**Edwin Ario Abdiwijaya**

**2440062155 – LC01**

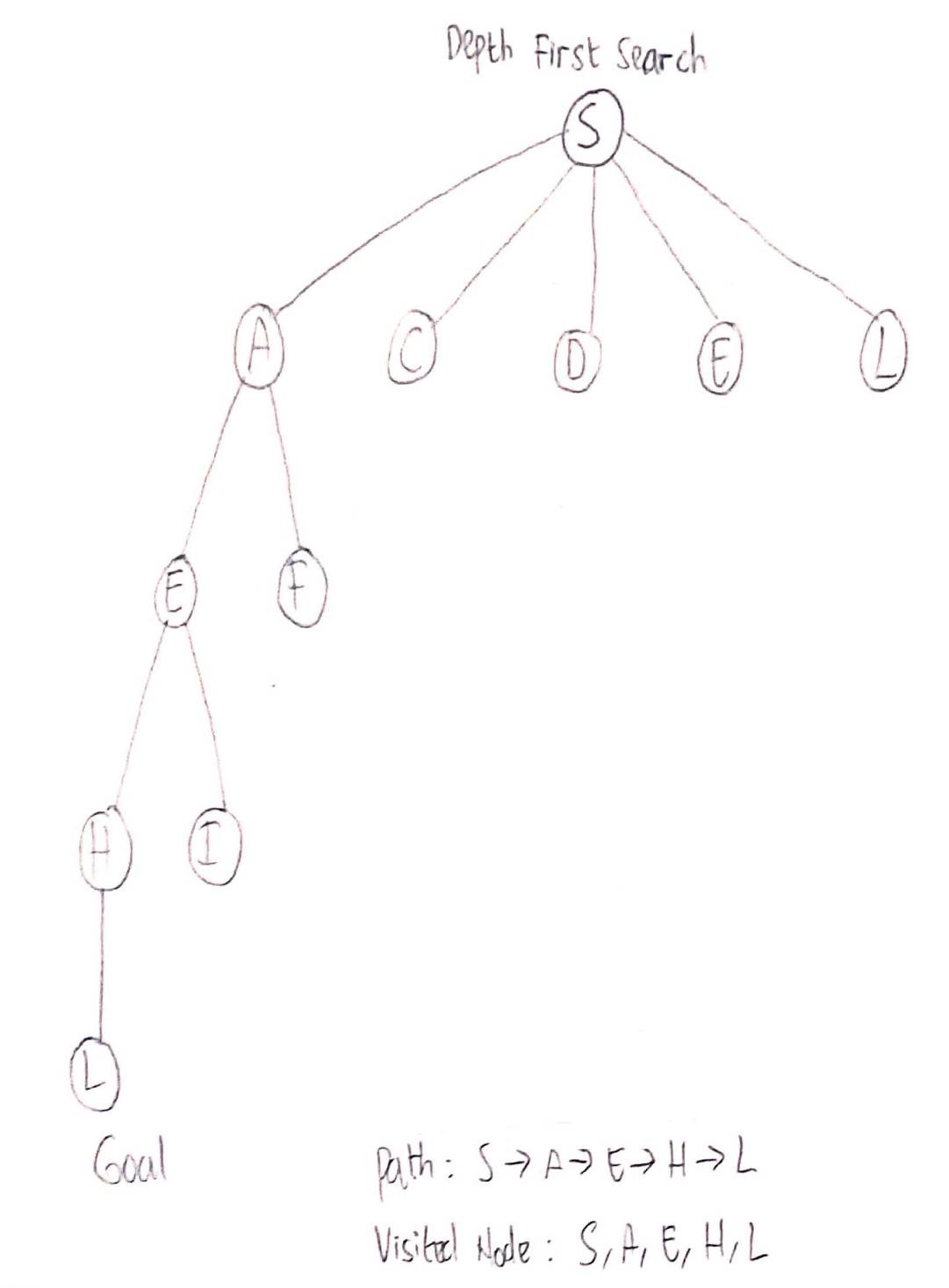
**COMP6639001 – Artificial Intelligence – Mid Exam**

1. a) The solution for find to move from node S to node L with Depth First Search algorithm below,

* Root Node = S
* Stack = {S}
* Visit : S, Stack = {JS, ES, DS, CS, AS}
* Visit : AS, Stack = {JS, ES, DS, CS, FA, EA}
* Visit : EA, Stack = {JS, ES, DS, CS, FA, IE, HE}
* Visit : HE, Stack = {JS, ES, DS, CS, FA, IE, LH}
* Visit : LH, Stack = {JS, ES, DS, CS, FA, IE}

The path = S 🡪 A 🡪 E 🡪 H 🡪 L

Visited Node = S, A, E, H, L

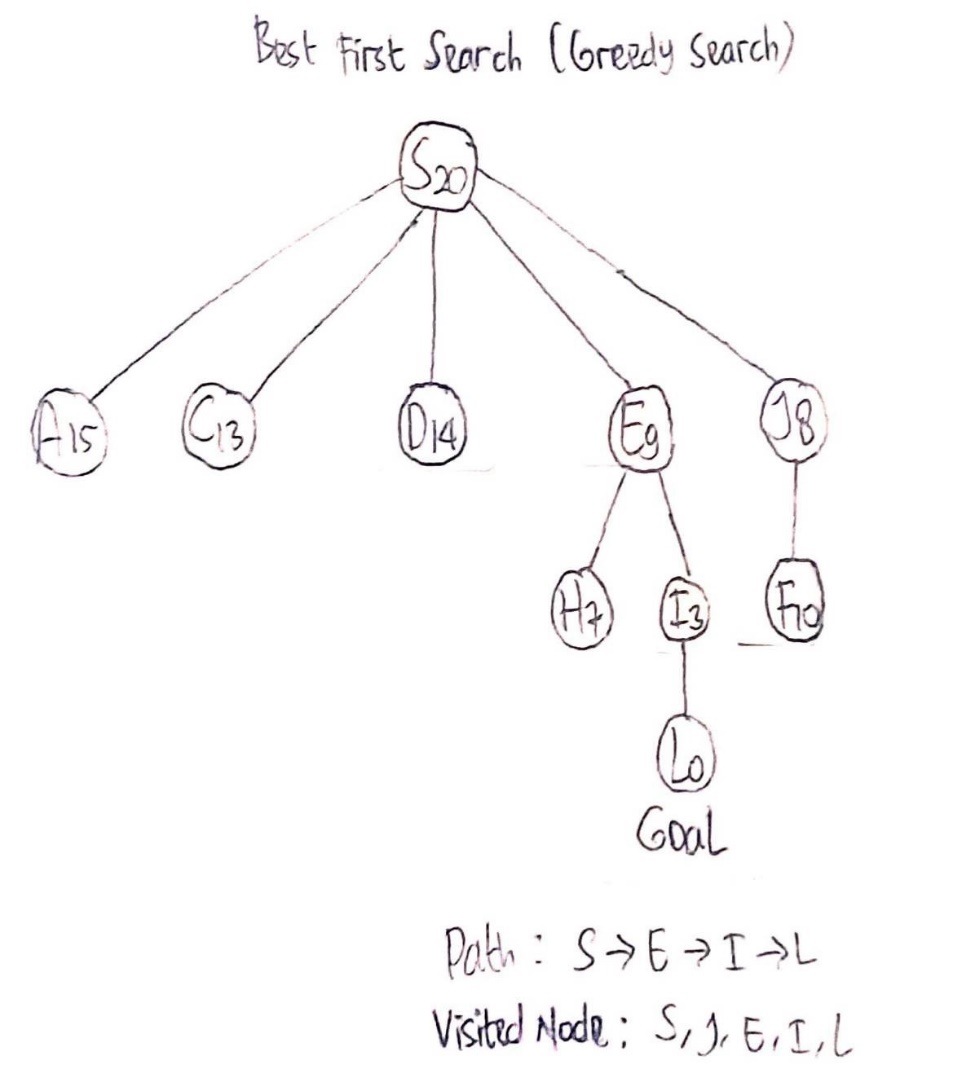


b) The solution for find to move from node S to node L with Best First Search(Greedy search) algorithm below,

* Root Node = S
* Priority Queue = {S}
* Visit : S, Priority Queue = {JS, ES, CS, DS, AS} // 8, 9, 13, 14, 15
* Visit : JS, Priority Queue = {ES, FJ, CS, DS, AS} // 9, 10, 13, 14, 15
* Visit : ES, Priority Queue = {IE, HE, FJ, CS, DS, AS} // 3, 7, 10, 13, 14, 15
* Visit : IE, Priority Queue = {LI, HE, HI, FJ, CS, DS, AS} // 0, 7, 7, 10, 13, 14, 15
* Visit : LI, Priority Queue = {HE, HI, FJ, CS, DS, AS} // 7, 7, 10, 13, 14, 15

The Path = S 🡪 E 🡪 I 🡪 L

Visited Node = S, J, E, I, L



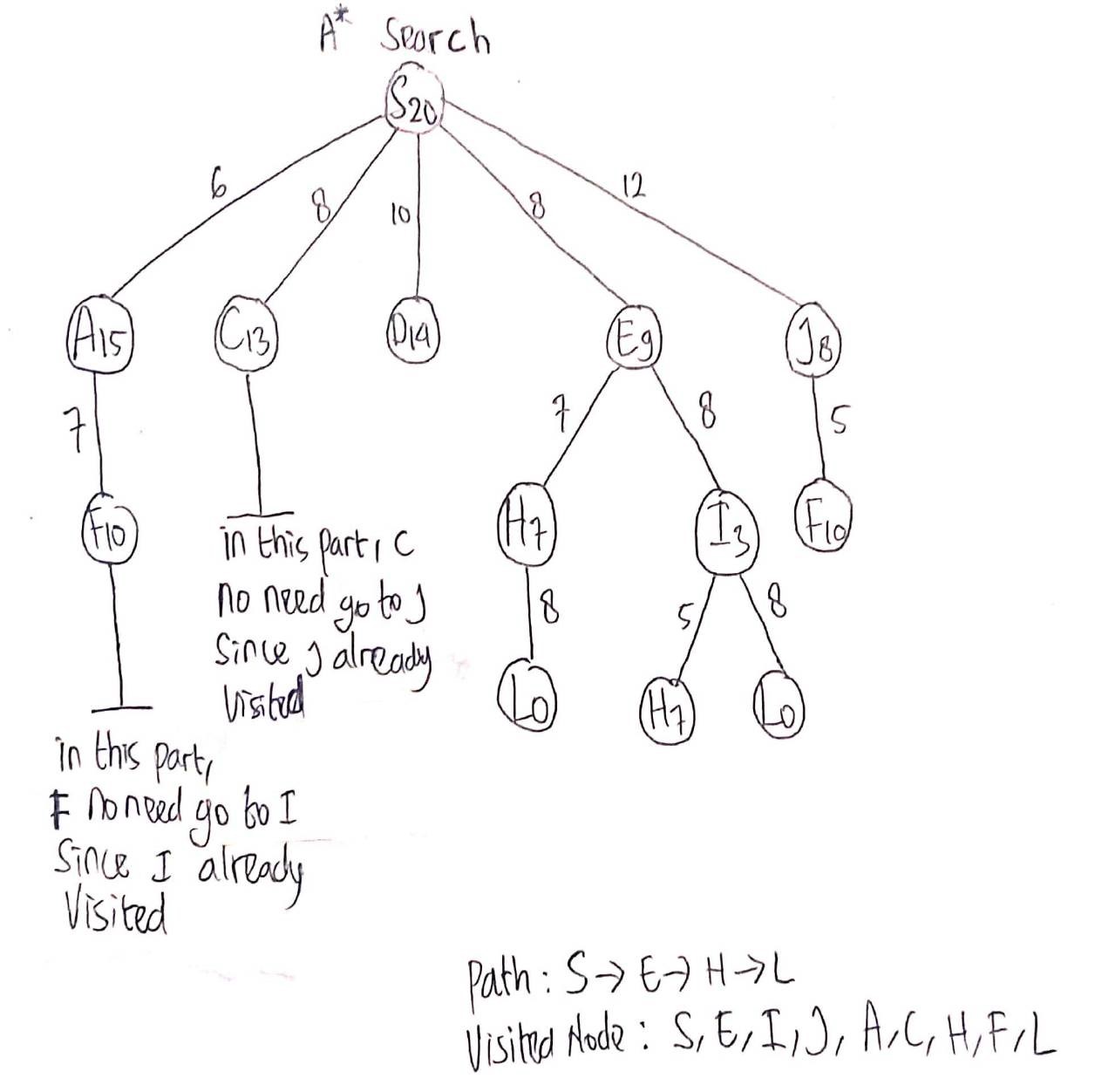
The number next to Alphabets inside circle is Heuristic cost, for example S20 mean h(S) = 20, A15 mean h(A) = 15, and so on.

c) The solution for find to move from node S to node L with A\* Search algorithm below,

* Root Node = S
* Priority Queue = {S}
* Visit : S, Priority Queue = {ES, JS, AS, CS, DS} // 17, 20, 21, 21, 24
* Visit : ES, Priority Queue = {IE, JS, AS, CS, HE, DS} // 19, 20, 21, 21, 22, 24
* Visit : IE, Priority Queue = {JS, AS, CS, HE, DS, LI, HI} // 20, 21, 21, 22, 24, 24, 28
* Visit : JS, Priority Queue = {AS, CS, HE, DS, LI, FJ, HI } // 21, 21, 22, 24, 24, 27, 28
* Visit : AS, Priority Queue = {CS, HE, FA, DS, LI, FJ, HI } // 21, 22, 23, 24, 24, 27, 28
* Visit : CS, Priority Queue = {HE, FA, DS, LI, FJ, HI } // 22, 23, 24, 24, 27, 28
* Visit : HE, Priority Queue = {FA, LH, DS, LI, FJ, HI } // 23, 23, 24, 24, 27, 28
* Visit : FA, Priority Queue = {LH, DS, LI, FJ, HI } // 23, 24, 24, 27, 28
* Visit : LH, Priority Queue = {DS, LI, FJ, HI } // 24, 24, 27, 28

The Path = S 🡪 E 🡪 H 🡪 L

Visited Node = S, E, I, J, A, C, H, F, L



The number next to Alphabets inside circle is Heuristic cost, for example S20 mean h(S) = 20, A15 mean h(A) = 15, and so on.

1. a) Andi is a professor.

FOL : professor(Andi)

CNF : professor(Andi)

b) All professors are people.

FOL : ∀x (professor(x) 🡪 people(x))

CNF : ¬profressor(x) ∨ people(x)

c) Ani is the dean.

FOL : dean(Ani)

CNF : dean(Ani)

d) All Deans are professors.

FOL : ∀x (dean(x) 🡪 professor(x))

CNF : ¬dean(x) ∨ professor(x)

e) All professors consider the dean a friend or don’t know him.

FOL : ∀x∀y (professor(x) ∧ dean(y) 🡪 friend(x,y)

CNF : ¬professor(x) ∨ ¬dean(y) ∨ friend(x,y)

f) Everyone is a friend of someone.

FOL : ∀x∃y (friend(x,y))

CNF : friend(x,y)

g) People only criticize people that are not their friends.

FOL : ∀x∀y (people(x) ∧ people(y) ∧ criticized(x,y) 🡪 ¬friend(x,y))

CNF : ¬people(x) ∨ ¬people(y) ∨ ¬criticized(x,y) ∨ ¬friend(x,y)

h) Andi criticized Ani.

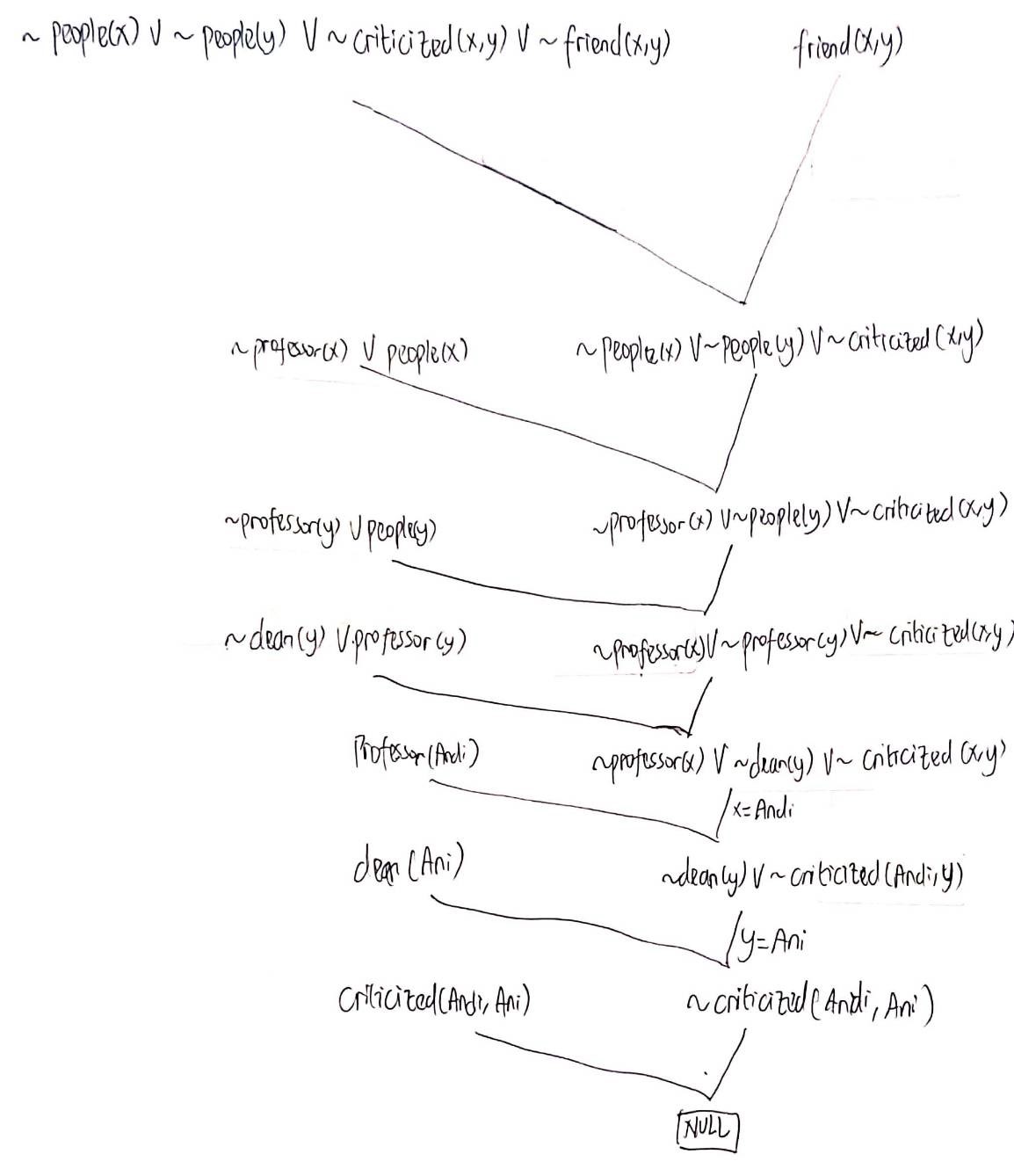
FOL : criticized(Andi, Ani)

CNF : criticized(Andi, Ani)

Prove that: Ani is not Andi’s friend. With The knowledge in CNF :

* ¬people(x) ∨ ¬people(y) ∨ ¬criticized(x,y) ∨ ¬friend(x,y)
* ¬professor(x) ∨ ¬dean(y) ∨ friend(x,y)
* ¬profressor(x) ∨ people(x)
* ¬dean(x) ∨ professor(x)
* friend(x,y)
* professor(Andi)
* dean(Ani)
* criticized(Andi, Ani)

The resolution below,



Because the end result is NULL, therefore Ani is not Andi’s friend is proved.

1. a) Prob(F, E, C, D, ¬A, ¬B)

* Prob(F, E, C, D, ¬A, ¬B) = Prob(¬A) \* Prob(¬B) \* Prob (C | ¬A, ¬B) \* Prob(D | ¬B) \* Prob(E | C, D) \* Prob(F | E)
* Prob(F, E, C, D, ¬A, ¬B) = 0.80 \* 0.60 \* 0.10 \* 0.55 \* 0.70 \* 0.70
* Prob(F, E, C, D, ¬A, ¬B) = 0.0012936

So, the answer for Prob(F, E, C, D, ¬A, ¬B) = 0.0012936

b) Prob(F, E, ¬C, D, A, ¬B)

* Prob(F, E, ¬C, D, A, ¬B) = Prob(A) \* Prob(¬B) \* Prob (¬C | A, ¬B) \* Prob(D | ¬B) \* Prob(E | ¬C, D) \* Prob(F | E)
* Prob(F, E, ¬C, D, A, ¬B) = 0.20 \* 0.60 \* 0.50 \* 0.55 \* 0.45 \* 0.70
* Prob(F, E, ¬C, D, A, ¬B) = 0.010395

So, the answer for Prob(F, E, ¬C, D, A, ¬B) = 0.010395

c) Prob (C | B)

* Prob(C | B) = Prob(C, B) / Prob(B)
* Prob(C | B) = (Prob(C, A, B) + Prob(C, ¬A, B)) / Prob(B)
* Prob(C | B) = (Prob(C | A, B) \* Prob(A) \* Prob(B) + Prob (C | ¬A, B) \* Prob(¬A) \* Prob(B)) / Prob(B)
* Prob(C | B) = (Prob(C | A, B) \* Prob(A) + Prob (C | ¬A, B) \* Prob(¬A)) Prob(B) / Prob(B)
* Prob(C | B) = (Prob(C | A, B) \* Prob(A) + Prob (C | ¬A, B) \* Prob(¬A))
* Prob(C | B) = 0.55 \* 0.2 + 0.45 \* 0.8
* Prob (C | B) = 0.47

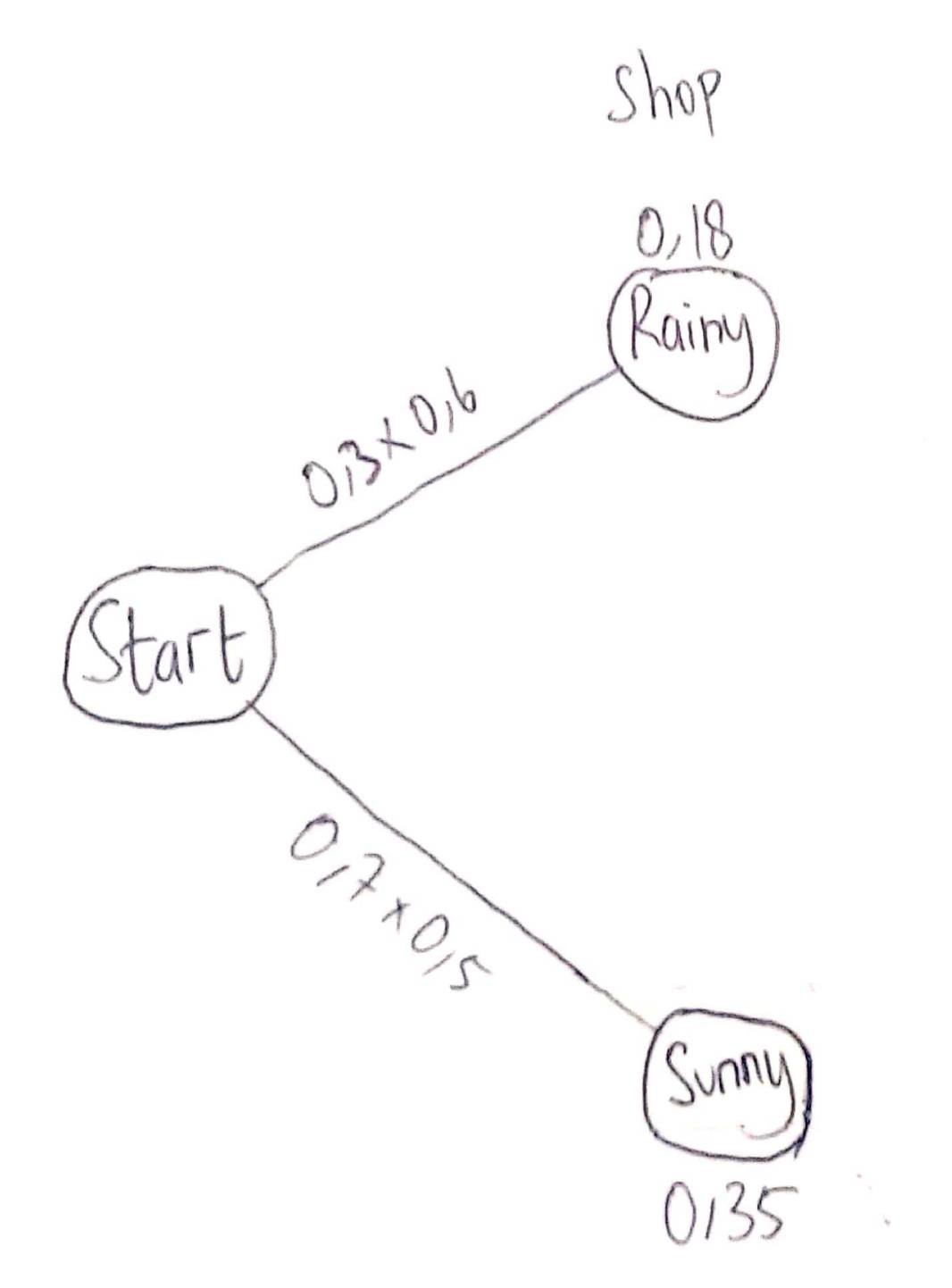
So, the answer for Prob(C | B) = 0.47

d) Prob (C | ¬A)

* Prob(C | ¬A) = Prob(C, ¬A) / Prob(¬A)
* Prob(C | ¬A) = (Prob(C, ¬A, B) + Prob(C, ¬A, ¬B)) / Prob(¬A)
* Prob(C | ¬A) = (Prob(C | ¬A, B) \* Prob(¬A) \* Prob(B) + Prob (C | ¬A, ¬B) \* Prob(¬A) \* Prob(¬B)) / Prob(¬A)
* Prob(C | ¬A) = (Prob(C | ¬A, B) \* Prob(B) + Prob (C | ¬A, ¬B) \* Prob(¬B)) Prob(¬A) / Prob(¬A)
* Prob(C | ¬A) = (Prob(C | ¬A, B) \* Prob(B) + Prob (C | ¬A, ¬B) \* Prob(¬B))
* Prob(C | ¬A) = 0.45 \* 0.4 + 0.1 \* 0.6
* Prob(C | ¬A) = 0.24

So, the answer for Prob(C | ¬A) = 0.24

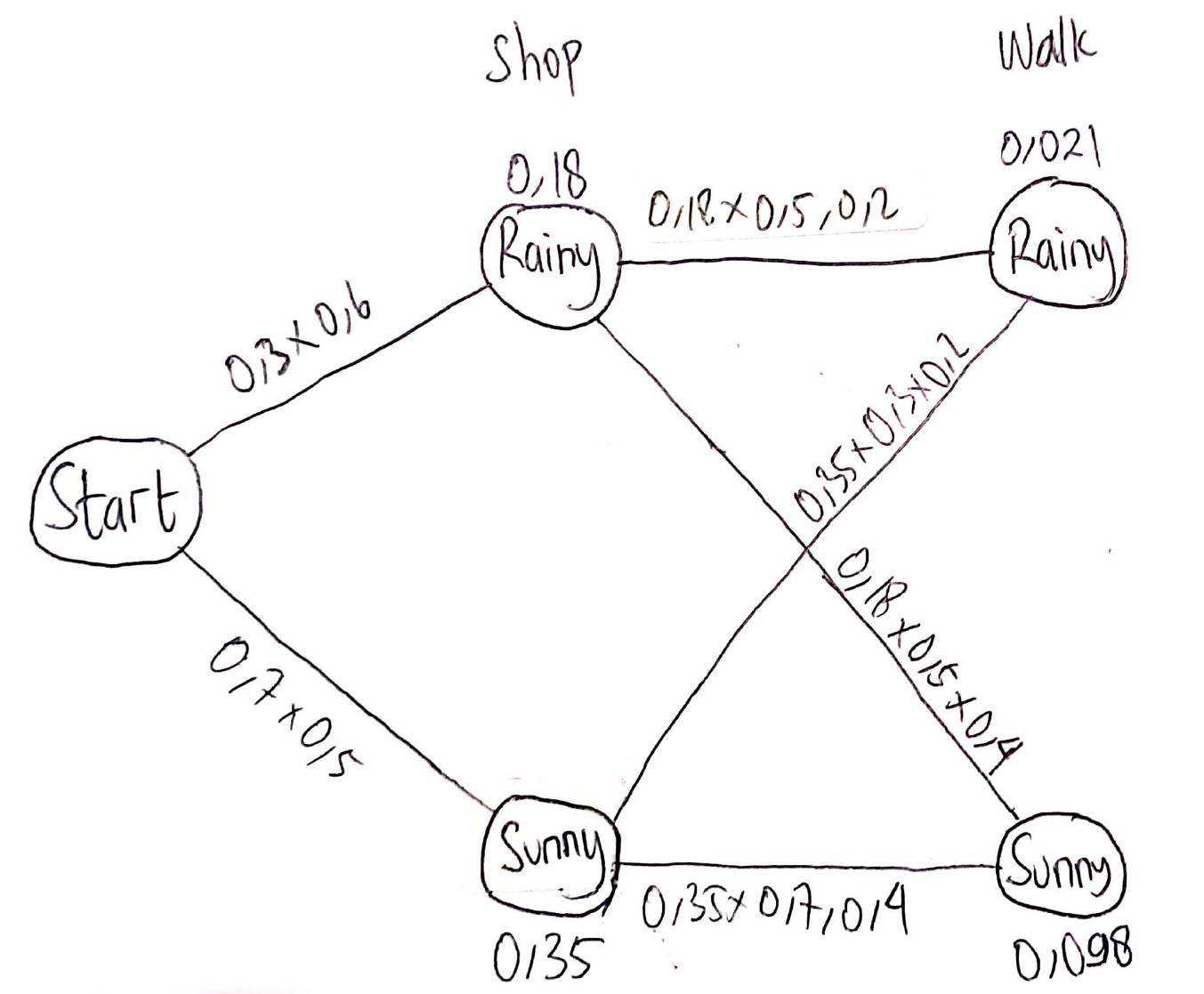
1. From the question, someone has the following activity sequence: shop, walk, and clean. From the above statement, using the Viterbi Algorithm and using the Trellis diagram, the answer below,



As the first image above, the first activity sequence is Shop. From Start 🡪 Rainy is 0.3 and Rainy 🡪 Shop is 0.6. Therefore, Start 🡪 Rainy 🡪 Shop is 0.3 \* 0.6 = 0.18

From Start 🡪 Sunny is 0.7 and Sunny 🡪 Shop is 0.5. Therefore, Start 🡪 Rainy 🡪 Sunny is 0.7 \* 0.5 = 0.35

From the 2 operations above, there are 2 results. Since there are 2 paths and there are 2 results, 1 path 1 result, then the Viterbi Algorithm can’t compare and forced to choose the only result.



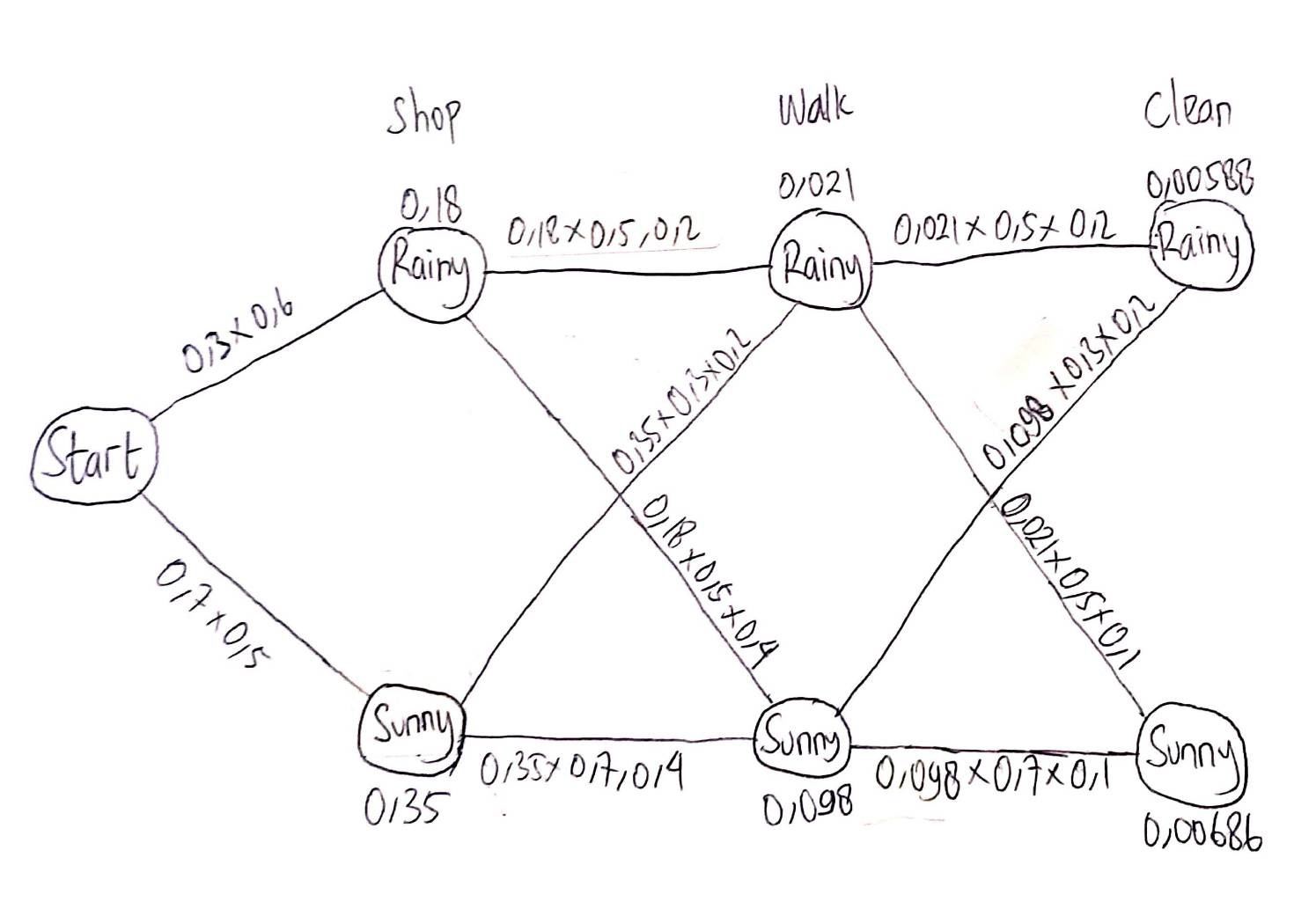
As the second image above, the second activity sequence is Walk. From the last activity, which is, Start 🡪 Rainy 🡪 Shop is 0.18. Rainy 🡪 Rainy is 0.5 and Rainy 🡪 Walk is 0.2. Therefore, Start 🡪 Rainy 🡪 Shop 🡪 Rainy 🡪 Walk is 0.18 \* 0.5 \* 0.2 = 0.018

From the last activity, which is, Start 🡪 Rainy 🡪 Shop is 0.18. Rainy 🡪 Sunny is 0.5 and Sunny 🡪 Walk is 0.4. Therefore, Start 🡪 Rainy 🡪 Shop 🡪 Sunny 🡪 Walk is 0.18 \* 0.5 \* 0.4 = 0.036

From the last activity, which is, Start 🡪 Sunny 🡪 Shop is 0.35. Sunny 🡪 Rainy is 0.3 and Rainy 🡪 Walk is 0.2. Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk is 0.35 \* 0.3 \* 0.2 = 0.021

From the last activity, which is, Start 🡪 Sunny 🡪 Shop is 0.35. Sunny 🡪 Sunny is 0.7 and Sunny 🡪 Walk is 0.4. Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk is 0.35 \* 0.7 \* 0.4 = 0.098

From the 4 operations above, there are 4 results. Since there are 2 paths and there are 4 results, 2 paths 1 result. The Viterbi Algorithm will compare them and find the biggest numbers. First compare is Start 🡪 Rainy 🡪 Shop 🡪 Rainy 🡪 Walk with Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk, 0.021 is more than 0.018, therefore the Viterbi Algorithm will choose 0.021 (Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk). Second compare is Start 🡪 Rainy 🡪 Shop 🡪 Sunny 🡪 Walk with Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk, 0.098 is more than 0.036, therefore the Viterbi Algorithm will choose 0.098(Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk).



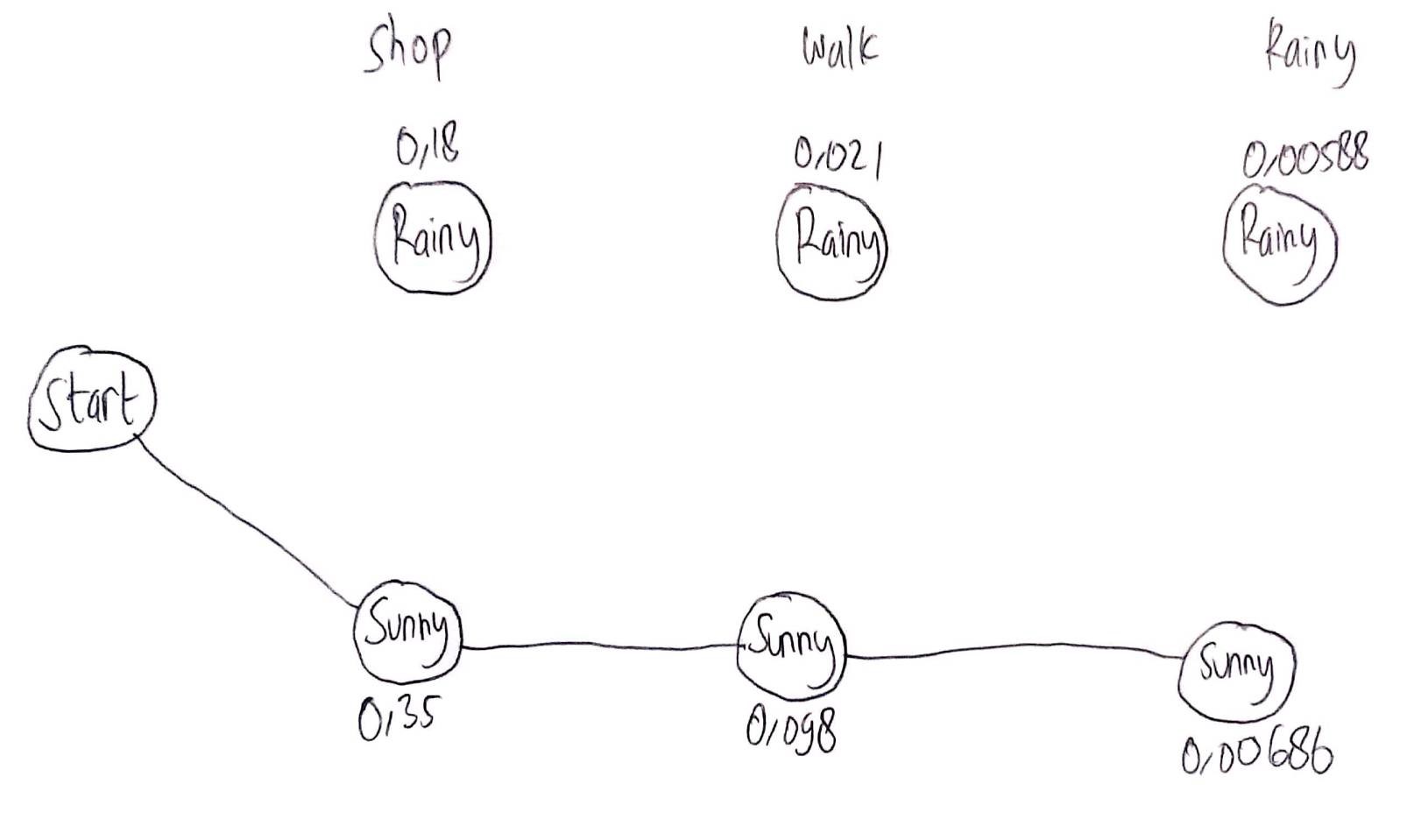
As the third image above, the third activity sequence is Clean. From the last activity, which is, Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk is 0.021. Rainy 🡪 Rainy is 0.5 and Rainy 🡪 Clean is 0.2. Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk 🡪 Rainy 🡪 Clean is 0.021 \* 0.5 \* 0.2 = 0.0021

From the last activity, which is, Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk is 0.021. Rainy 🡪 Sunny is 0.5 and Sunny 🡪 Clean is 0.1. Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk 🡪 Sunny 🡪 Clean is 0.021 \* 0.5 \* 0.1 = 0.00105

From the last activity, which is, Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk is 0.098. Sunny 🡪 Rainy is 0.3 and Rainy 🡪 Clean is 0.2. Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk 🡪 Rainy 🡪 Clean is 0.098 \* 0.3 \* 0.2 = 0.00588

From the last activity, which is, Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk is 0.098. Sunny 🡪 Sunny is 0.7 and Sunny 🡪 Clean is 0.1. Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk 🡪 Sunny 🡪 Clean is 0.35 \* 0.7 \* 0.1 = 0.0686

From the 4 operations above, there are 4 results. Since there are 2 paths and there are 4 results, 2 paths 1 result. The Viterbi Algorithm will compare them and find the biggest numbers. First compare is Therefore, Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk 🡪 Sunny 🡪 Clean with Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk 🡪 Rainy 🡪 Clean, 0.00588 is more than 0.0021, therefore the Viterbi Algorithm will choose 0.00588(Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk 🡪 Rainy 🡪 Clean). Second compare is Start 🡪 Sunny 🡪 Shop 🡪 Rainy 🡪 Walk 🡪 Sunny 🡪 Clean with Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk 🡪 Sunny 🡪 Clean, 0.00686 is more than 0.00105, therefore the Viterbi Algorithm will choose 0.00686(Start 🡪 Sunny 🡪 Shop 🡪 Sunny 🡪 Walk 🡪 Sunny 🡪 Clean).



And lastly, the fourth image above is the result of Trellis Diagram with Viterbi Algorithm. The answer is Sunny-Sunny-Sunny

1. Given, Food Quality = 6.0 and Service = 8.0

When the Fuzzy Logics draw line in food quality graph at x = 6.0 toward the y-axis, the graph will only intersect with Decent and Great only. Therefore, we just need to compute the Decent and the Great.

* Decent = (10 – 6) / (10 – 5)
* Decent = 4 / 5
* Decent = 0.8
* Great = (6 – 5) / (10 – 5)
* Great = 1 / 5
* Great = 0.2

So, the Food Quality :

* Bad = 0.0
* Decent = 0.8
* Great = 0.2

When the Fuzzy Logics draw line in service quality graph at x = 8.0 toward the y-axis, the graph will only intersect with Amazing and Acceptable only. Therefore, we just need to compute the Amazing and the Acceptable.

* Amazing = (8 – 5) / (10 – 5)
* Amazing = 3 / 5
* Amazing = 0.6
* Acceptable = (10 – 8) / (10 – 5)
* Acceptable = 2 / 5
* Acceptable = 0.4

So, the Service Quality :

* Poor = 0.0
* Acceptable = 0.4
* Amazing = 0.6

Now, there are 3 simple rules :

1. food\_quality(bad) ∨ service\_quality(poor) 🡪 tip(low)

Since the food\_quality(bad) is 0 and service\_quality(poor) is 0, the Fuzzy Logics will ignore this rule.

1. service\_quality(acceptable) 🡪 tip(medium)

service\_quality(0.4) 🡪 tip(0.4)

1. food\_quality(great) ∨ service\_quality(amazing) 🡪 tip(high)

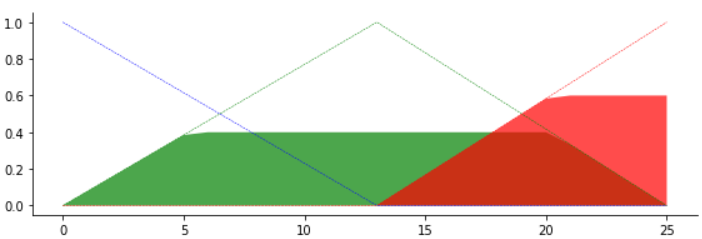
Since the operator that the rule used is OR. Then, the Fuzzy Logics will compare and find the maximum value. In this case, 0.6 > 0.2. Therefore, the Fuzzy Logics will choose 0.6

food\_quality(0.2) ∨ service\_quality(0.6) 🡪 tip(0.6)

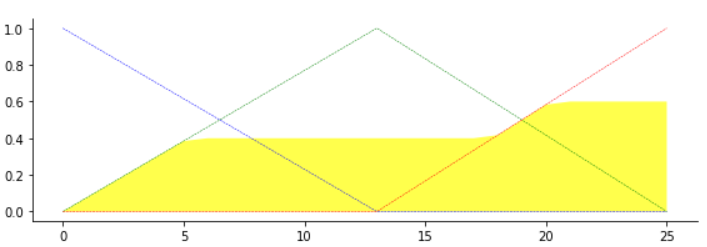
So, the Tip :

* Low = 0.0
* Medium = 0.4
* High = 0.6

When the medium tip = 0.4, and the high tip = 0.6, the graph,



And, when the area green and red become one, the graph,



When the Fuzzy Logics draw line in Tip Membership Function graph at y = 0.4 toward the x-axis or the tip amount, the line will only touch the nearest Medium line, then draw the line 90 degree down to the tip amount and the Fuzzy Logics will make temporary variable, let’s say x1.

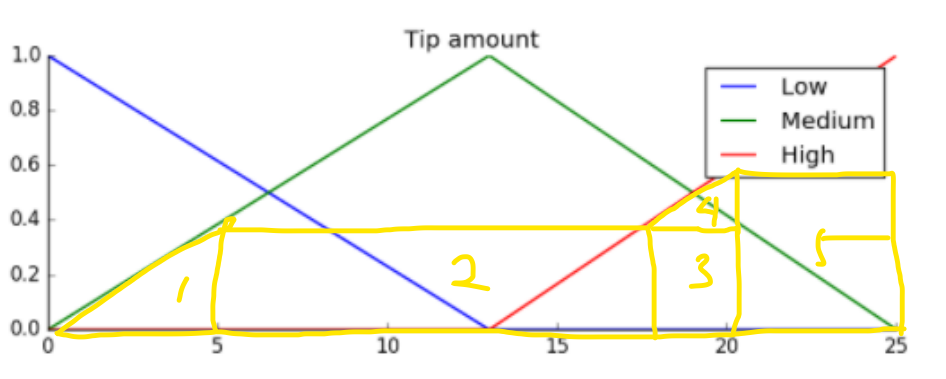
* y = (x1 – 0) / (12.5 – 0)
* 0.4 = x1 / 12.5
* x1 = 5

Same as High, When the Fuzzy Logics draw line in Tip Membership Function graph at y = 0.6 toward the x-axis or the tip amount, the line will only touch the nearest High line, then draw the line 90 degree down to the tip amount and the Fuzzy Logics will make temporary variable, let’s say x2.

* y = (x2 – 12.5) / (25 – 12.5)
* 0.6 = (x2 – 12.5) / 12.5
* 7.5 = x2 – 12.5
* x2 = 20

With x1, x2, tip Medium, and tip High. The tip that the Fuzzy Logics will give is,

First, the Fuzzy Logics will search area, each area represented by numbers, the image shown below,



* area–1 = 0.4 \* 5 / 2 = 1
* area–2 = (17.5 – 5) \* 0.4 = 5
* area–3 = 2.5 \* 0.4 = 1
* area–4 = (20 – 17.5) \* (0.6 – 0.4) / 2 = 0.25
* area–5 = 5 \* 0.6 = 3

Second, the Fuzzy Logics will search Centroid of each areas,

* centroid\_area–1 = (0 + 5 + 5) / 3 = 10 / 3 = 3.33
* centroid\_area–2 = (5 + 5 + 17.5 + 17.5) / 4 = 45 / 4 = 11.25
* centroid\_area–3 = (17.5 + 17.5 + 20 + 20) / 4 = 75 / 4 = 18.75
* centroid\_area–4 = (17.5 + 20 + 20) / 3 = 57.5 / 3 = 19.16
* centroid\_area–5 = (20 + 20 + 25 + 25) / 4 = 22.5

Lastly, to get the tip, the Fuzzy Logics will multiply area with the same representative numbers, divided by the sum of all areas

Tip = (area–1 \* centroid\_area–1 + area–2 \* centroid\_area–2 + area–3 \* centroid\_area–3 + area–4 \* centroid\_area–4 + area–5 \* centroid\_area–5) / (area–1 \* area–2 \* area–3 \* area–4 \* area–5)

Tip = (1 \* 3.33 + 5 \* 11.25 + 1 \* 18.75 + 0.25 \* 19.16 + 3 \* 22.5) / (1 + 5 + 1 + 0.25 + 3)

Tip = (3.33 + 56,25 + 18.75 + 4.79 + 67.5) / 10.25

Tip = 150.62 / 10.25

Tip = 14.69

The answer is Tip = 14.69 and the 14.69 in the graph would looks like below image,

