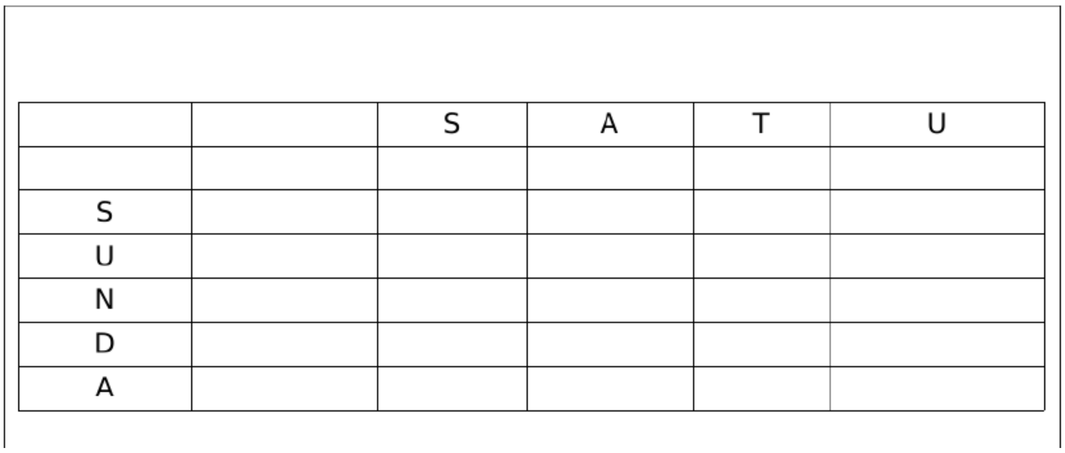
**QUESTION 1**

Compute the *Edit Distance* (also known as *Levenshtein Distance*) between the strings “SATU” and “SUNDA” by completing the Dynamic Programming table below. We assume that all edit operations cost one unit.

(10 marks)



| | | S | A | T | U |
|---|---|---|---|---|---|
| | | | | | |
| S | | | | | |
| U | | | | | |
| N | | | | | |
| D | | | | | |
| A | | | | | |

Running the wk12-13/SOLUTION_DP.py

The table M is initialized as,

```
M =
[[ 0.  1.  2.  3.  4.]
 [ 1.  0.  0.  0.  0.]
 [ 2.  0.  0.  0.  0.]
 [ 3.  0.  0.  0.  0.]
 [ 4.  0.  0.  0.  0.]
 [ 5.  0.  0.  0.  0.]]
```

The final M :

**This is the answer to the question**

```
[[ 0.  1.  2.  3.  4.]
 [ 1.  0.  1.  2.  3.]
 [ 2.  1.  1.  2.  2.]
 [ 3.  2.  2.  2.  3.]
 [ 4.  3.  3.  3.  3.]
 [ 5.  4.  3.  4.  4.]]
```

**The edit distance is therefore 4**

Edit operations

```
['s', 'u', 'n', 'd', 'a']
['s', '-', 'a', 't', 'u']
```

**QUESTION 2**

Let $h_1(s)$ be an admissible A* heuristic. Let $h_2(s) = 2\, h_1(s)$

(a) Is the solution found by A* tree search with $h_2$ guaranteed to be an optimal solution? Justify your answer.

No, as h2 is not necessarily admissible (for example, if h1=h*), there is no guarantee that A* tree search will return an optimal solution.

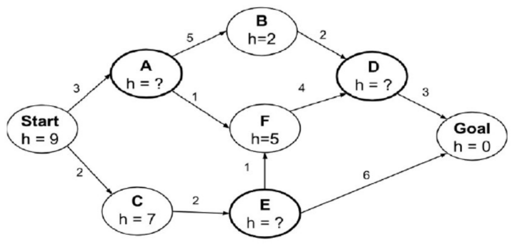
(b) Is the solution found by A* tree search with $h_2$ guaranteed to have a cost at most twice as much as the optimal path? Justify your answer.

(2 marks)

Yes, it is. Let G2 be the state graph obtained from the original state graph G1 by multiplying each weight by 2. The function h2 is admissible with G2.
For all node n in the search tree, g1(n)+h2(n) <= 2*g1(n)+h2(n).
This is true in particular for the goal node ng reached using A* with 2*g1+h2 in G2.
This goal node ng is also optimal in G1 with respect to g1+h1.
The goal node ng' reached with A* in G1 with g1+h2 satisfies g1(ng')+h2(ng')<=g1(ng)+h2(ng).
Therefore the cost of ng' is at most twice the cost of an optimal solution of G1.

(c) Consider the state space graph shown below in which some of the states are missing a heuristic value. Determine the possible range for each missing heuristic value so that the heuristic is admissible and consistent. If this isn't possible, write so.

(6 marks)



| State | Range for h(s) |
|---|---|
| A | $\leq h(A) \leq$ |
| D | $\leq h(D) \leq$ |
| E | $\leq h(E) \leq$ |

We have h(A)<= min(5+h(B), 1+h(F)) = min(7,6) = 6 and h(S)<=3+h(A)
Therefore **6=9-3 <= h(A) <= 6**

We have max(h(B)-2, h(F)-4)<=h(D) and h(D)<=3+h(G)
Therefore **max(0,1)=1 <= h(D) <= 3**

We have h(E)<= min(1+h(F), 6+h(G)) = min(6,6) = 6 and h(C)<=2+h(E)
Therefore **5=7-2 <= h(E) <= 6**

**QUESTION 3**

Suppose that you have implemented a program for Reinforcement Learning (RL) using the Q-learning algorithm. To check this program, you will compute by hand a test case. Consider a small toy example where the environment can be in four different states ***s*** (1 , 2 , 3 and 4), and the agent in each time step chooses one out of three actions ***a*** (1, 2 and 3). The Q-function is stored as a table in the program, and you initialize it with the values below:

| | **s = 1** | **s = 2** | **s = 3** | **s = 4** |
|---|---|---|---|---|
| **a = 1** | 0.1 | 0.2 | 0.3 | 0.4 |
| **a = 2** | 0.5 | 0.6 | 0.7 | 0.8 |
| **a = 3** | 0.9 | 1.0 | 1.1 | 1.2 |

The discount of future rewards $\gamma$ is set to 0.9. The learning rate $\lambda$ is 0.2 (that is, the Q-value is adapted by 20% of the temporal difference value).

The obervation sequence is S, A, R, S, A, R, S = 3, 1, 1, 2, 2, 0, 4
The update for Q-learning is
Q[s,a] = (1-lr)*Q[s,a] + lr*(r+ df* max( Q[sn,an] for an in actions(sn) ) )

(a) State which Q-values have changed and by how much after the agent has completed the first step (take Action 1 in State 3 and arrive in State 2 with a reward of 1). Give the relevant formula update(s) for the action values.

Q(3,1) will be updated.

**Q(3,1) = (1-0.2) * 0.3 + 0.2 *(1+0.9*max(0.2,0.6,1.0)) = 0.62**

(b) State which Q-values have changed and by how much after the agent has completed the second step (take Action 2 in State 2 and arrive in State 4 with a reward of 0). Give the relevant formula update(s) for the action values.

Q(2,2) will be updated.

**Q(2,2) = (1-0.2) * 0.6 + 0.2 *(0+0.9*max(0.4,0.8,1.2)) = 0.696**

(c) Explain what happens in reinforcement learning if the agent always chooses the action that maximizes the Q-value.

The agent will not explore its action space and won’t be able to discover better actions than the greedy actions of the current policy derived from Q. The agent will fail to learn a good policy.

(d) Suggest a way to force a RL agent to explore its policy space.

The agent can explore its action space by selecting a random action 10% of the time. It can also use the UCB formula to guide its choice of actions.

**QUESTION 4**

(a) Consider the following cancer testing scenario:

- *1% of women have breast cancer (and therefore 99% do not).*
- *80% of mammograms detect breast cancer when it is there (and therefore 20% miss it).*
- *9.6% of mammograms detect breast cancer when it is not there (and therefore 90.4% correctly return a negative result).*

Now suppose that Alice gets a positive test result. What are the chances Alice has cancer?
Show your work below.

Let's define some variables
C: has cancer
T: test positive
From the text, we derive that
P(C) = 1% , P(not C) = 99%
P(T|C) = 80% , P(not T|C) = 20%
P(T|not C) = 9.6% and P(not T|not C)= 90.4%

In the case of Alice, we are interested in P(C|T)

P(C|T) = P(T|C) * P(C) /P(T) where

P(T) = P(T,C)+P(T, not C) = P(T|C) * P(C) + P(T|not C) * P(not C)

Numerically,
P(T) = 0.8*0.01 +0.096*0.99 = 0.008 + 0.09504

**P(C|T) = 0.008/(0.008 + 0.09504) = 0.0776**

That is, **Alice has only about 7.8% chance to have cancer given that the test result**.

Question 4 continued

(b) Given the table below,

| Ex # | A | B | C | D | Y |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 | 0 | 0 |
| 7 | 0 | 1 | 0 | 1 | 1 |

Apply Naïve Bayes rule to calculate the ratio $\frac{P(Y=1|A=0,B=1,C=0,D=0)}{P(Y=0|A=0,B=1,C=0,D=0)}$

P(Y|A,B,C,D) = P(A,B,C,D|Y)*P(Y)/P(A,B,C,D)

Using Naive Bayes hypothesis,
we approximate P(A,B,C,D|Y) with P(A|Y)*P(B|Y)*P(C|Y)*P(D|Y)

The ratio becomes
P(Y=1)* P(A=0|Y=1)*P(B=1|Y=1)*P(C=0|Y=1)*P(D=0|Y=1)
over
P(Y=0)* P(A=0|Y=0)*P(B=1|Y=0)*P(C=0|Y=0)*P(D=0|Y=0)

Numerically
(3/7)*(2/3)*(3/3)*(2/3)*(1/3) = 0.0634920
over
(4/7)*(1/4)*(2/4)*(1/4)*(3/4) = 0.01339285

**The ratio equals 4.74074**