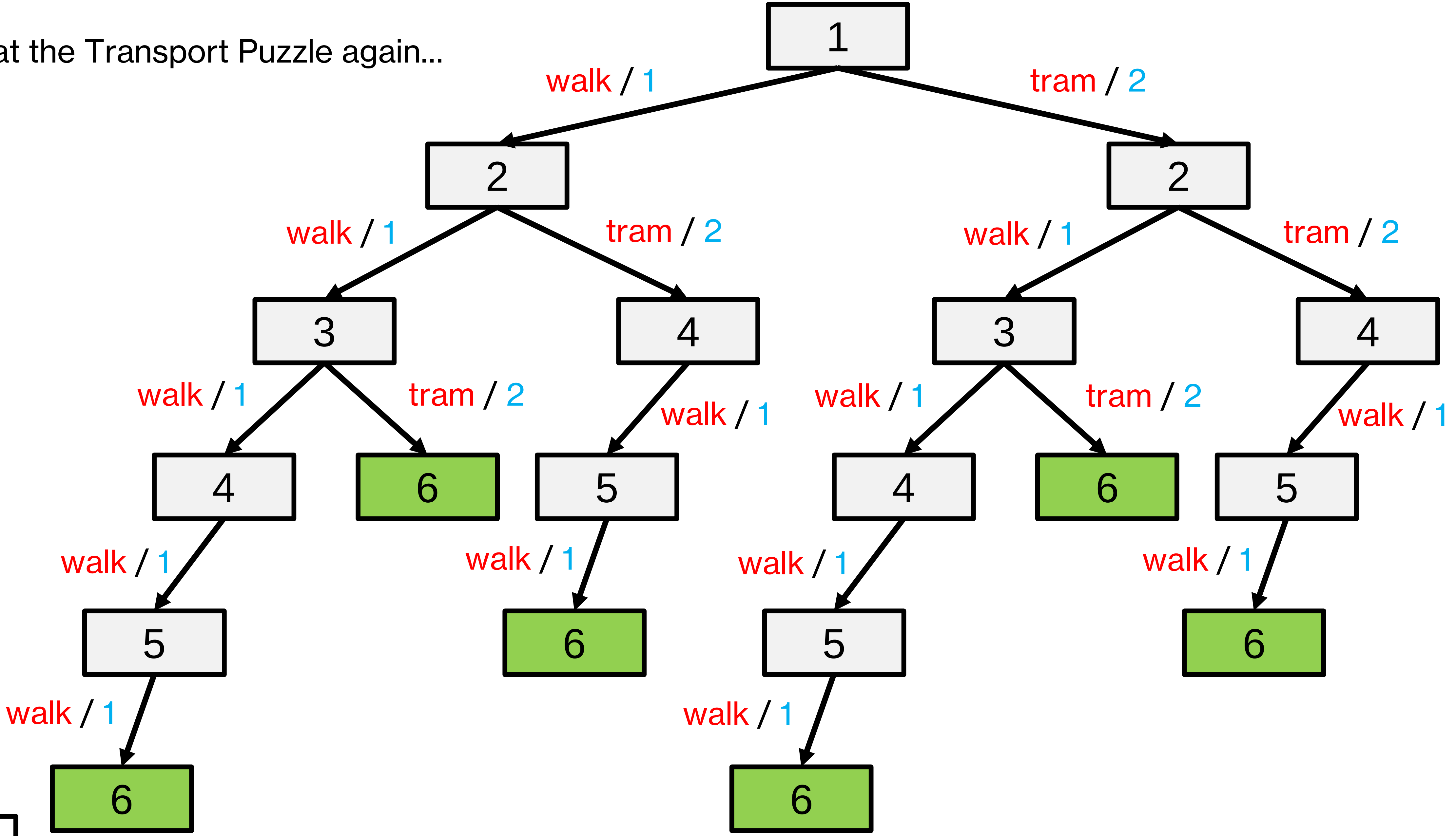
UNIFORM COST SEARCH

Thomas Tiam-Lee, PhD

College of Computer Studies

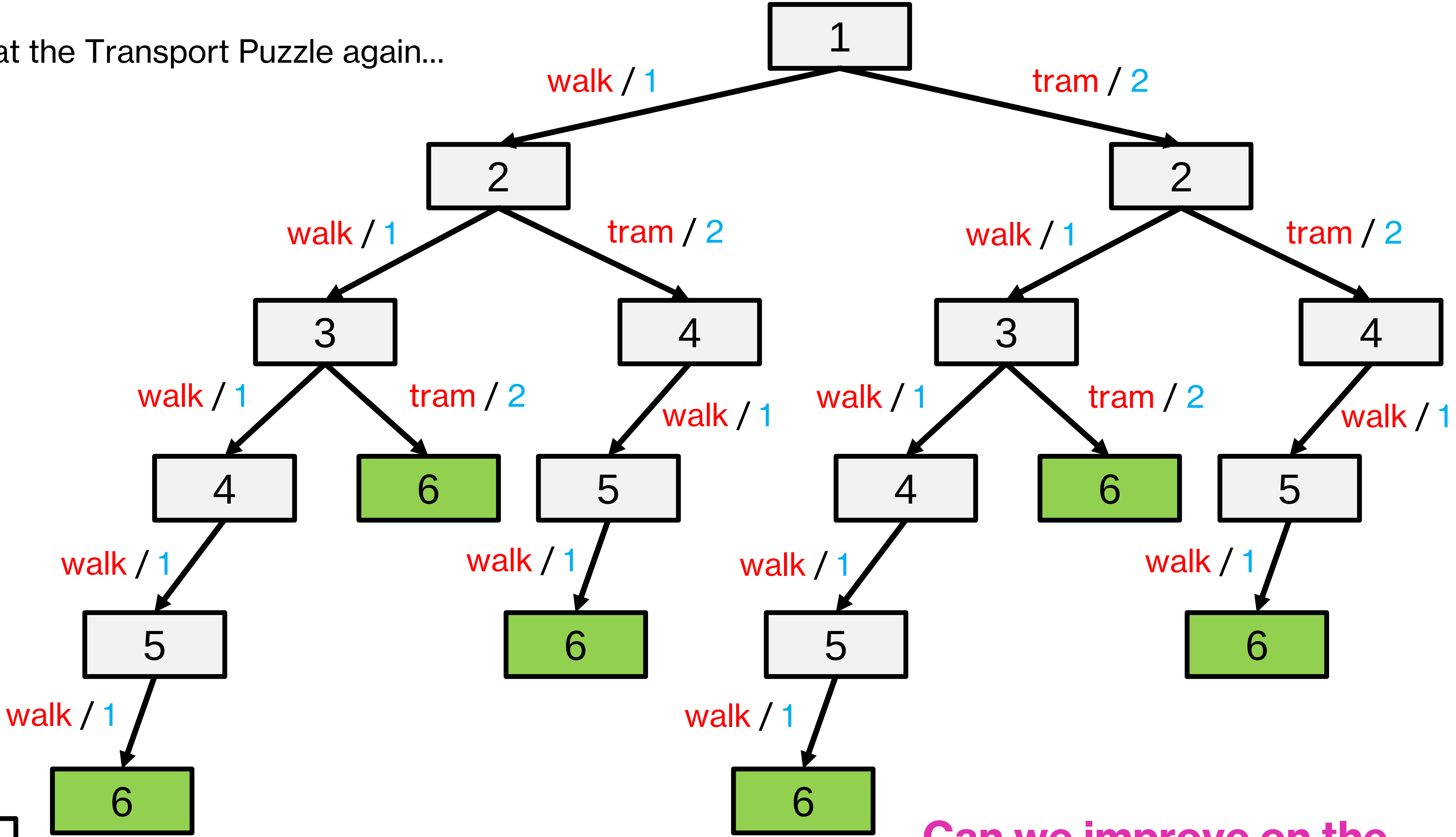
Let's look at the Transport Puzzle again...

$n = 6$



Let’s look at the Transport Puzzle again…

$n = 6$

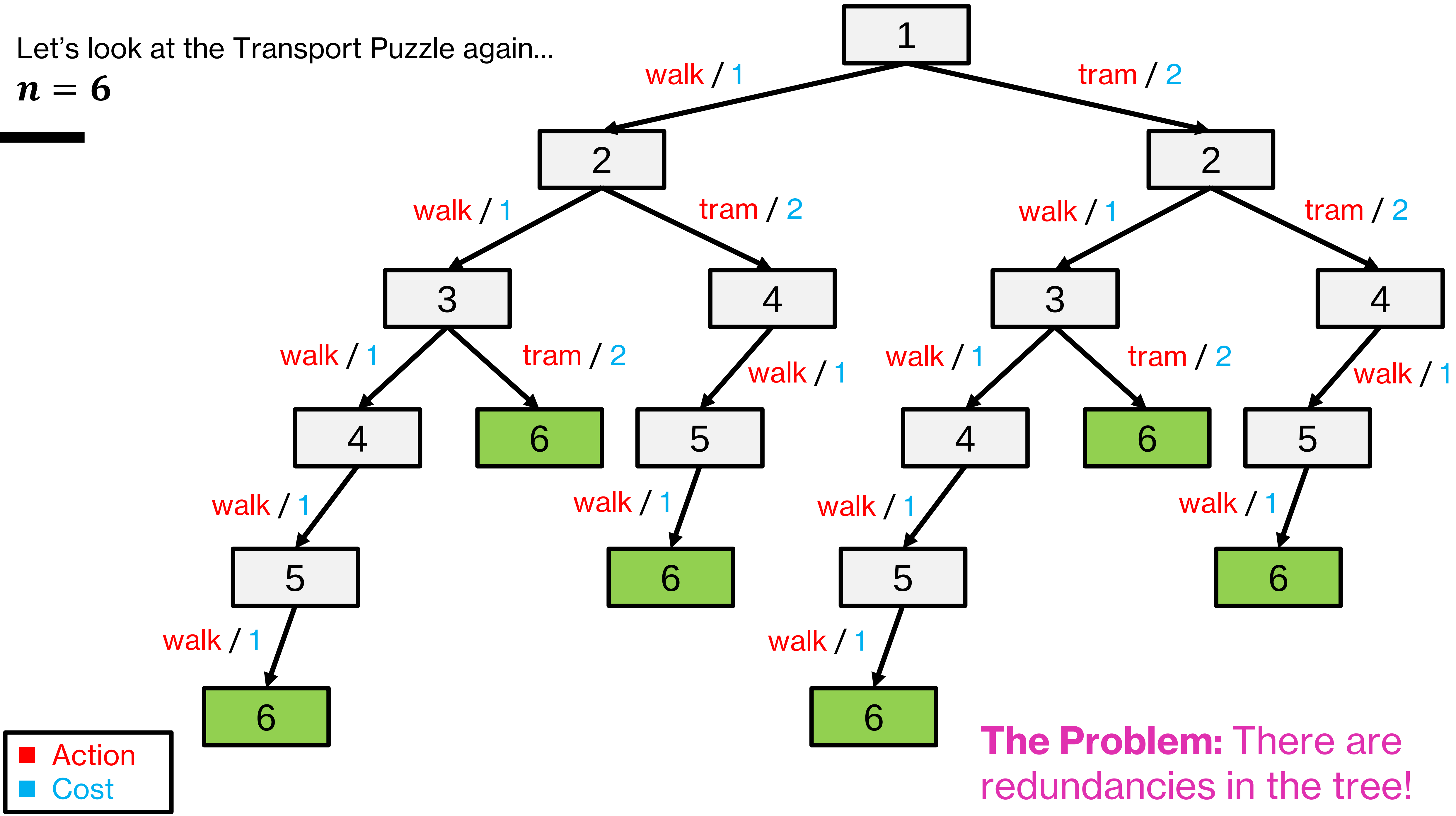


Action
Cost

Can we improve on the backtracking solution?

Let's look at the Transport Puzzle again…

$n = 6$



The Problem: There are redundancies in the tree!

Dynamic Programming

- Programming paradigm in which **solutions to subproblems are stored so they can easily be retrieved later**.
- Goal: reduce the number of states that must be explored (to reduce the execution time)

$$MinCost(s) = \begin{cases} 0 & IsEnd(s) \\ \min\limits_{a \in Actions} Cost(s,a) + MinCost(Succ(s,a)) & otherwise \end{cases}$$

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6
					0

- If you are already at the goal, then the minimum cost from the current state to the goal is 0 (base case).

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6
				1	0

- MinCost(5) = 1 + MinCost(6)
  = 1 + 0
  = 1

From block 5, no choice but to walk.

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6
			2	1	0

- MinCost(4) = 1 + MinCost(5)

  = 1 + 1

  = 2

From block 4, no choice but to walk.

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6
		2	2	1	0

- MinCost(3) = min(1 + MinCost(4), 2 + MinCost(6))
  = min(1 + 2, 2 + 0)
  = min(3, 2) = 2

From block 3, it's better to take the tram

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6
	3	2	2	1	0

- MinCost(2) = min(1 + MinCost(3), 2 + MinCost(4))
  = min(1 + 2, 2 + 2)
  = min(3, 4) = 3

From block 2, it's better to walk

Dynamic Programming in the Transport Puzzle ($n = 6$)

MinCost cache

1	2	3	4	5	6
4	3	2	2	1	0

- MinCost(1) = min(1 + MinCost(2), 2 + MinCost(2))

  = min(1 + 3, 2 + 3)

  = min(4, 5) = 4

From block 1, it's better to walk

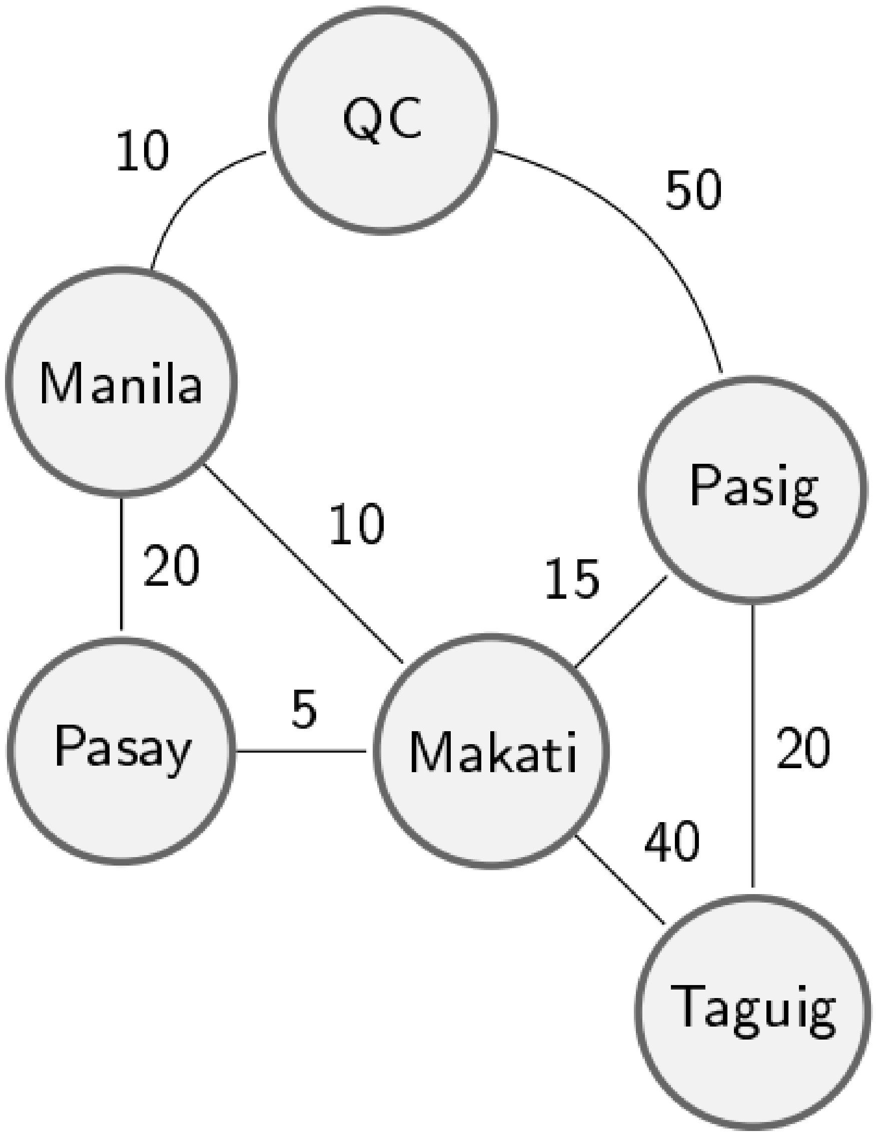
Performance Gains

Total number of states to explore (Transport Puzzle)

n	Backtracking	Dynamic Programming
6	19	6
10	59	10
50	9827	50
100	205657	100
1000	7389571	1000

Routing Problem

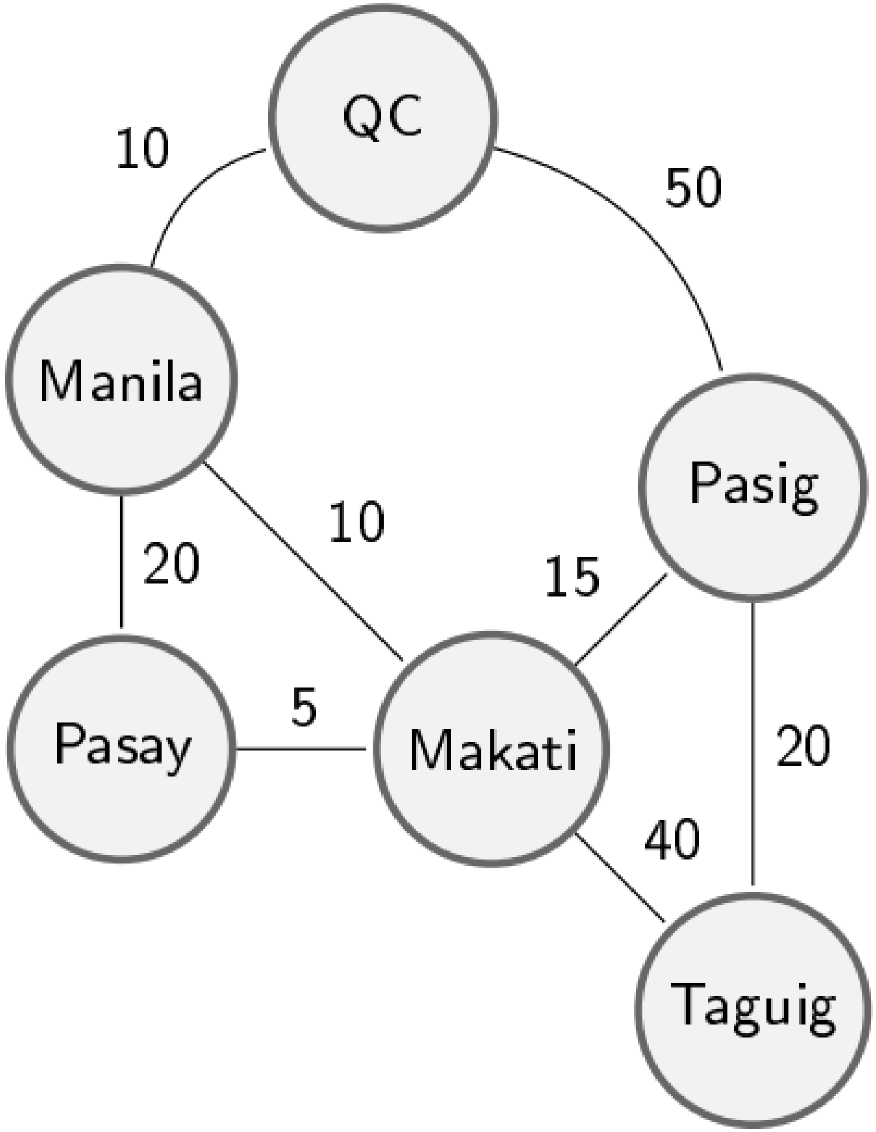
- Start at QC
- Want to travel to Taguig
- Want to take the path with the least traffic
- The weights of the edges in the graph represent the amount of traffic between the cities
- **Can dynamic programming handle this problem?**



Routing Problem

- Dynamic programming cannot handle this because there are cycles in the state transitions!

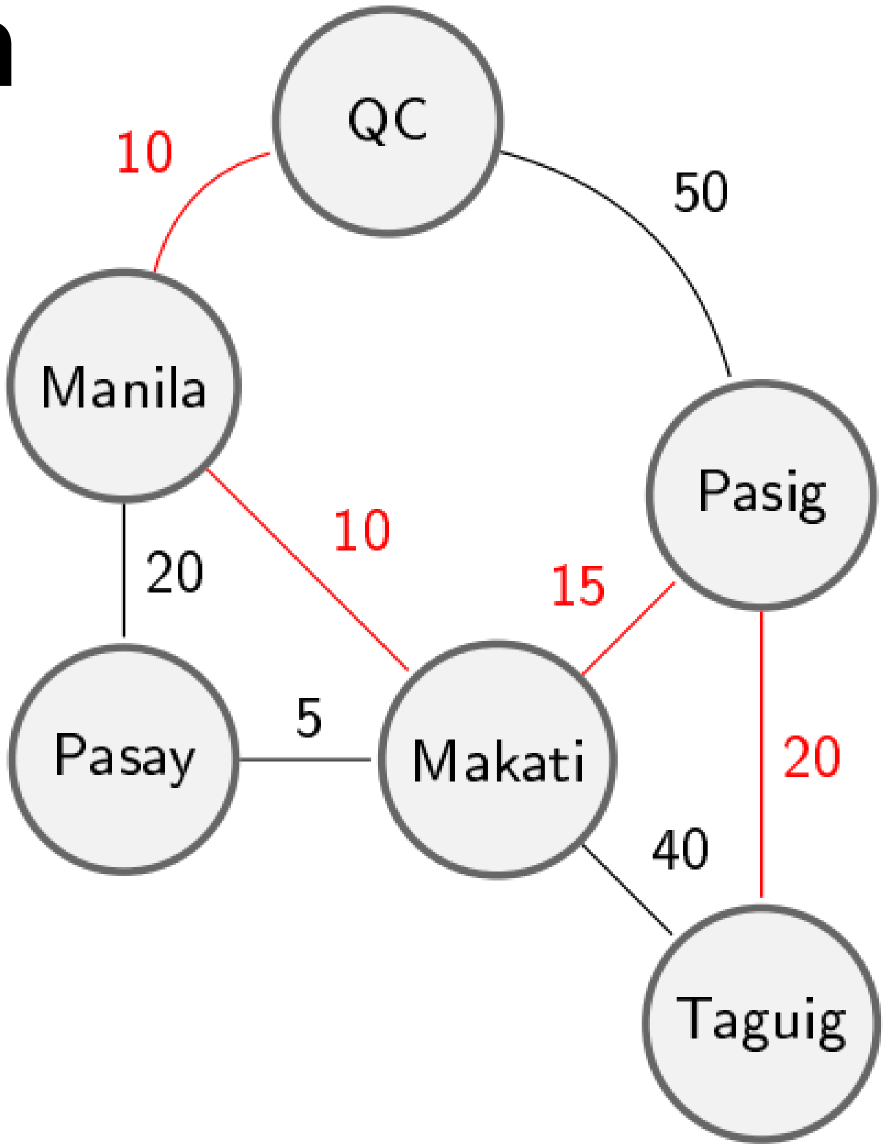
- *(from Pasay you can go to Makati, but from Makati you can also go to Pasay, and either one may come first)*



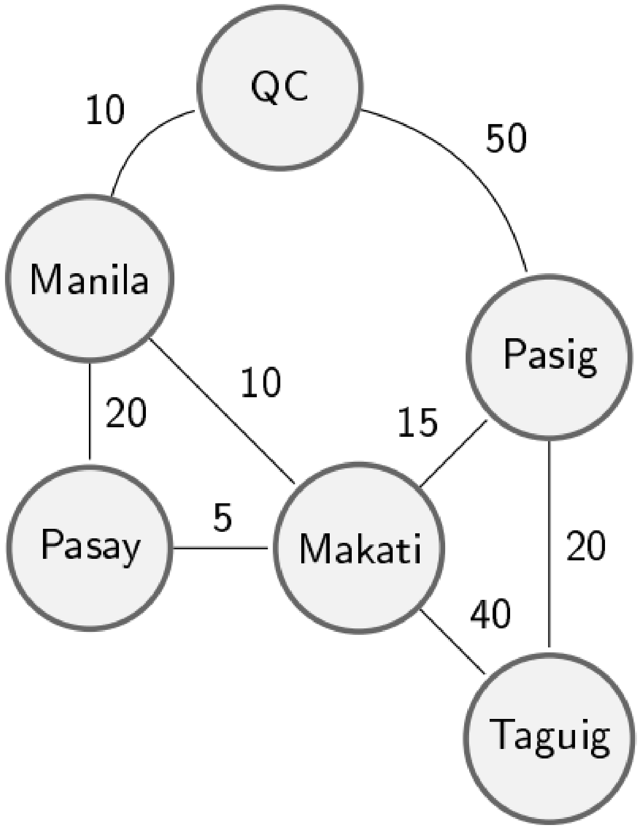
Uniform Cost Search

- **Observation:** Prefixes of the optimal path are also optimal.

- Key Idea: maintain a frontier list, and **always explore the state with the lowest cost from the start state**.



Uniform Cost Search Example

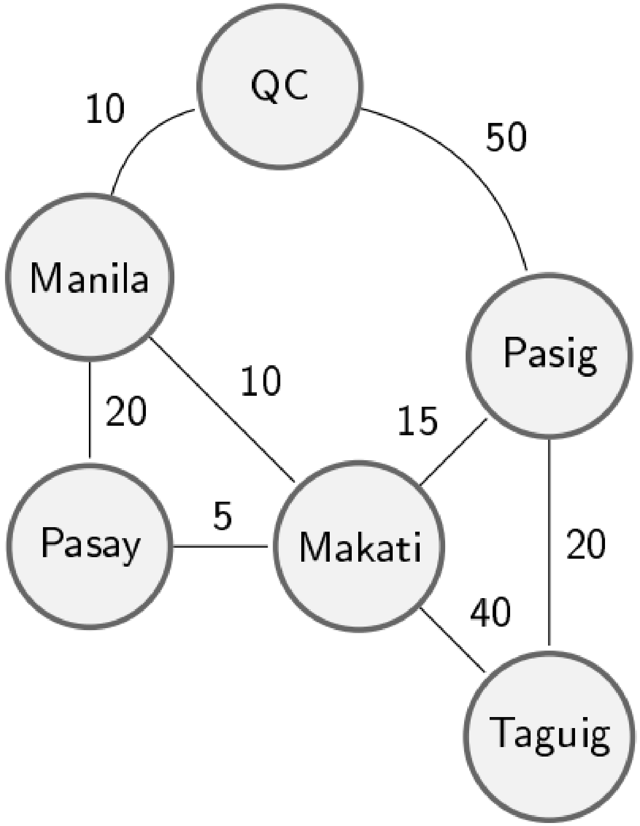


FRONTIER

QC (0)

EXPLORED

Uniform Cost Search Example

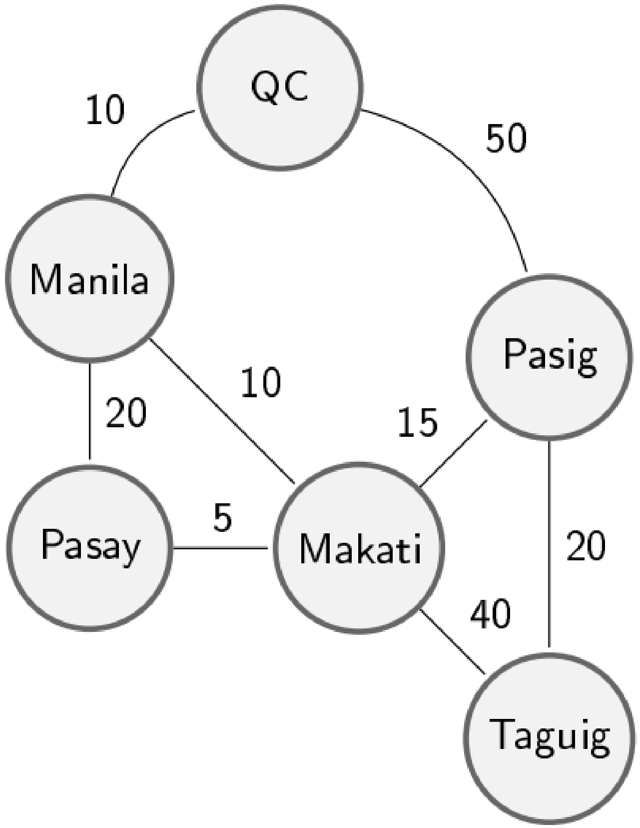


FRONTIER

QC (0)

EXPLORED

Uniform Cost Search Example



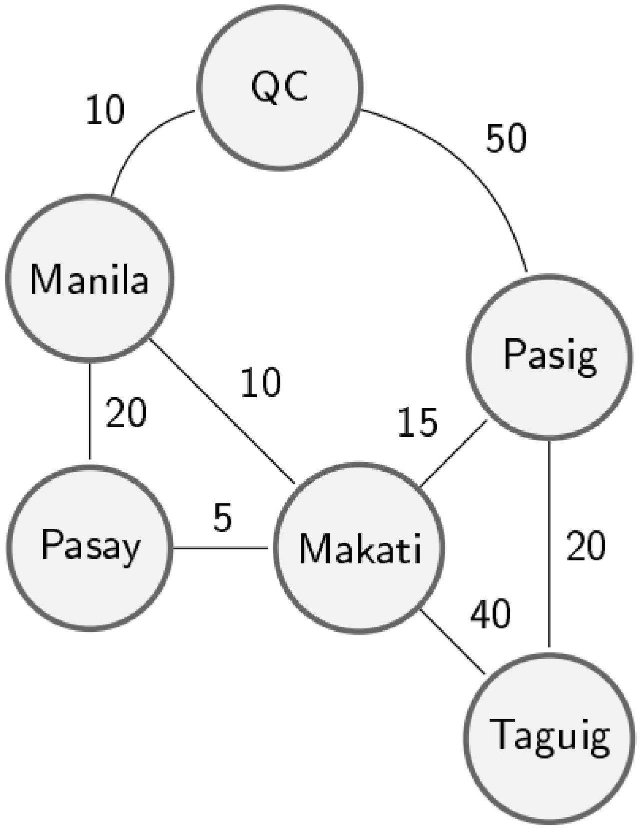
FRONTIER

Manila (10 from QC)
Pasig (50 from QC)

EXPLORED

QC (0)

Uniform Cost Search Example



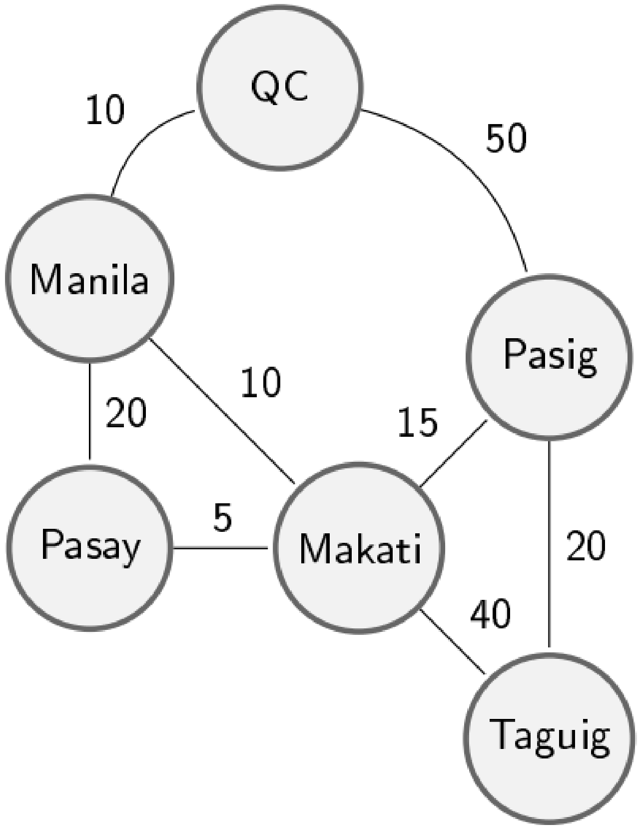
FRONTIER

Manila (10 from QC)
Pasig (50 from QC)

EXPLORED

QC (0)

Uniform Cost Search Example



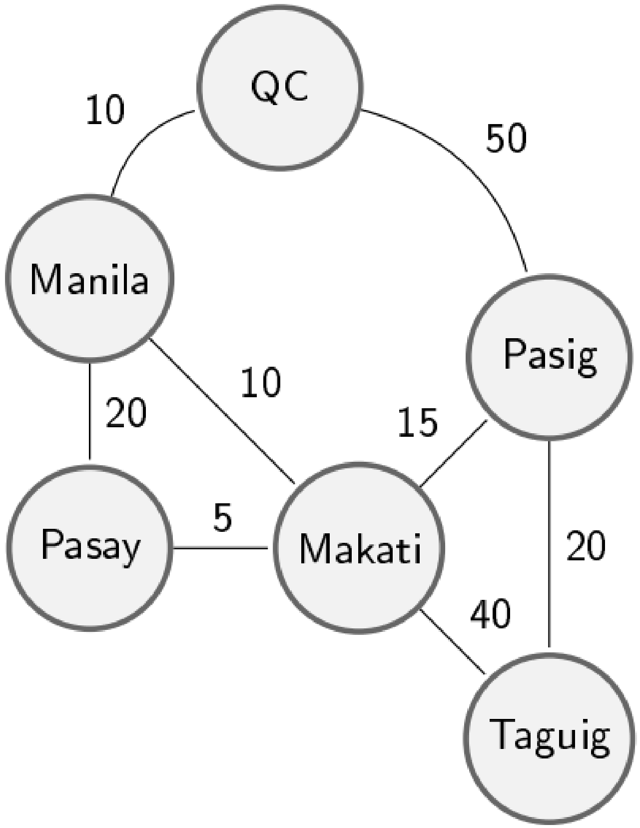
FRONTIER

Pasig (50 from QC)
Pasay (30 from Manila)
Makati (20 from Manila)

EXPLORED

QC (0)
Manila (10 from QC)

Uniform Cost Search Example



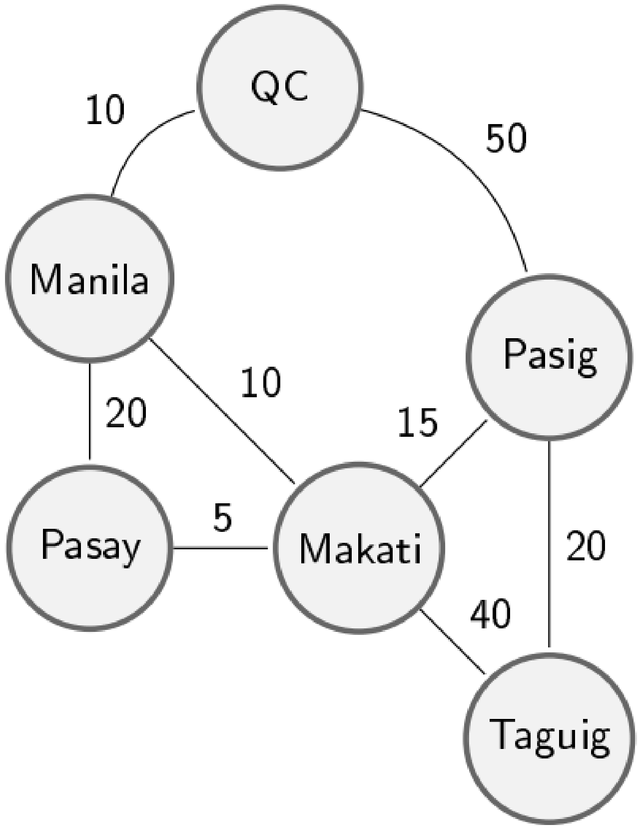
FRONTIER

Pasig (50 from QC)
Pasay (30 from Manila)
Makati (20 from Manila)

EXPLORED

QC (0)
Manila (10 from QC)

Uniform Cost Search Example



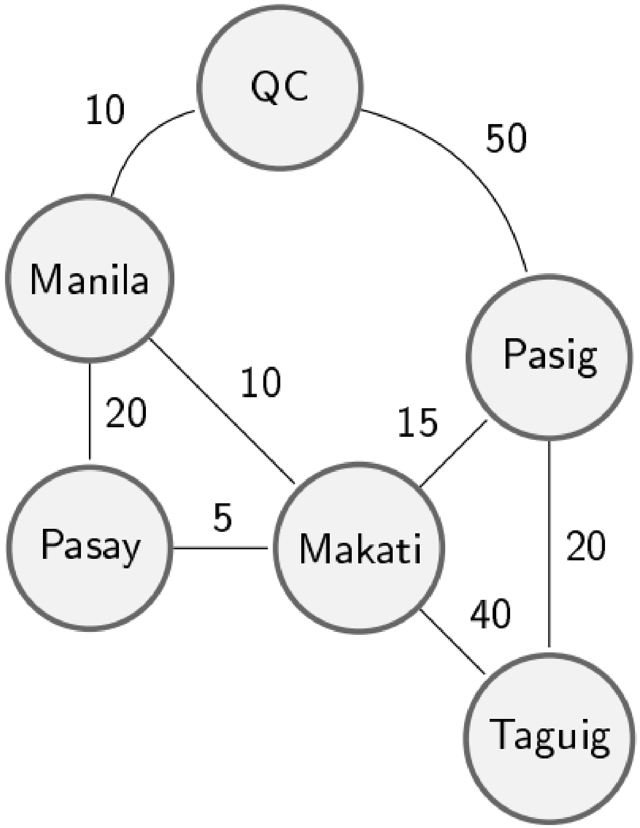
FRONTIER

Pasay (25 from Makati)
Pasig (35 from Makati)
Taguig (60 from Makati)

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)

Uniform Cost Search Example



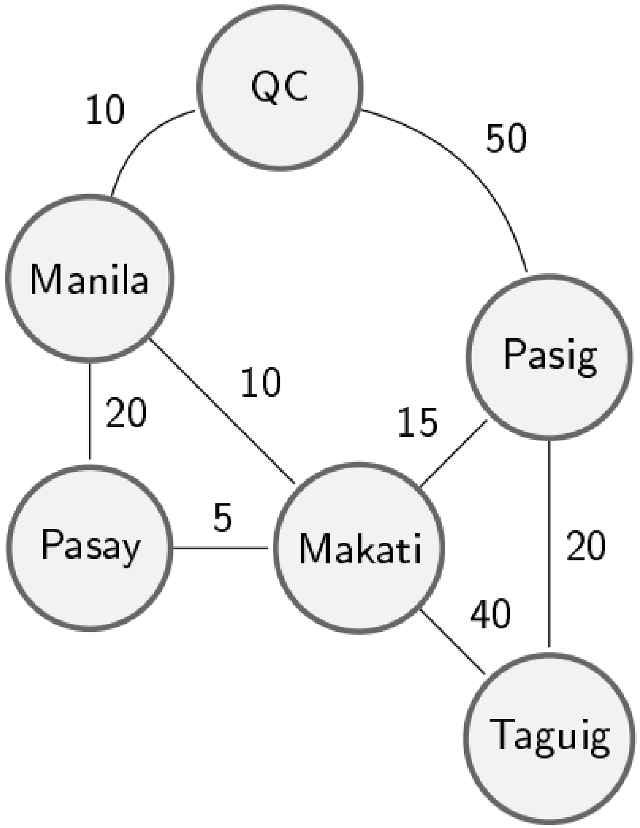
FRONTIER

Pasay (25 from Makati)
Pasig (35 from Makati)
Taguig (60 from Makati)

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)

Uniform Cost Search Example



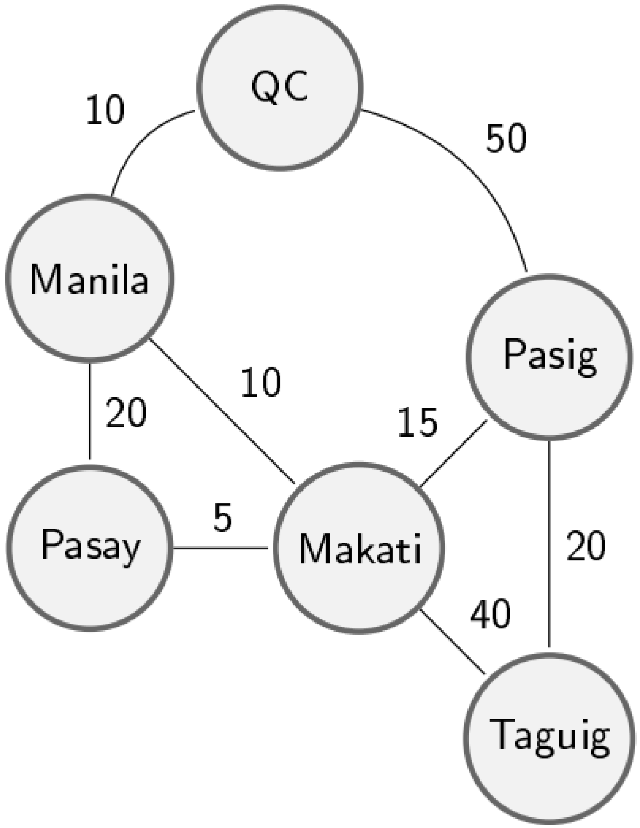
FRONTIER

Pasig (35 from Makati)
Taguig (60 from Makati)

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)
Pasay (25 from Makati)

Uniform Cost Search Example



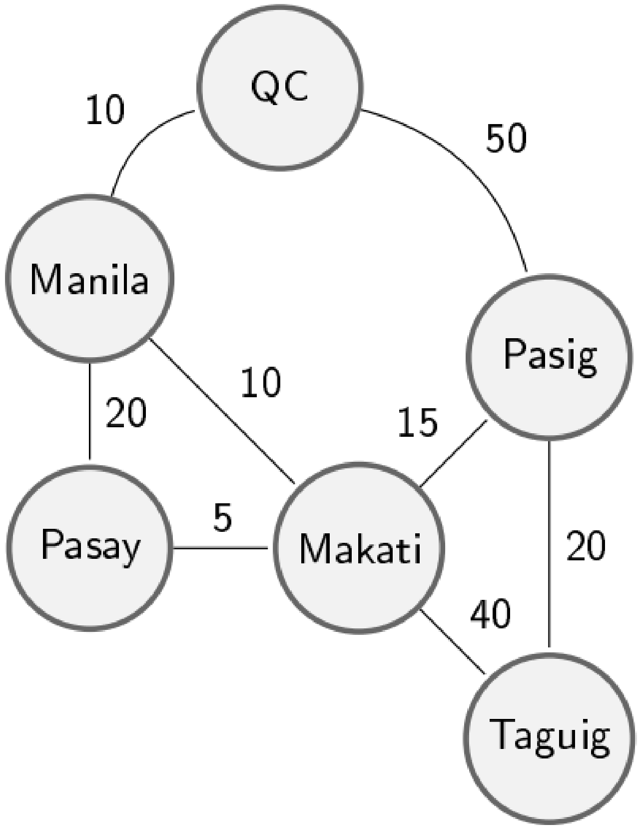
FRONTIER

Pasig (35 from Makati)
Taguig (60 from Makati)

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)
Pasay (25 from Makati)

Uniform Cost Search Example



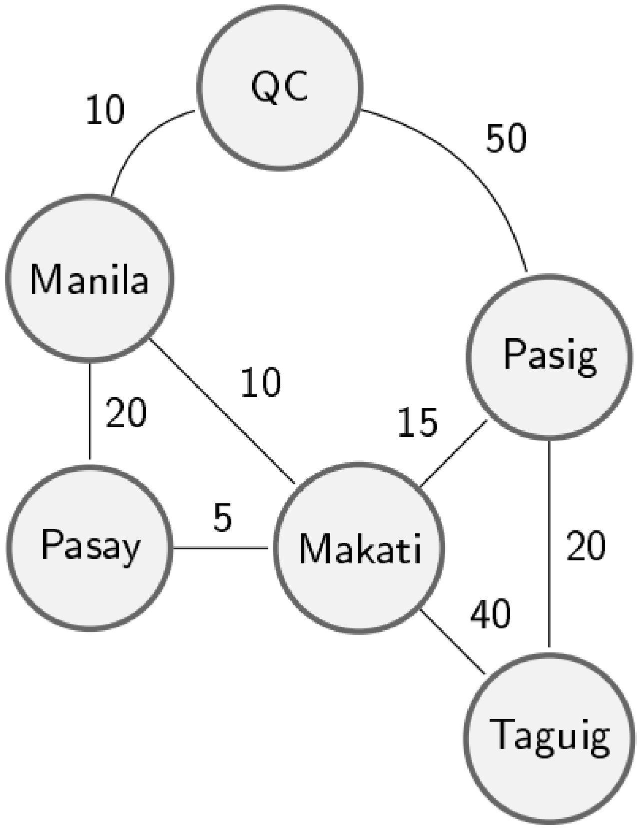
FRONTIER

Taguig (55 from Pasig)

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)
Pasay (25 from Makati)
Pasig (35 from Makati)

Uniform Cost Search Example



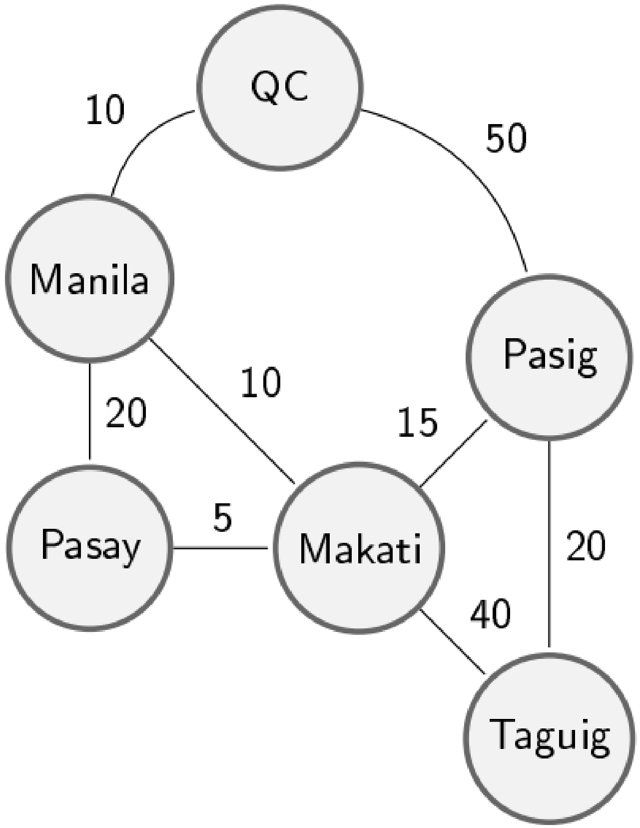
FRONTIER

Taguig (55 from Pasig)

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)
Pasay (25 from Makati)
Pasig (35 from Makati)

Uniform Cost Search Example

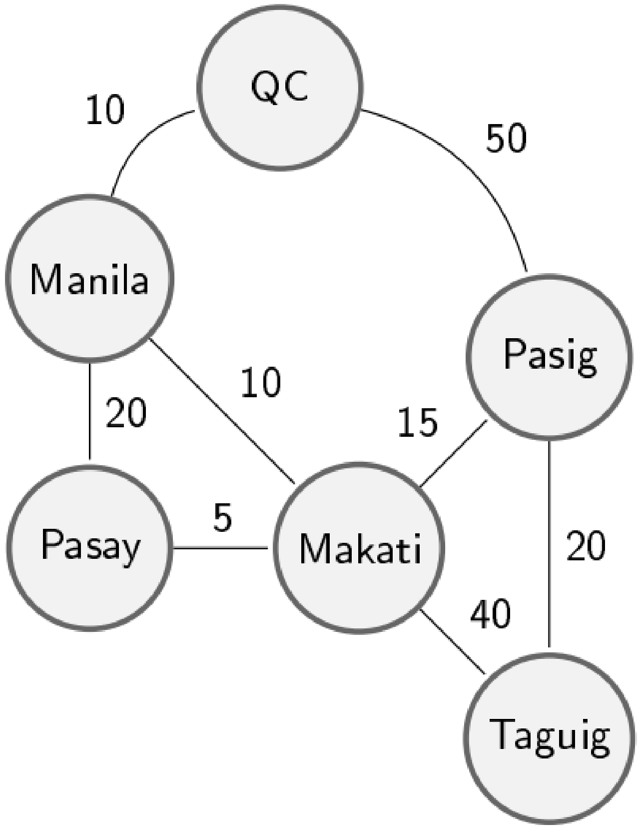


FRONTIER

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)
Pasay (25 from Makati)
Pasig (35 from Makati)
Taguig (55 from Pasig)

Uniform Cost Search Example



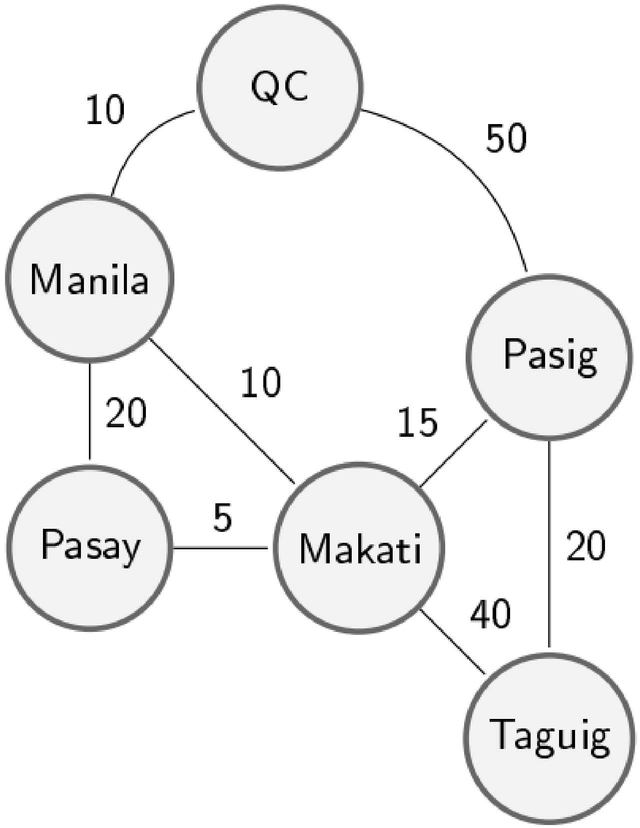
FRONTIER

EXPLORED

QC (0)
Manila (10 from QC)
Makati (20 from Manila)
Pasay (25 from Makati)
Pasig (35 from Makati)
Taguig (55 from Pasig)

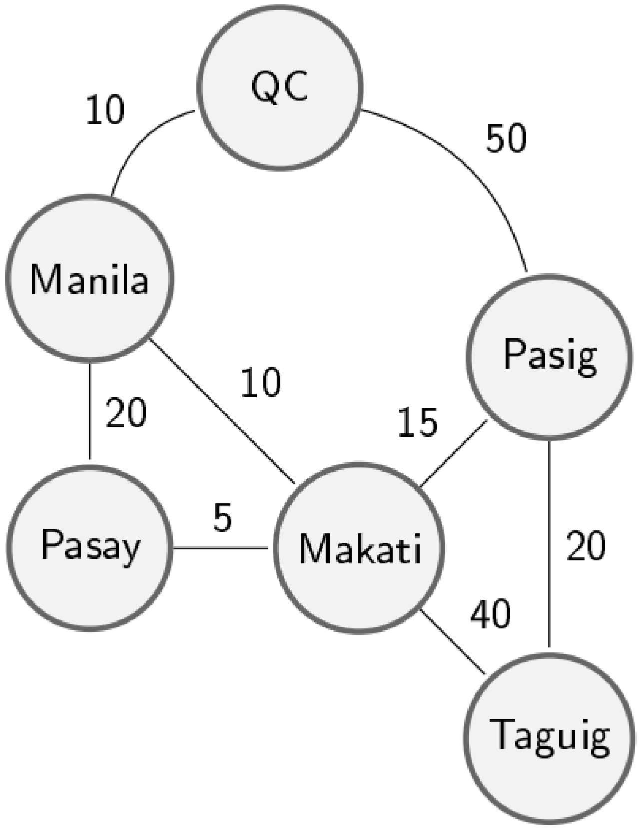
Uniform Cost Search Example

QC (0)

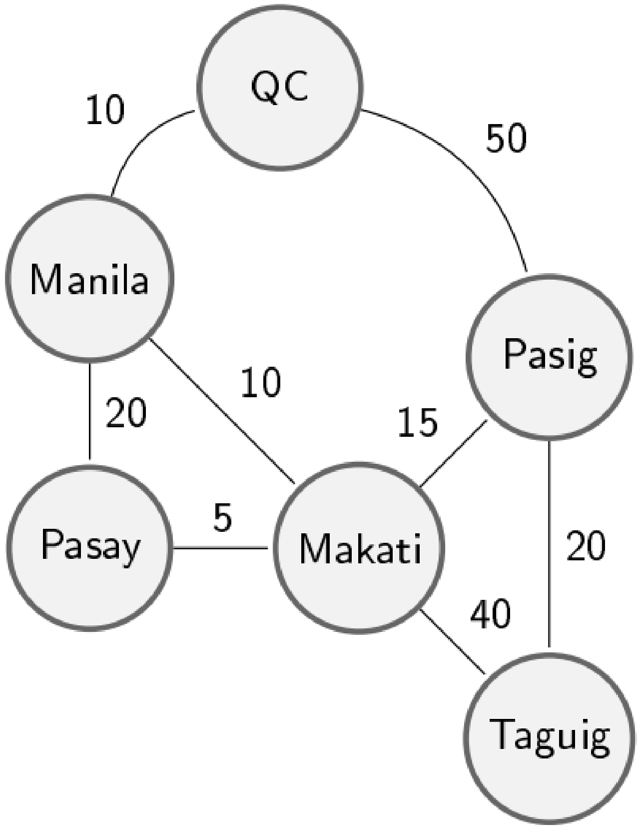


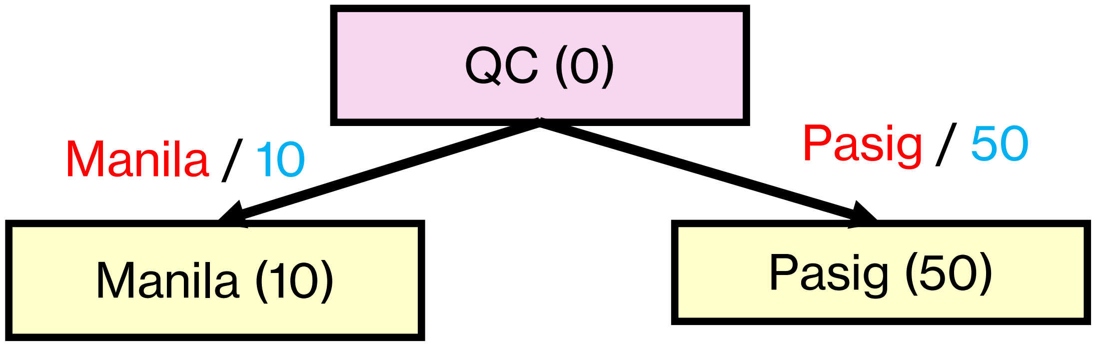
Uniform Cost Search Example

QC (0)

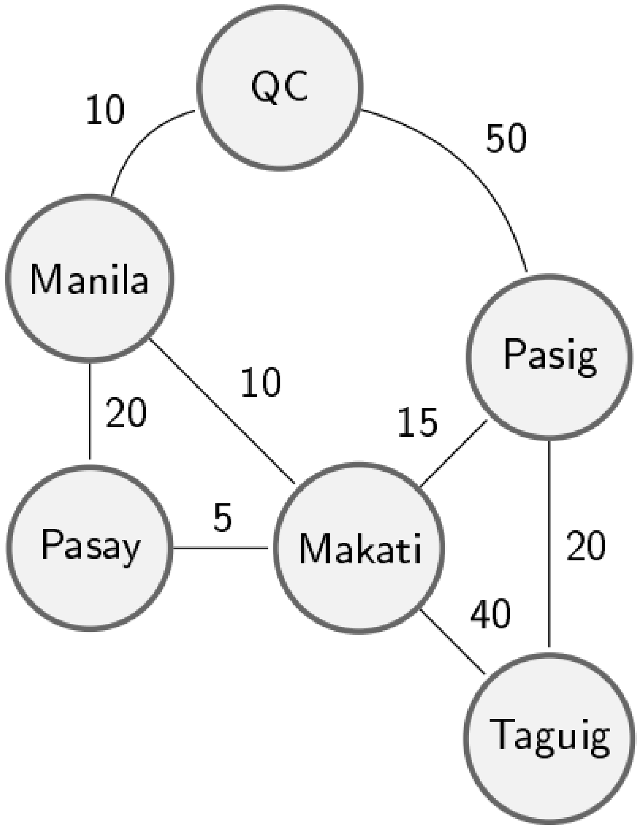


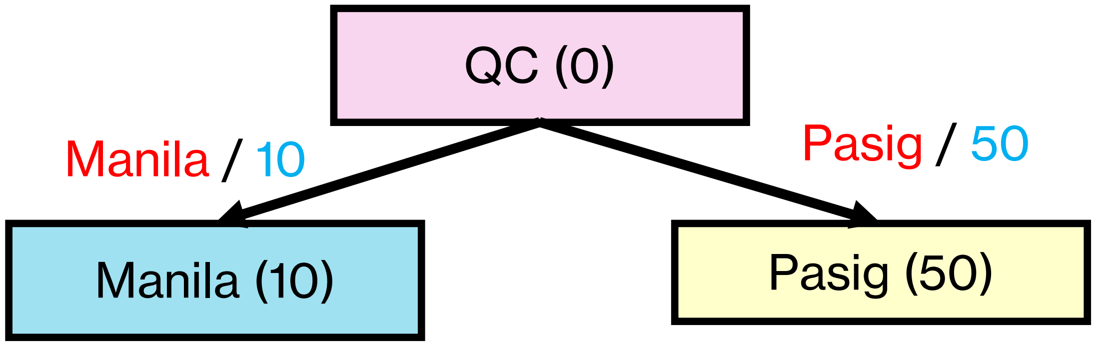
Uniform Cost Search Example



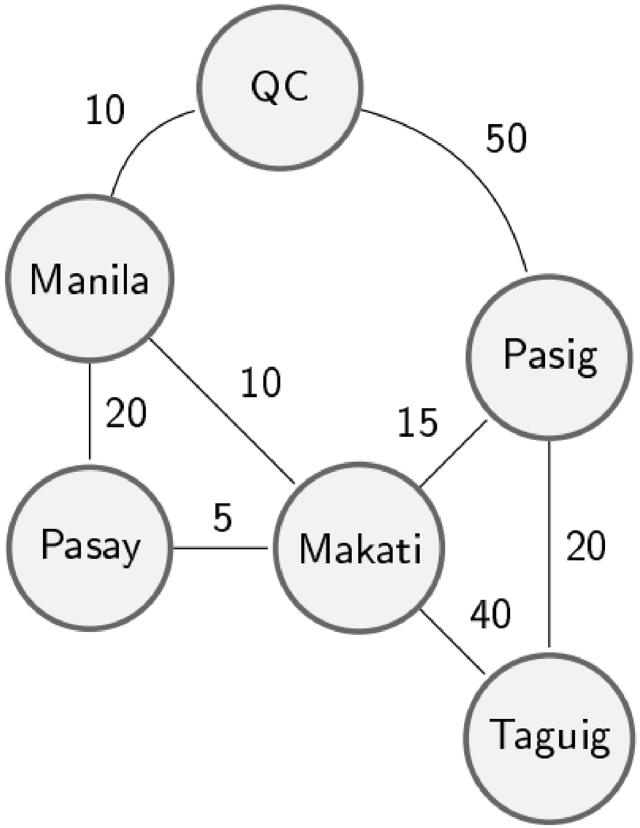


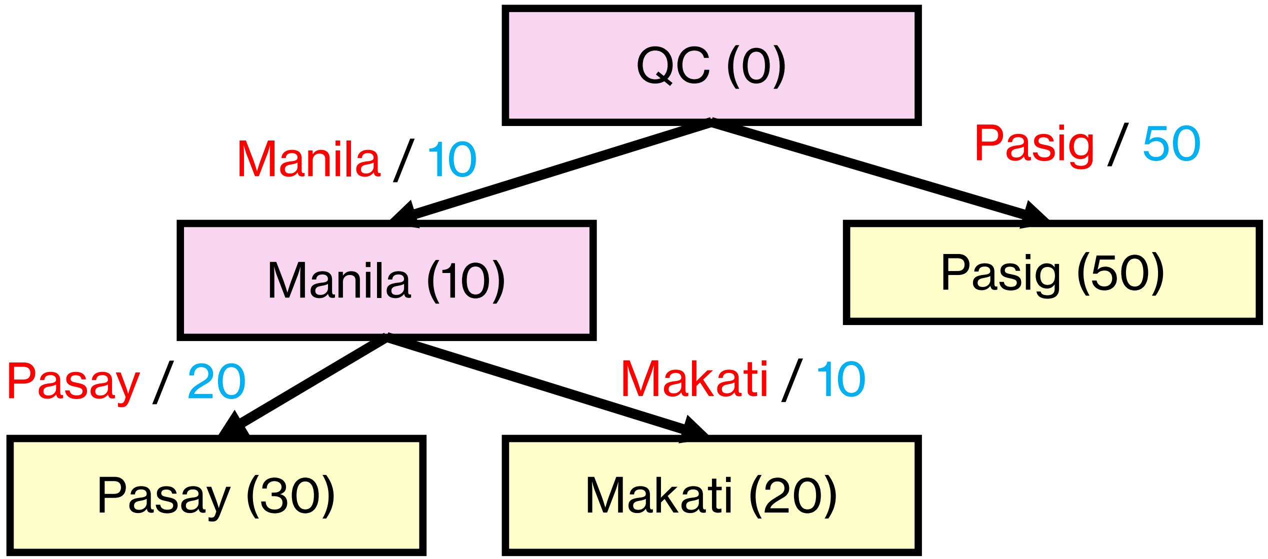
Uniform Cost Search Example



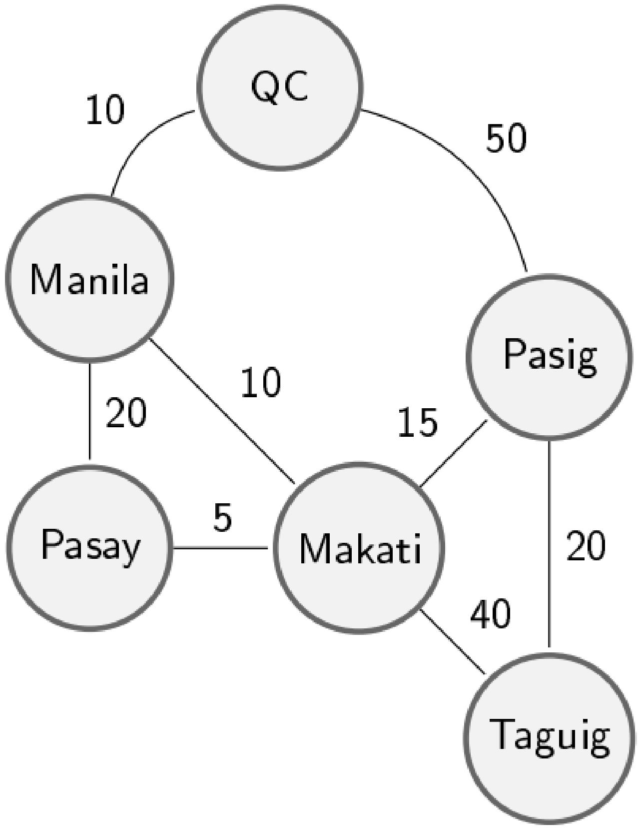


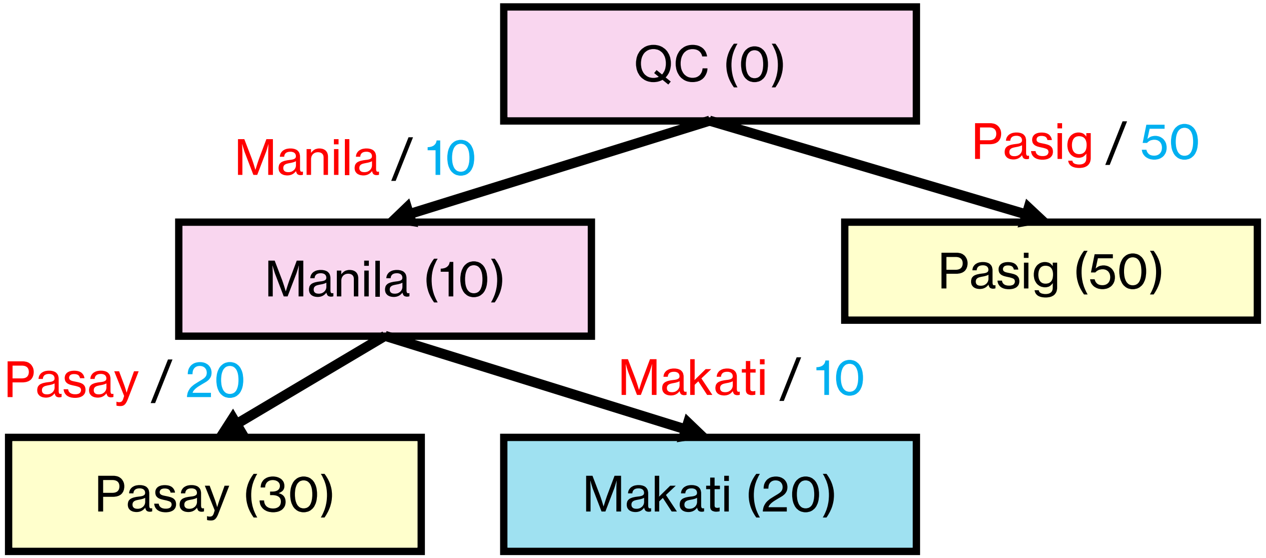
Uniform Cost Search Example



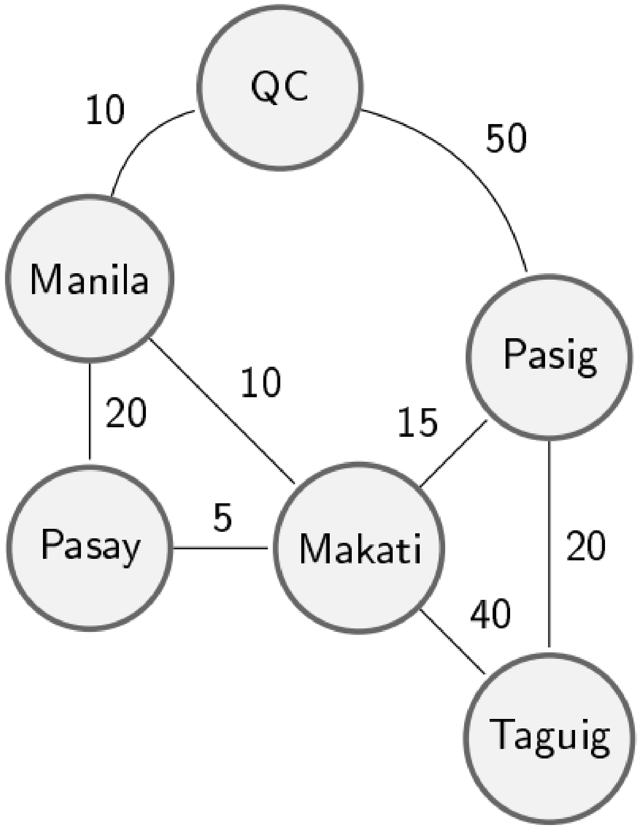


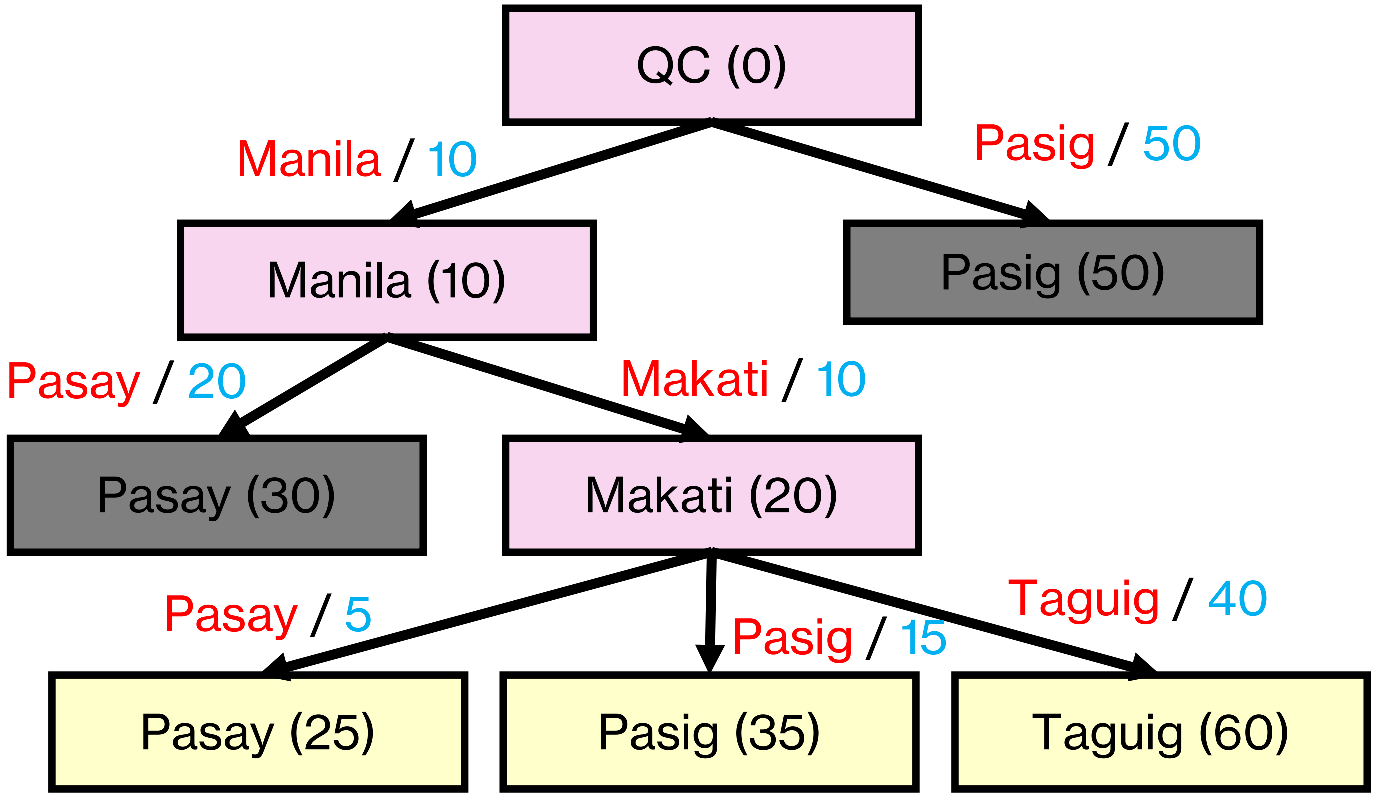
Uniform Cost Search Example



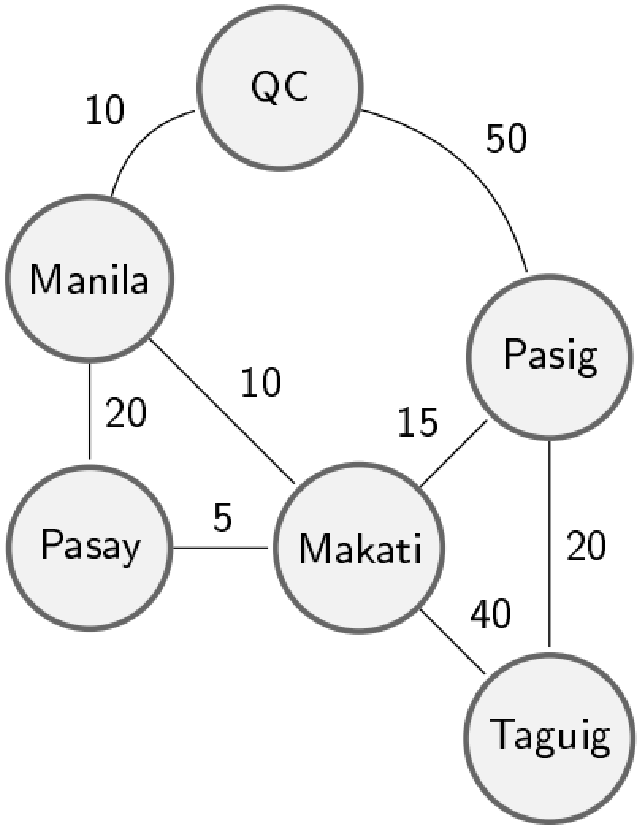


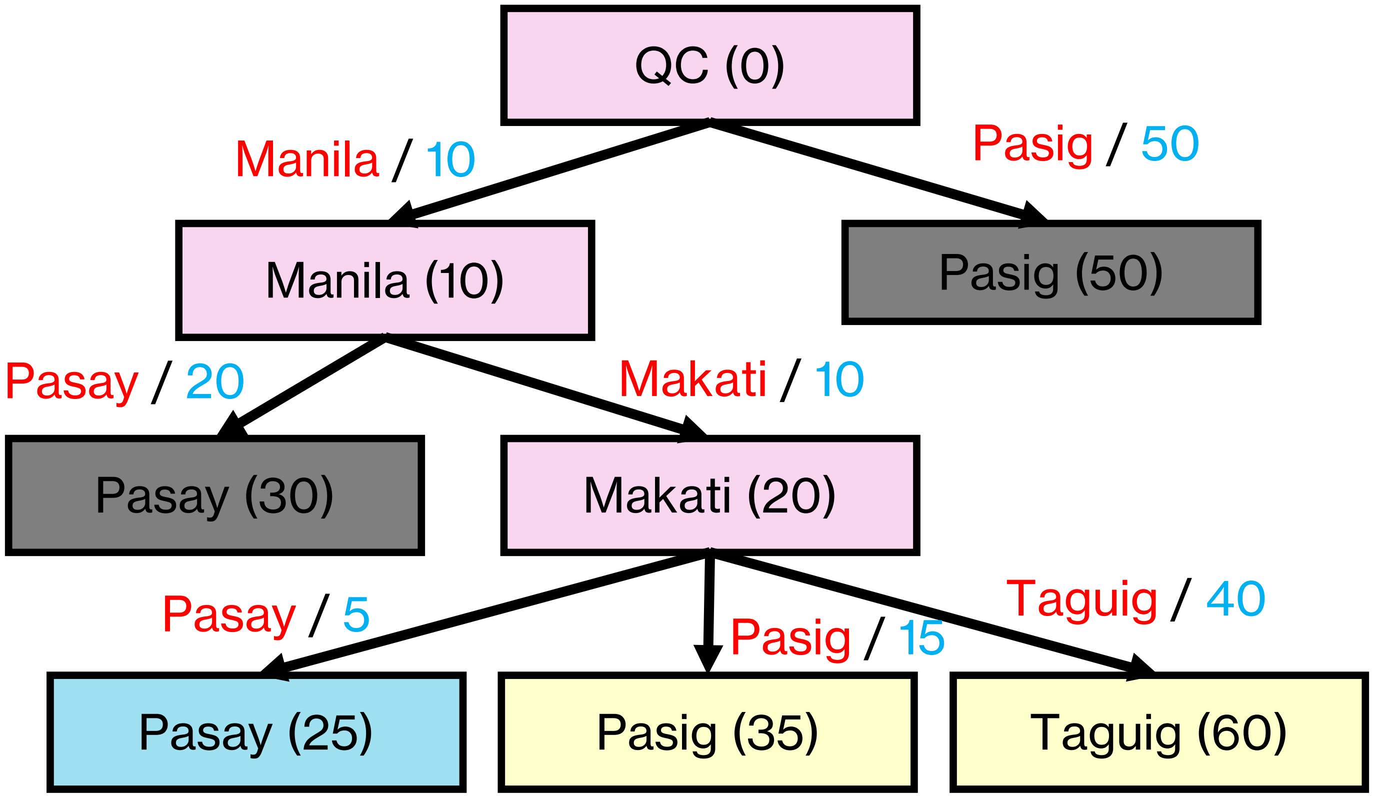
Uniform Cost Search Example



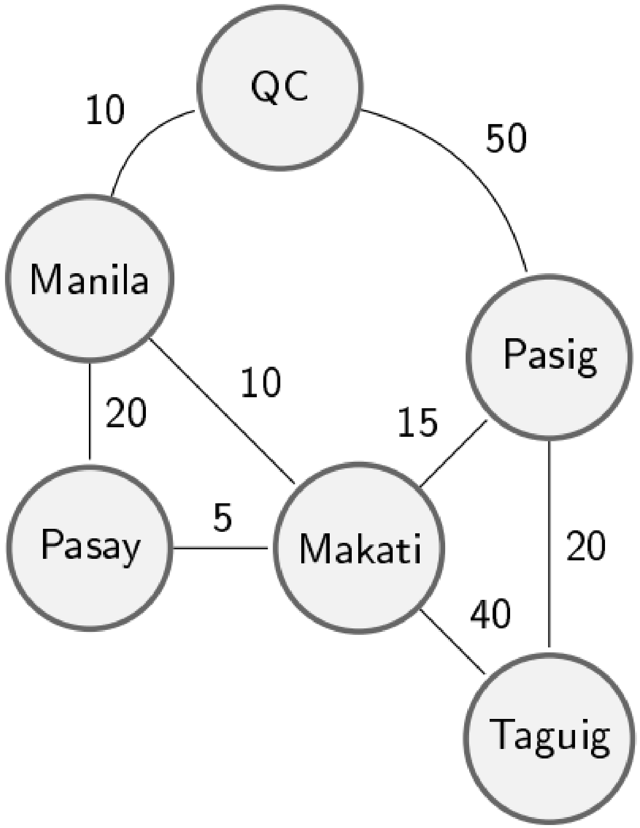


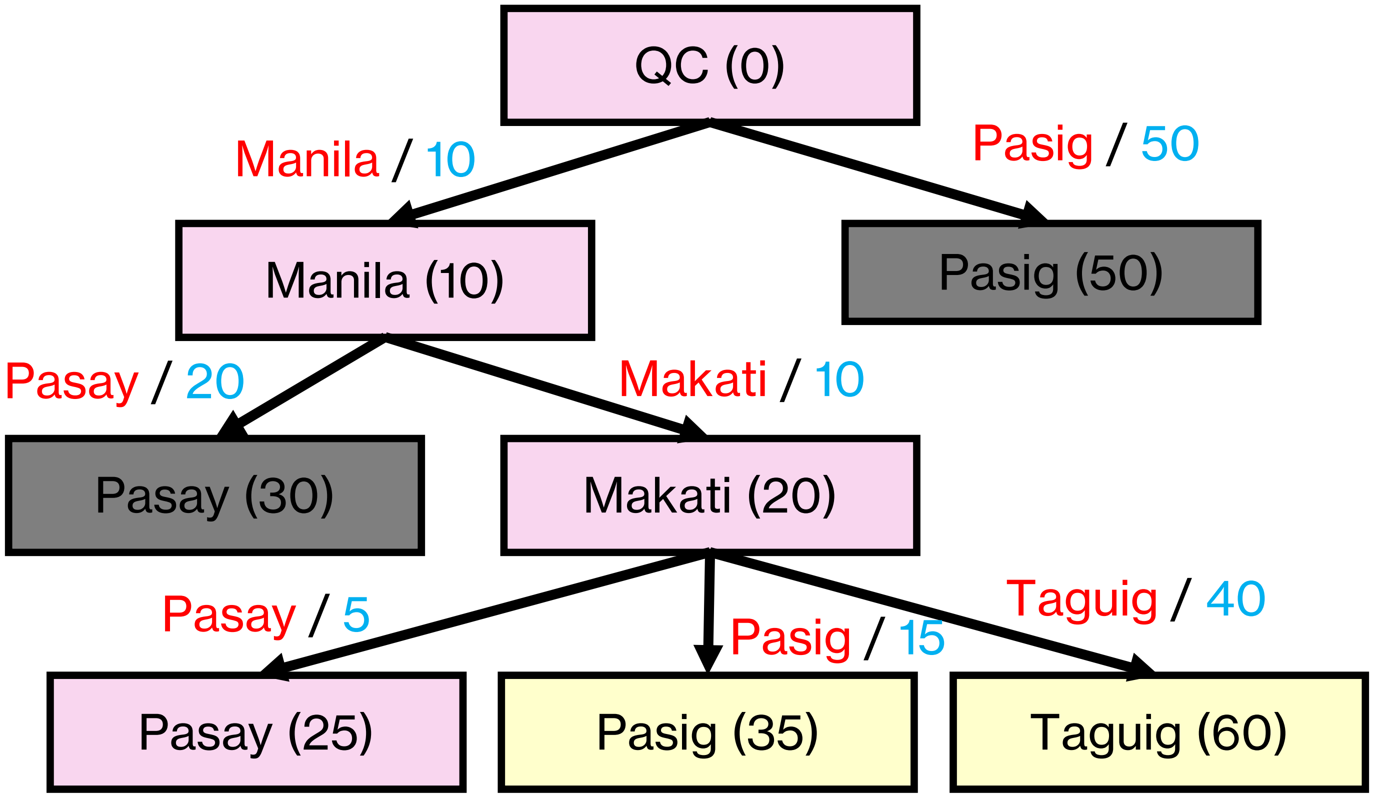
Uniform Cost Search Example



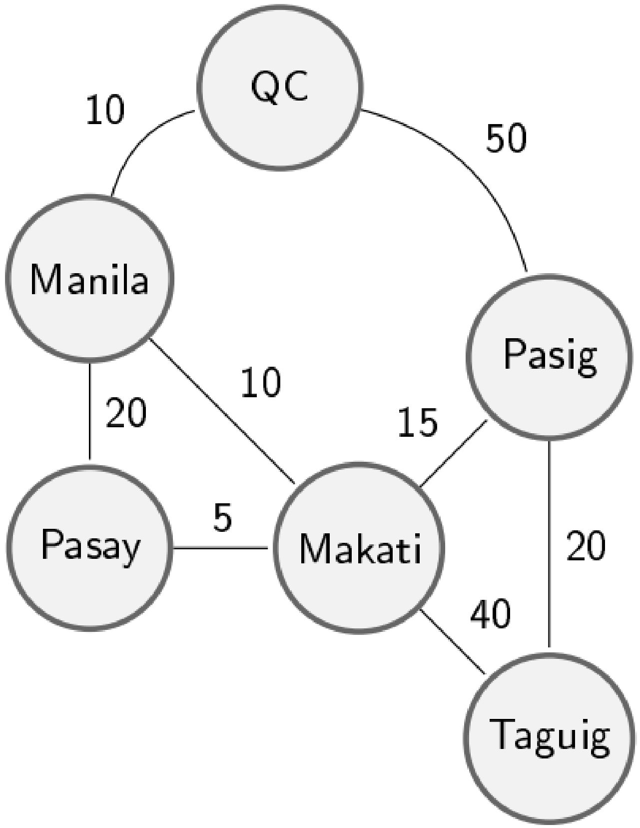


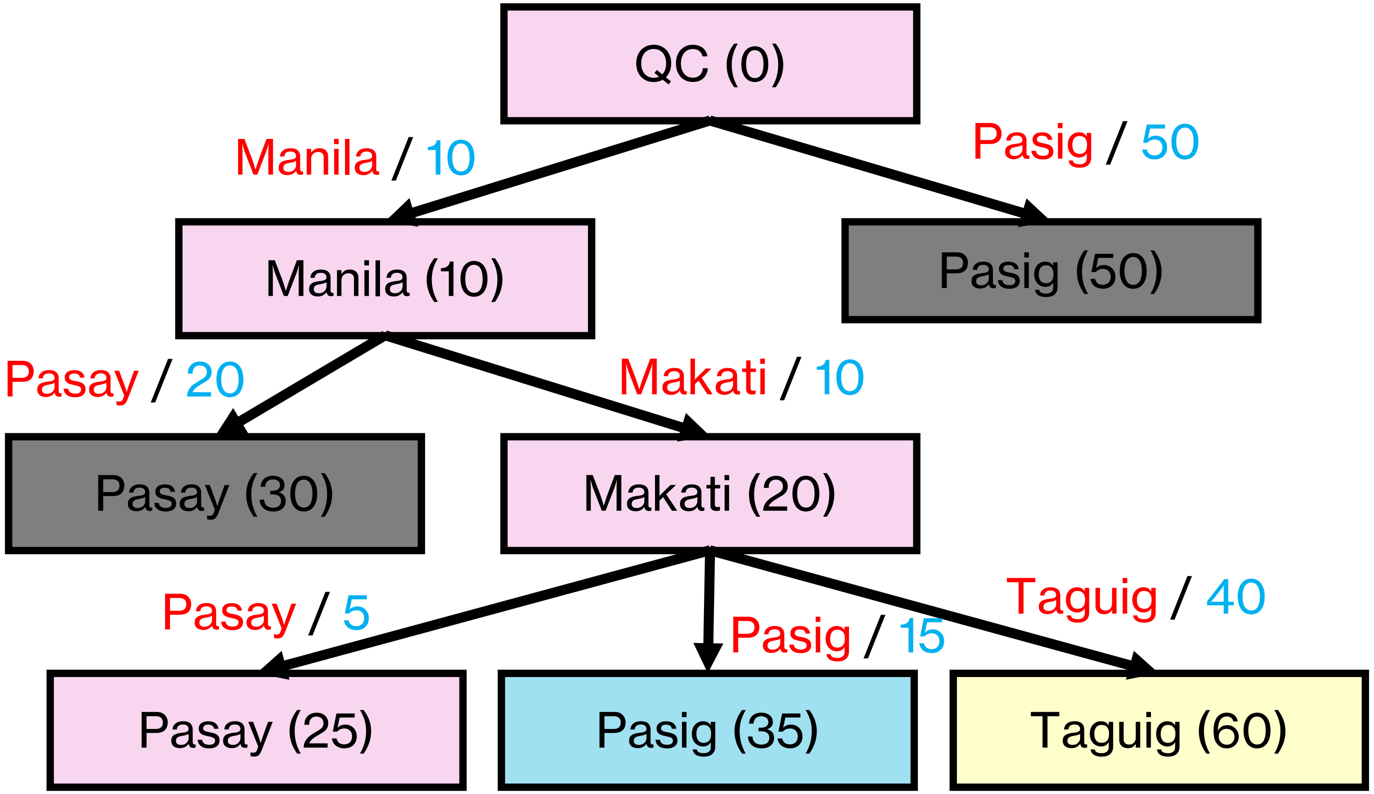
Uniform Cost Search Example



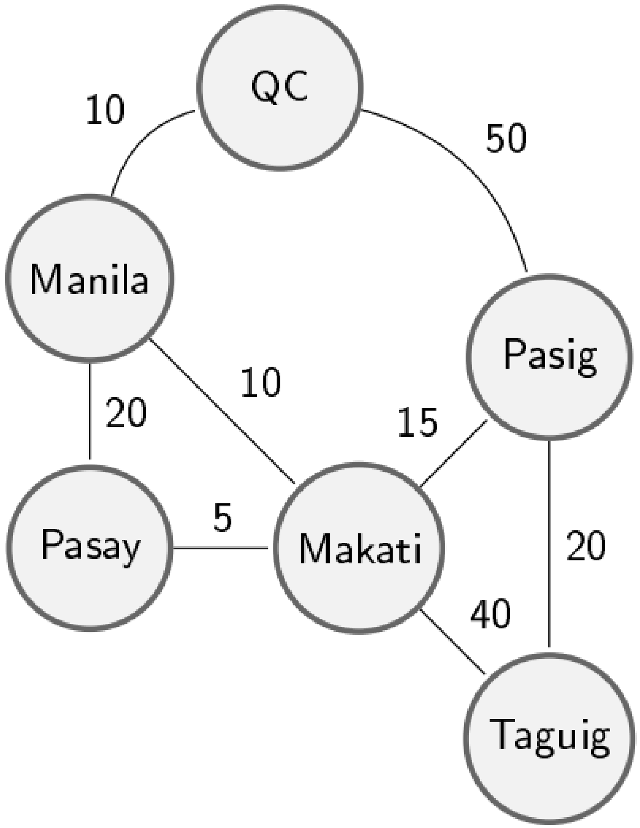


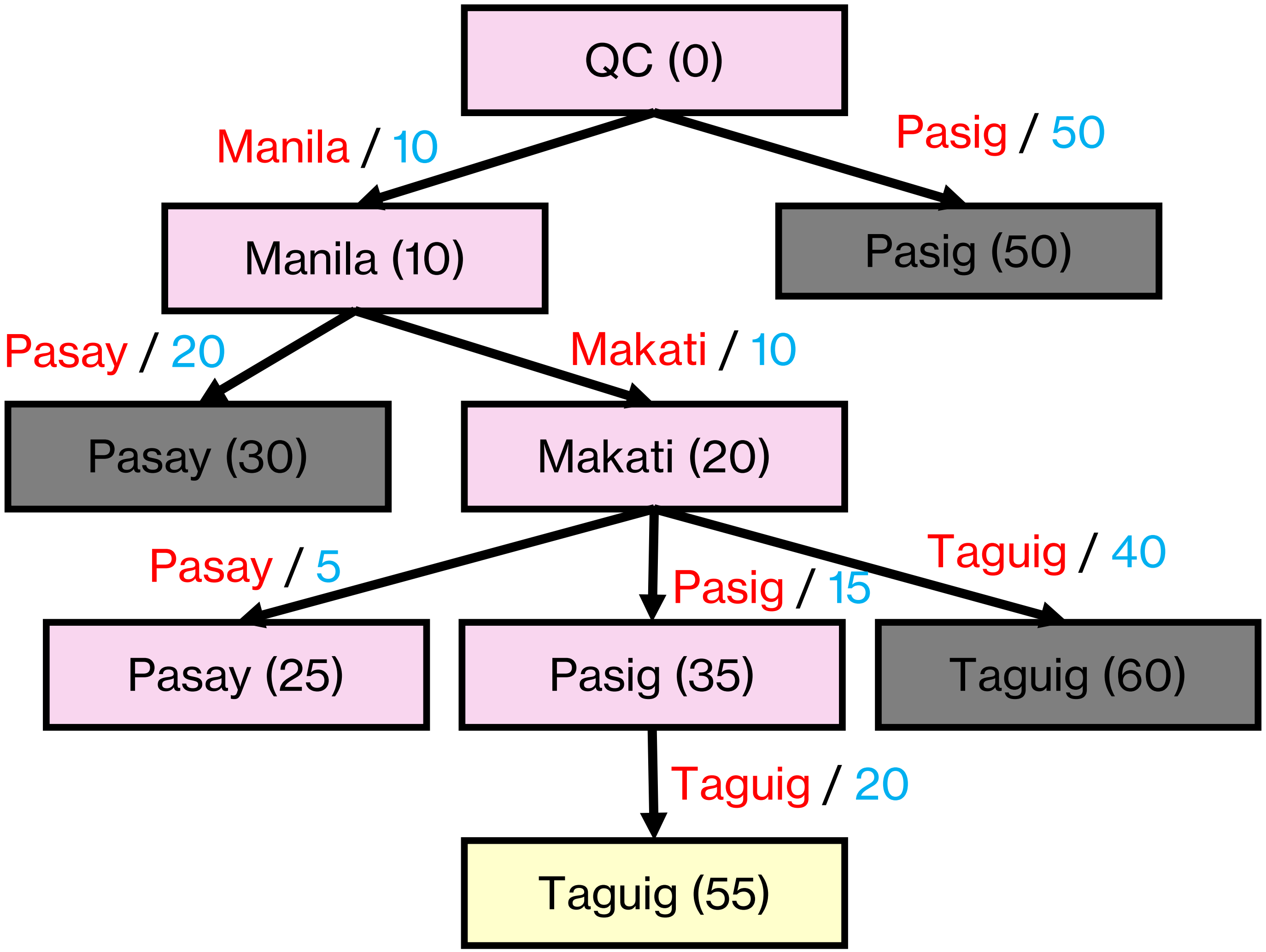
Uniform Cost Search Example



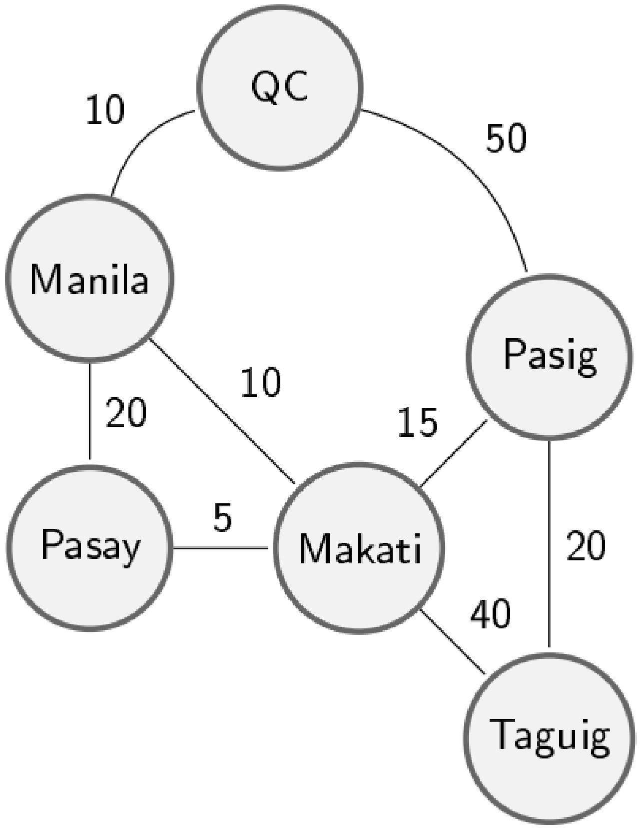


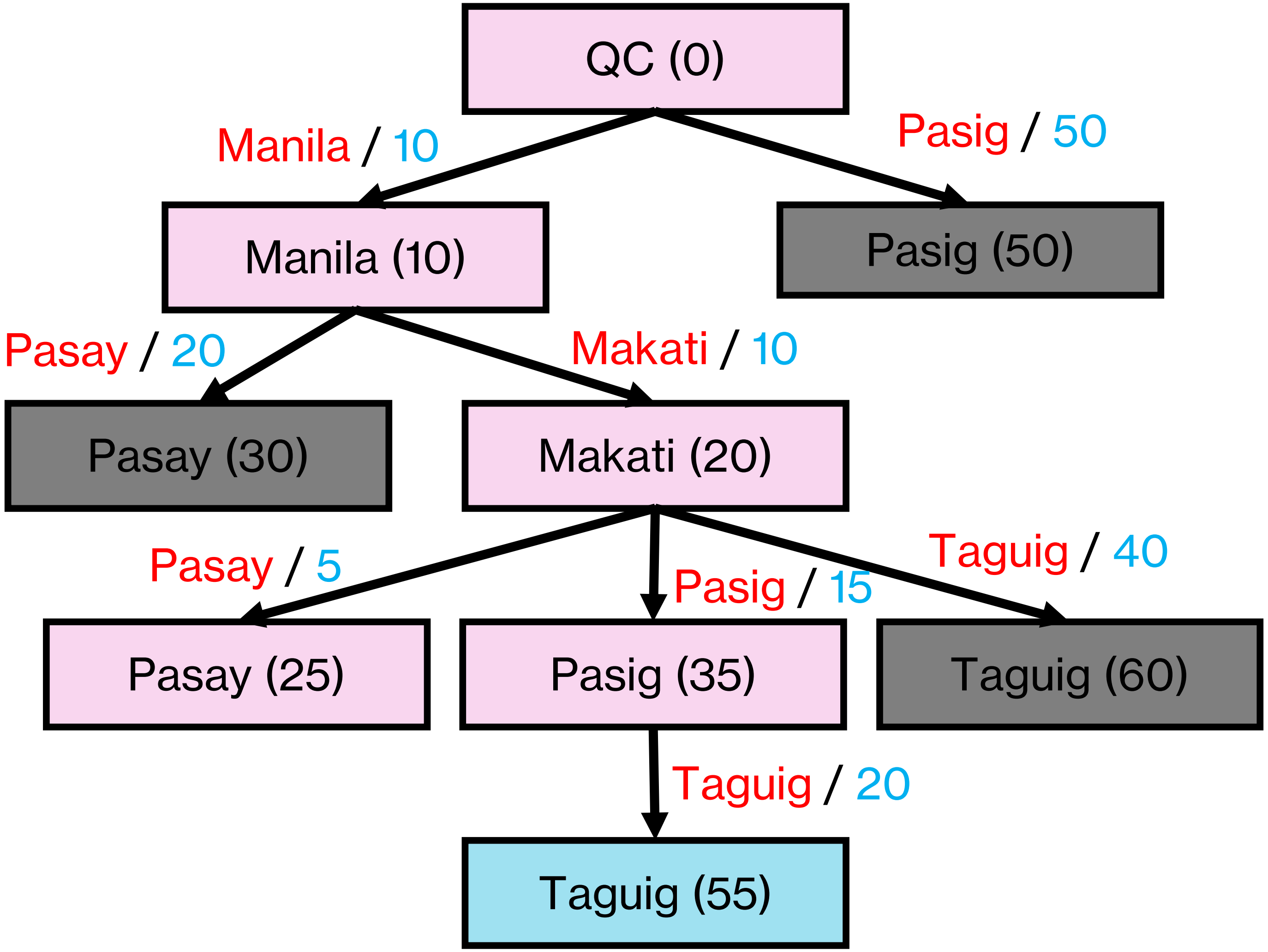
Uniform Cost Search Example



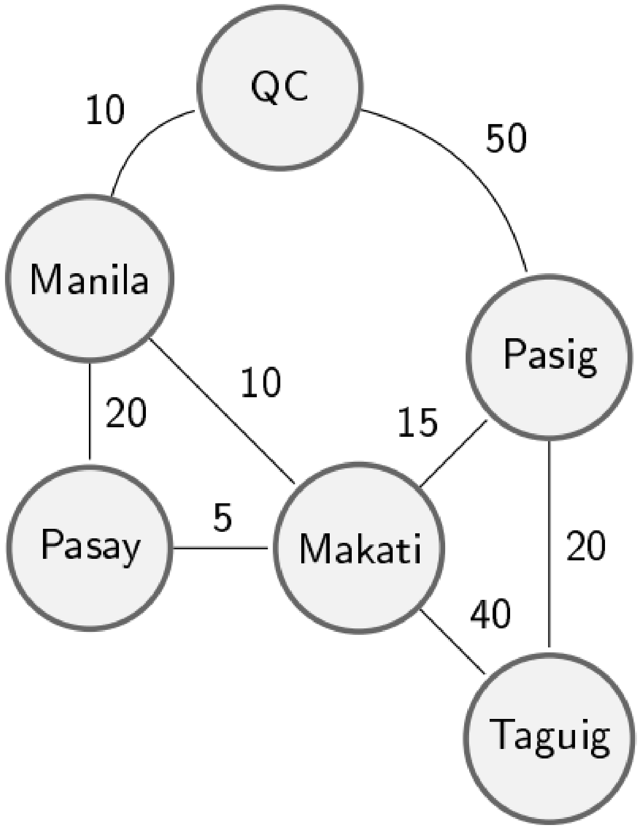


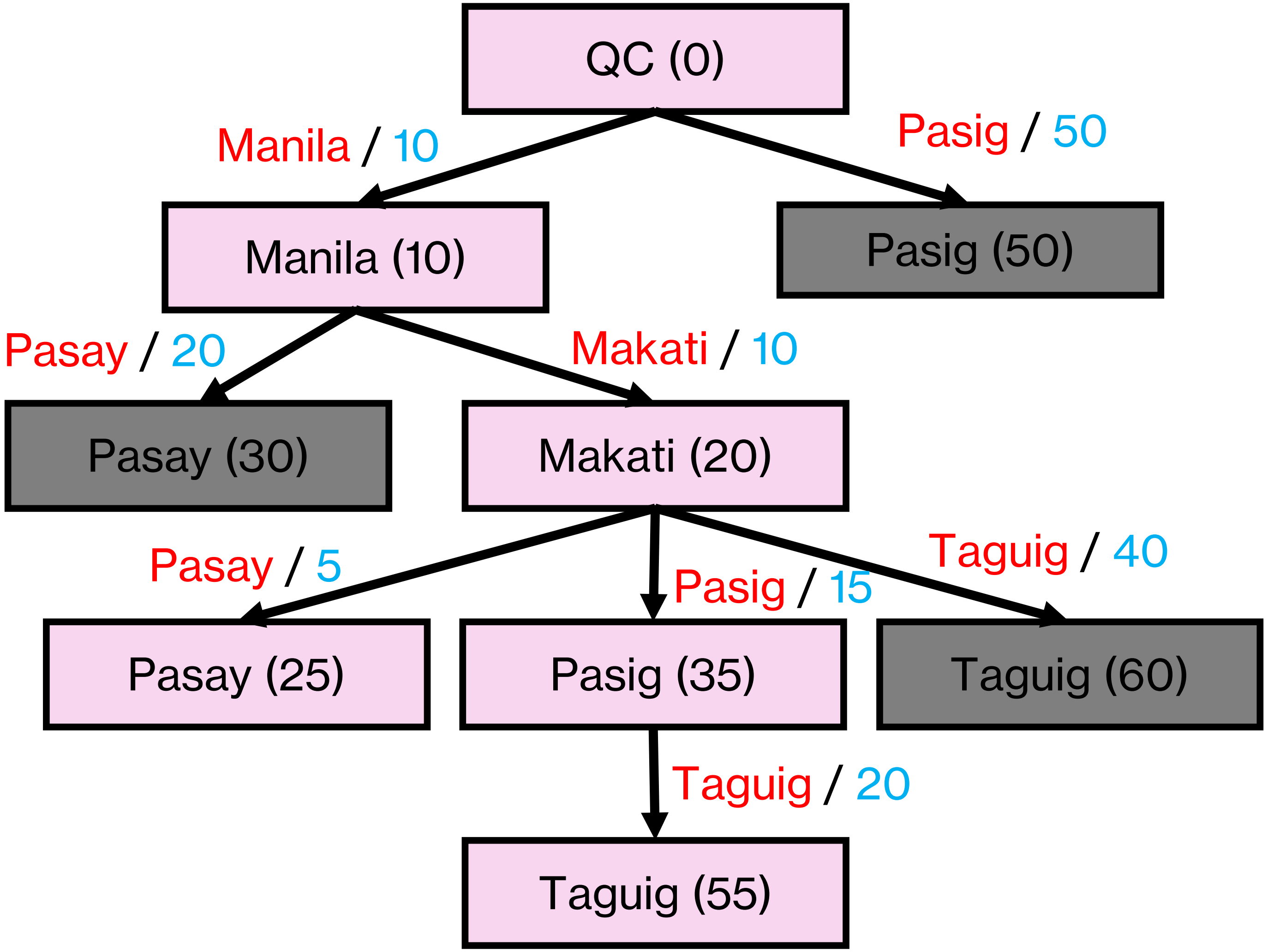
Uniform Cost Search Example





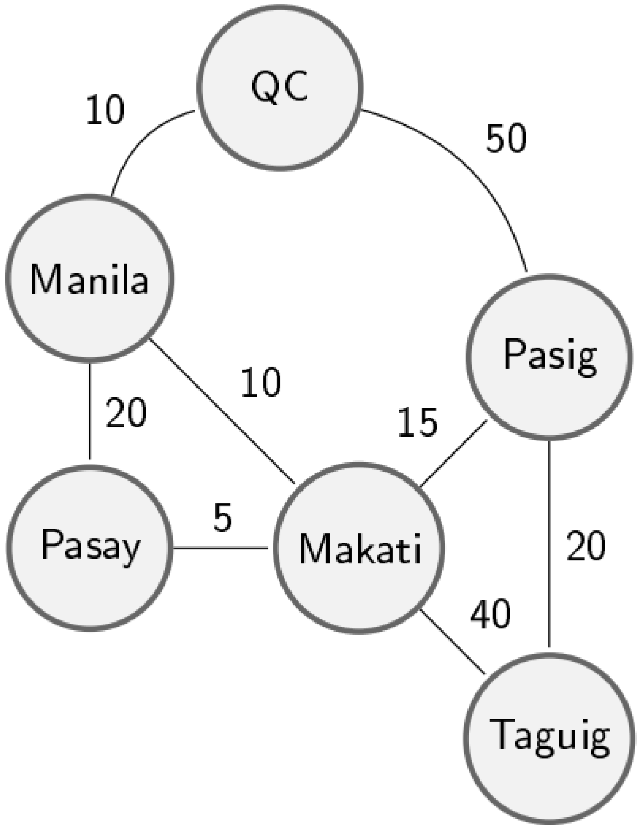
Uniform Cost Search Example

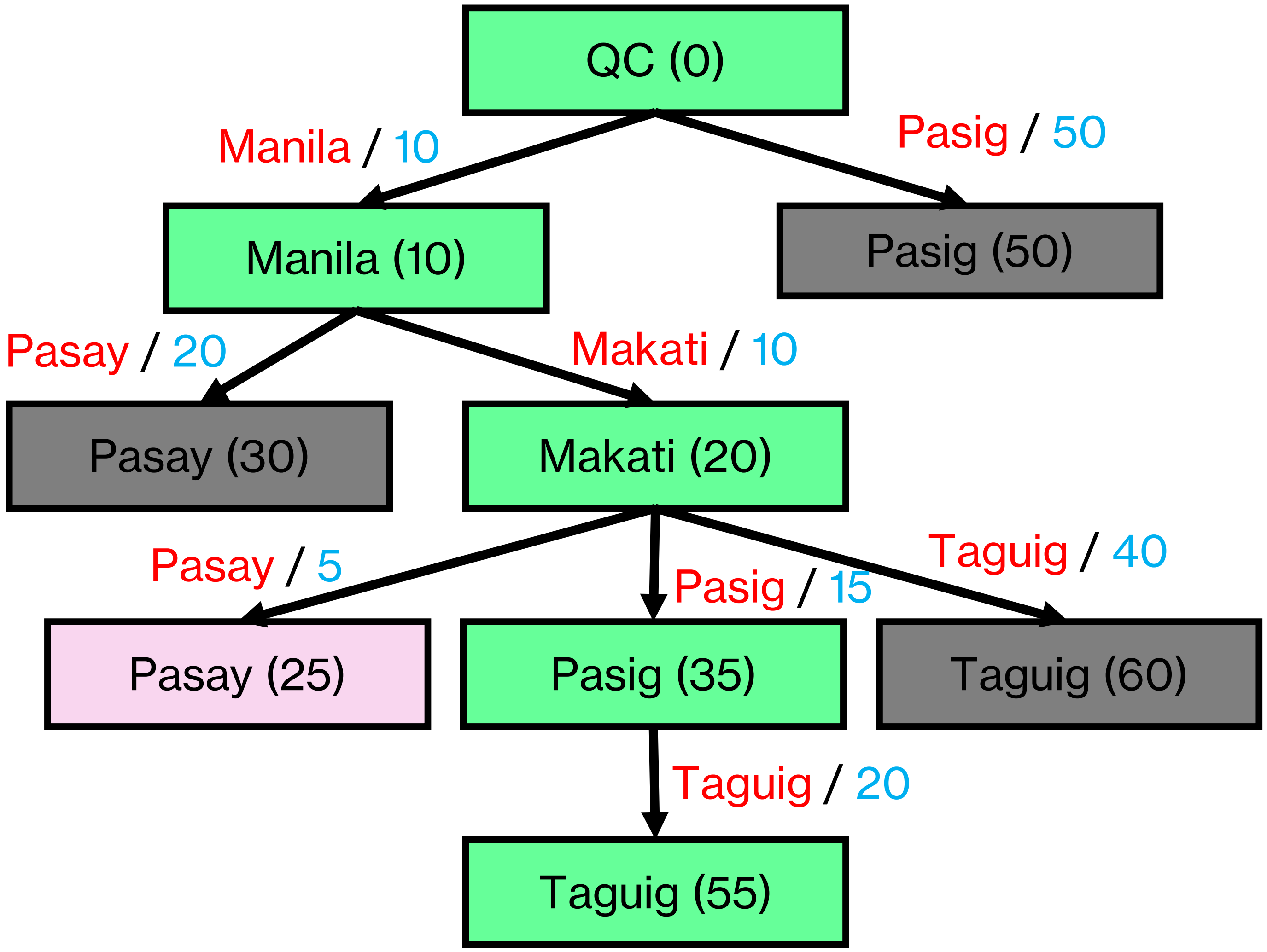




Solution found!

Uniform Cost Search Example





Solution found!

Uniform Cost Search

ALGORITHM

Add s_{start} to FRONTIER with priority 0.
While FRONTIER is not empty do
    Remove s with the lowest priority p from FRONTIER
    if IsEnd(s) then
        return solution
    Add s to EXPLORED
    for each $a \in Actions(s)$ do
        Get successor $s' \leftarrow Succ(s, a)$
        if s' not yet in EXPLORED
            Update s' in FRONTIER with priority $p + Cost(s, a)$

Characteristics of Uniform Cost Search

- Cannot handle negative costs.
- If state space is finite, it is **complete**.
- It is **optimal**.

- Time and space complexity: $\boldsymbol{O(n \log n)}$, where n is the number of states that are closer to the start state than the goal state.

DP vs UCS

- N total states, n of which are closer to start state than the goal.

Algorithm	Cycles?	Action Costs	Time / Space Complexity
Dynamic Programming	no	any	$O(N)$
Uniform Cost Search	yes	≥ 0	$O(n \log n)$

Acknowledgments

- Stanford University CS221 Autumn 2021 course. Available online at: https://stanford-cs221.github.io/autumn2021
- Previous CSINTSY slides by the following instructors:
  - Raymund Sison, PhD
  - Judith Azcarraga, PhD
  - Merlin Suarez, PhD
  - Joanna Pauline Rivera

Readings

- https://www.youtube.com/watch?v=dRMvK76xQJI
- https://www.educba.com/uniform-cost-search/
- https://www.youtube.com/watch?v=9iE9Mj4m8jk