IF100 – Fall 2017
Homework 1
Due October 18th 2017 Wednesday 23:55 (Sharp Deadline)

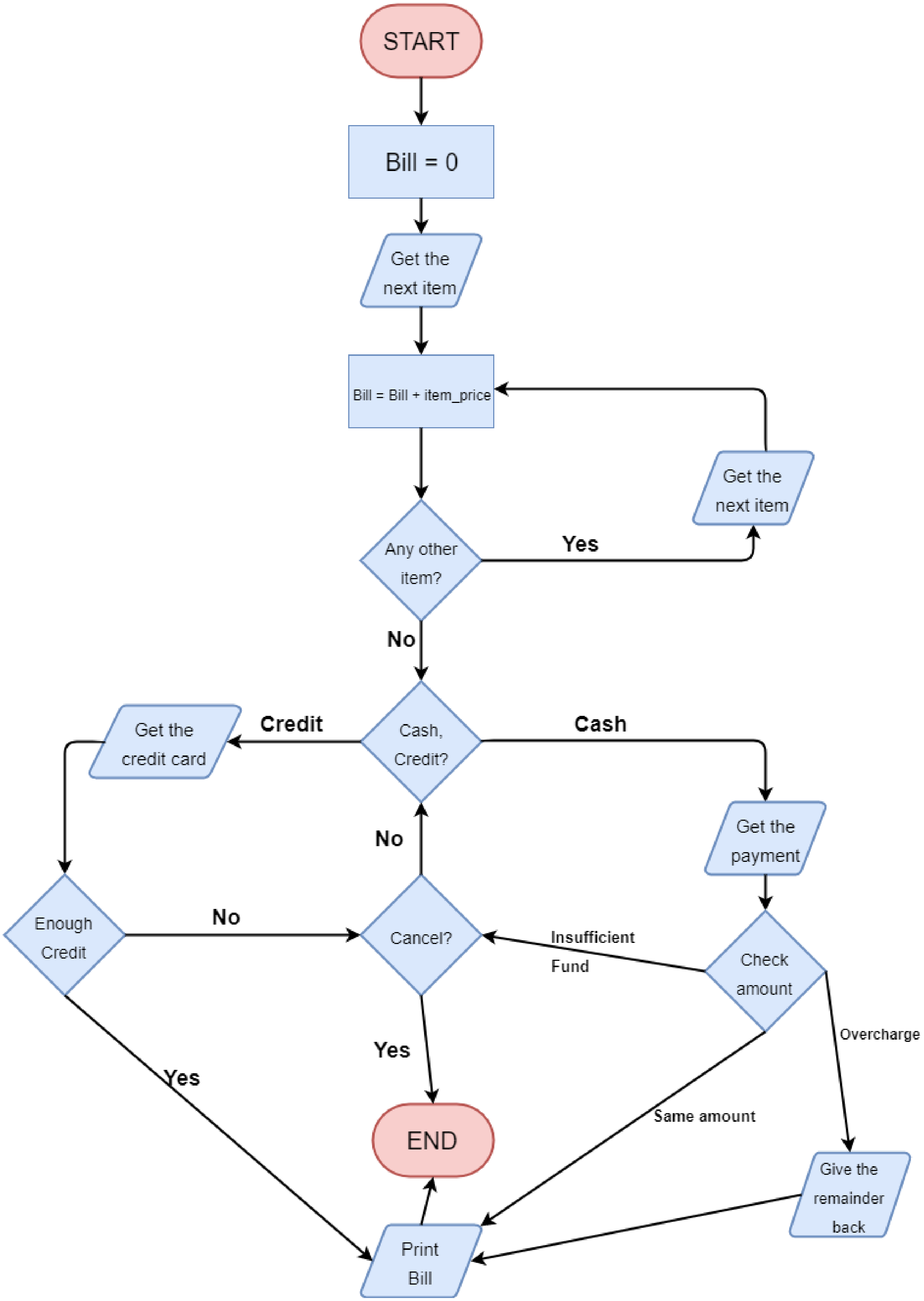
Description

In this homework, you will answer the following three questions.

1. Draw the flowchart of an algorithm followed by a supermarket cashier, whose specifications are given below. The system should accept different types of products, cash and credit card as inputs, and it should supply the requested products and cash balance as outputs.

   The user will pass the products that (s)he wants to buy to the cashier, one by one. At each iteration, the cashier will get an item and update the bill, until there are no more products left to be bought. Then, the cashier will ask the user whether (s)he wants to pay in cash or with a credit card. If the user wants to pay in cash, then the cashier will get the payment and check the paid amount. If the payable balance is equal to the paid amount, then the cashier will print the bill and hand over the products to the user. If it is an overcharge, then the cashier will first give the remainder (change) to the user, and then (s)he will print the bill and hand over the products to the user. Otherwise, i.e. if it is an insufficient fund, the cashier will ask the user whether (s)he wants to cancel his/her transaction (shopping) or not. If the user wants to cancel, then the cashier gives the paid amount back to the user and (s)he is done with that customer. If the user wants to continue, then the cashier will ask whether the user wants to pay the remaining amount by cash or credit card, and depending on the answer of the user, the above-mentioned procedures will be carried out.

   On the other hand, if the user wants to pay with a credit card, then the cashier will get the credit card of the user and check whether there is enough credit or not. If there is, then (s)he will print the bill and hand over the products to the user. Otherwise, the cashier will ask the user whether to cancel the transaction or not, and depending on his/her answer, the procedures to be followed will be as described above.


START
Bill = 0
Get the next item
Bill = Bill + item_price
Get the next item
Any other item?
Yes
No
Cash, Credit?
Credit
Cash
Get the credit card
Get the payment
No
Enough Credit
No
Cancel?
Insufficient Fund
Check amount
Overcharge
Yes
Yes
Same amount
END
Give the remainder back
Print Bill

2. Boza is a popular fermented beverage in some countries like Turkey, Kazakhstan,... etc. Assume that you have an unlimited amount of Boza in a very big storage. Additionally, you have two containers (let's say A and B) where their sizes are 3 litres and 8 litres respectively (containers are initially empty). Your goal is to have exactly 4 litres of Boza in one of these two containers.

   You have a set of instructions which are given below. You can only use these instructions to solve the question.

Instruction	**Meaning**
Fill A	Fill container A with 3 litres of Boza
Fill B	Fill container B with 8 litres of Boza
Empty A	Empty the container A
Empty B	Empty the container B
A → B	Move the available amount of Boza from container A to container B.
B → A	Move the available amount of Boza from container B to container A.

   An example to consecutive set of instructions is given below:

Instruction	**Volume in A**	**Volume in B**
Fill B	0	8
B → A	3	5
Empty B	3	0

   At the end, you will have a table like above to reach your goal. ***Additionally, you MUST clearly indicate how you utilized the component(s) of computational thinking, otherwise you will NOT get any credits EVEN if your answer is true.***

This question was inspired by the movie "Die Hard with a Vengeance (Die Hard 3)". If you haven't watched the movie, you can watch the related scene :) https://www.youtube.com/watch?v=6cAbgAaEOVE

Solution 1

The first fact is that we can only get 4 liters of Boza by having 4 liters in B (since we cannot keep 4 liters in A). So, the question is how can we have 4 liters in B? There are several ways actually. One of the ways is to have 7 liters in B, so we can apply "B → A" and B will have 4 liters at the end. The other way is to have 1 liter in B, so we can "Fill A", then apply "A → B" and we have 4 liters in B. Actually, we convert our problems into relatively smaller problems which helps us to solve the main problem (Decomposition). Let's choose the second way (to have 1 liter in B). How can we get 1 liter in B? If we manage to have 1 liter in A and then we can apply "A → B". How can we get 1 liter in A? If we manage to have 6 liters in B and 3 liters in A (we can always have 3 liters in A), then we can apply "A → B". How can we get 6 liters in B? We can fill from A into B twice. As you will notice, we represent our problem with another subproblems and these subproblems are represented by other subproblems. Decomposition helped us to solve this question.

So the order of the instructions can be as follows (surely there are other set of instructions to have 4 liters in B, they are also okay):

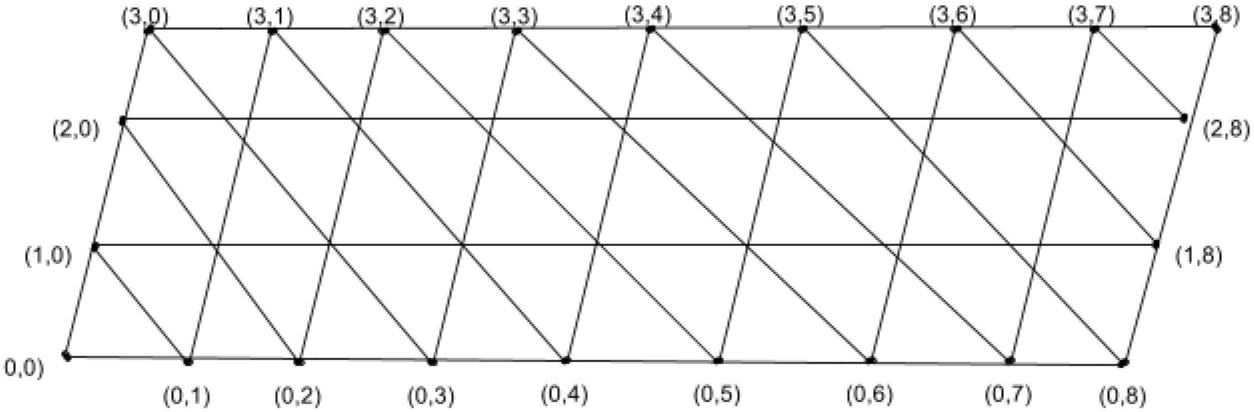
Instruction	**Volume in A**	**Volume in B**
Fill A	3	0
A → B	0	3
Fill A	3	3
A → B	0	6
Fill A	3	6
A → B	1	8
Empty B	1	0

A → B	0	1
Fill A	3	1
A → B	0	4

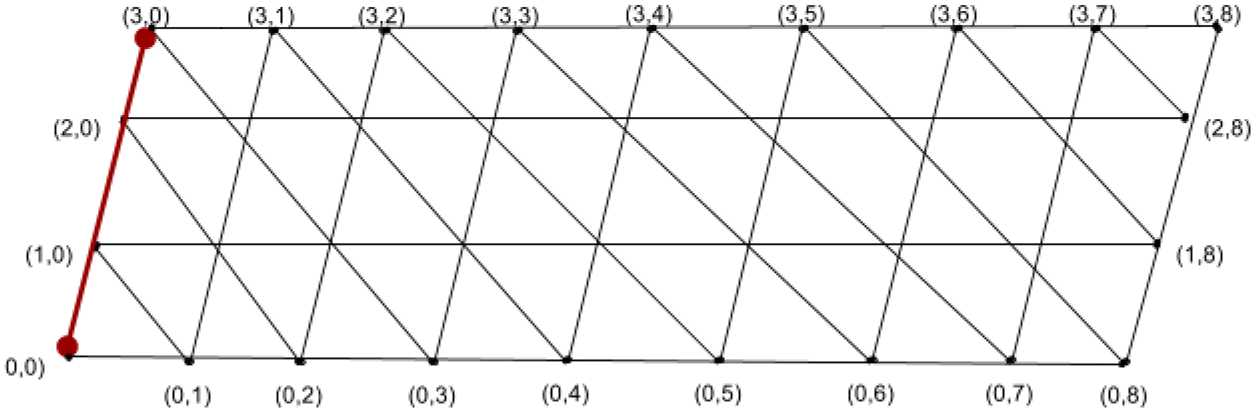
Solution 2

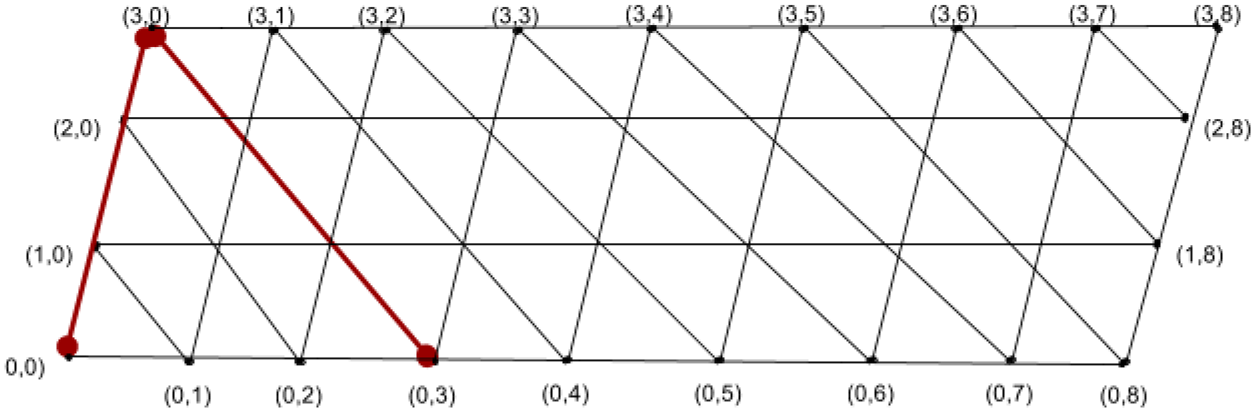
As the second solution technique, we can use abstraction component of computational thinking. We can define our problem in a different representations. Let's say that we have nodes and these nodes represented as (A,B). For instance, the node (3,0) represents the situation that A has 3 liters and B is empty. Let's say two nodes are connected to each other if we can reach from one to another with just 1 instruction.

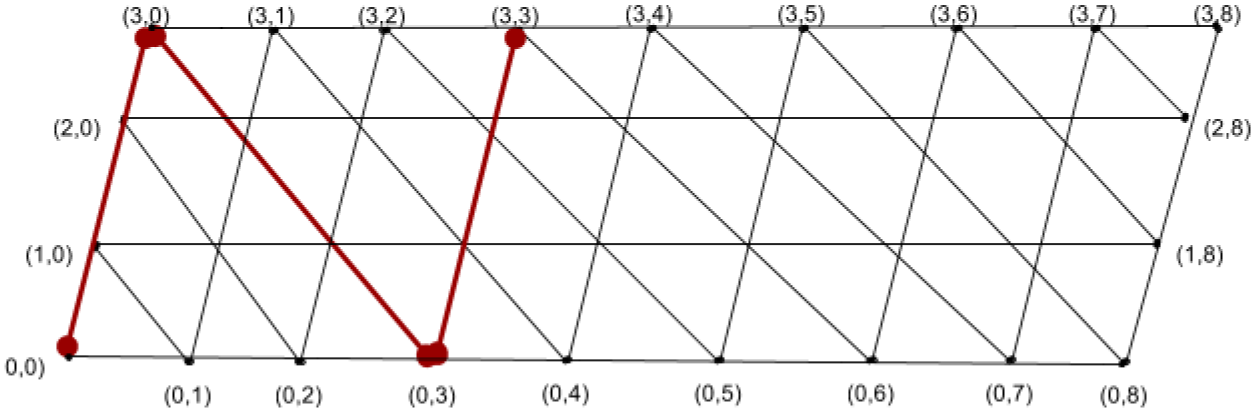
For example, (3,0) is connected to (0,0), (3,5) and (0,3).

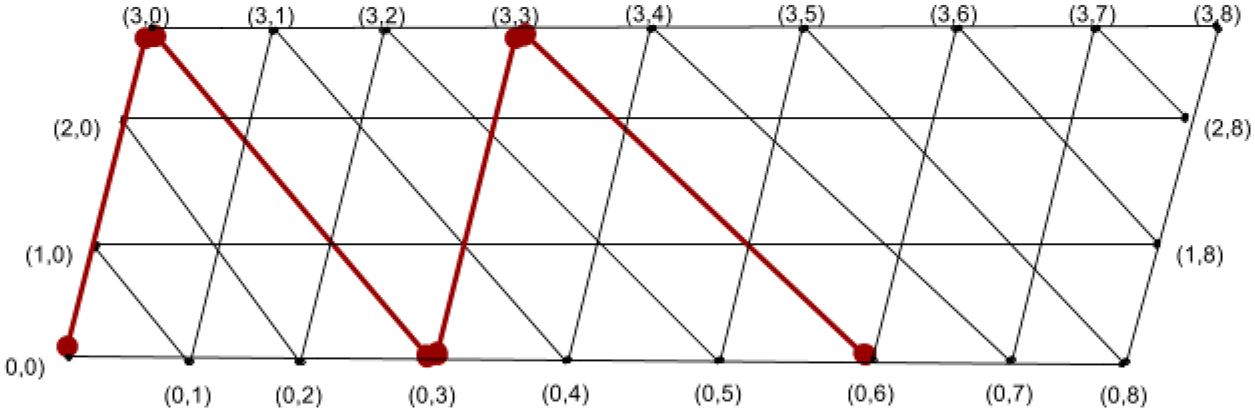


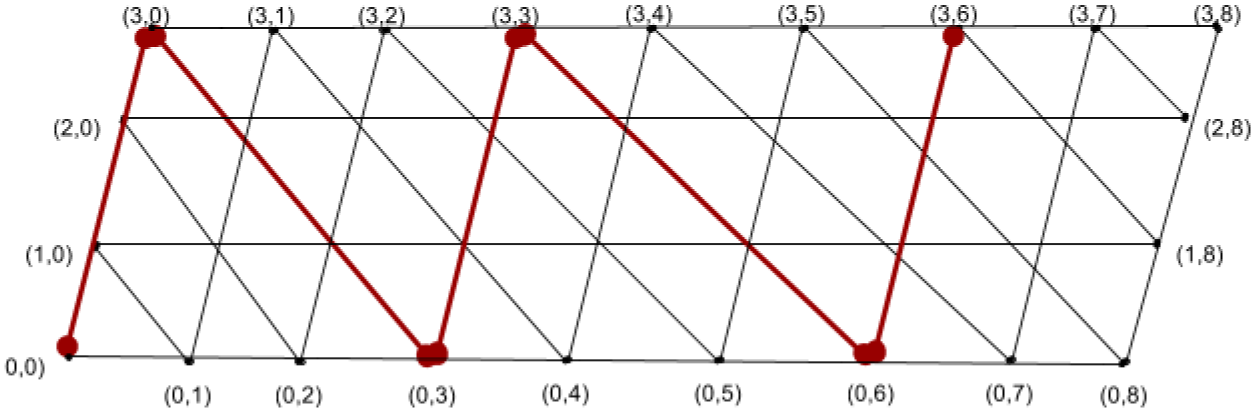
We will start from (0,0) and we want to end up in a node which has 4 B or a node which represented as (x, 4) where x is any value for A. We can reach (3,0) or (0,8) from (0,0). Let's choose (3,0). Then it will be like playing billiard. One of the paths can be as follows: (0,0), (3,0), (0,3), (3,3), (0,6), (3,6), (1,8), (1,0), (0,1), (3,1), (0,4)

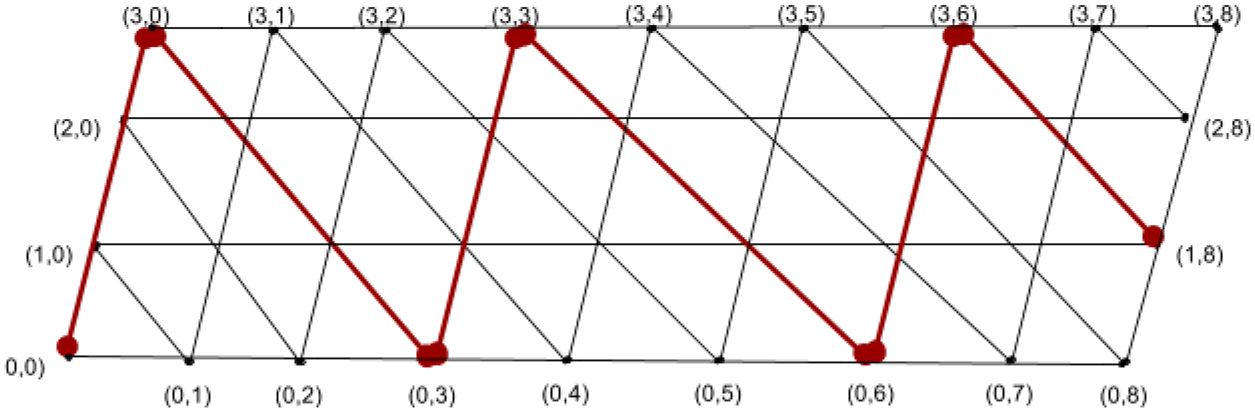

(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)

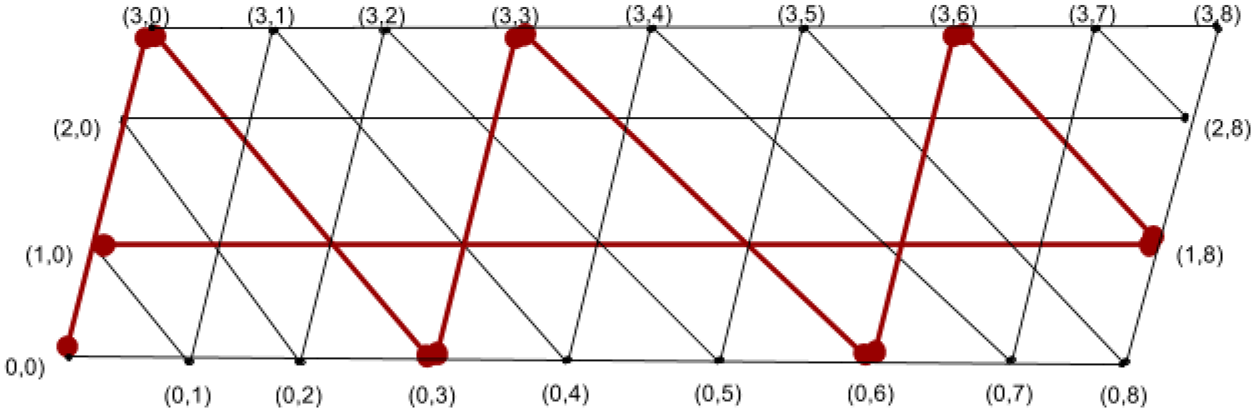

(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)

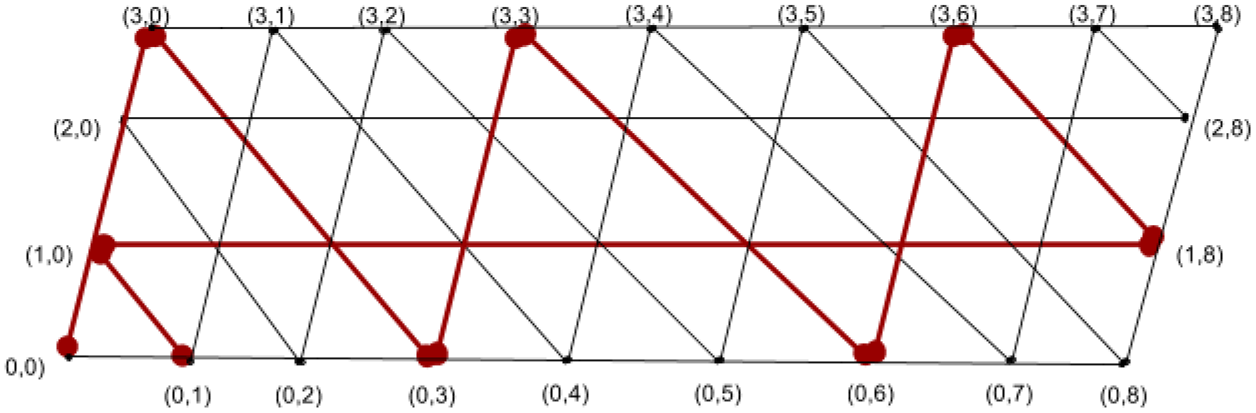

(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)

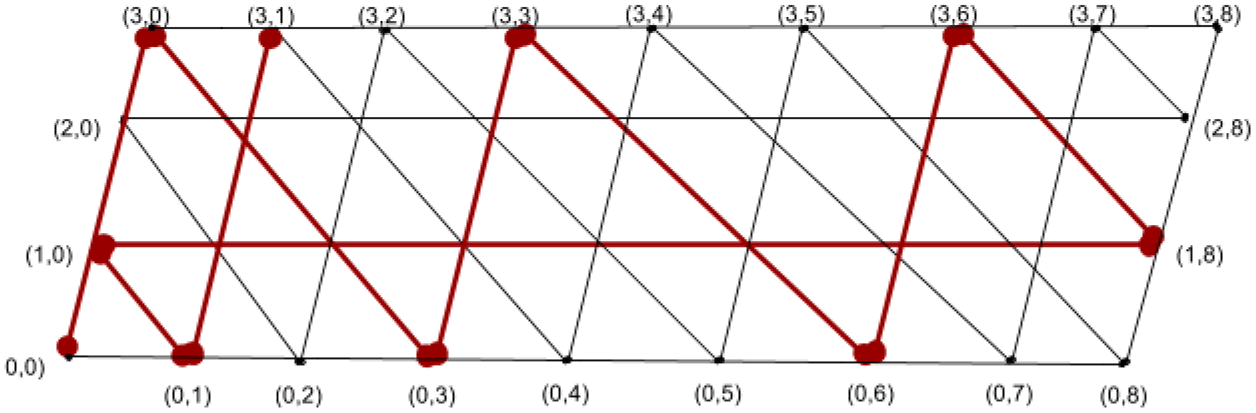

(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)

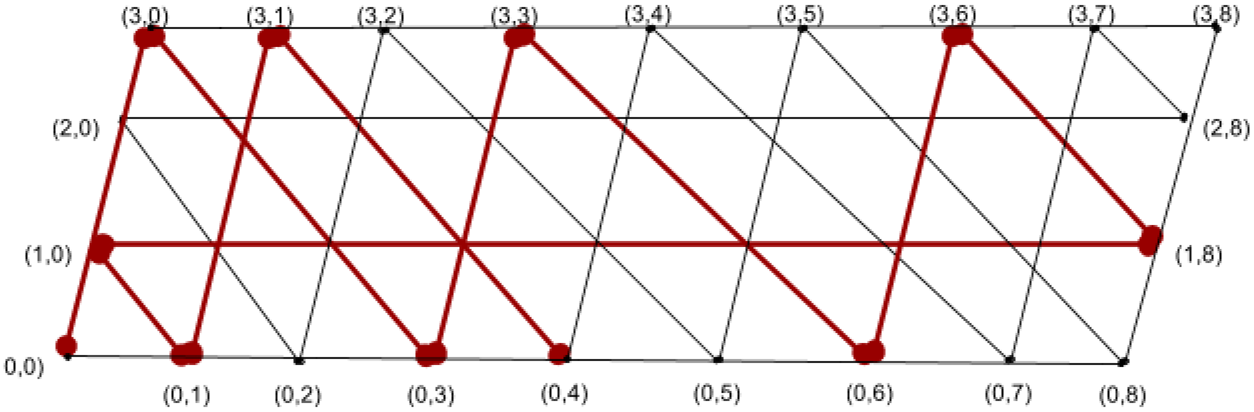

(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)


(3,0)
(3,1)
(3,2)
(3,3)
(3,4)
(3,5)
(3,6)
(3,7)
(3,8)
(2,0)
(2,8)
(1,0)
(1,8)
0,0)
(0,1)
(0,2)
(0,3)
(0,4)
(0,5)
(0,6)
(0,7)
(0,8)



3. Assume that you are a tourist in Istanbul and you want to visit some regions in both Anatolian and European sides on a single day. We defined some regions below; assume that you are in Region 1 and you want to visit all the other regions, i.e. Region 2, Region 3, Region 4, and Region 5. Given below is the map of Istanbul, in which the areas of interest and the bridges/tunnels between them are marked. You have to use a bridge or a tunnel while going from one region to another. In other words, you are not allowed to walk or use public transportation in between the regions that are not connected via a bridge or a tunnel, i.e. you cannot visit Region 5 (Eyüp) after visiting Region 4 (Topkapı). *Each consecutive district should be reachable through either a bridge or a tunnel*. Additionally, you have to use one of the connected bridges/tunnels, i.e. from Region 2 you can only use Yavuz Sultan Selim Bridge or Fatih Sultan Mehmet Bridge. You can find the related regions and connection list below.

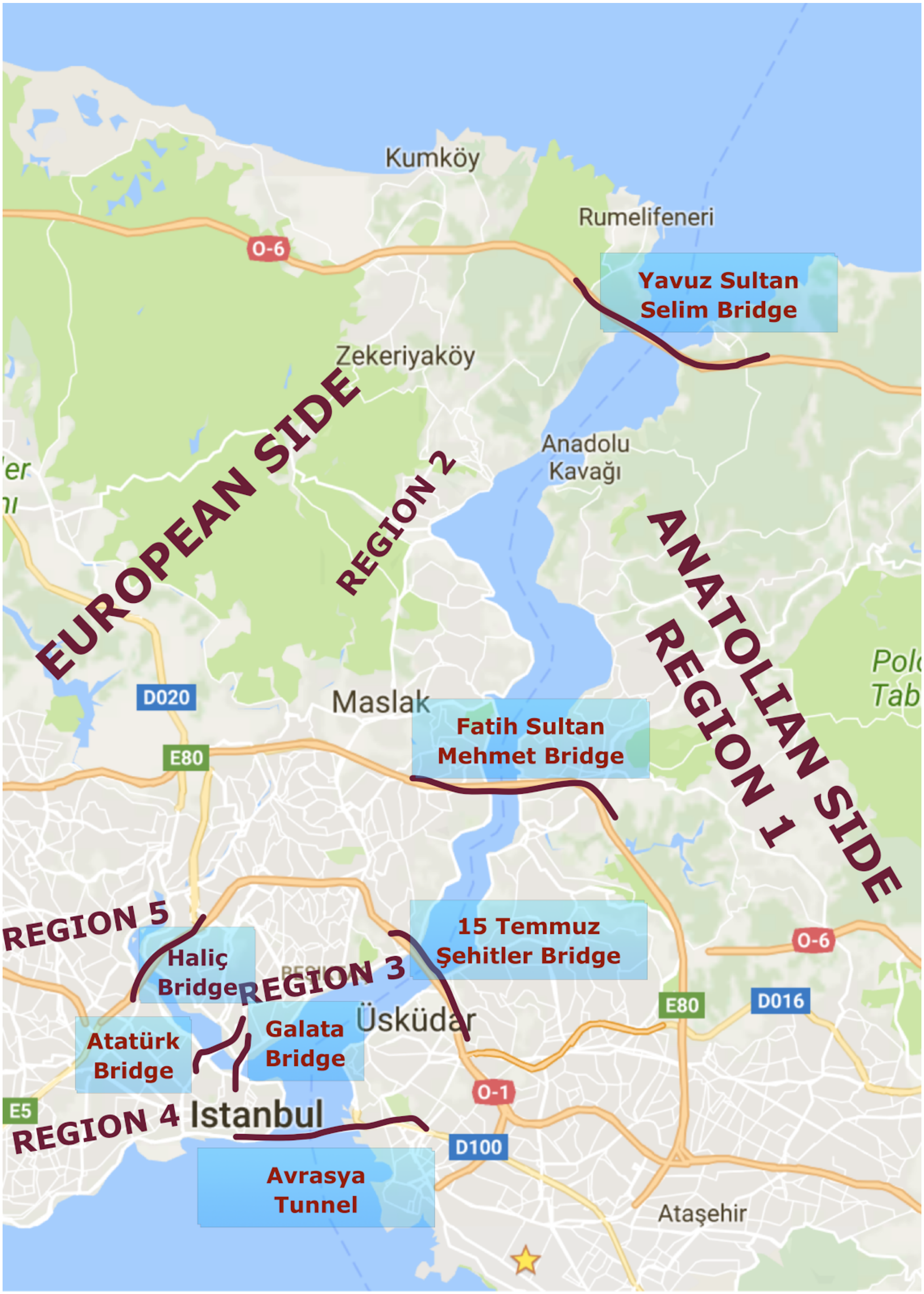
   Can you design a tour satisfying the above-mentioned restriction in which you pass each one of the bridges and the tunnel exactly once starting from Region 1 (you can visit a specific region several times)? If so, provide the path; otherwise, describe why it is not possible and suggest minimum number of new bridge(s) and/or tunnel(s) between the regions so that the path can be constituted. ***You MUST clearly indicate how you utilized the component(s) of computational thinking, otherwise you will NOT get any credits EVEN if your answer is true.***

   Following list defines the regions

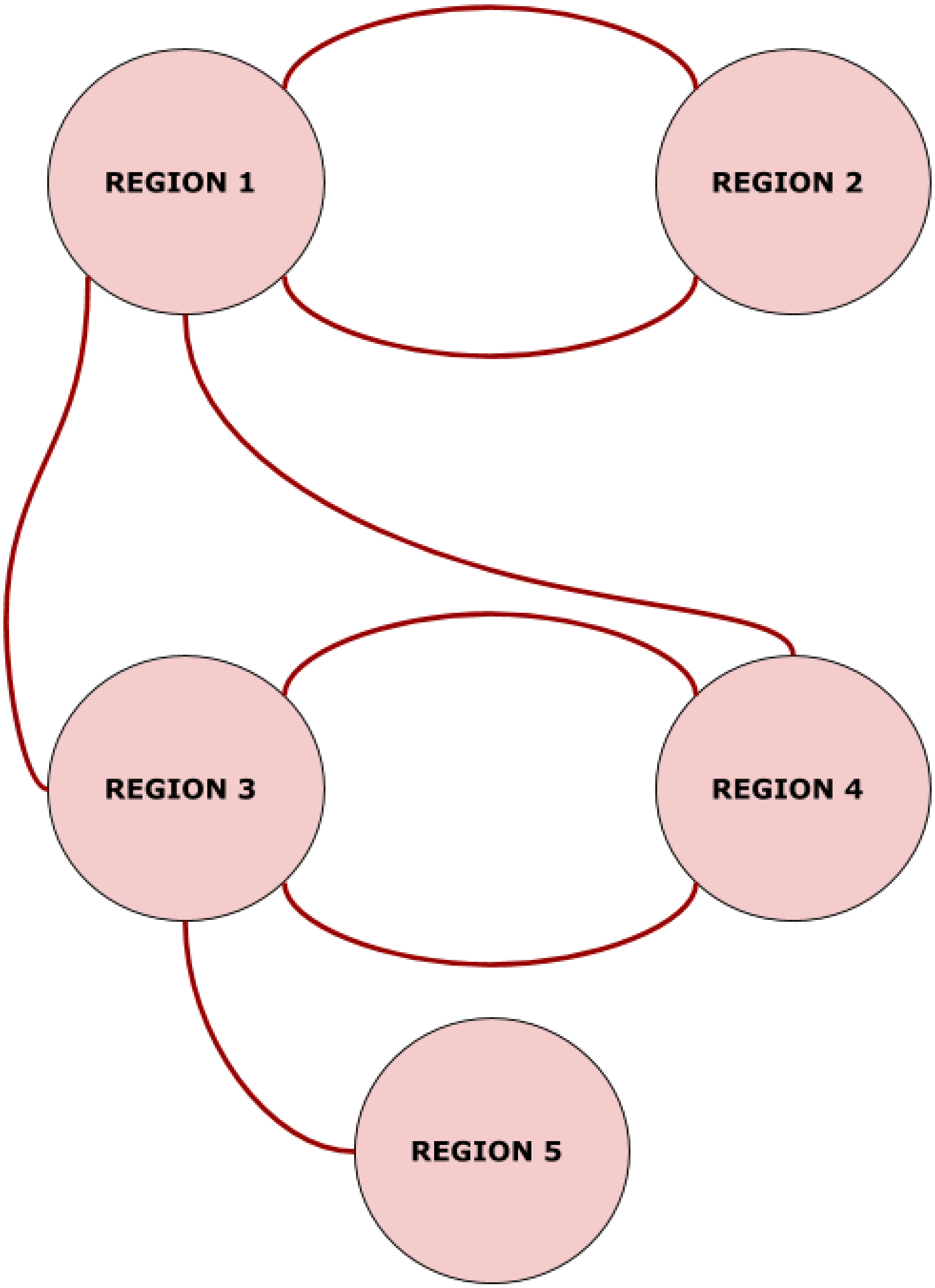
Region 1 → Kadıköy, Üsküdar, Kavacık
Region 2 → Sarıyer, Emirgan
Region 3 → Karaköy, Beşiktaş
Region 4 → Eminönü, Topkapı, Sirkeci
Region 5 → Eyüp, Balat

<u>Following list defines which bridge/tunnel to be taken from which region</u>

- REGION 1 → 15 Temmuz Şehitler Bridge / Avrasya Tunnel / Fatih Sultan Mehmet Bridge / Yavuz Sultan Selim Bridge
- REGION 2 → Fatih Sultan Mehmet Bridge / Yavuz Sultan Selim Bridge /
- REGION 3 → 15 Temmuz Şehitler Bridge / Atatürk Bridge / Galata Bridge / Haliç Bridge
- REGION 4 → Avrasya Tunnel / Atatürk Bridge / Galata Bridge
- REGION 5 → Haliç Bridge


Kumköy
Rumelifeneri
Yavuz Sultan Selim Bridge
O-6
Zekeriyaköy
EUROPEAN SIDE
REGION 2
Anadolu Kavağı
ANATOLIAN SIDE
REGION 1
D020
Maslak
Fatih Sultan Mehmet Bridge
E80
REGION 5
Haliç Bridge
REGION 3
15 Temmuz Şehitler Bridge
O-6
Atatürk Bridge
Galata Bridge
Üsküdar
E80
D016
E5
O-1
REGION 4
Istanbul
D100
Avrasya Tunnel
Ataşehir

We can use abstraction to convert the given map representation into a graph representation. This way, we can get rid of the details that we do not need to know about.



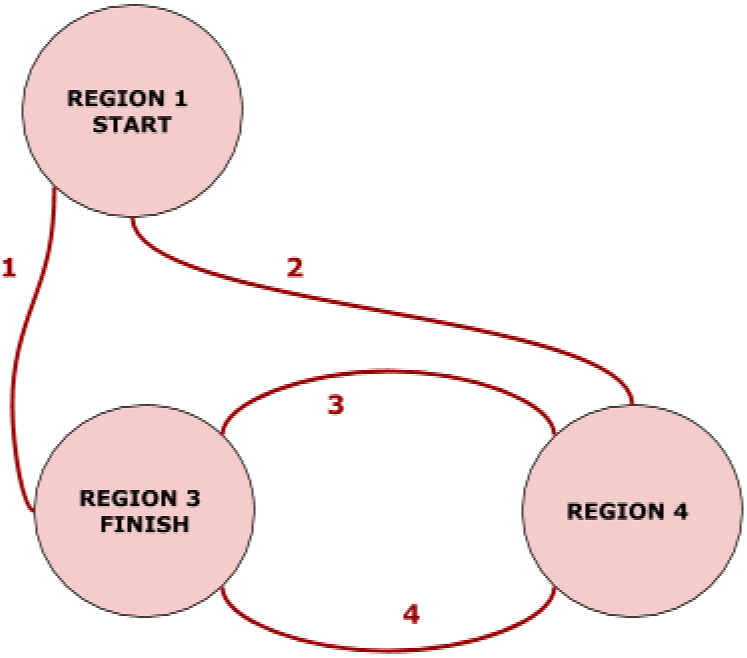
We cannot visit each one of the bridges and the tunnel exactly once starting from Region 1. We can use decomposition here:

- We will start at Region 1 and we should finish at Region 5, since we cannot get back when we go to Region 5.
  Region 1 → …. → Region 5

- The only way to get into Region 5 is to go through Region 3.
  Region 1 → …. → Region 3 → Region 5

- Region 2 is kind of a neutral element, we can loop through Region 1.
  Region 1 → Region 2 → Region 1 → …. → Region 3 → Region 5

- What's left is "start from Region 1 and finish in Region 3 by passing over exactly 4 bridges/tunnels". We can try the possibilities here since there are small number of regions and small number of bridges/tunnels.
  Region 1 → Region 4 → Region 3 → Region 4
  (cannot get back to R.3 & edge #1 missing)
  Region 1 → Region 4 → Region 3 → Region 1
  (cannot get back to R.3 & edge #3 or edge #4 missing)
  Region 1 → Region 3 → Region 4 → Region 1
  (cannot get back to R.3 & edge #3 or edge #4 missing)
  Region 1 → Region 3 → Region 4 → Region 3
  (edge #2 missing)



- Now, we need to check whether we can complete the cycle by adding 1 bridge/tunnel to either of the below given paths

  Region 1 → Region 4 → Region 3 → Region 4 **→ Region 1 → Region 3**

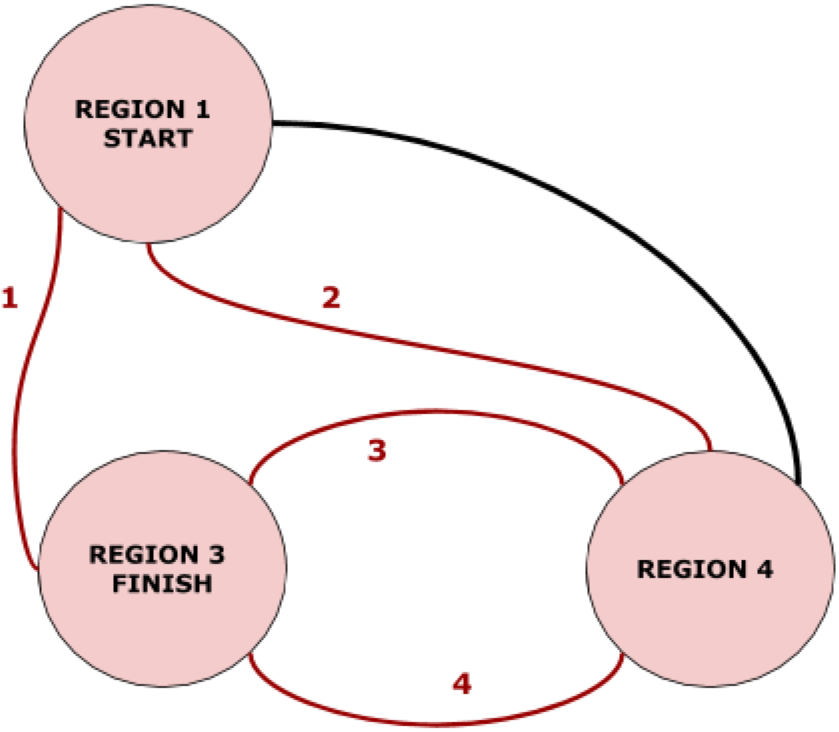
  Region 1 → Region 4 → Region 3 → Region 1 **→ Region 4 → Region 3**

  Region 1 → Region 3 → Region 4 → Region 1 **→ Region 4 → Region 3**

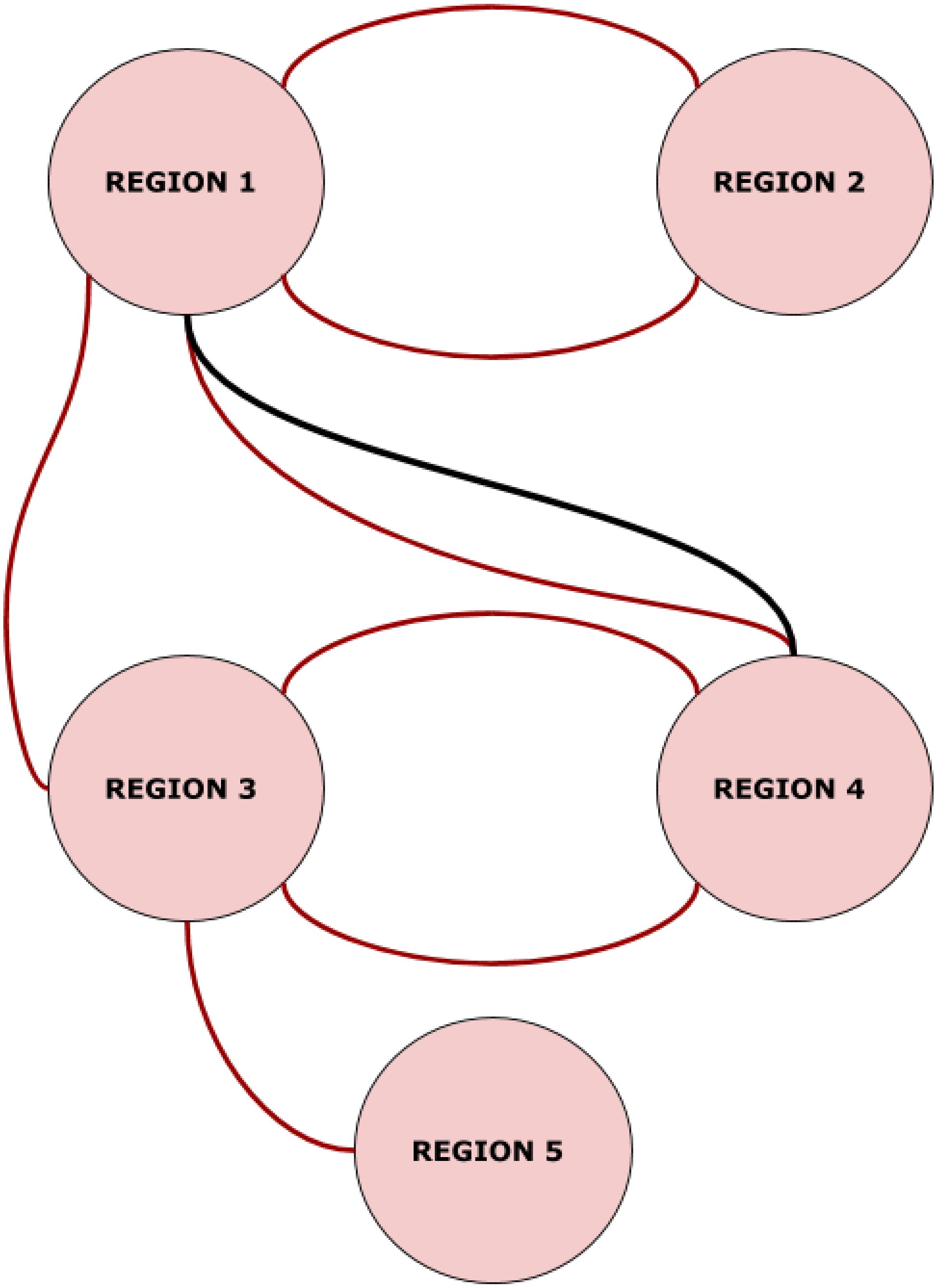
  <u>Thus, we need to add an edge between Region 1 and Region 4.</u>

  Then, there exists another possible path as well:

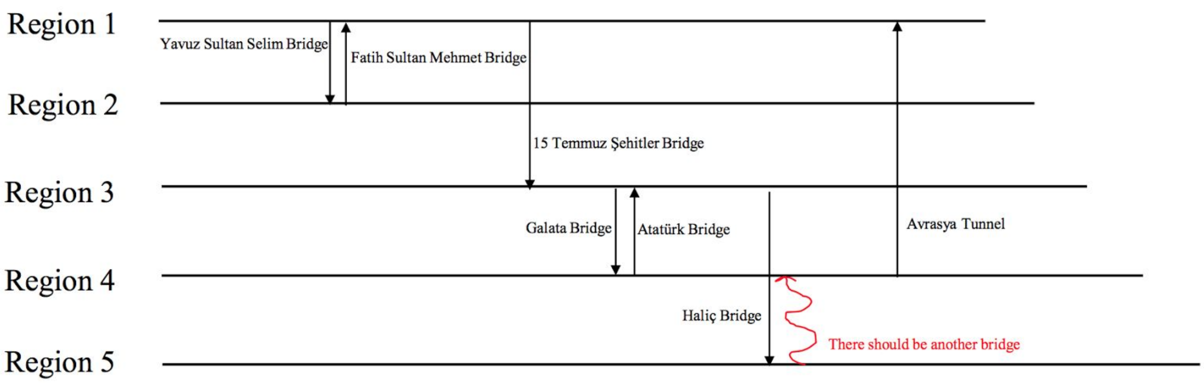
  Region 1 → Region 4 → Region 1 → Region 3 → Region 4 → Region 3



Indeed, the overall graph will look like the following:



One of your friends had a different representation to show the bridge which was needed. We really like the solution and want to share it with you.


Region 1
Region 2
Region 3
Region 4
Region 5
Yavuz Sultan Selim Bridge
Fatih Sultan Mehmet Bridge
15 Temmuz Şehitler Bridge
Galata Bridge
Atatürk Bridge
Avrasya Tunnel
Haliç Bridge
There should be another bridge

What and where to submit?

You should submit a ***pdf*** file that includes all of your solutions. Besides, your solutions should **NOT** be handwritten. Otherwise, your submission will be considered as an *unsuccessful submission*.

You can check the following link to save a *word* file (for Microsoft Office users) as a *pdf* file:
https://support.office.com/en-us/article/Save-or-convert-to-PDF-d85416c5-7d77-4fd6-a216-6f4bf7c7c110#ID0EAADAAA=2016,_2013,_2010

You can check the following link to save a *pages* file (for MacOS users) as a *pdf* file:
https://support.apple.com/en-us/HT202227

The name of the *pdf* file should be as follows: *username_hwNumber.pdf*

For example: if your username is “inancarin”, then the name of the *pdf* file should be: *inancarin_hw1.pdf*

You may visit the office hours if you have any questions regarding submissions.

General Homework Rules

- Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.
- There is NO late submission. You need to submit your homework before the deadline. Please be careful that SuCourse time and your computer time may

have a 1-2 minutes differences. You need to take this time difference into consideration.

- Do NOT submit your homework via email or in hardcopy! SuCourse is the only way that you can submit your homework.
- (For coding homeworks) If your code does not compile, then we cannot grade it and your grade will be 0.
- Plagiarism will not be tolerated. Please check our plagiarism policy given in syllabus of the course.

Good luck!
İnanç Arın, Duygu Karaoğlan Altop, Hüsnü Yenigün