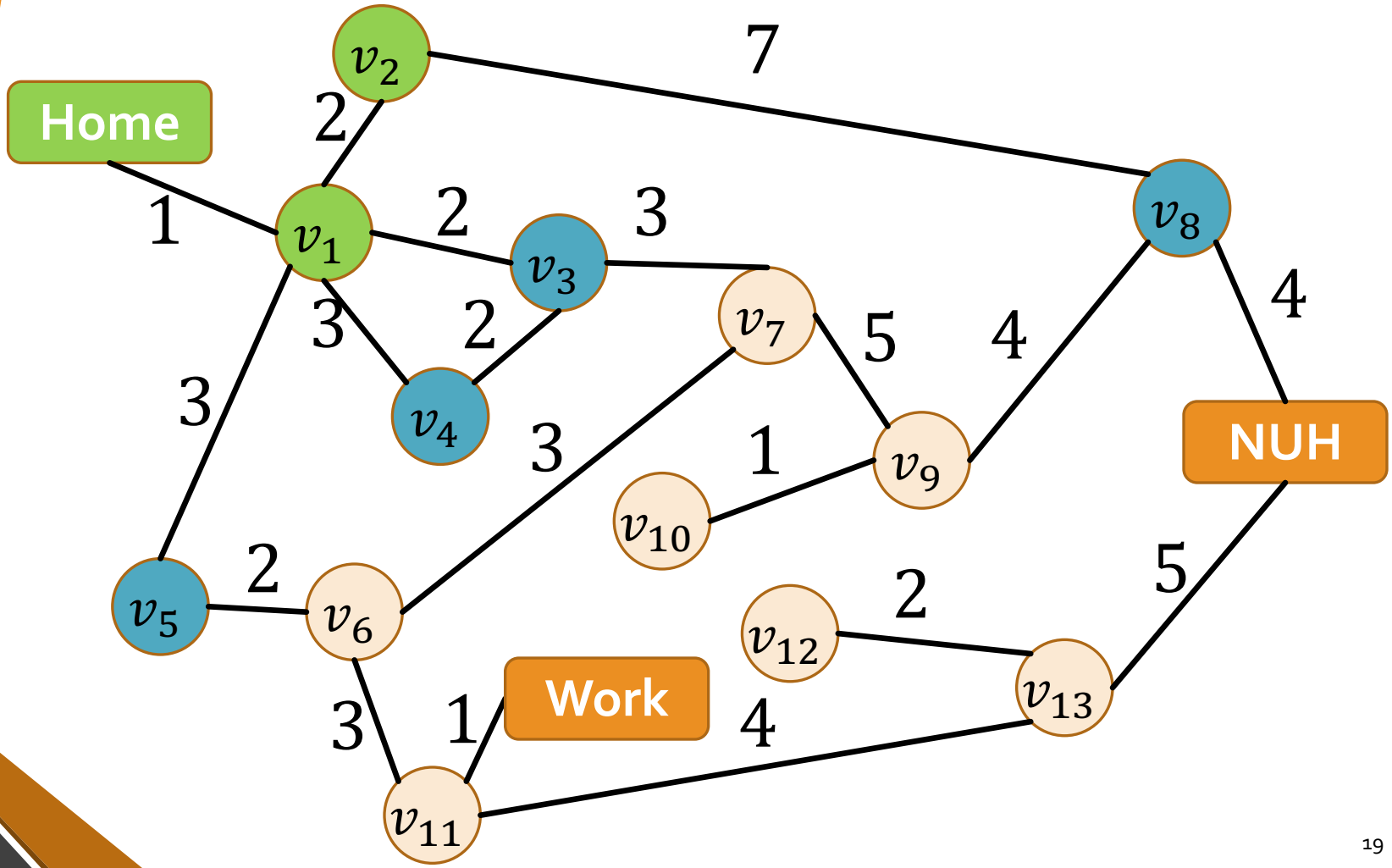


$\langle Home, g = 0, p = nil \rangle$

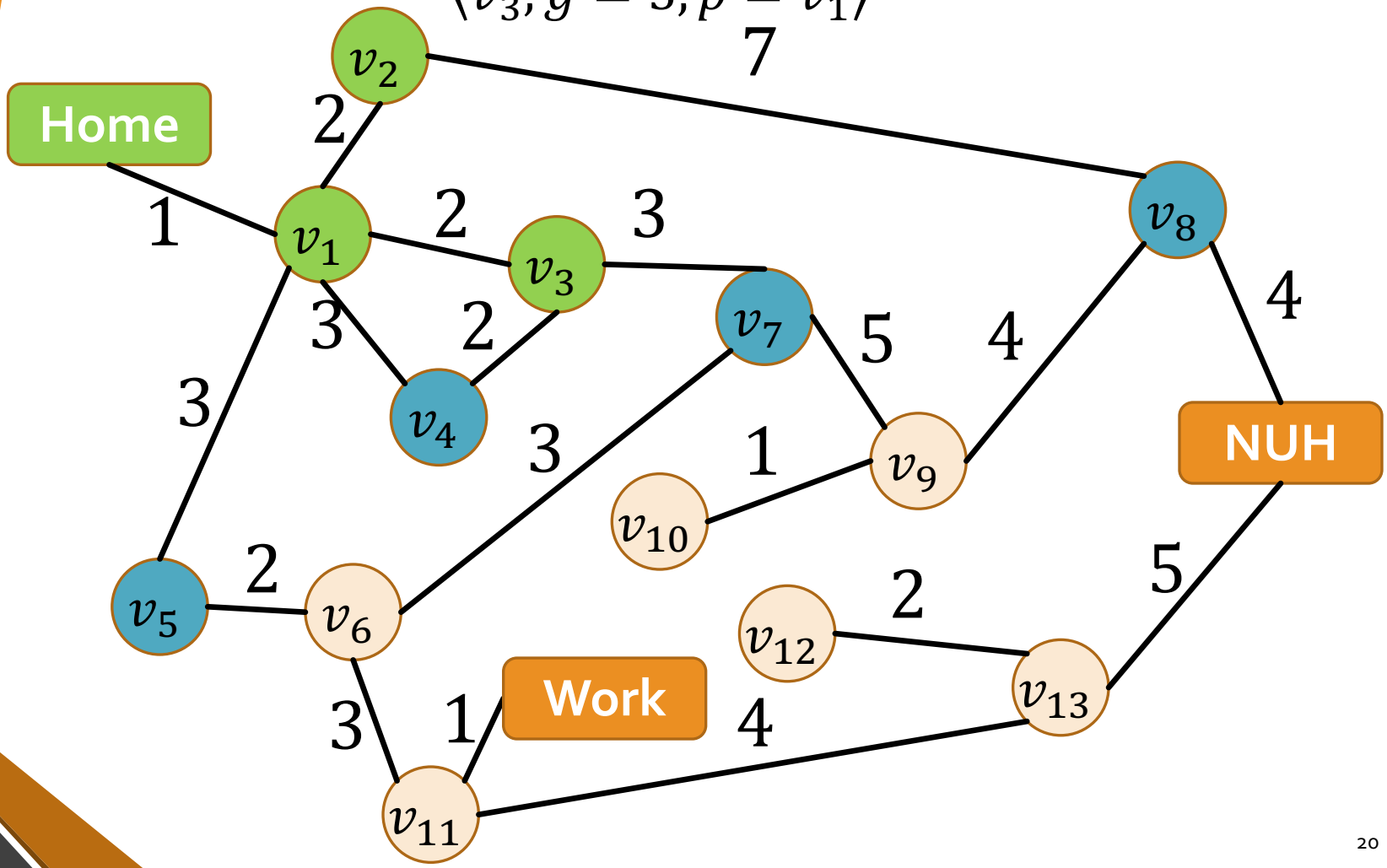
$\langle v_1, g = 1, p = Home \rangle$



$\langle Home, g = 0, p = nil \rangle$
$\langle v_1, g = 1, p = Home \rangle$
$\langle v_2, g = 3, p = v_1 \rangle$



$\langle Home, g = 0, p = nil \rangle$
$\langle v_1, g = 1, p = Home \rangle$
$\langle v_2, g = 3, p = v_1 \rangle$
$\langle v_3, g = 3, p = v_1 \rangle$



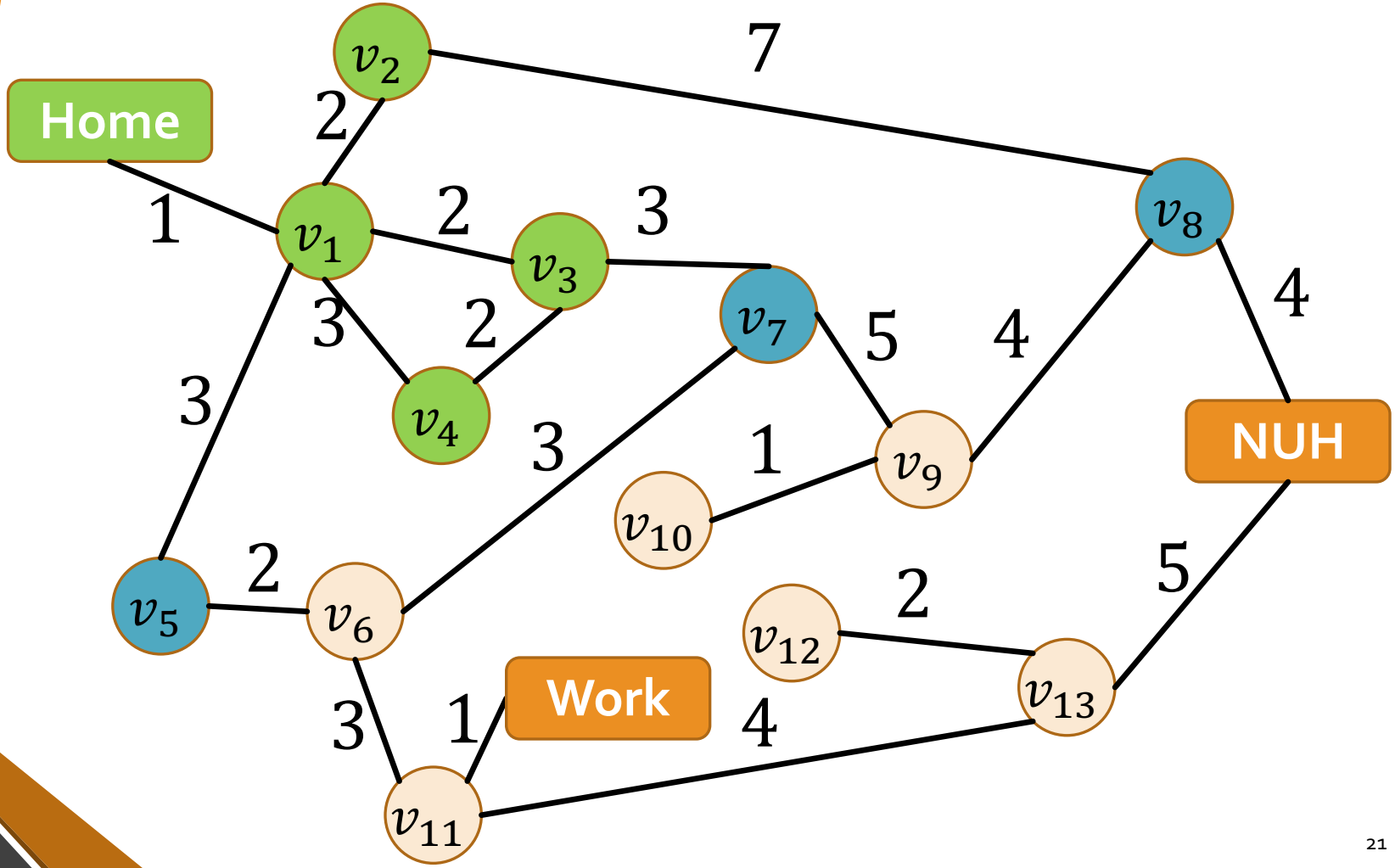
$\langle Home, g = 0, p = nil \rangle$

$\langle v_1, g = 1, p = Home \rangle$

$\langle v_2, g = 3, p = v_1 \rangle$

$\langle v_3, g = 3, p = v_1 \rangle$

$\langle v_4, g = 4, p = v_1 \rangle$



Search Strategies

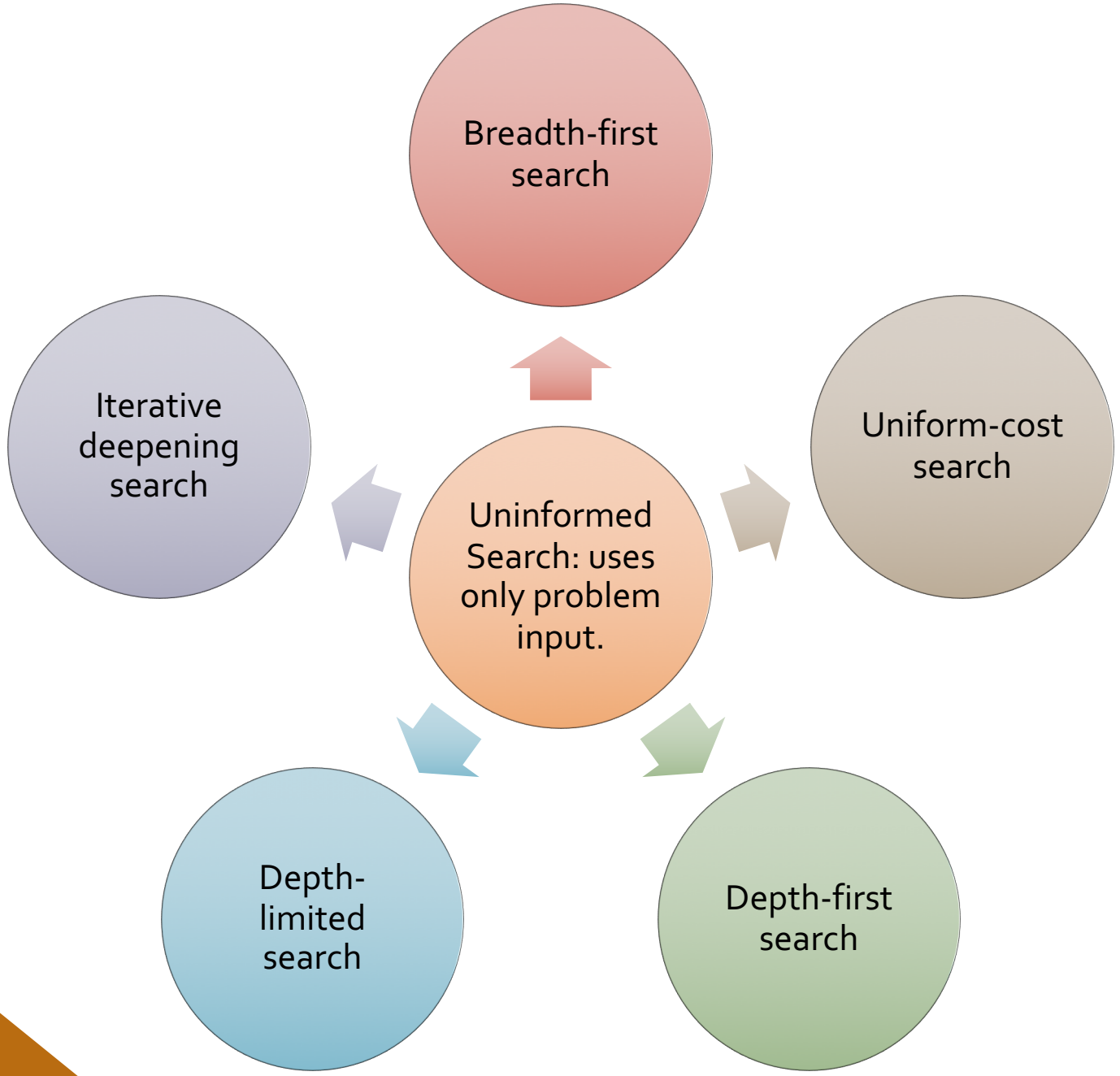
Any deterministic search algorithm is determined by the order of node expansion.

Evaluation Criteria

- completeness: always find a solution if exists
- optimality: find a least-cost solution
- time complexity: number of nodes generated
- space complexity: max. number of nodes in memory

Problem Parameters

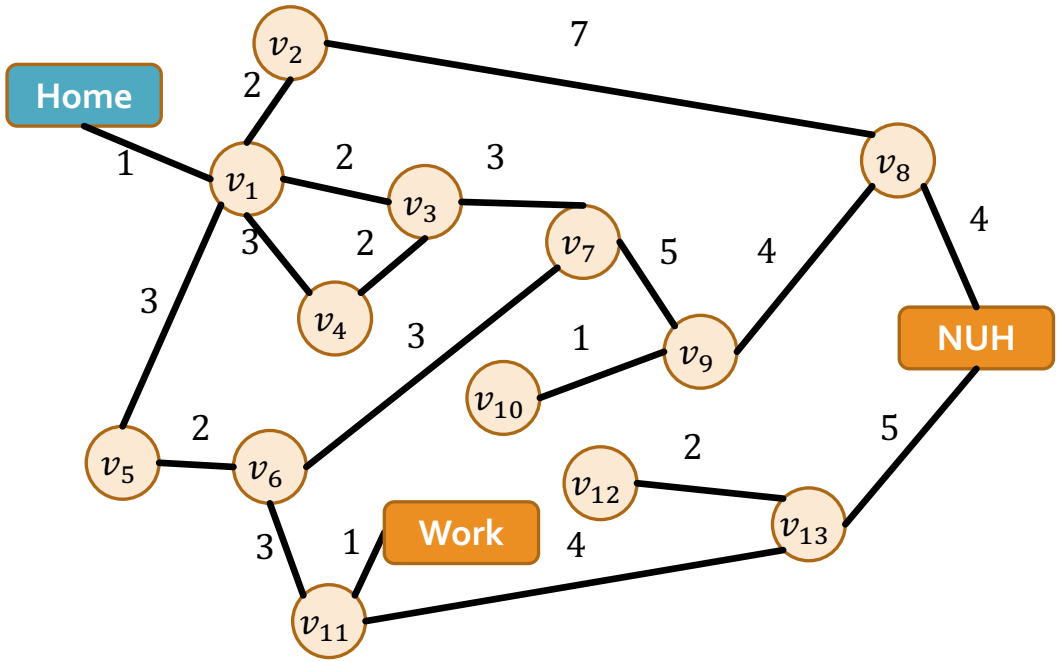
- b: maximum # of successors of any node (may be ∞)
- d: depth of shallowest **goal** node
- m: maximum depth of search tree (may be ∞)


Breadth-first search
Uniform-cost search
Depth-first search
Depth-limited search
Iterative deepening search
Uninformed Search: uses only problem input.

Breadth-First Search (BFS)

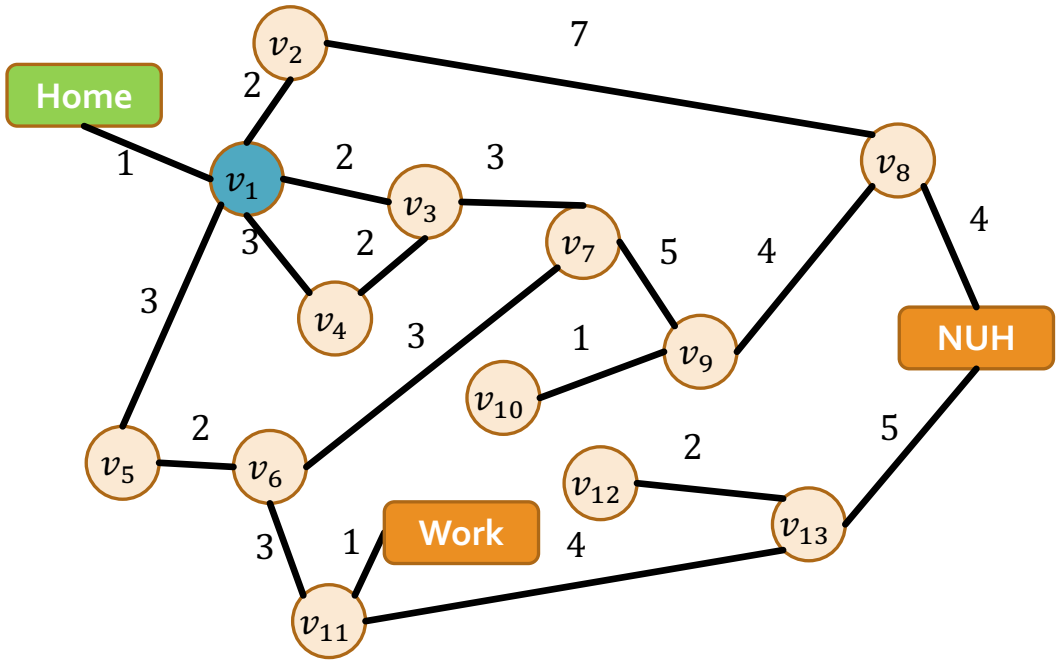
- Idea: Expand shallowest unexpanded node
- Implementation: Frontier is a FIFO queue, i.e., insert new successors at the end

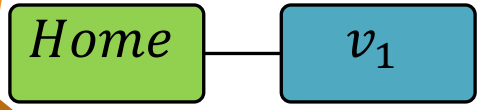
$[Home]$


Home
v_2
7
2
1
v_1
2
3
v_8
v_3
v_7
4
3
2
5
4
3
v_4
3
1
NUH
v_9
v_{10}
2
5
v_5
v_6
2
v_{12}
Work
v_{13}
3
1
4
v_{11}

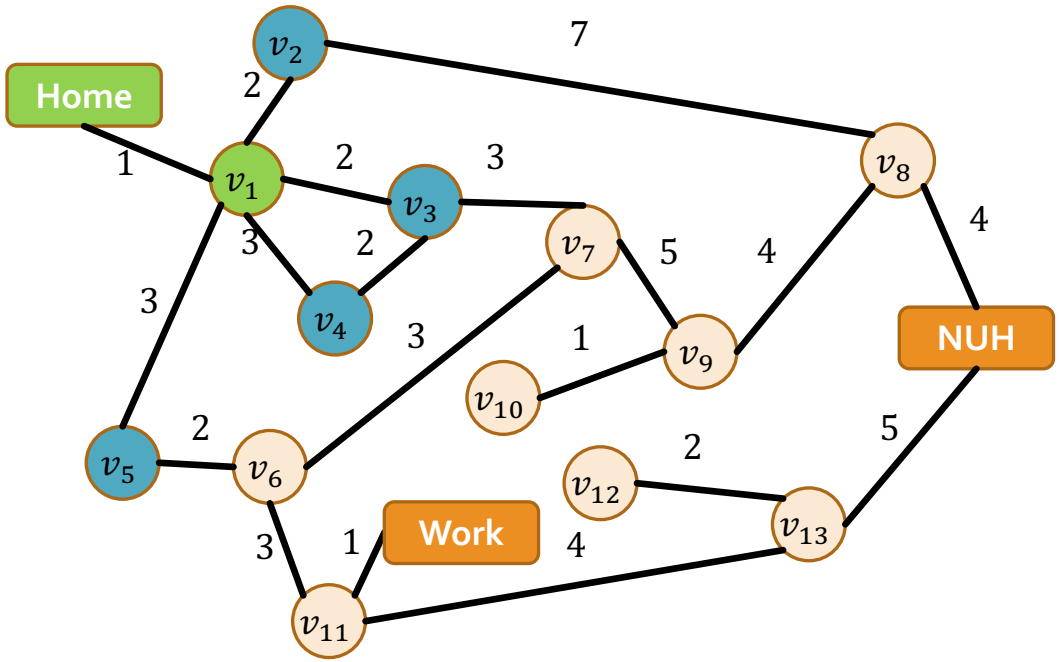
Home

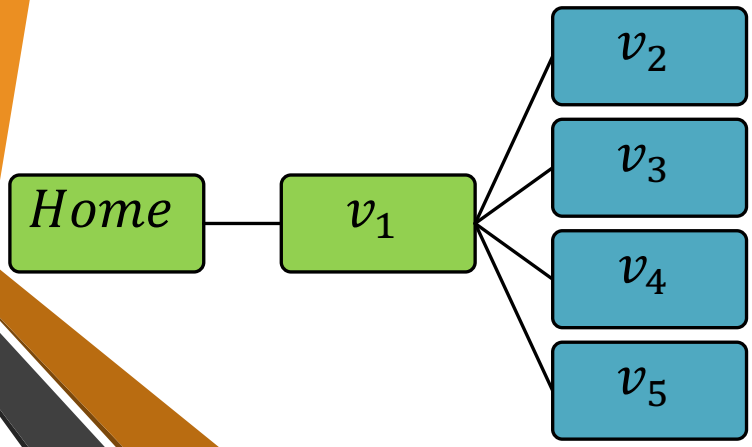
$[v_1]$


Home
v_2
7
2
1
v_1
2
3
v_8
4
3
2
v_7
5
4
v_3
3
v_4
3
1
v_9
NUH
v_{10}
2
5
v_5
v_6
2
v_{12}
v_{13}
3
1
Work
4
v_{11}

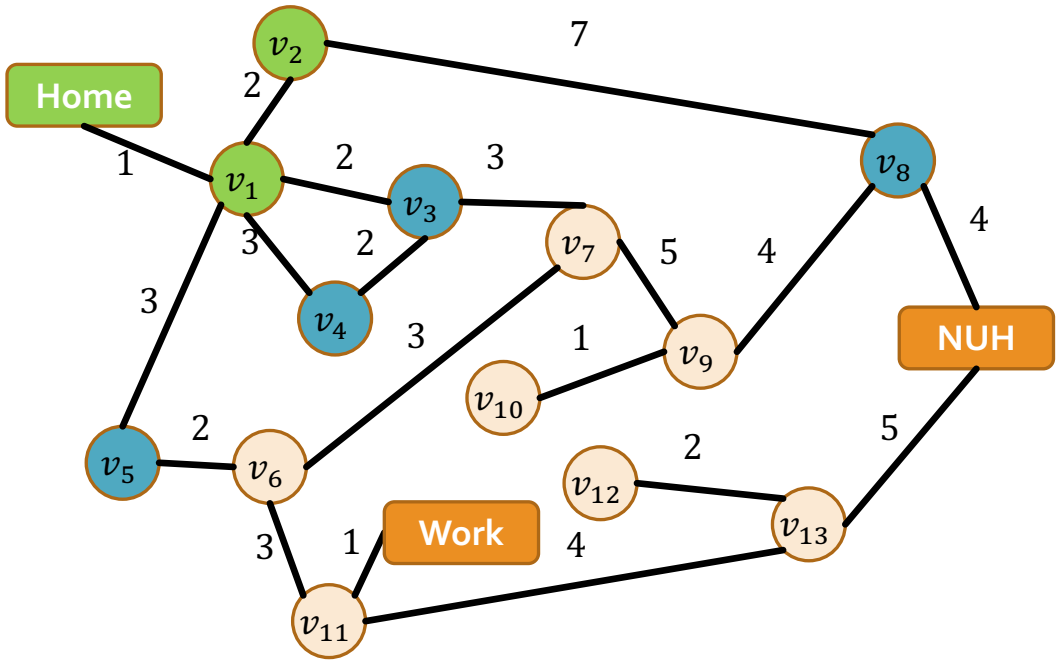

Home
v_1

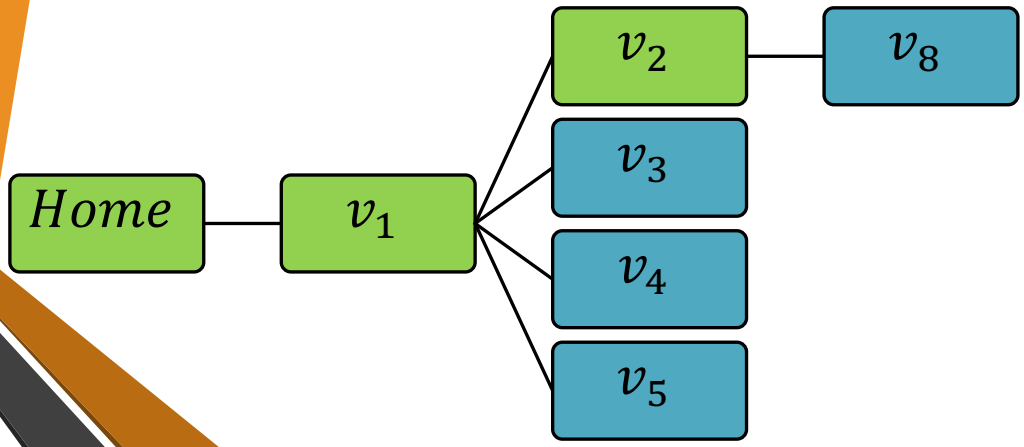
$[v_2, v_3, v_4, v_5]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
7
2
1
2
3
3
2
3
3
1
5
4
4
2
5
2
3
1
4

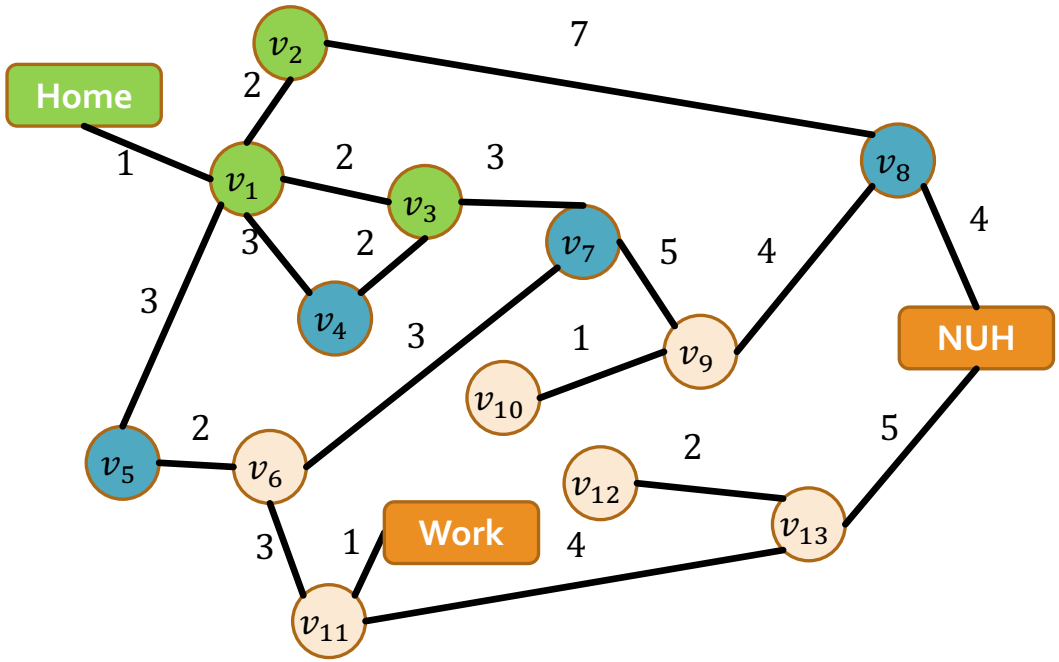

Home
v_1
v_2
v_3
v_4
v_5

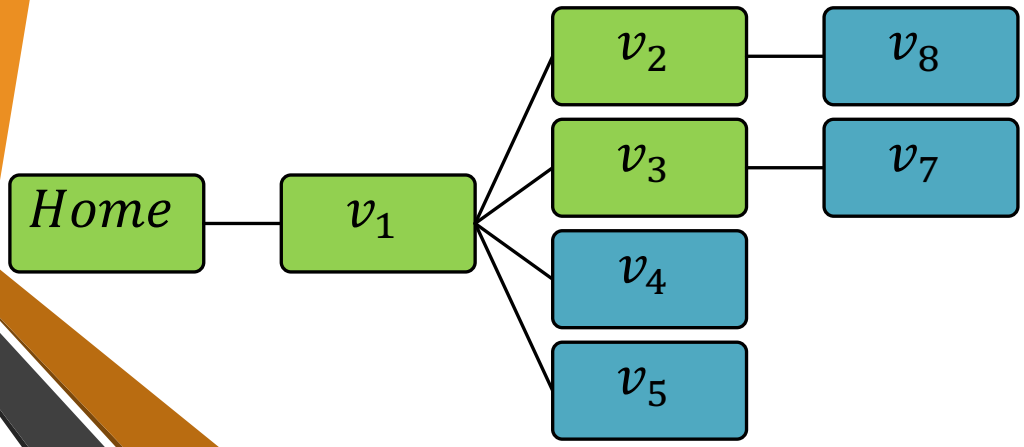
$[v_3, v_4, v_5, v_8]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7

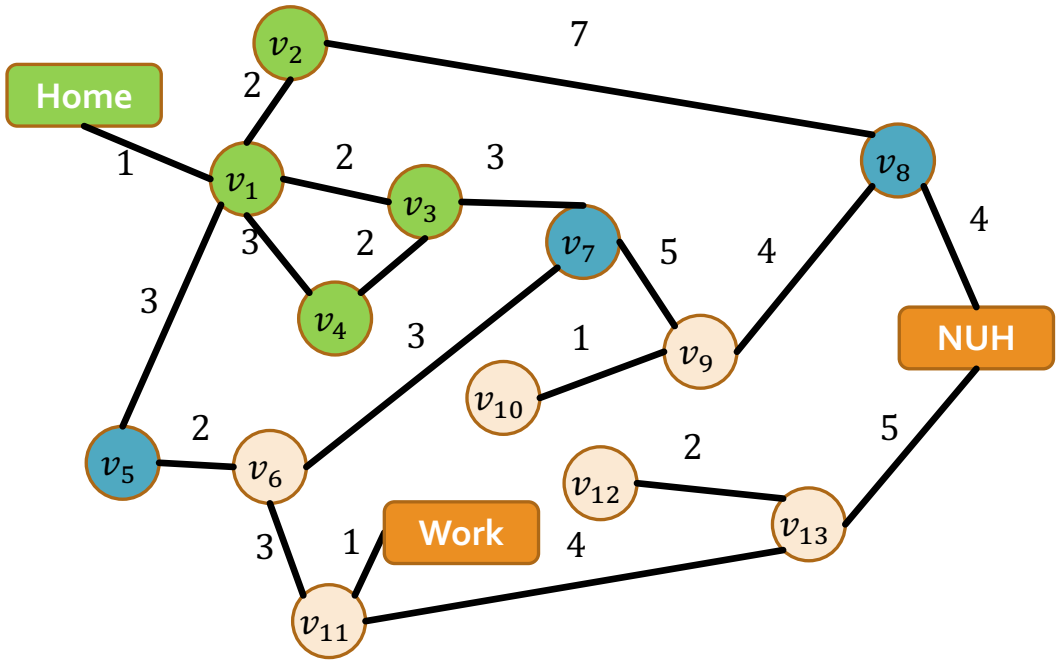

Home
v_1
v_2
v_3
v_4
v_5
v_8

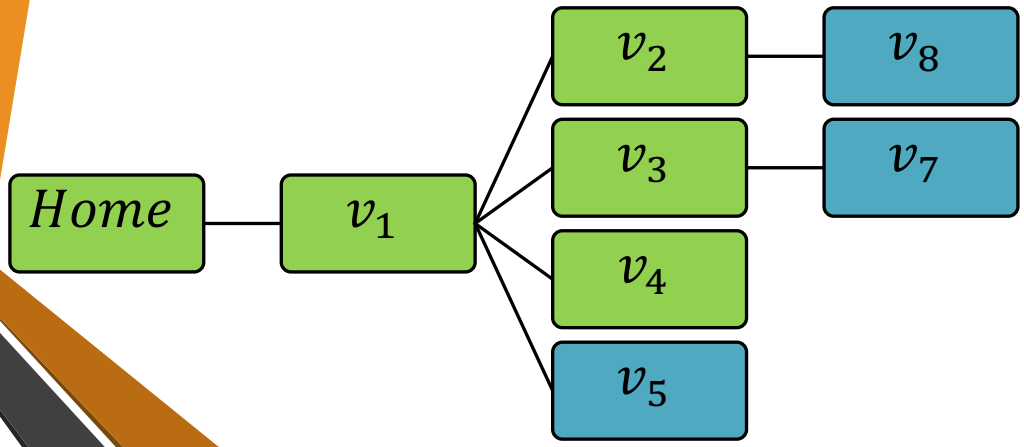
$[v_4, v_5, v_8, v_7]$


Home
v_2
v_1
v_3
v_8
v_7
v_4
v_9
NUH
v_{10}
v_5
v_6
v_{12}
v_{13}
Work
v_{11}
7
2
1
2
3
3
2
5
4
4
3
3
1
2
5
2
3
1
4

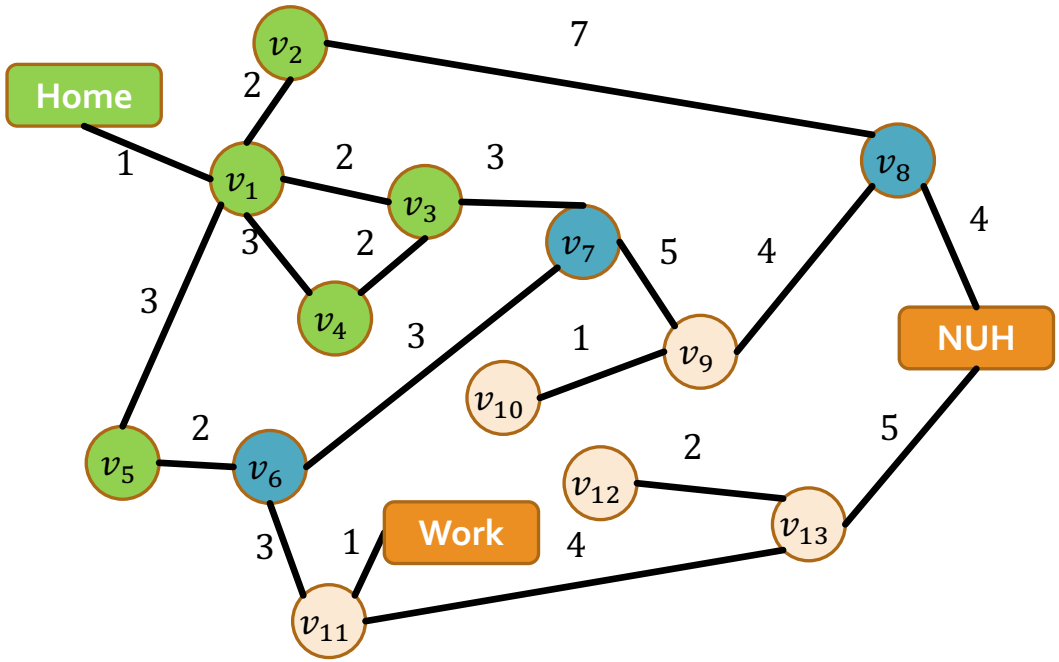

Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7

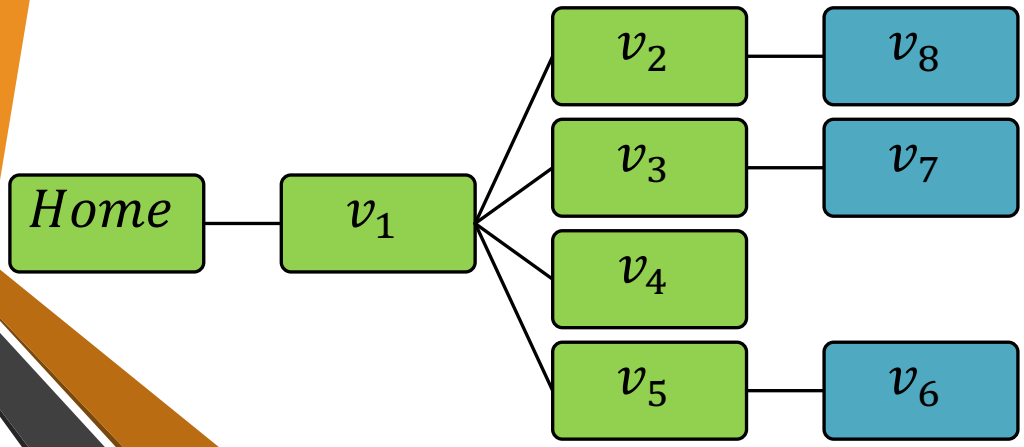
$[v_5, v_8, v_7]$


Home
v_2
v_1
v_3
v_8
v_7
v_4
v_9
NUH
v_{10}
v_5
v_6
v_{12}
v_{13}
Work
v_{11}
7
2
1
2
3
3
2
5
4
4
3
3
1
2
5
2
3
1
4

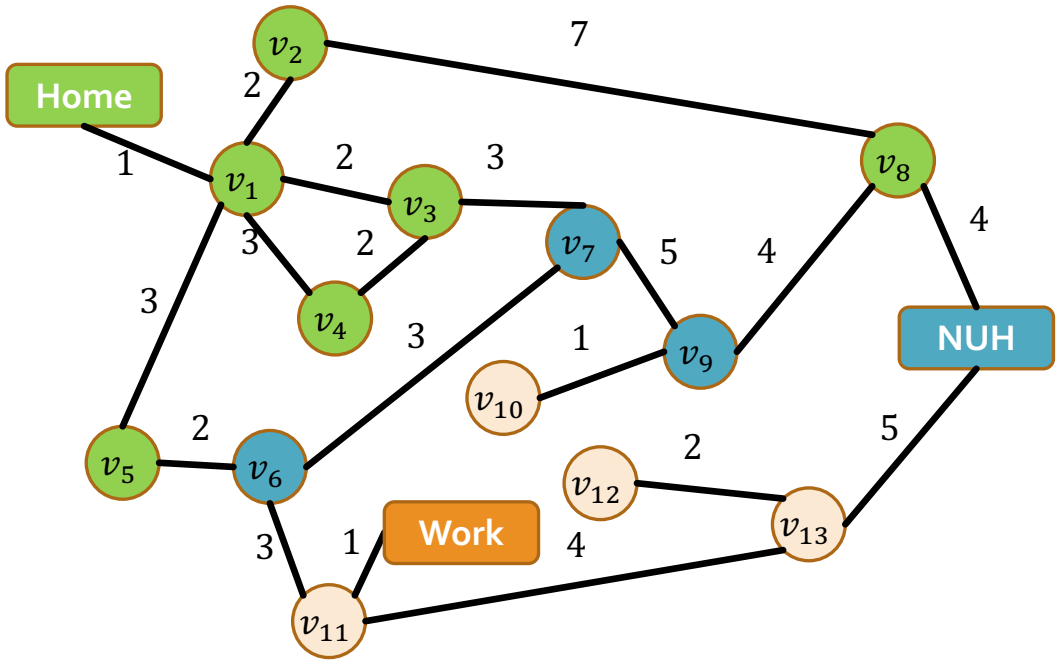

Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7

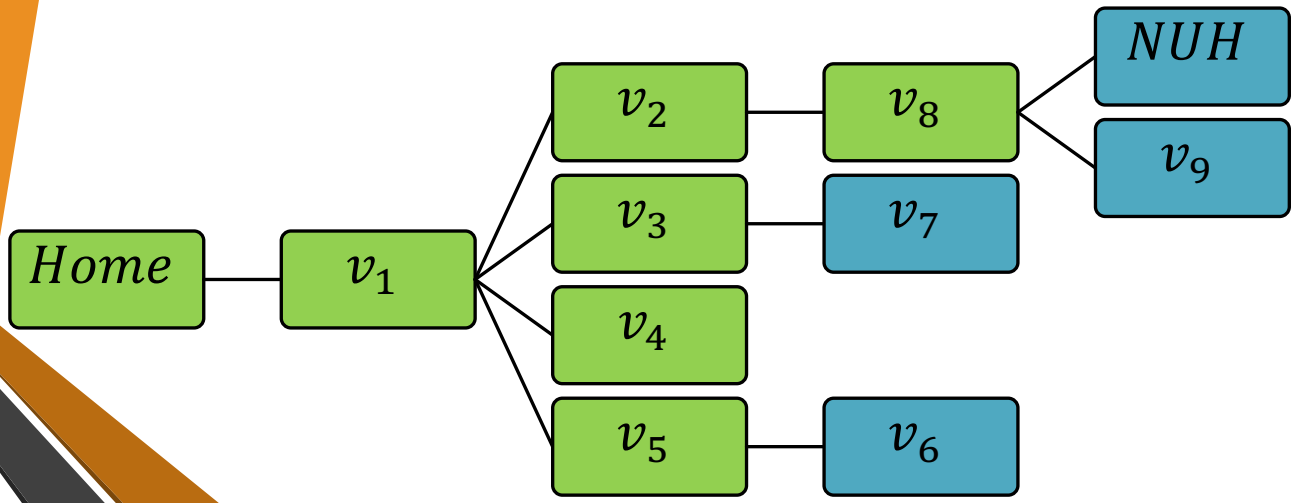
$[v_8, v_7, v_6]$


Home
v_2
v_1
v_3
v_8
v_7
v_4
v_9
NUH
v_{10}
v_5
v_6
v_{12}
v_{13}
Work
v_{11}
7
2
1
2
3
3
2
5
4
4
3
3
1
2
5
2
3
1
4

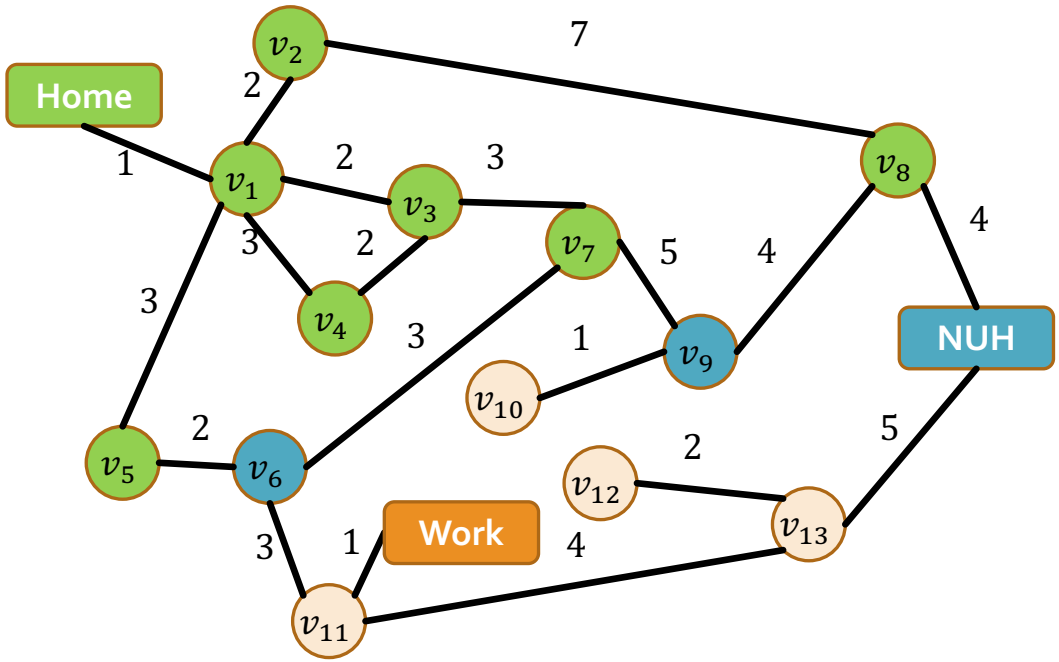

Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7
v_6

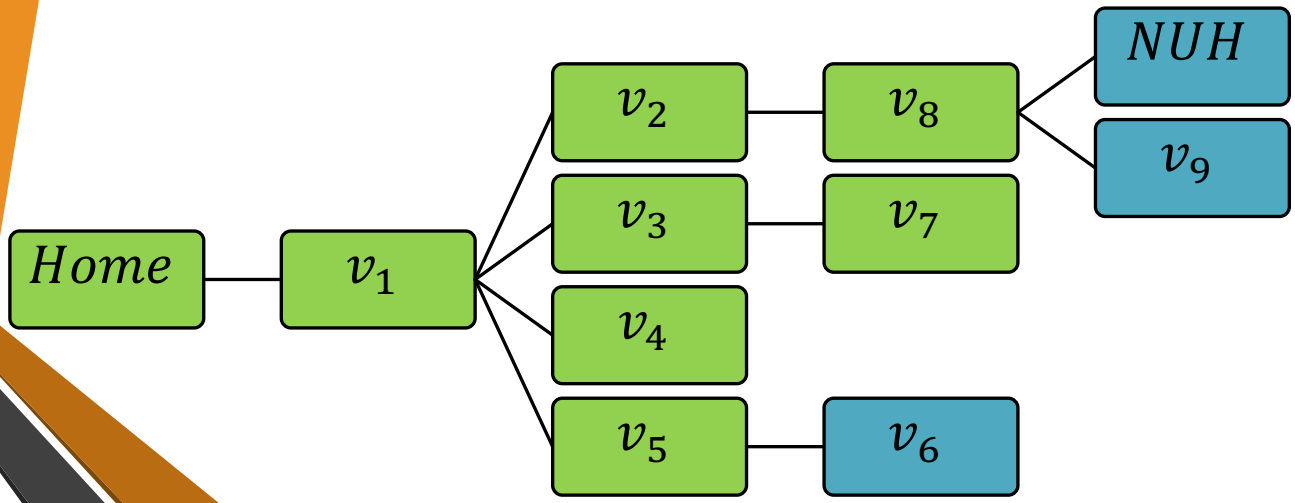
$[v_7, v_6, NUH, v_9]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7

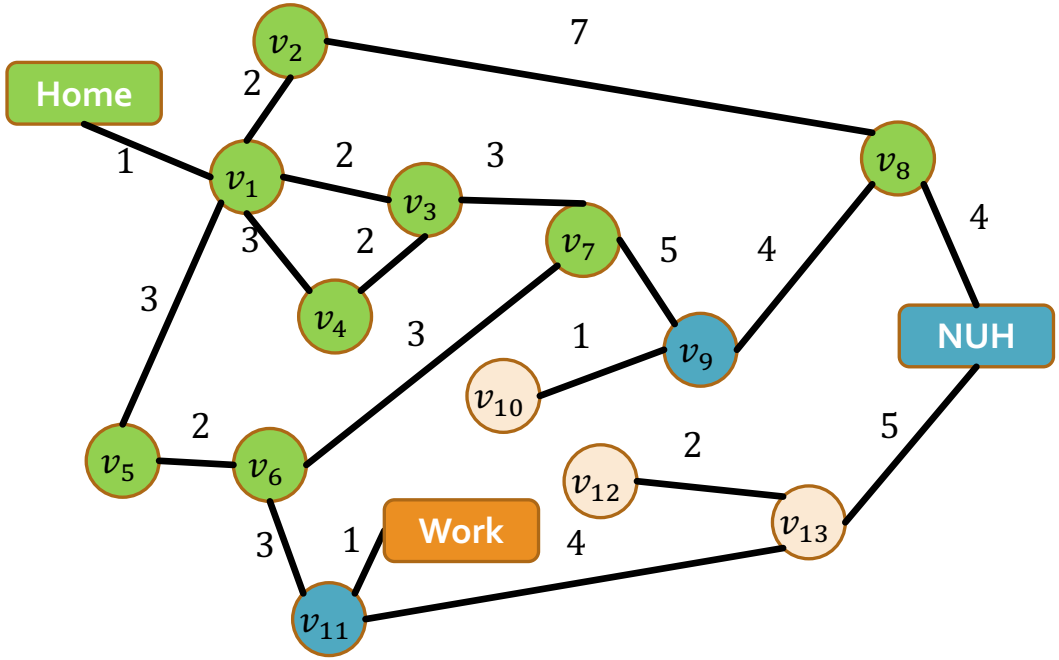

$Home$
v_1
v_2
v_3
v_4
v_5
v_8
v_7
v_6
NUH
v_9

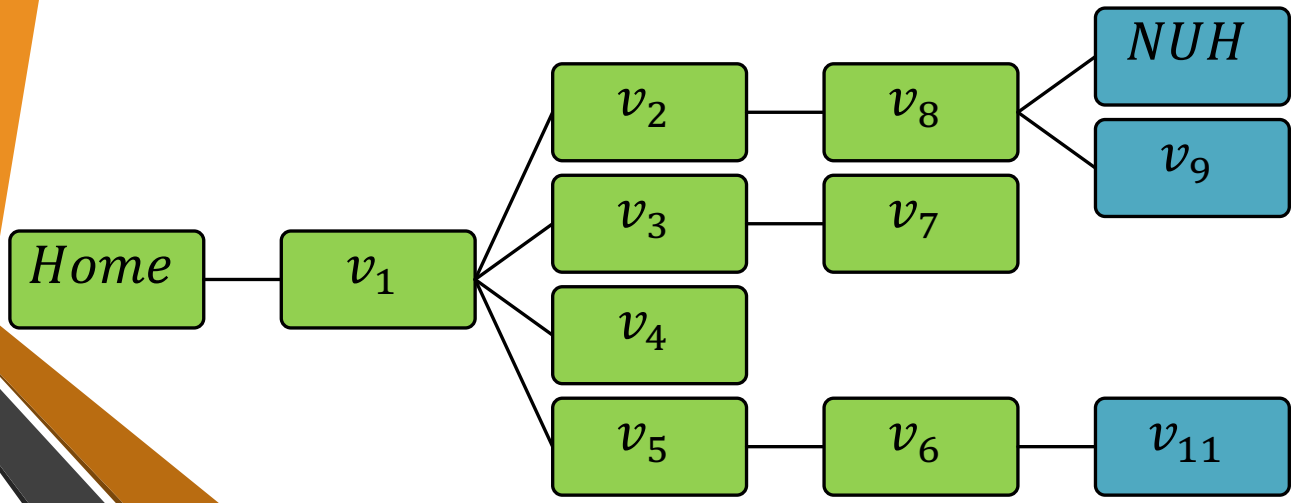
$[v_6, NUH, v_9]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7

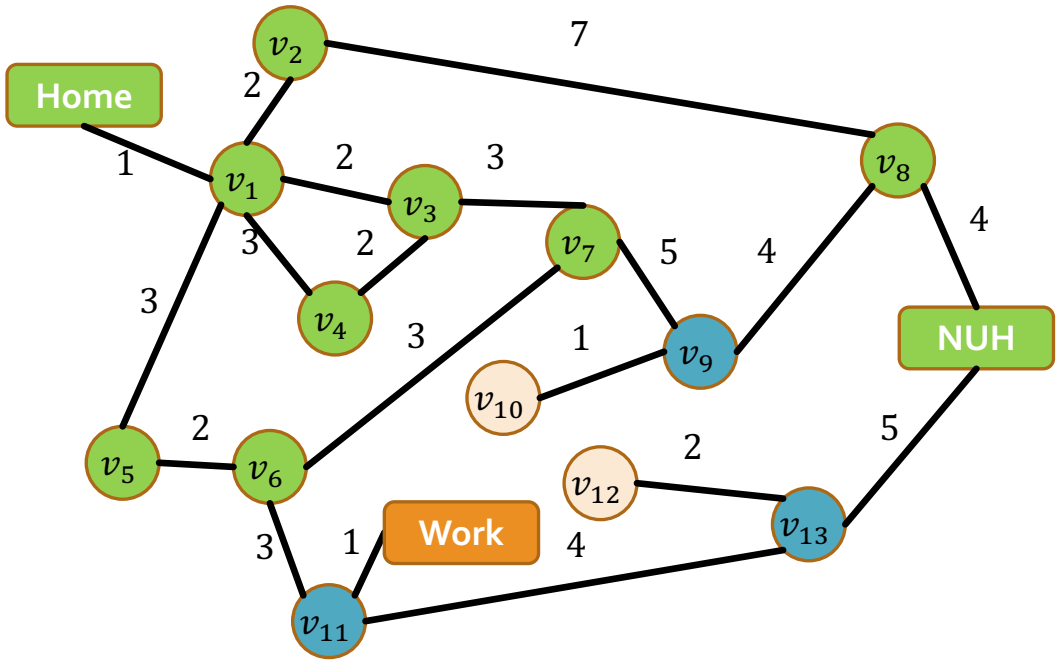

Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7
v_6
NUH
v_9

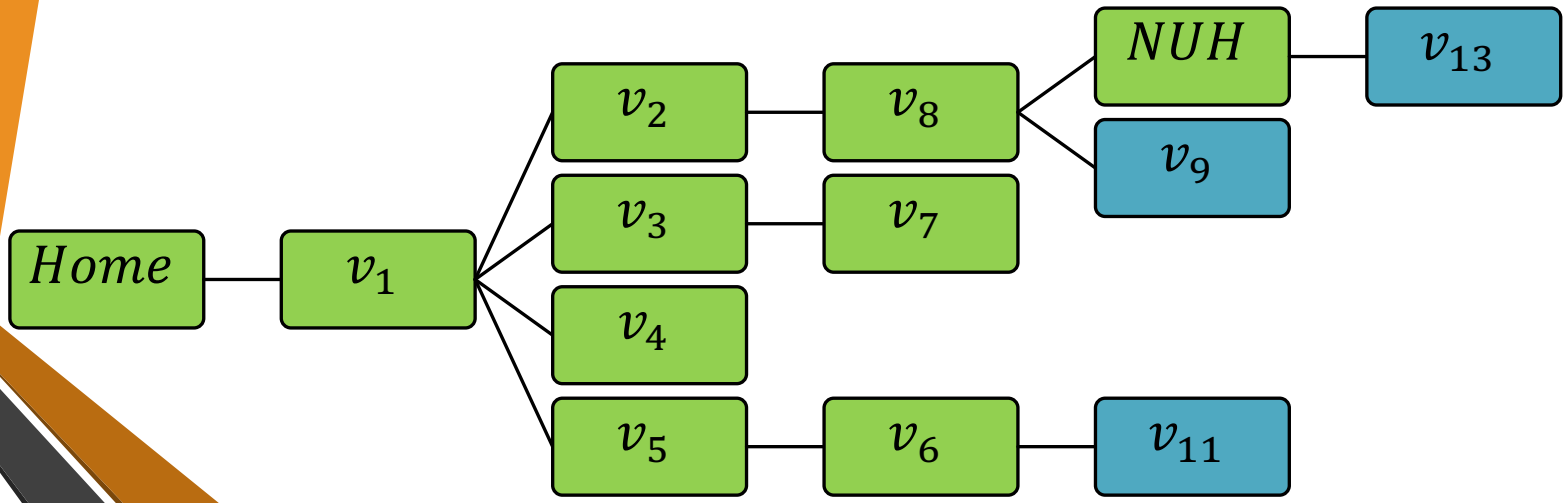
$[NUH, v_9, v_{11}]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7

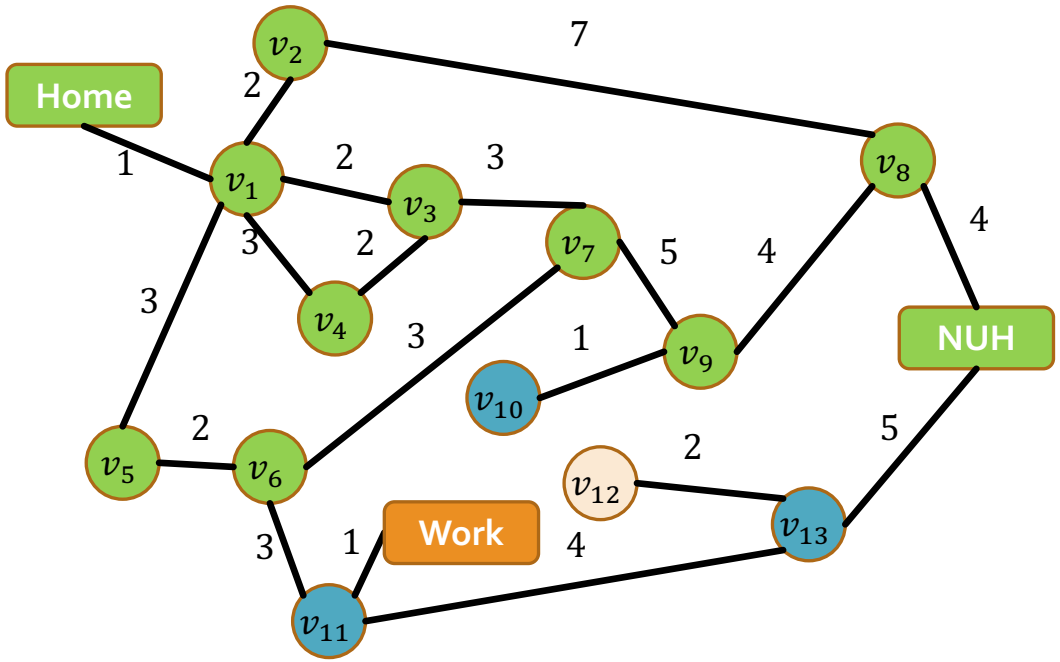

Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{11}
NUH

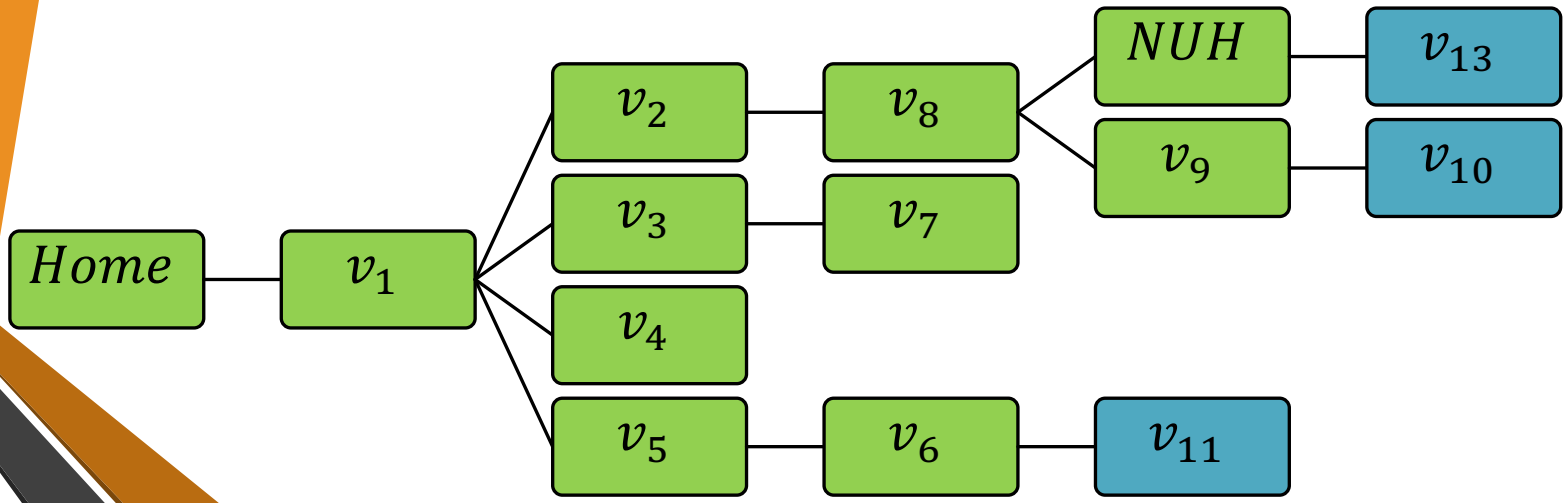
$[v_9, v_{11}, v_{13}]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7

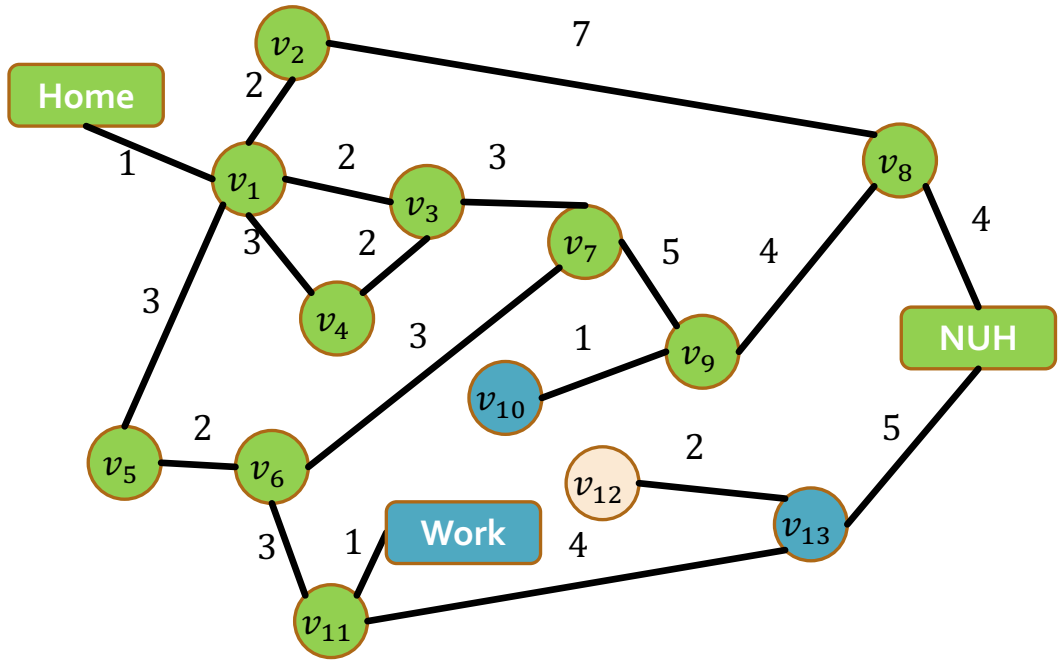

Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7
v_6
NUH
v_9
v_{13}
v_{11}

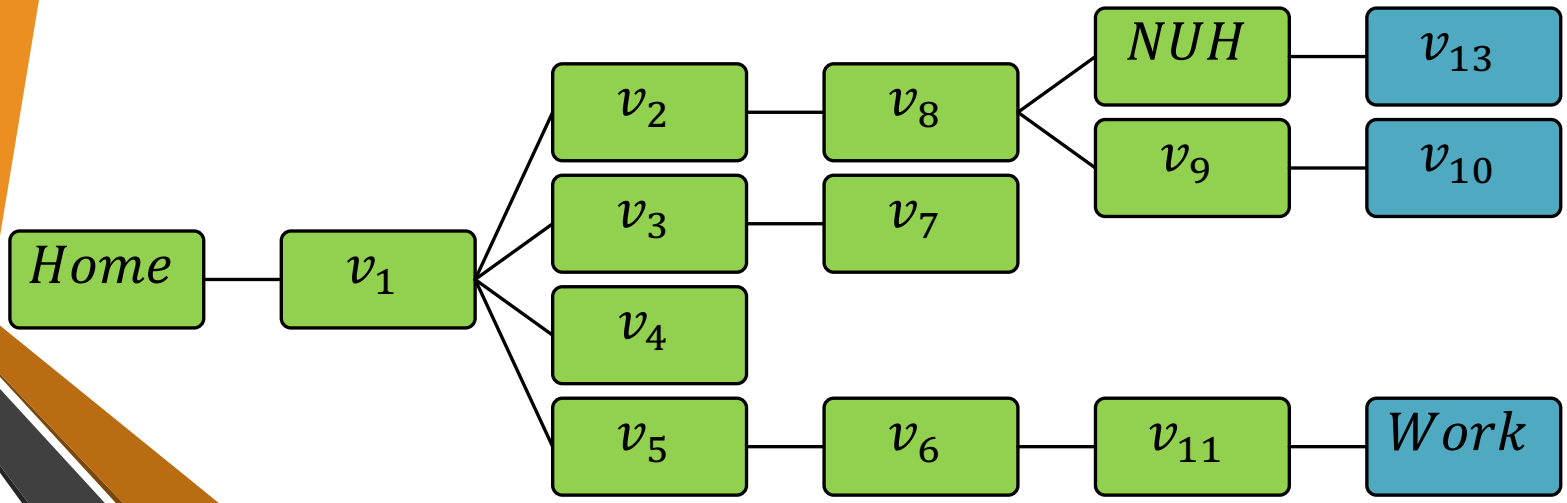
$[v_{11}, v_{13}, v_{10}]$


Home
NUH
Work
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
1
2
3
4
5
7

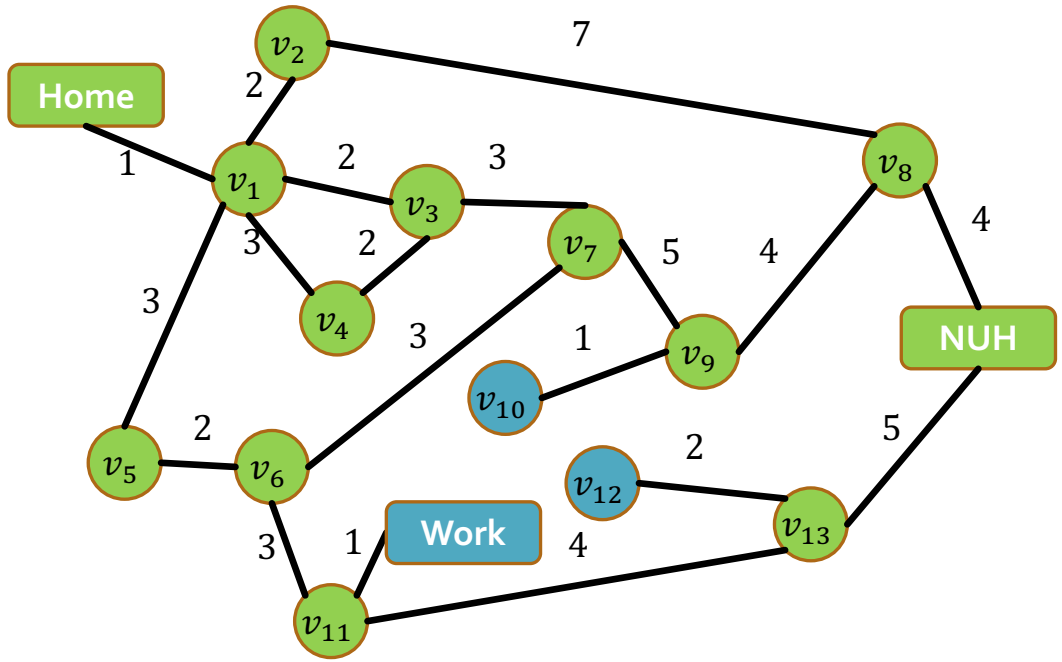

Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
NUH
v_{10}
v_{11}
v_{13}

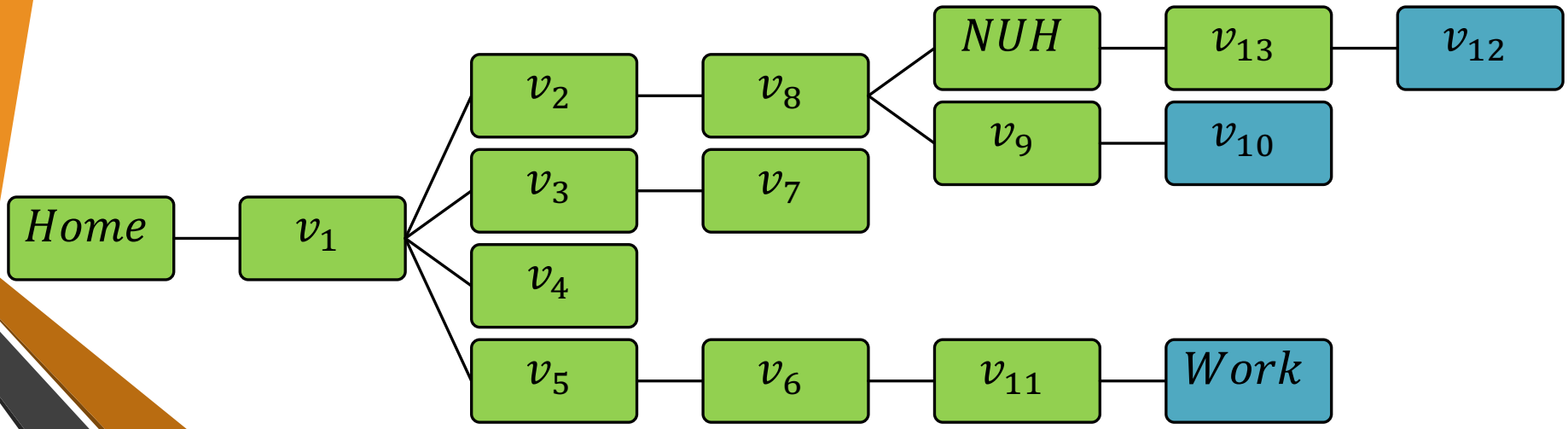
$[v_{13}, v_{10}, Work]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
7
2
1
2
3
3
2
3
3
1
4
5
2
2
5
3
1
4

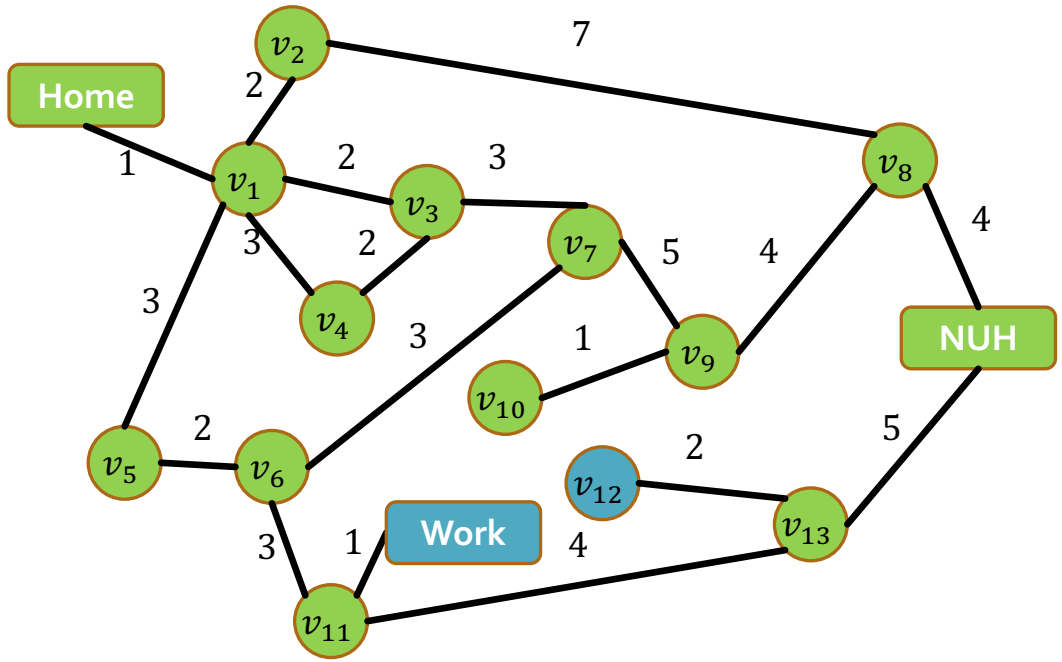

$Home$
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{13}
NUH
$Work$

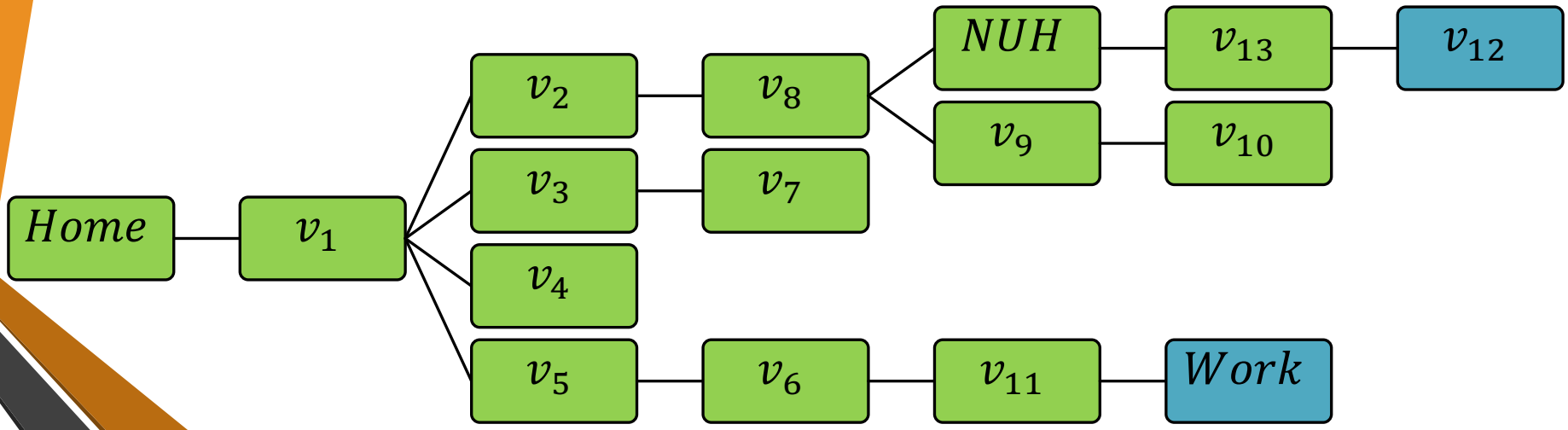
$[v_{10}, Work, v_{12}]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7

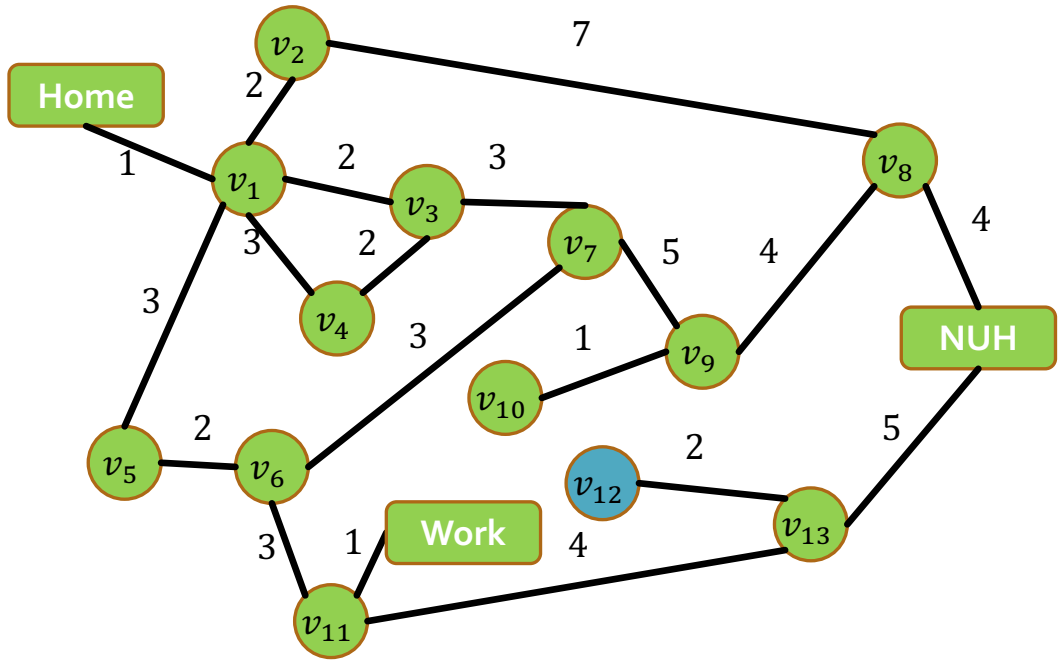

Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7
v_6
NUH
v_9
v_{13}
v_{10}
v_{12}
v_{11}
Work

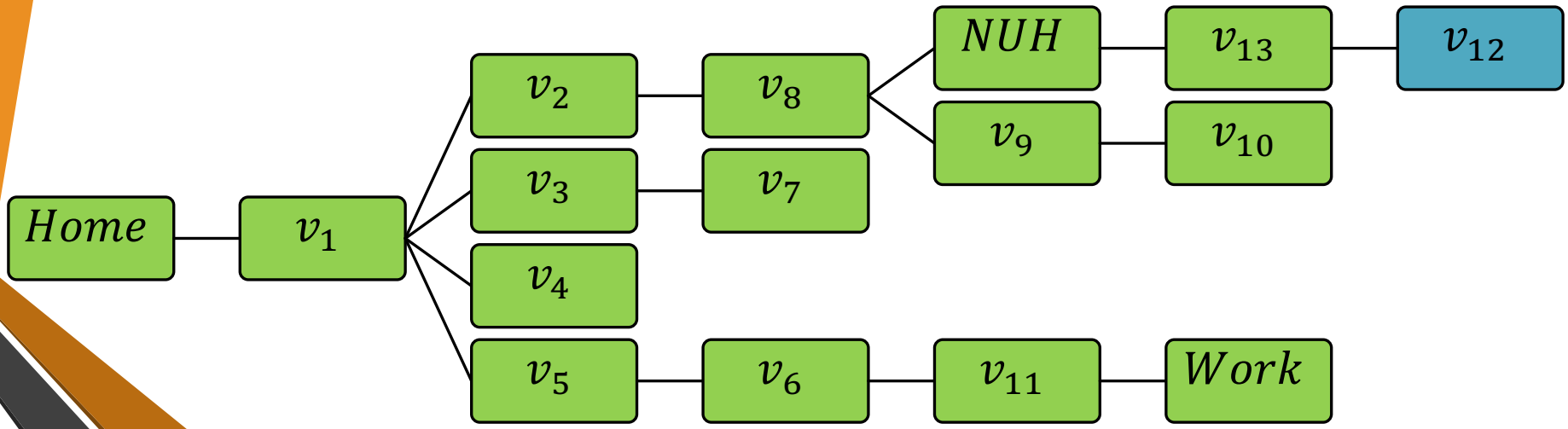
$[Work, v_{12}]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
7
2
1
2
3
3
2
3
5
4
4
3
1
2
2
5
3
1
4


Home
v_1
v_2
v_3
v_4
v_5
v_8
v_7
v_6
NUH
v_9
v_{13}
v_{10}
v_{12}
v_{11}
Work

$[v_{12}]$


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work
1
2
3
4
5
7


Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
NUH
Work

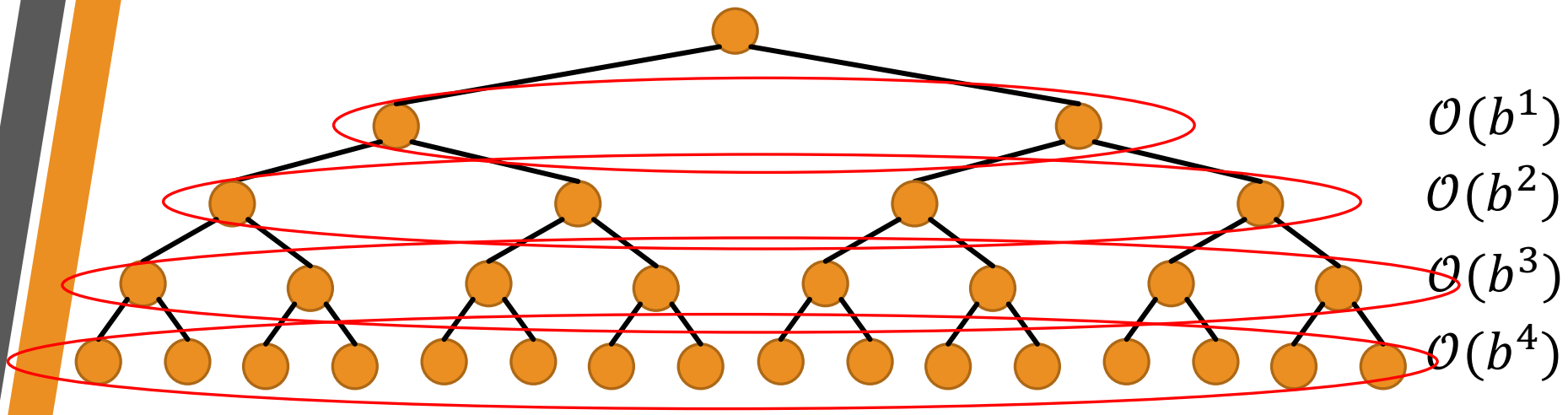
Properties of BFS

Property	
Complete?	Yes (if b is finite)
Optimal	No (unless step cost $= 1$)
Time	$\mathcal{O}(b) + \mathcal{O}(b^2) + \cdots + \mathcal{O}(b^d) = \mathcal{O}(b^d)$
Space	Max size of frontier $\mathcal{O}(b^d)$

Space is the bigger problem (more than time)

Properties of BFS

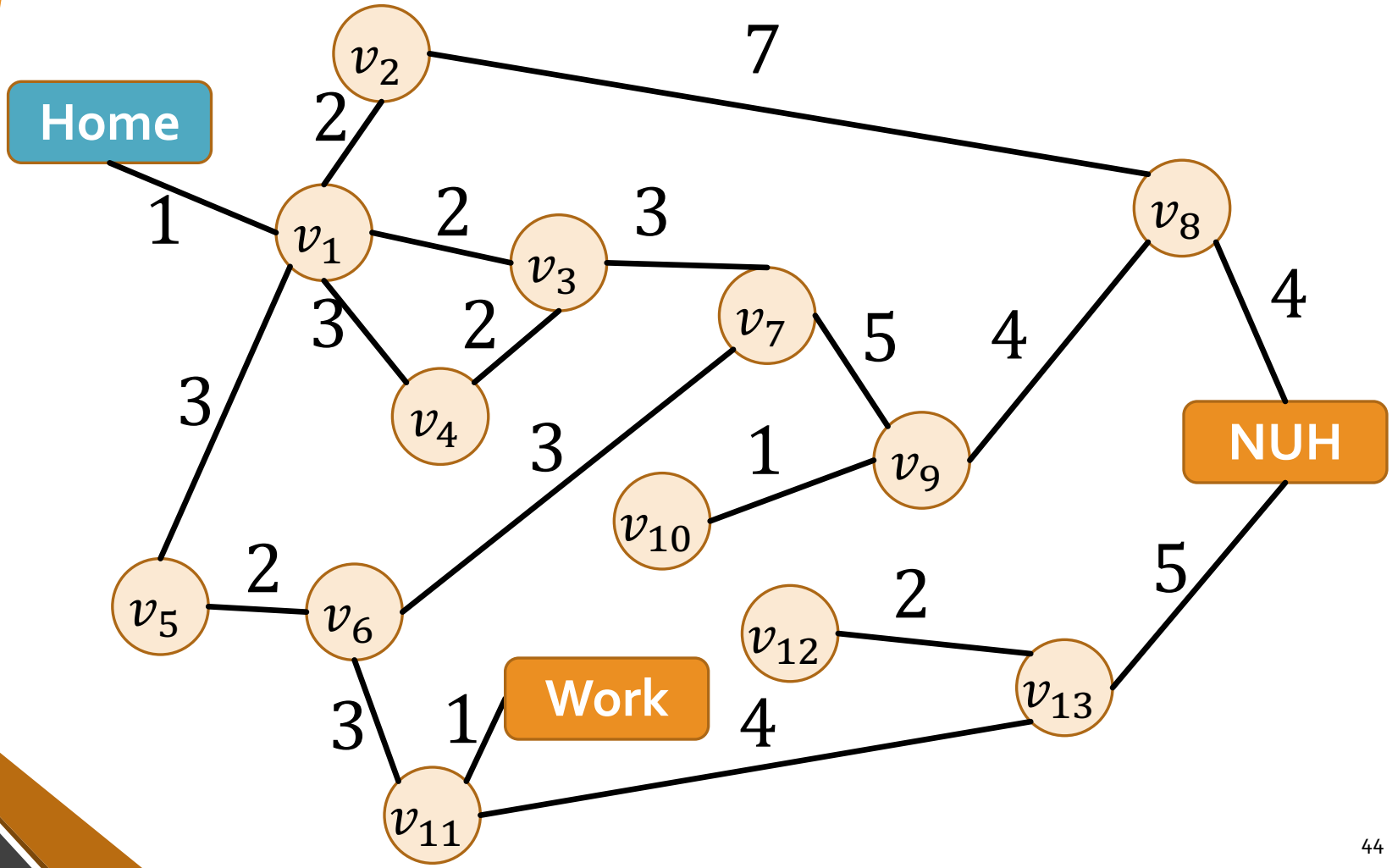
$b = 2$



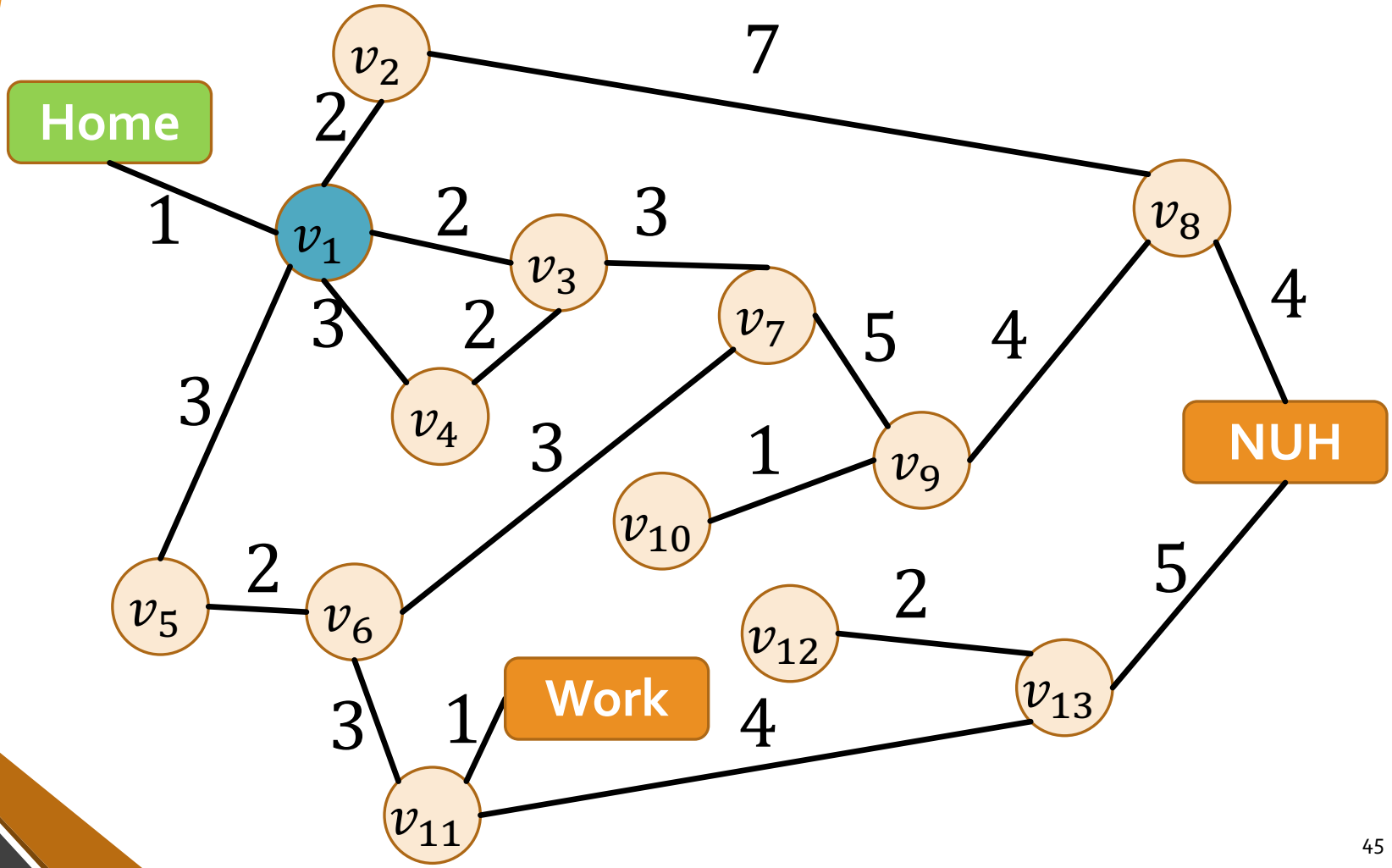
Uniform-Cost Search (UCS)

- Idea: expand least-path-cost unexpanded node
- Frontier: priority queue ordered by path cost g
- Equivalent to BFS if all step costs are equal

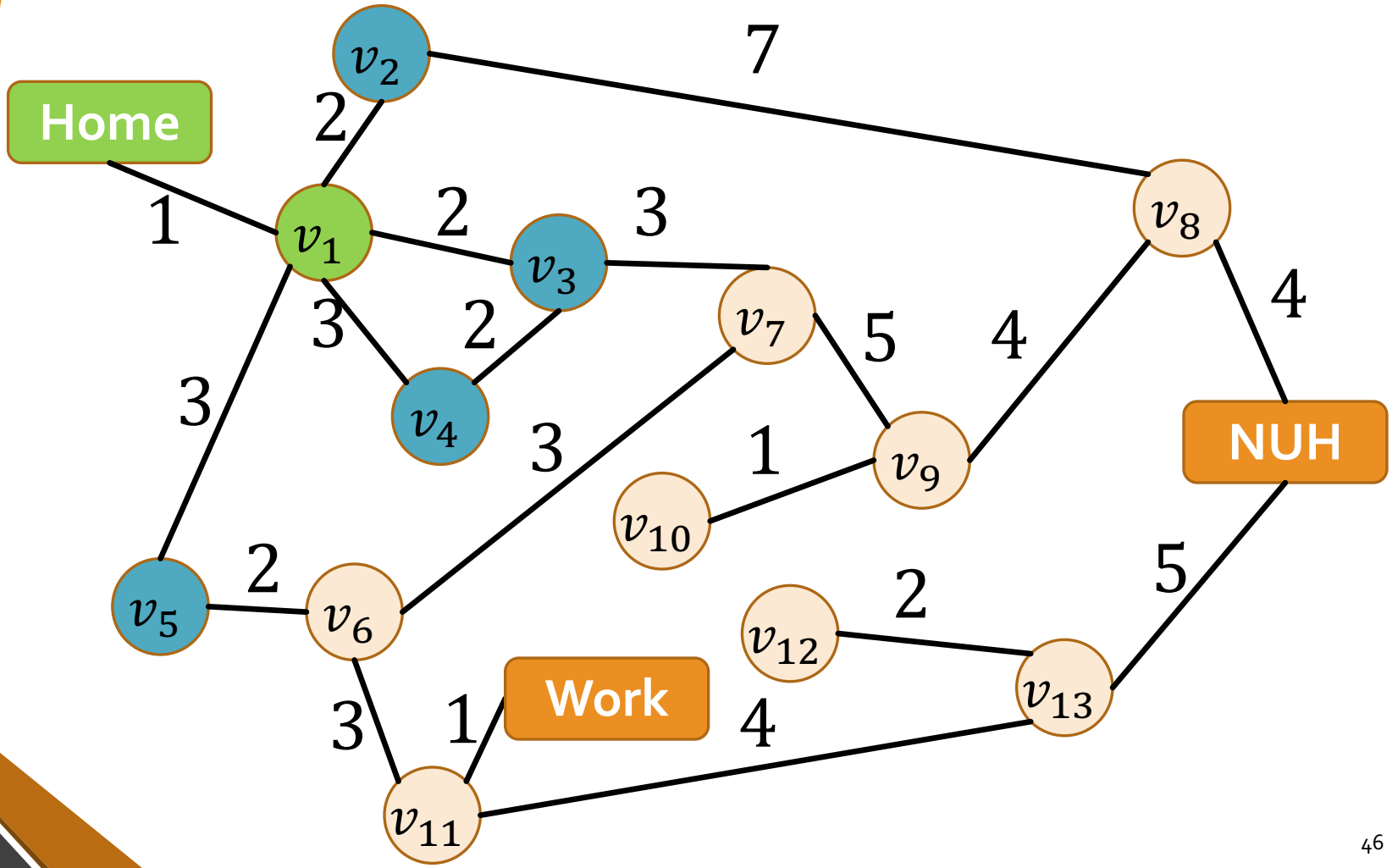
$[\langle Home, 0 \rangle]$



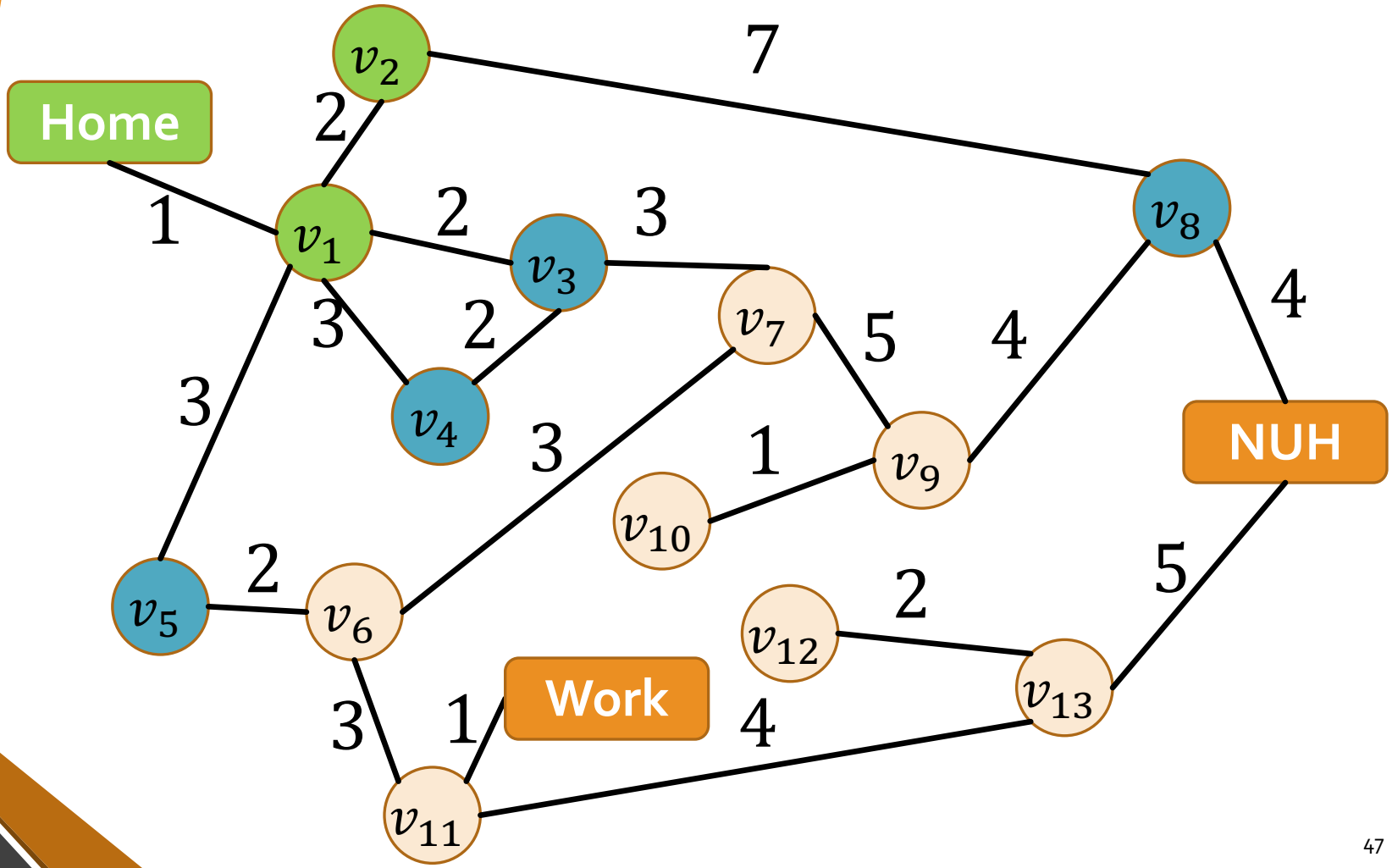
$[\langle v_1, 1 \rangle]$



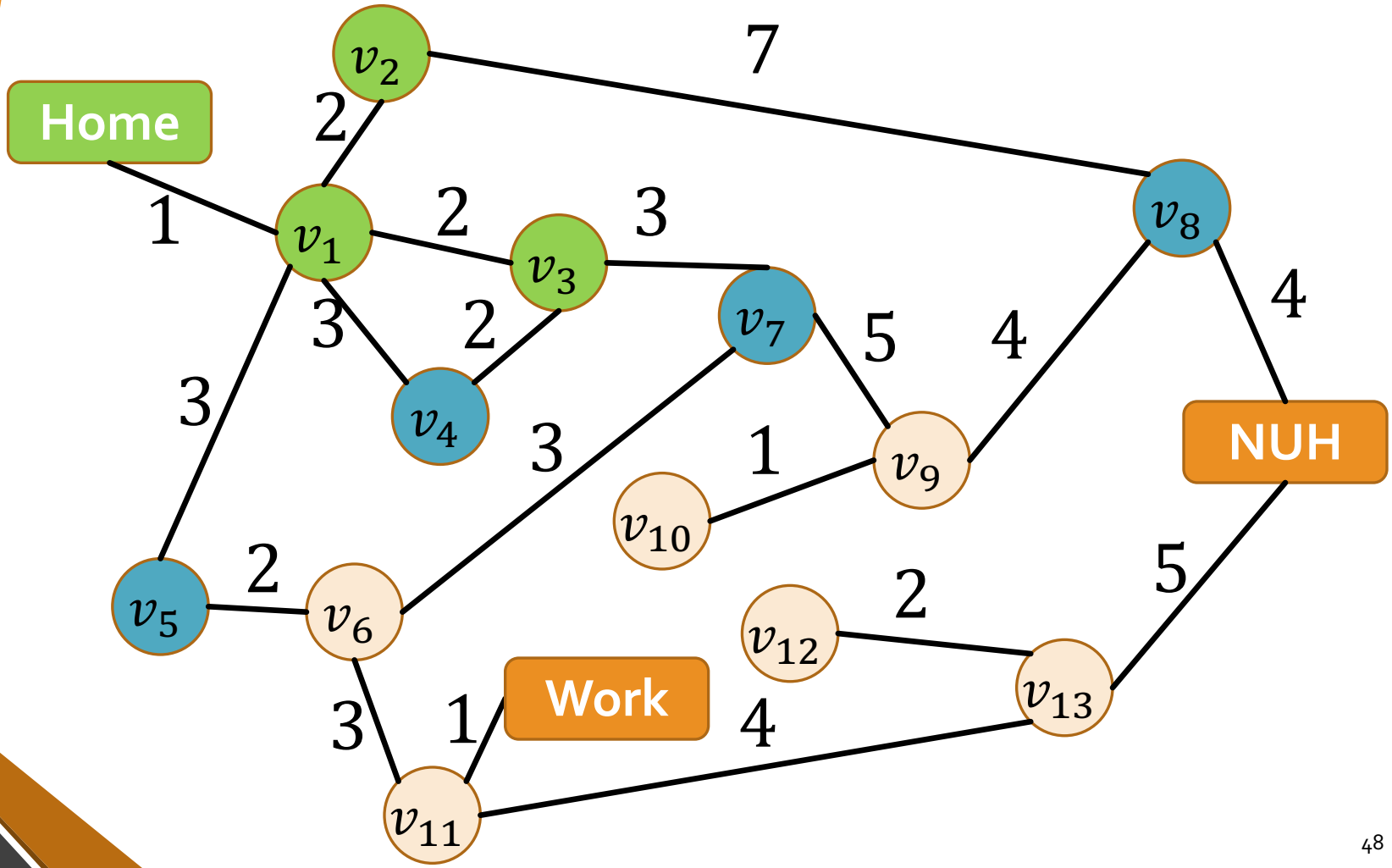
$$[\langle v_2, 3\rangle, \langle v_3, 3\rangle, \langle v_4, 4\rangle, \langle v_5, 4\rangle]$$



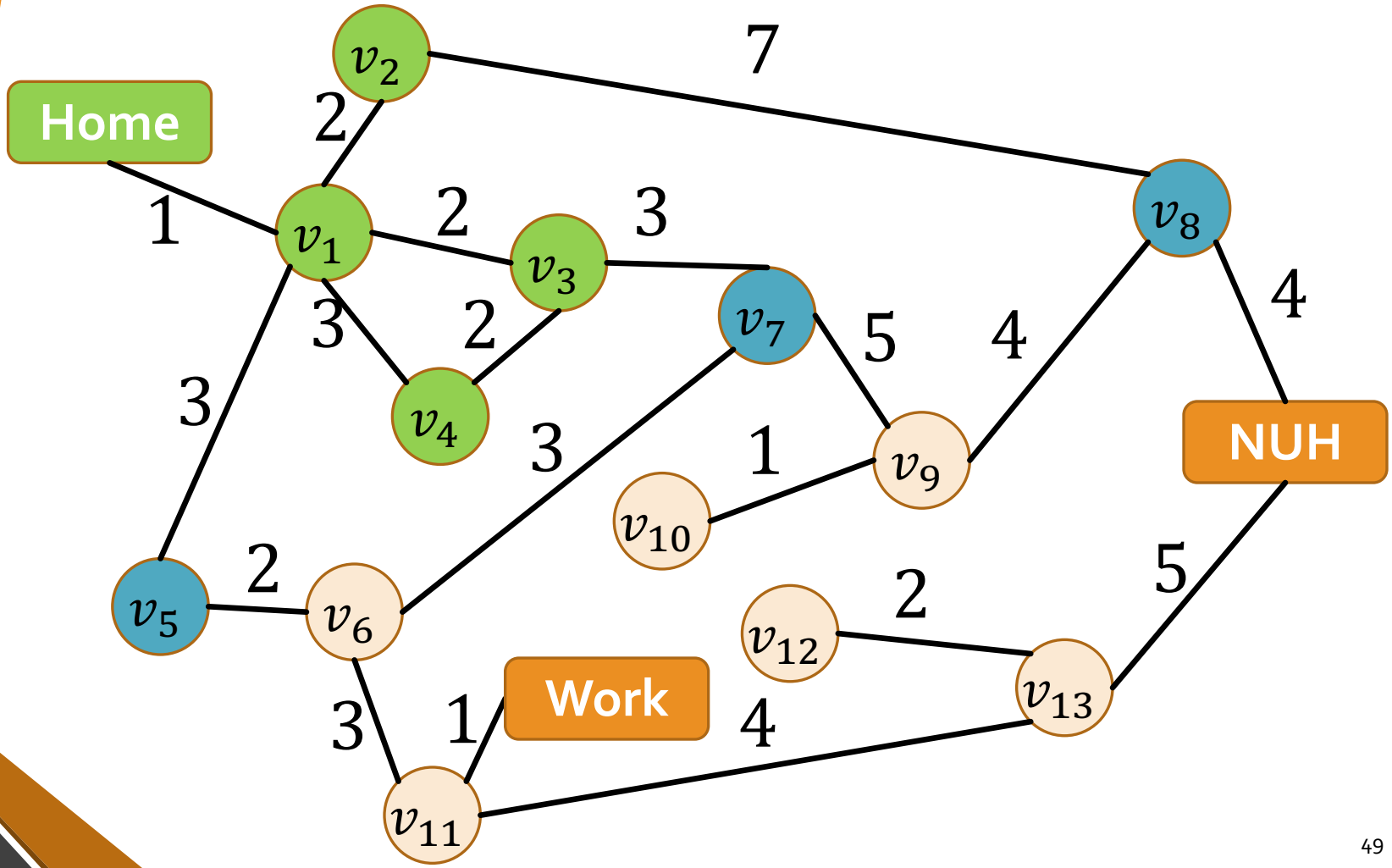
$$[\langle v_3, 3\rangle, \langle v_4, 4\rangle, \langle v_5, 4\rangle, \langle v_8, 10\rangle]$$



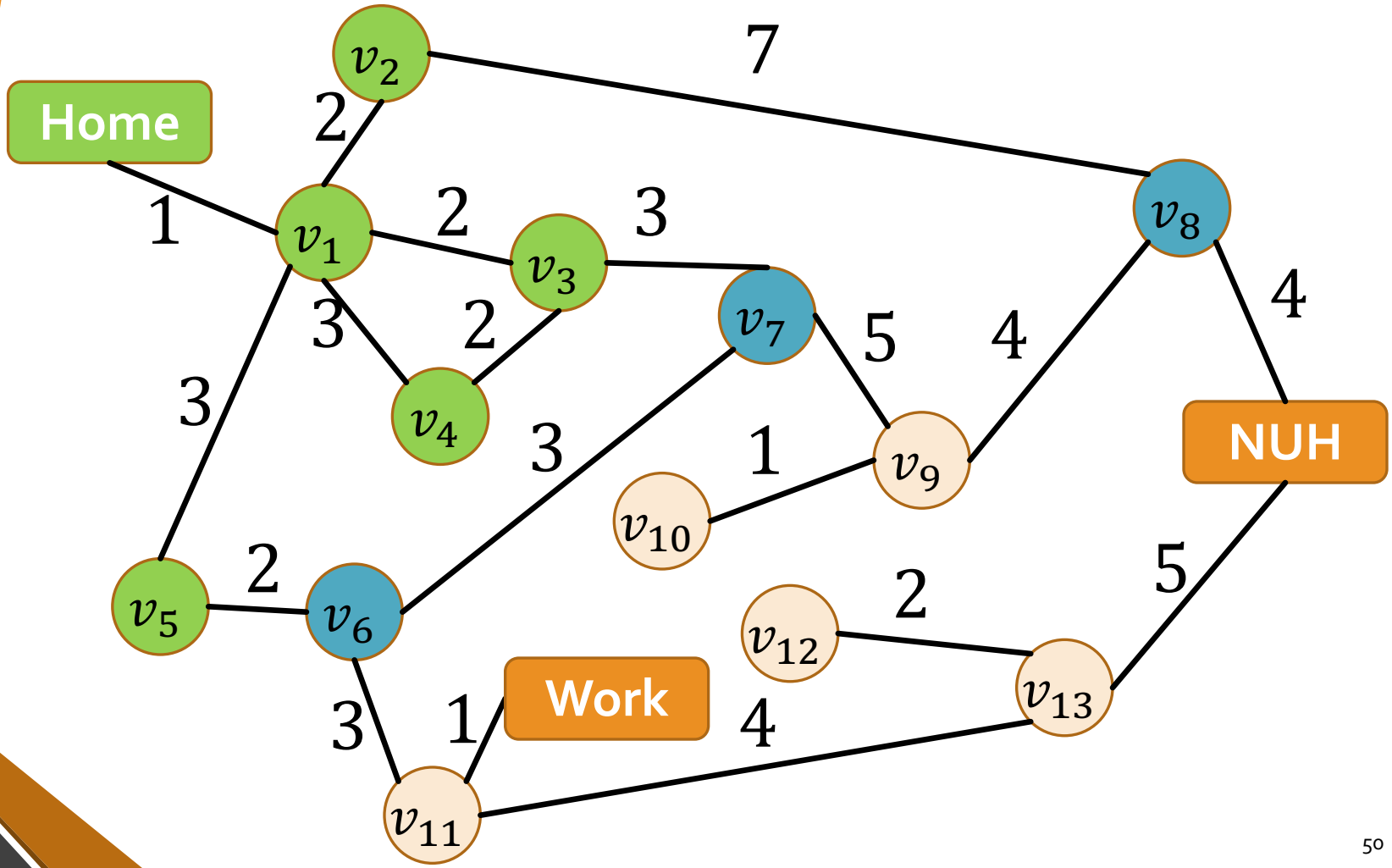
$[\langle v_4, 4\rangle, \langle v_5, 4\rangle, \langle v_7, 6\rangle, \langle v_8, 10\rangle]$



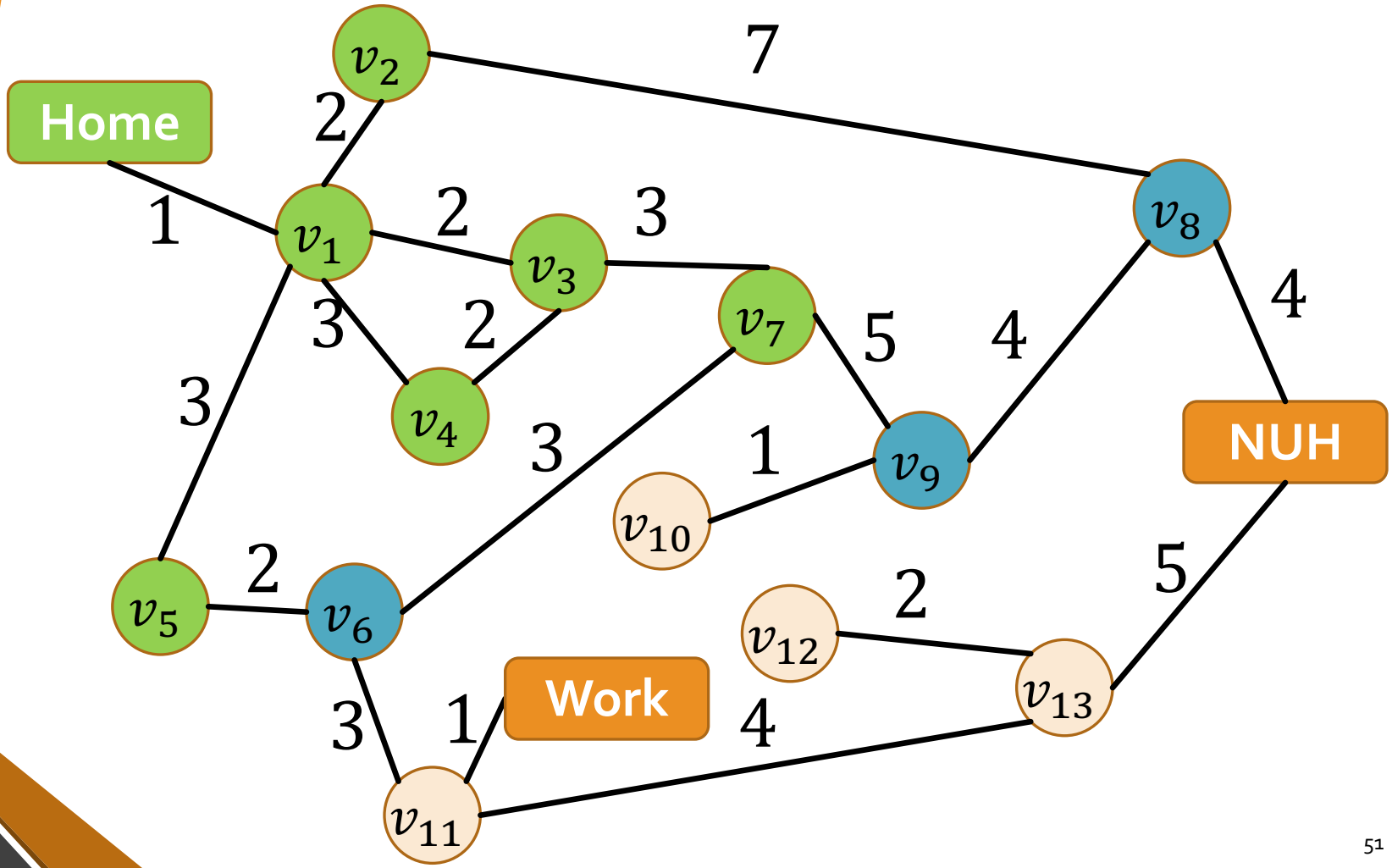
$$[\langle v_5, 4\rangle, \langle v_7, 6\rangle, \langle v_8, 10\rangle]$$



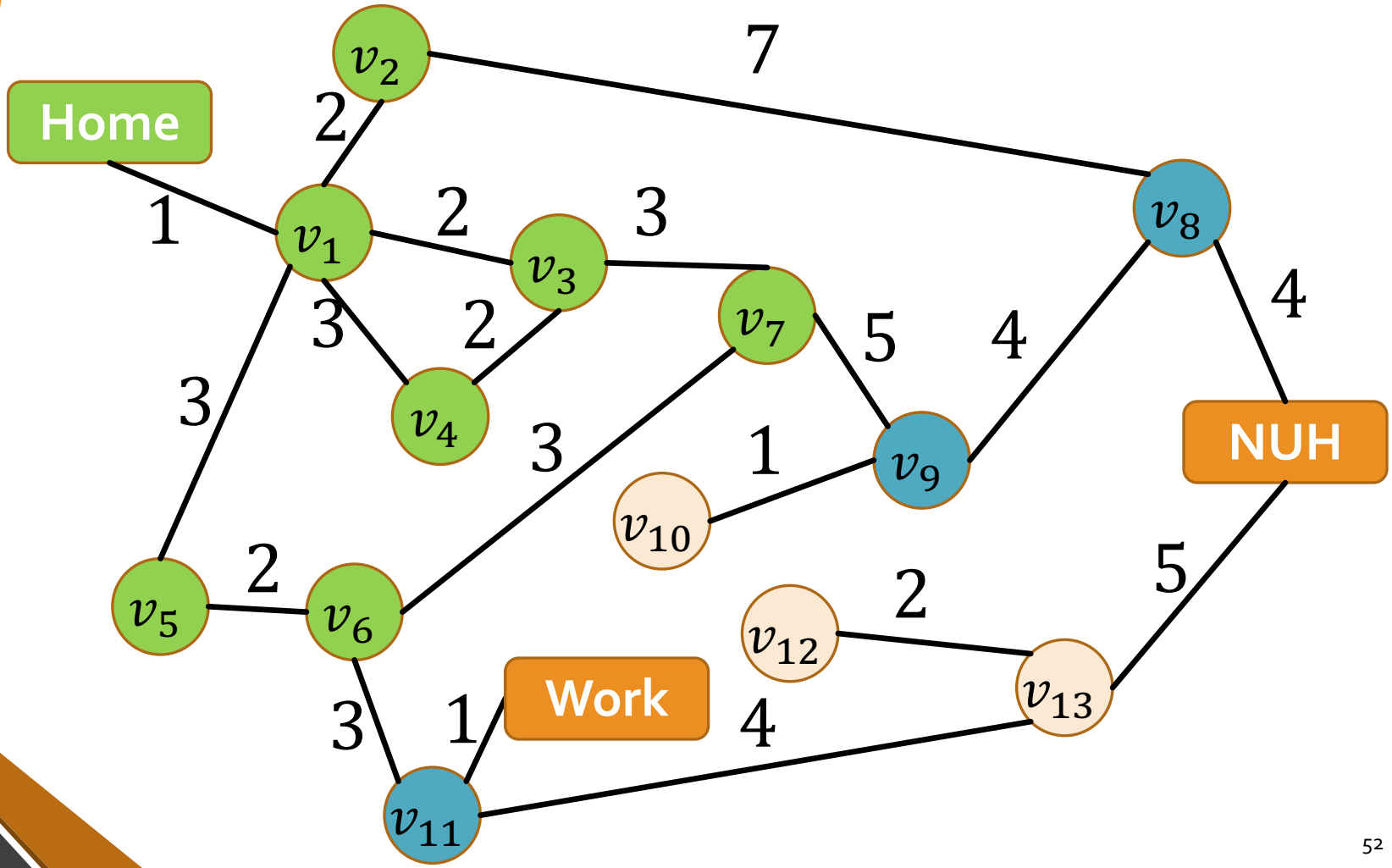
$$[\langle v_7, 6\rangle, \langle v_6, 6\rangle, \langle v_8, 10\rangle]$$



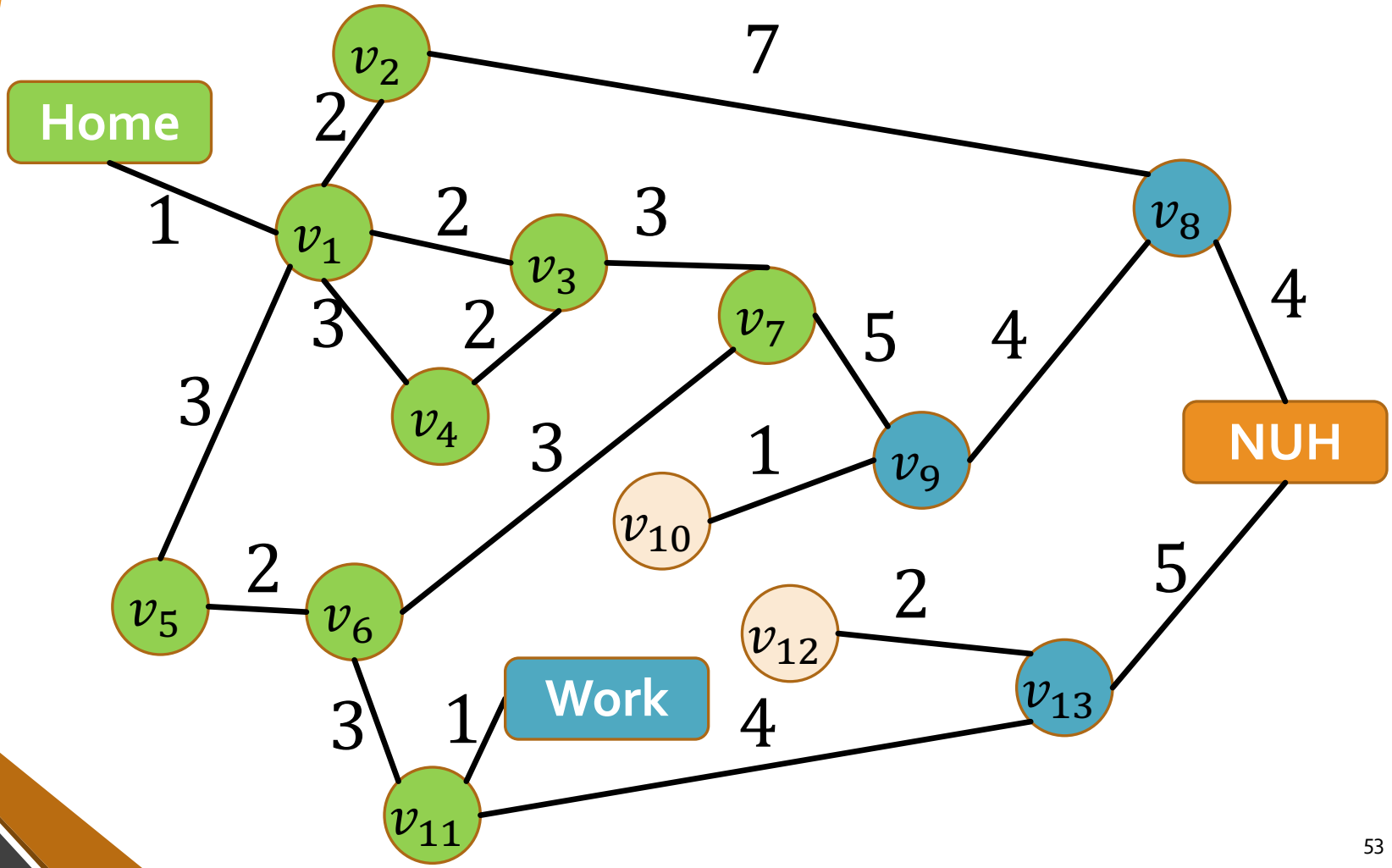
$$[\langle v_6, 6\rangle, \langle v_8, 10\rangle, \langle v_9, 11\rangle]$$



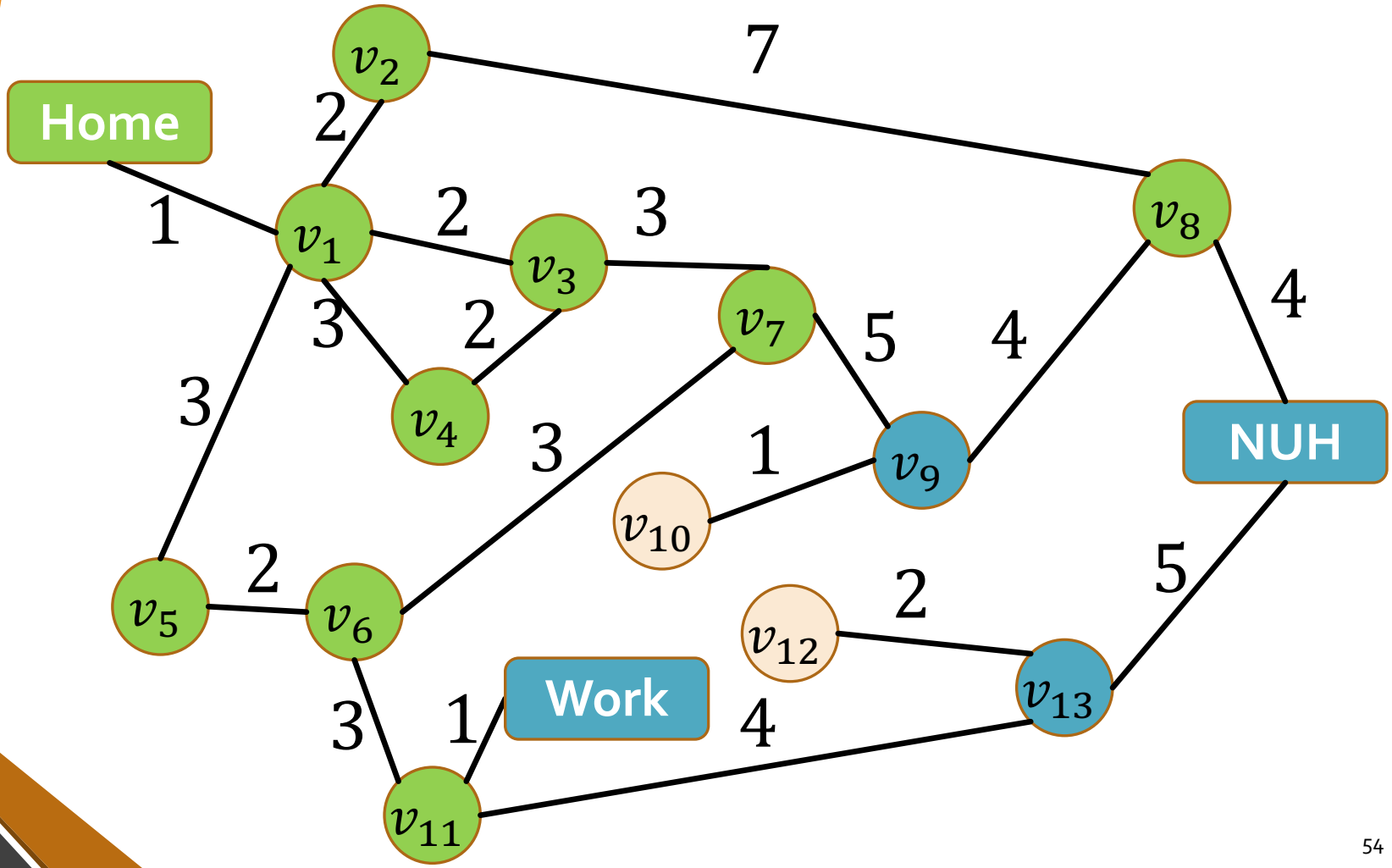
$$[\langle v_{11}, 9\rangle, \langle v_8, 10\rangle, \langle v_9, 11\rangle]$$



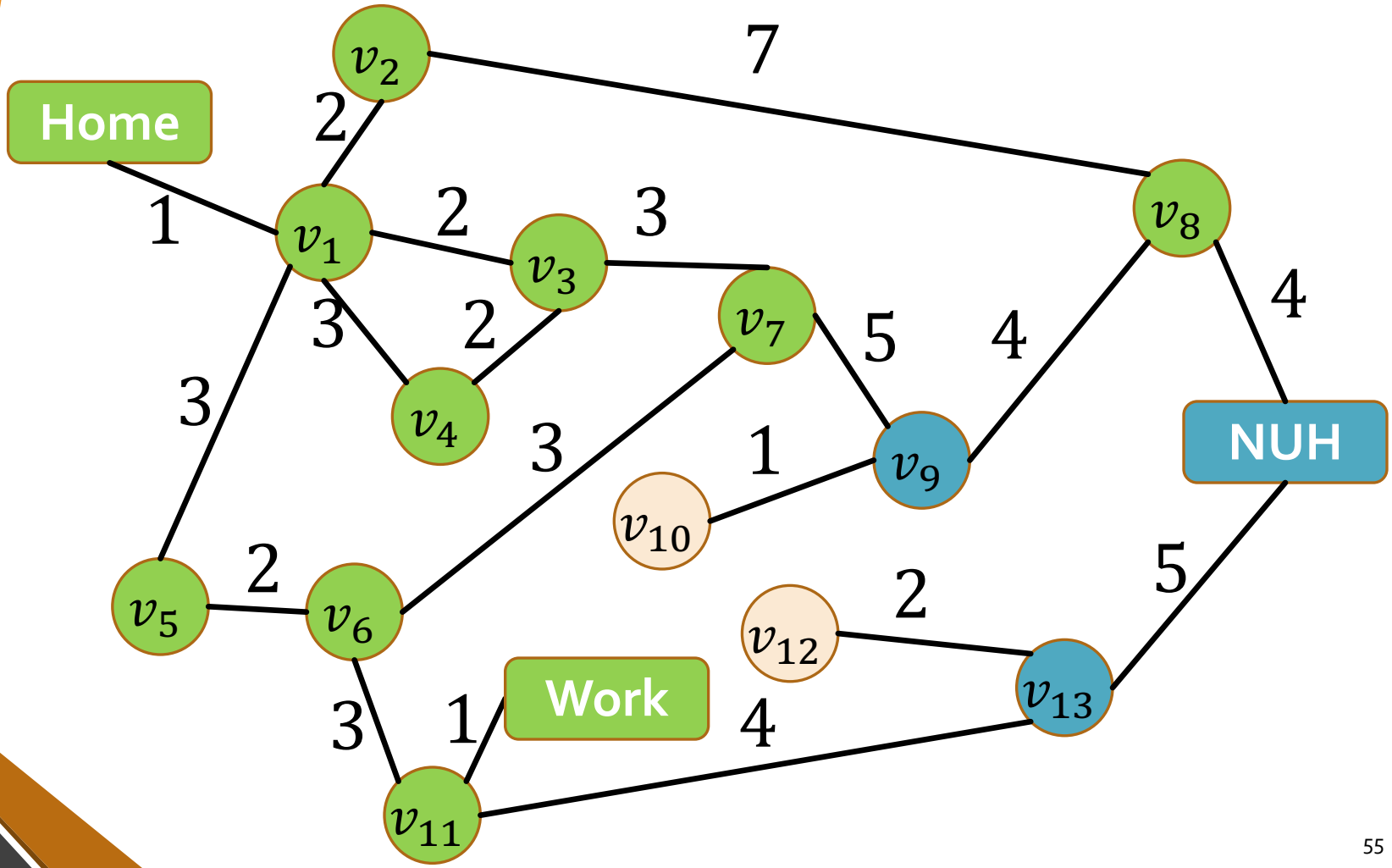
$[\langle v_8, 10\rangle, \langle Work, 10\rangle, \langle v_9, 11\rangle, \langle v_{13}, 13\rangle]$



$$[\langle Work, 10\rangle, \langle v_9, 11\rangle, \langle v_{13}, 13\rangle, \langle NUH, 14\rangle]$$



$$[\langle v_9, 11\rangle, \langle v_{13}, 13\rangle, \langle NUH, 14\rangle]$$



Properties of Uniform Cost Search

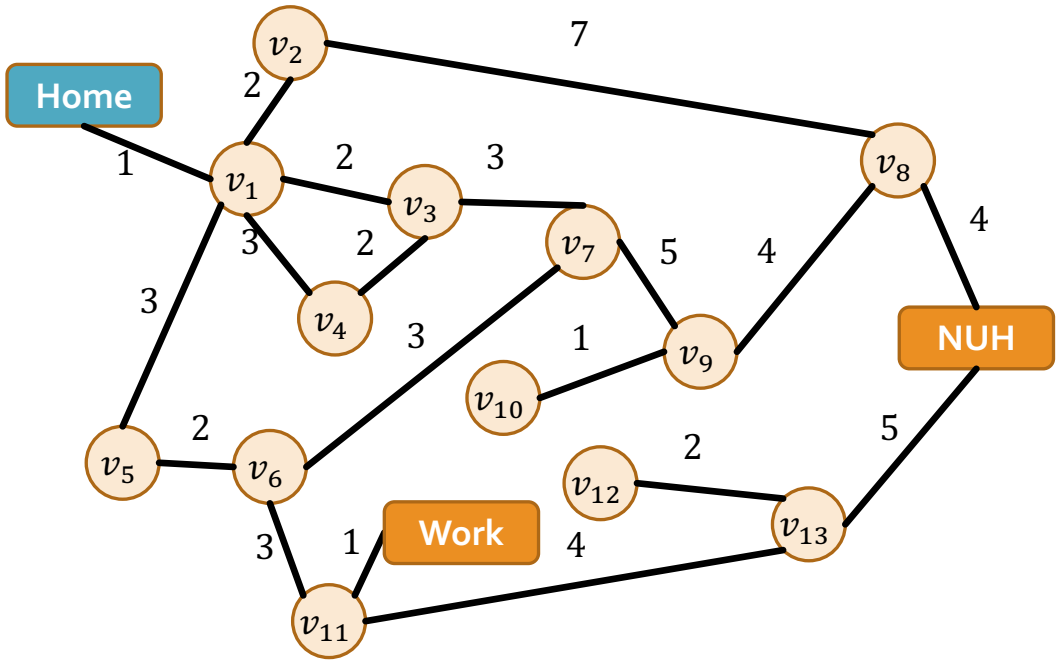
Property	
Complete?	Yes (if all step costs are $\geq \varepsilon$)
Optimal	Yes (shortest path nodes expanded first)
Time	$\mathcal{O}\left(b^{1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor}\right)$ where C^* is the optimal cost.
Space	$\mathcal{O}\left(b^{1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor}\right)$

Uniform Cost Search

- At every round we get at least a distance of ε closer to the goal.
- Reach nodes at distance $0, \varepsilon, 2\varepsilon, \ldots, \left\lfloor \frac{C^*}{\varepsilon} \right\rfloor \varepsilon$ of goal; total $\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor + 1$ steps.
- At step k (depth k at most): keep $\leq b^k$ nodes in frontier.

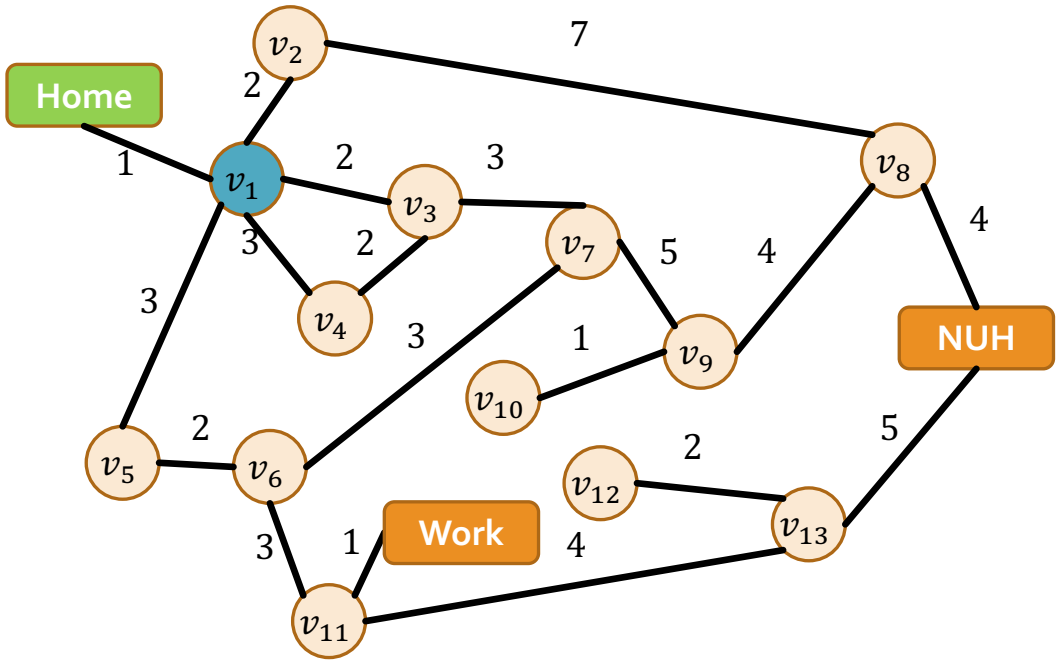
Depth-First Search

- Idea: Expand deepest unexpanded node
- Implementation: Frontier = LIFO stack, i.e., insert successors at the front

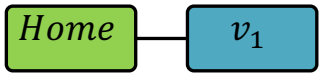

Home
v_2
7
2
1
v_1
2
3
v_3
v_8
4
3
2
v_7
5
4
v_4
3
3
1
v_9
NUH
v_{10}
2
5
v_5
v_6
2
v_{12}
Work
v_{13}
3
1
4
v_{11}

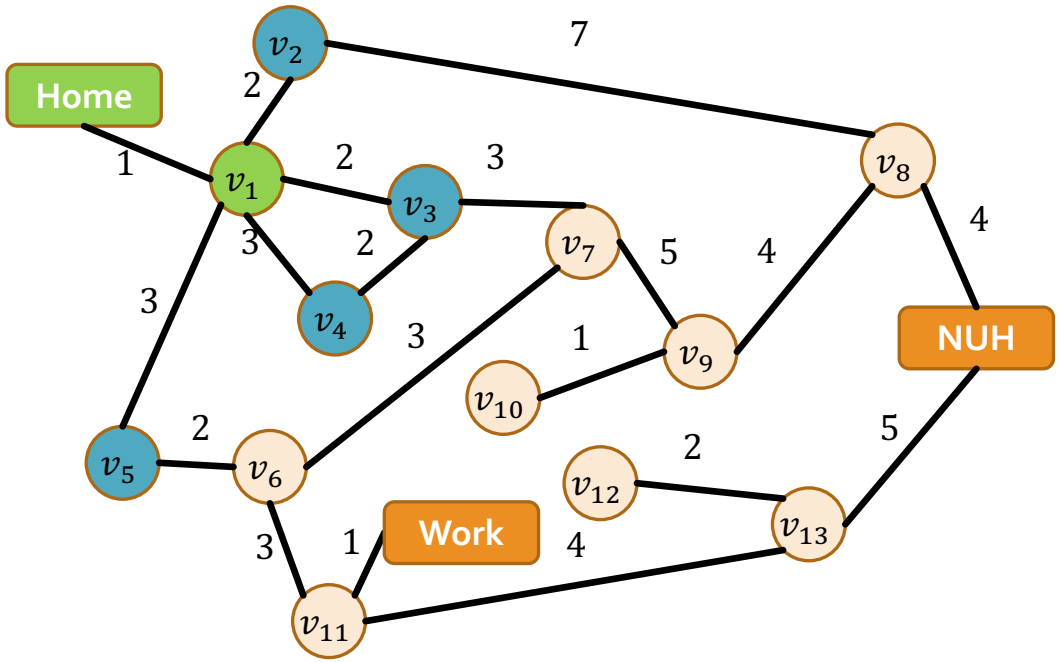
Home

Home

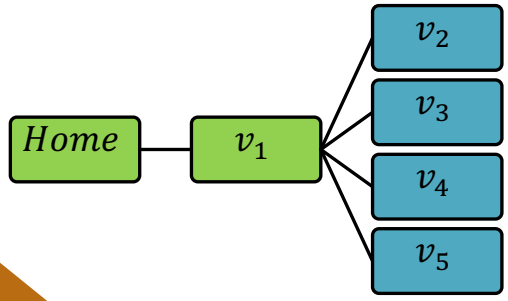

Home
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
Work
NUH
1
2
3
4
5
7

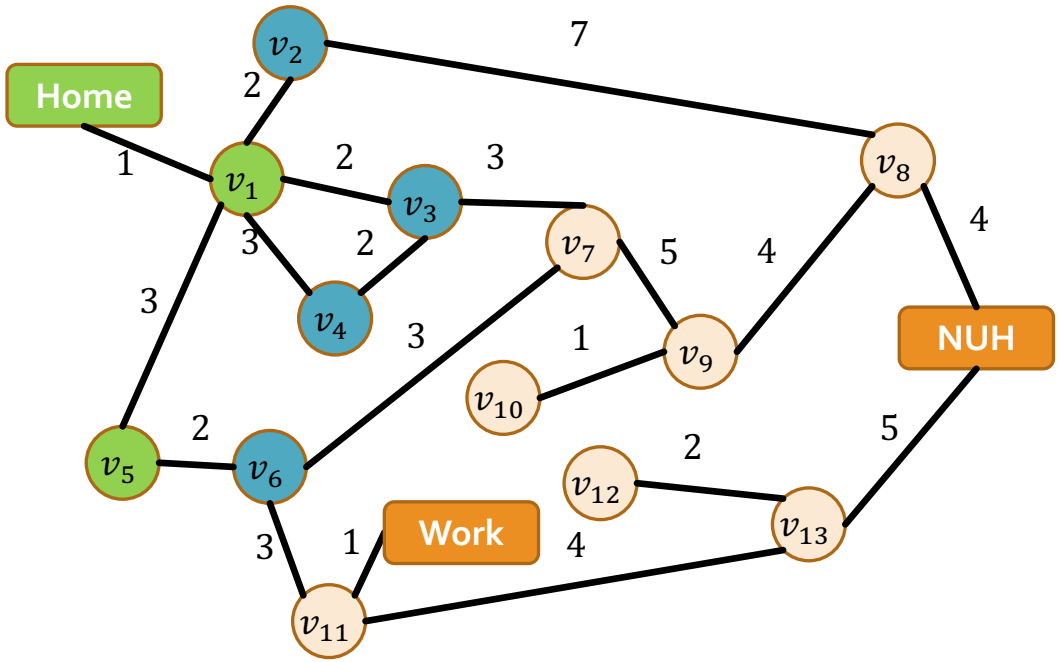
v_1


Home
v_1

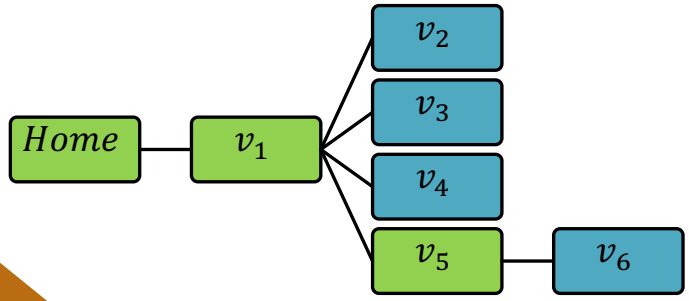

Home
v_2
v_1
v_3
v_8
v_7
v_4
v_9
NUH
v_{10}
v_5
v_6
v_{12}
v_{13}
Work
v_{11}
7
2
1
2
3
3
2
5
4
4
3
3
1
2
5
2
3
1
4

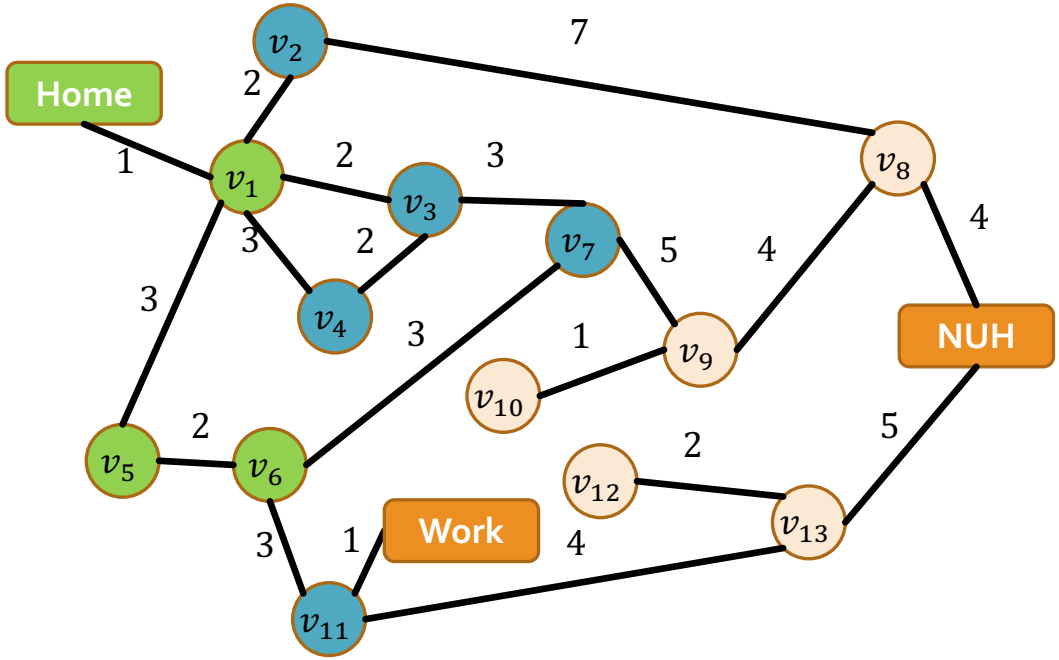
v_5
v_4
v_3
v_2


Home
v_1
v_2
v_3
v_4
v_5

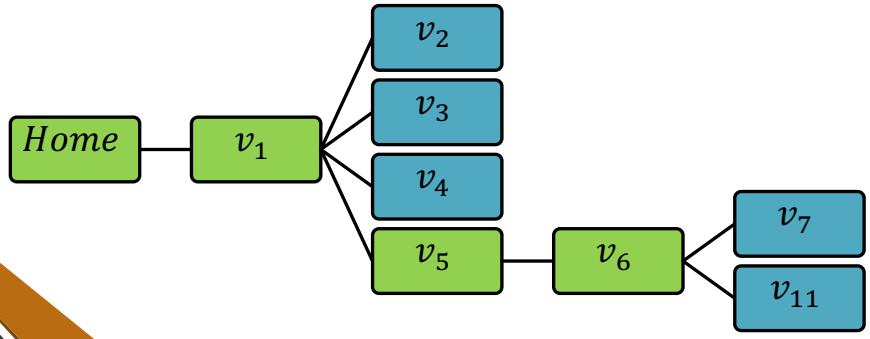

Home
NUH
Work
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
1
2
3
4
5
7

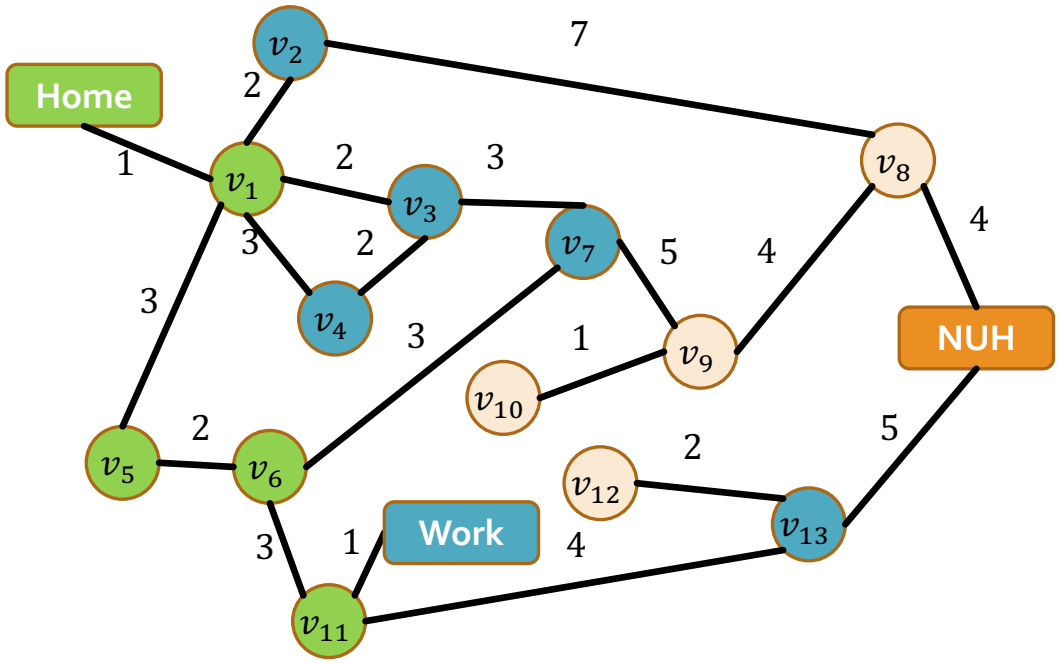
v_6
v_4
v_3
v_2


Home
v_1
v_2
v_3
v_4
v_5
v_6

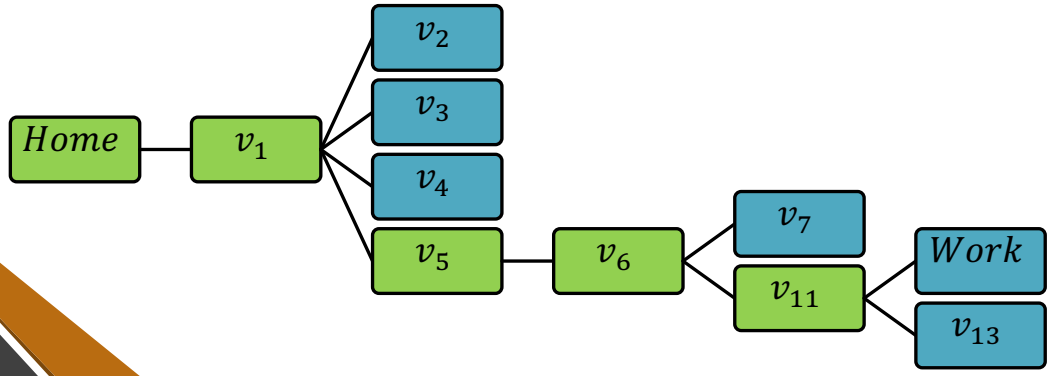


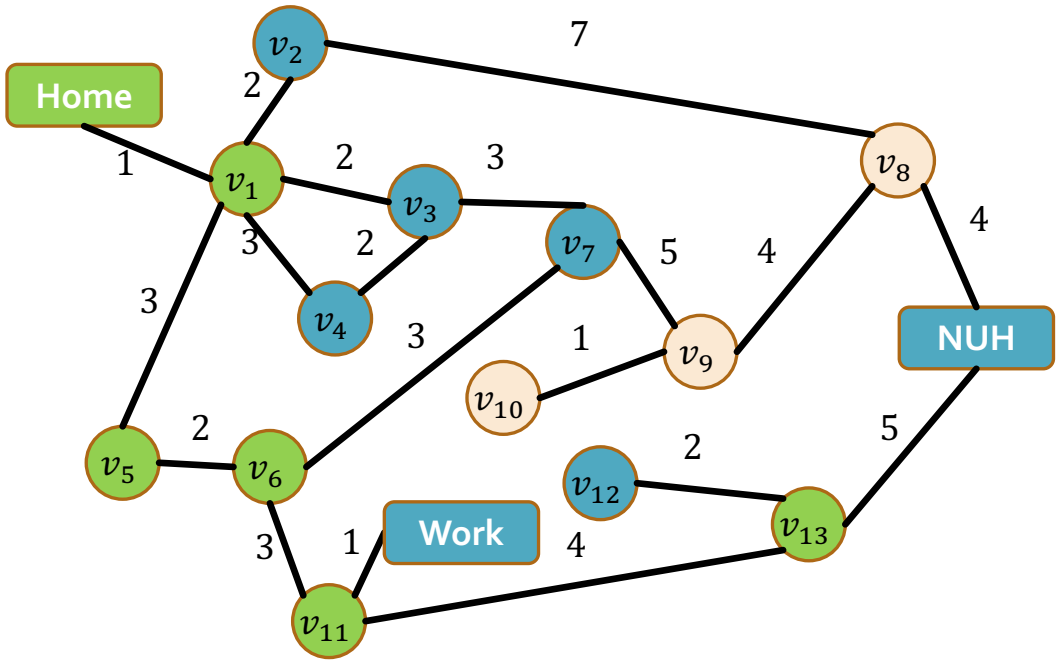
v_{11}
v_7
v_4
v_3
v_2



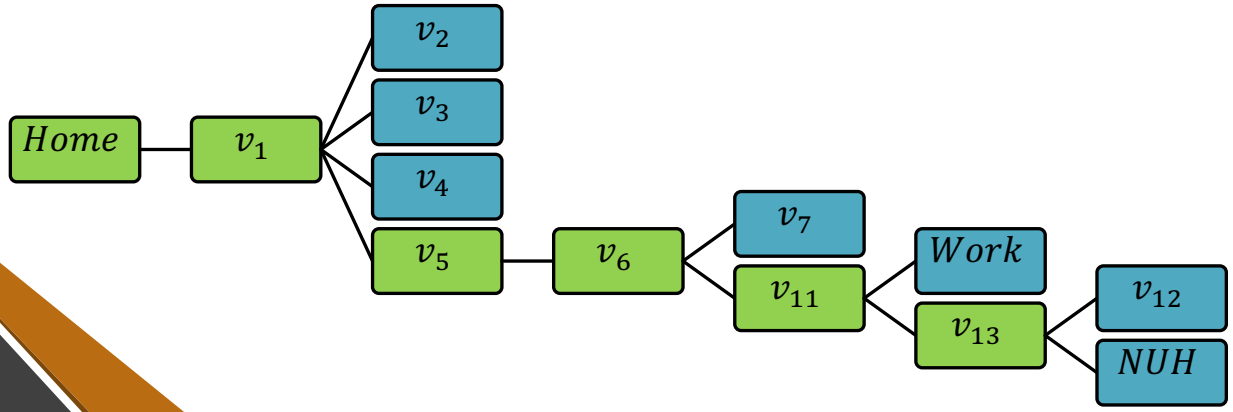


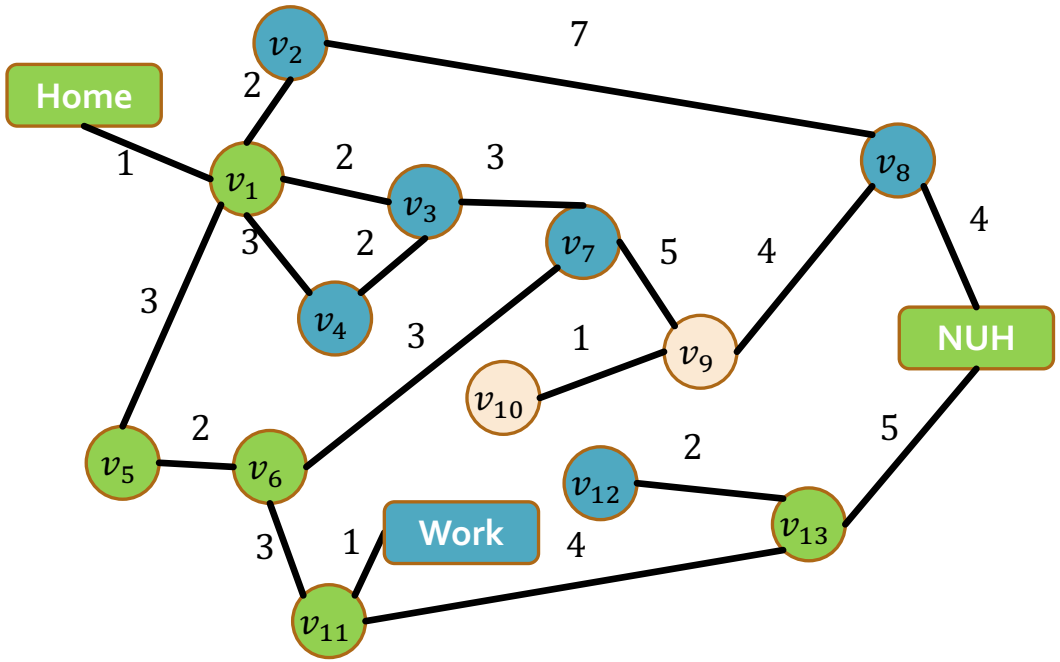
v_{13}
$Work$
v_7
v_4
v_3
v_2



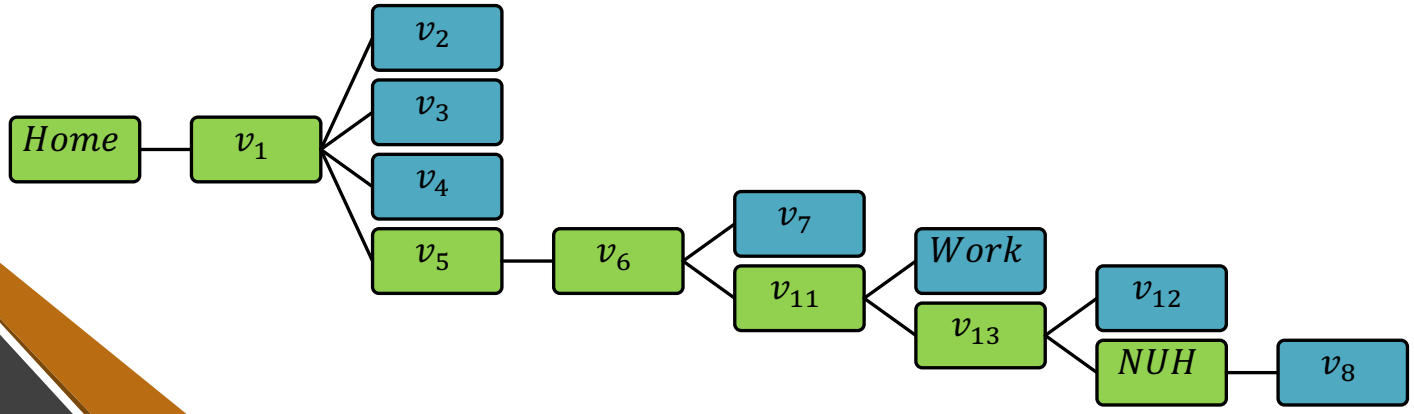


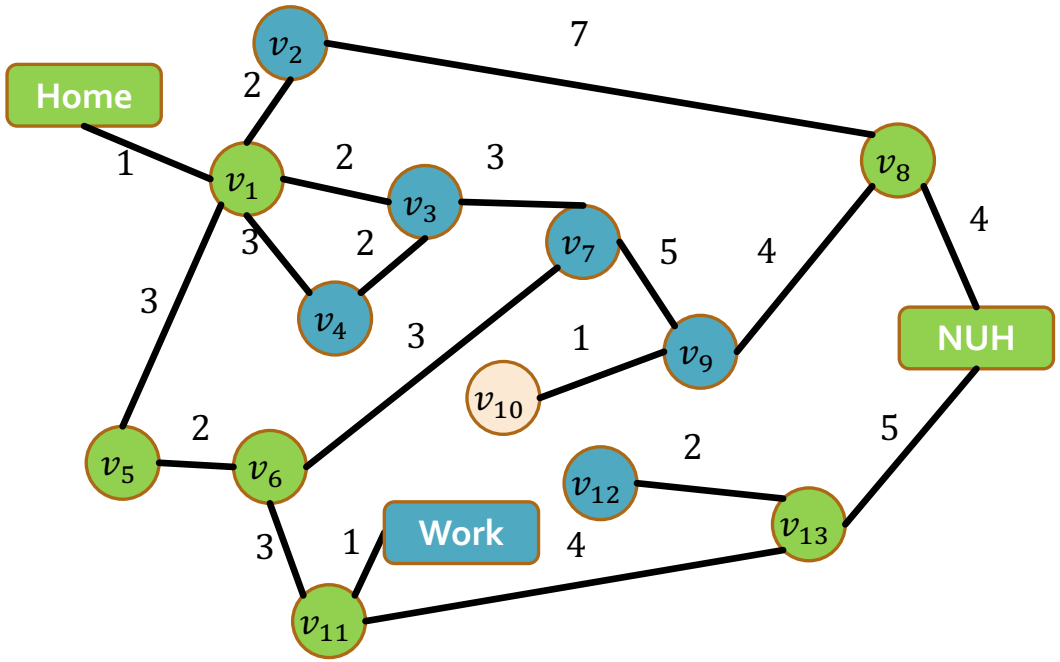
NUH
v_{12}
$Work$
v_7
v_4
v_3
v_2



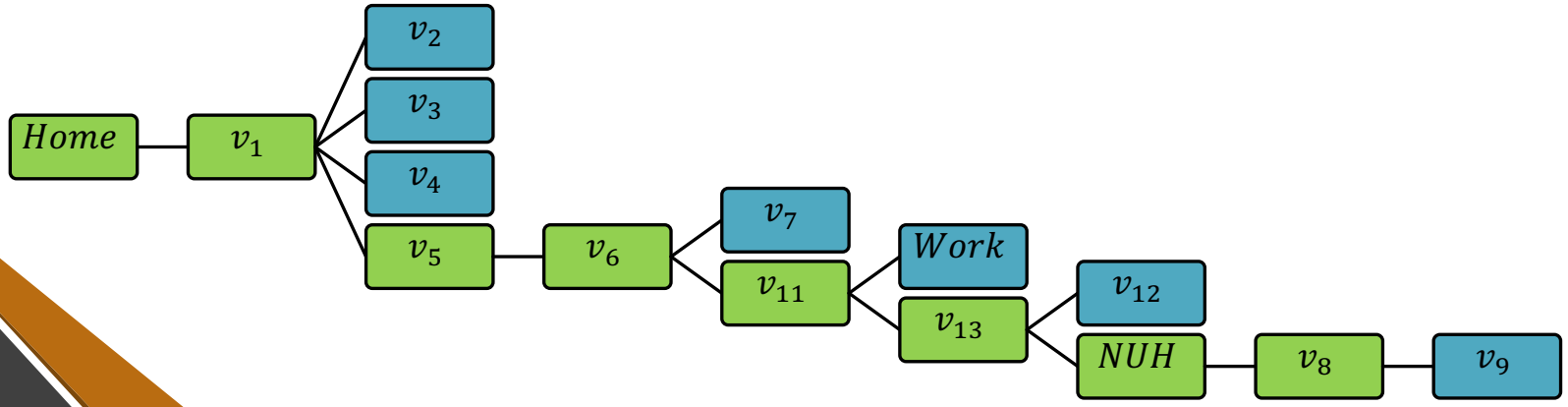


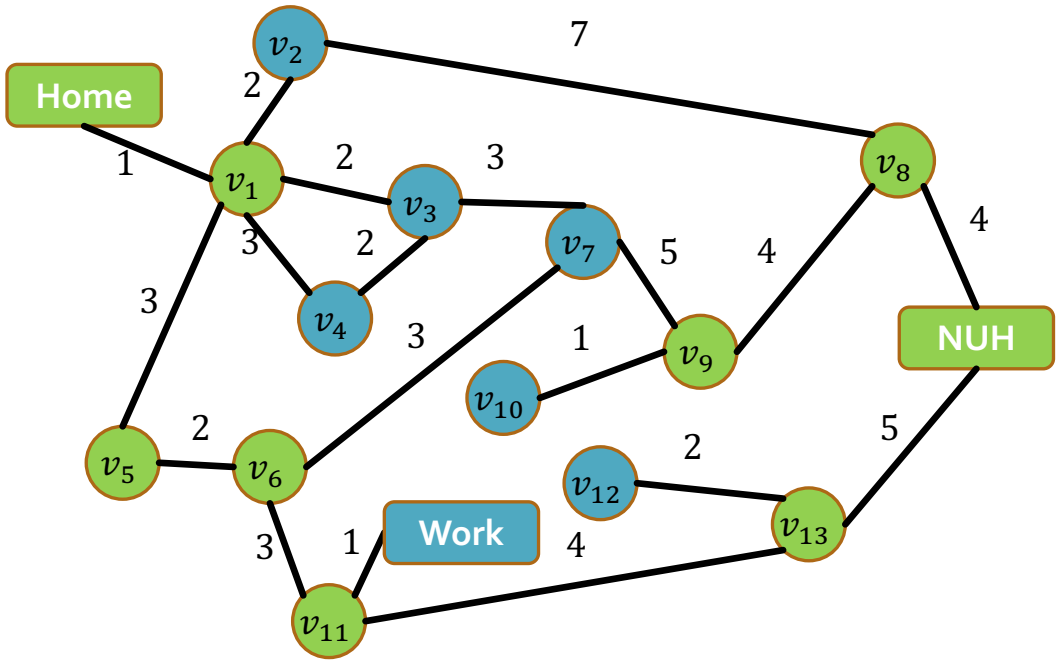
v_8
v_{12}
$Work$
v_7
v_4
v_3
v_2



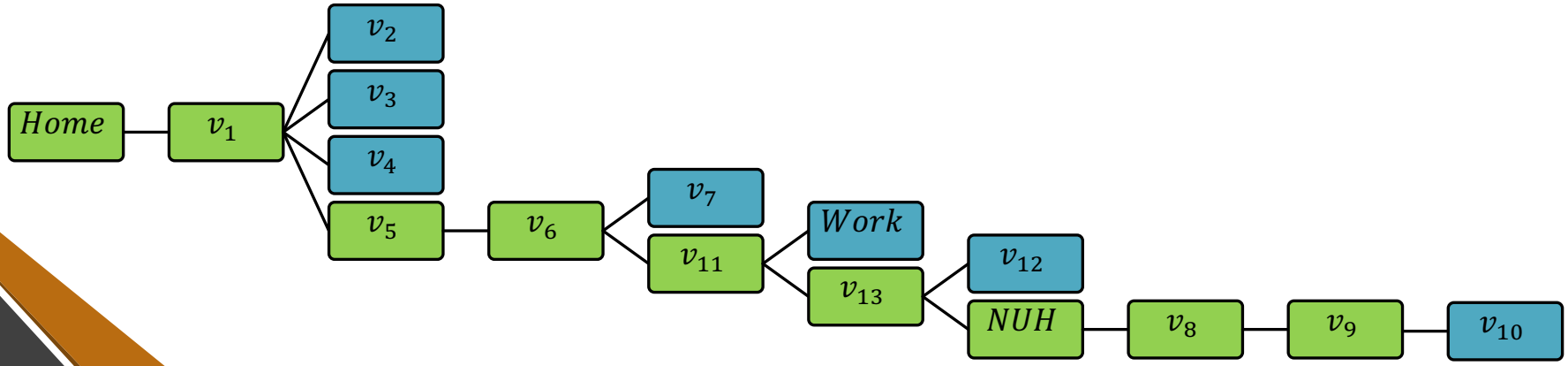


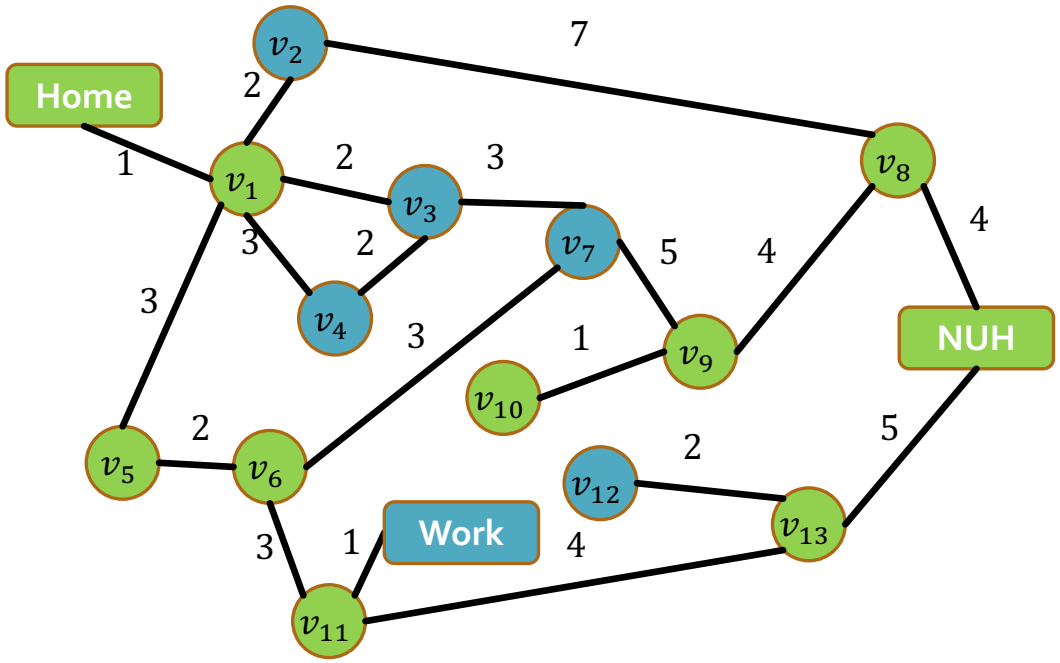
v_9
v_{12}
$Work$
v_7
v_4
v_3
v_2



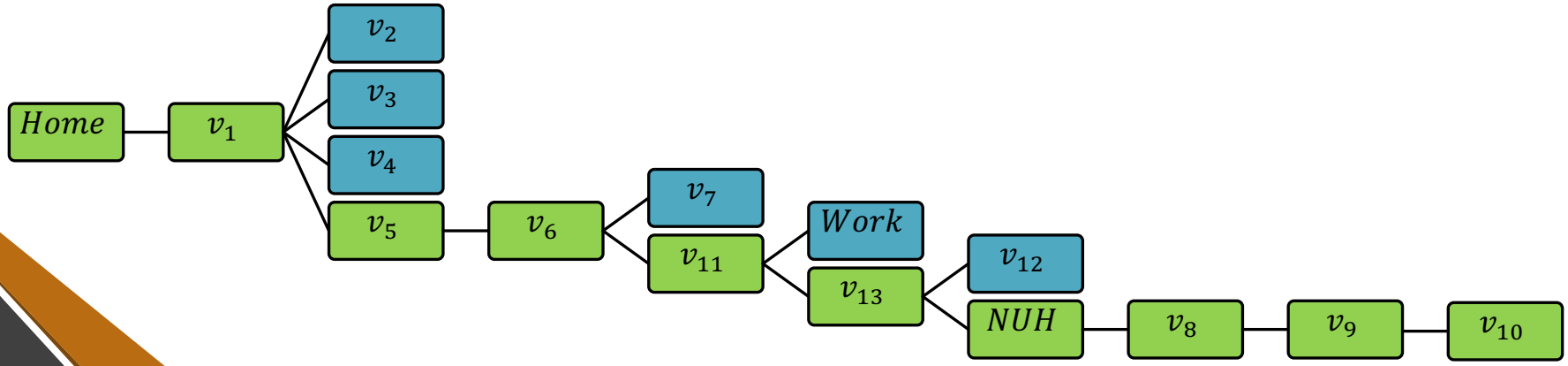


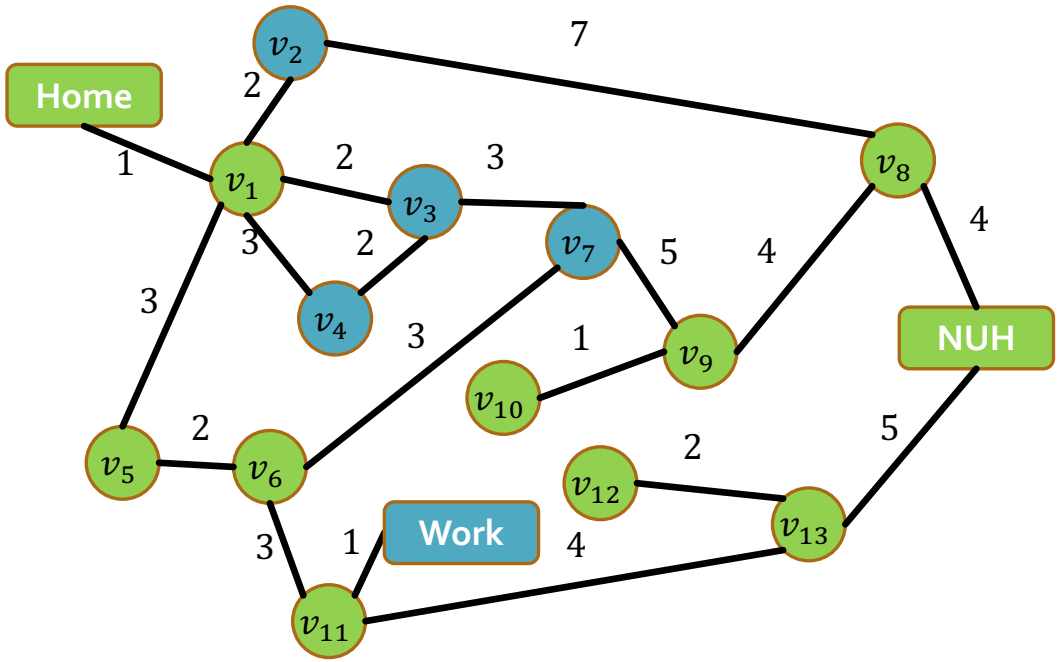
v_{10}
v_{12}
$Work$
v_7
v_4
v_3
v_2



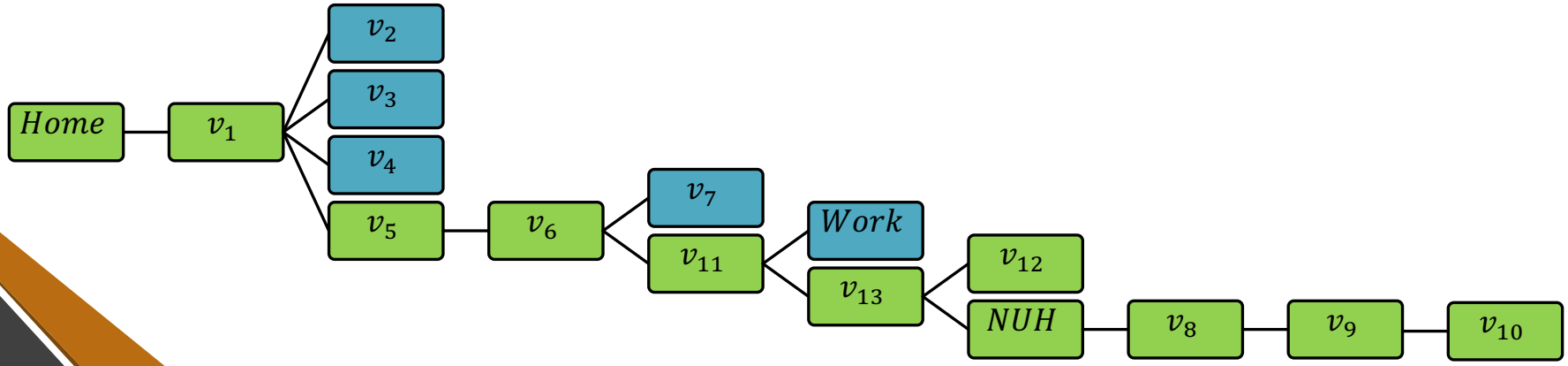


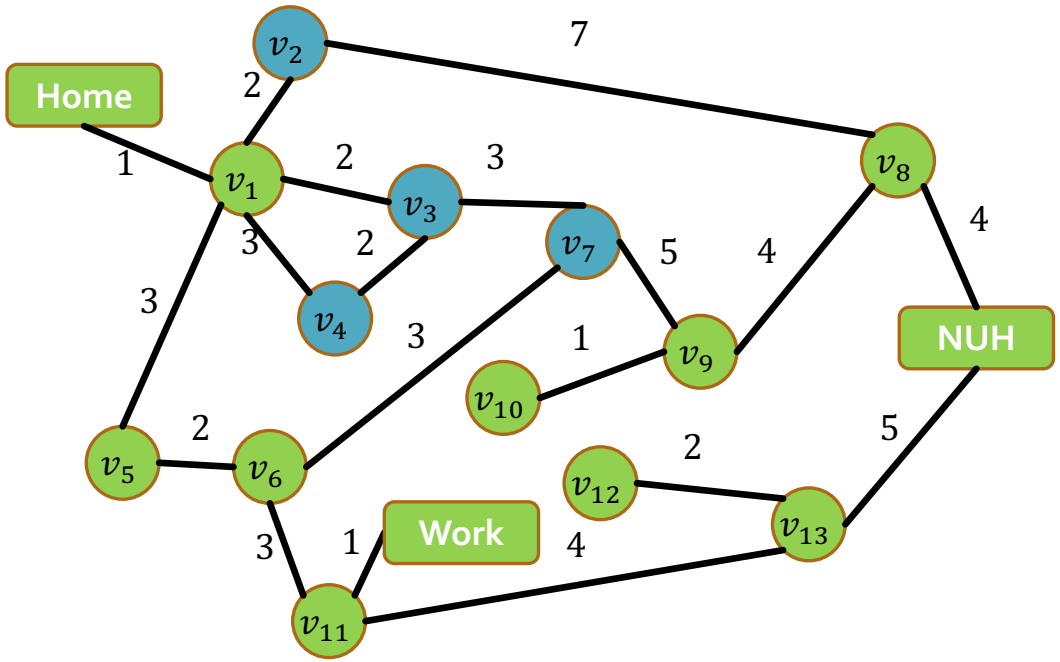
v_{12}
$Work$
v_7
v_4
v_3
v_2



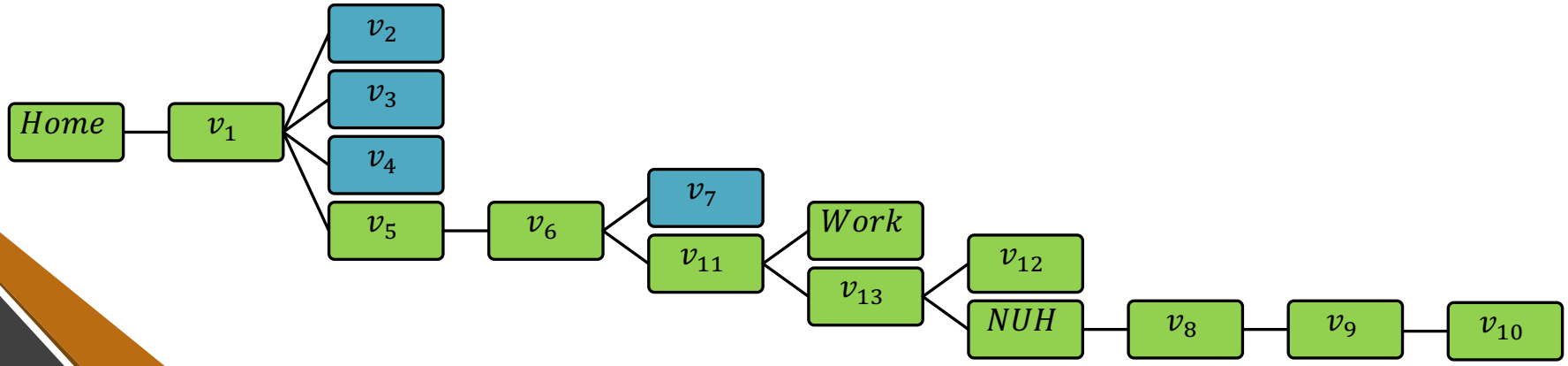


$Work$
v_7
v_4
v_3
v_2




Home
v_2
v_1
v_3
v_8
v_7
v_4
v_9
NUH
v_{10}
v_5
v_6
v_{12}
v_{13}
Work
v_{11}
7
2
1
2
3
3
2
5
4
4
3
3
1
2
5
2
3
1
4

v_7
v_4
v_3
v_2


$Home$
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_{11}
$Work$
v_{13}
v_{12}
NUH
v_8
v_9
v_{10}

Properties of DFS

Property	
Complete?	No on infinite depth graphs
Optimal	No
Time	$\mathcal{O}(b^m)$
Space	$\mathcal{O}(bm)$ (can be $\mathcal{O}(m)$)

Depth-First Search

- When checking a node v, we push at most b descendants to stack.
- We do so at most m times $\Rightarrow \mathcal{O}(bm)$ space.
- Do we really need to push **all** b descendants to the stack?

Depth-Limited Search (DLS)

- Idea: run DFS with depth limit ℓ, i.e., do not search at depth greater than ℓ.
- Same guarantees as DFS, with ℓ instead of m ($\mathcal{O}(b^{\ell})$ time; $\mathcal{O}(b\ell)$ space)

Iterative Deepening Search (IDS)

- Idea: Perform DLSs with increasing depth limit until goal node is found
- Better if state space is large and depth of solution is unknown
- Implementation:

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution, or failure
    for depth = 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH(problem, depth)
        if result ≠ cutoff then return result
```

How Wasteful is IDS?

- Number of nodes generated in DLS to depth ℓ with branching factor b:

$$\mathcal{O}(b^0) + \mathcal{O}(b^1) + \mathcal{O}(b^2) + \cdots + \mathcal{O}\left(b^{\ell-2}\right) + \mathcal{O}\left(b^{\ell-1}\right) + \mathcal{O}\left(b^{\ell}\right)$$

Number of nodes generated in IDS to depth d with branching factor b:

$$(d+1)\mathcal{O}(b^0) + d\mathcal{O}(b^1) + (d-1)\mathcal{O}(b^2) + \cdots 3\mathcal{O}\left(b^{d-2}\right) + 2\mathcal{O}\left(b^{d-1}\right) + \mathcal{O}(b^d)$$

- For $b = 10, d = 5$,
  - $N_{DLS} = 1 + 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 = 111{,}111$
  - $N_{IDS} = 6 + 50 + 400 + 3{,}000 + 20{,}000 + 100{,}000 = 123{,}456$
- Overhead $= \frac{123{,}456 - 111{,}111}{111{,}111} = 11\%$

Properties of IDS

Property	
Complete?	Yes (if b is finite)
Optimal	No (unless step cost is 1)
Time	$\mathcal{O}(b^d)$
Space	$\mathcal{O}(bd)$ (can be $\mathcal{O}(d)$)

Summary

Property	BFS	UCS	DFS	DLS	IDS
Complete	Yes[1]	Yes[2]	No	No	Yes[1]
Optimal	No[3]	Yes	No	No	No[3]
Time	$\mathcal{O}(b^d)$	$\mathcal{O}\left(b^{1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor}\right)$	$\mathcal{O}(b^m)$	$\mathcal{O}(b^\ell)$	$\mathcal{O}(b^d)$
Space	$\mathcal{O}(b^d)$	$\mathcal{O}\left(b^{1+\left\lfloor \frac{C^*}{\varepsilon} \right\rfloor}\right)$	$\mathcal{O}(bm)$	$\mathcal{O}(b\ell)$	$\mathcal{O}(bd)$

1. BFS and IDS are complete if b is finite.
2. UCS is complete if b is finite and step cost $\geq \varepsilon$
3. BFS and IDS are optimal if all step costs are identical