Lab 11 - SSSP

CS2040S Data Structures & Algorithms

AY20/21 Semester 1

Final Week

Reminders

- Don’t forget to give teaching feedback!
- Good luck for your finals!

One Day Assignment 10 Review

One-Day Assignment 10 - Lost Map

- n villages in the region (2 ≤ n ≤ 2500)
- There exists a network of roads that connects all villages, but the map is lost
- You know the shortest distance from each village to every other village
- Can you reconstruct the lost map?

One-Day Assignment 10 - Lost Map

“minimum number of roads have been constructed such that each village can reach every other village via a sequence of roads”

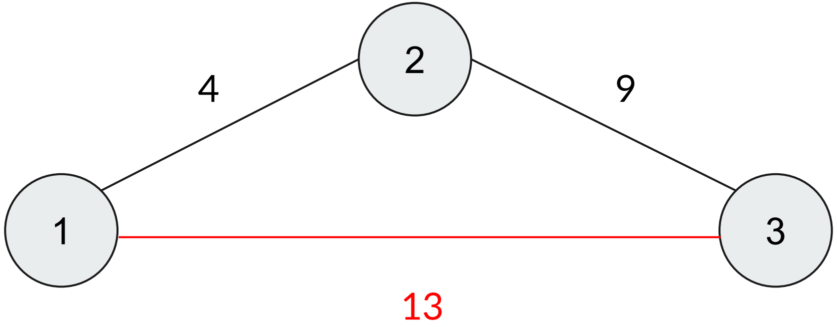
One-Day Assignment 10 - Lost Map

“minimum number of ~~roads~~ edges *have been constructed such that each ~~village~~* vertex *can reach every other ~~village~~* vertex *via a sequence of ~~roads~~* edges*”*

Road constructed forms a Tree!

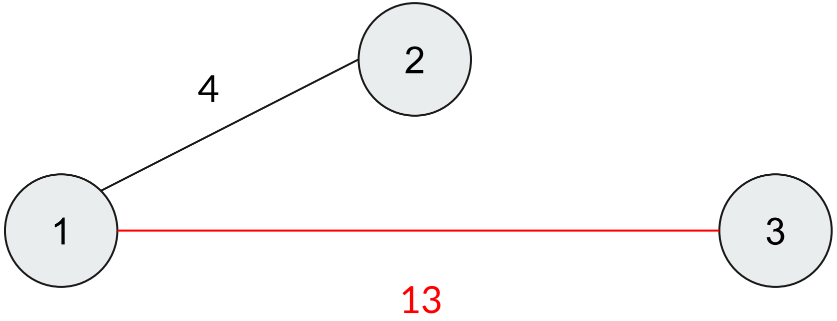
One-Day Assignment 10 - Lost Map

- Adjacency Matrix provided is created from the original road network
  - Given edge u -> v, it represents total distance needed to travel from edge u to edge v, even if there is no direct edge



One-Day Assignment 10 - Lost Map

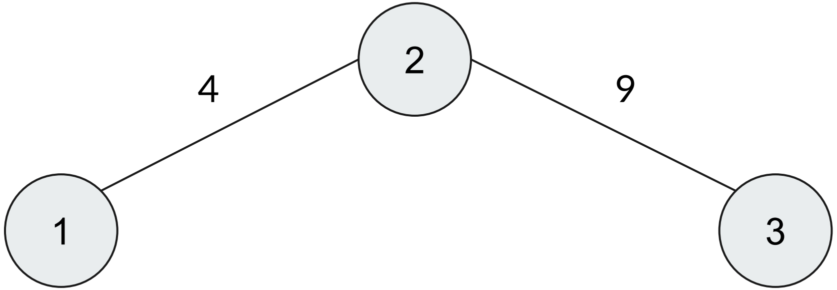
- This implies that any edge not part of the original road network, will connect two vertices with more length than necessary



Connected with total edge weight of 17

One-Day Assignment 10 - Lost Map

- This implies that any edge not part of the original road network, will connect two vertices with more length than necessary



Connected with total edge weight of 13

One-Day Assignment 10 - Lost Map

- This implies that any edge not part of the original road network, will connect two vertices with more length than necessary
- Original road network forms a Minimum Spanning Tree in the given adjacency matrix!

One-Day Assignment 10 - Lost Map

- Implement a MST algorithm of choice
  - Prim’s Algorithm
  - Kruskal’s

Visualgo Quiz

Assessment Details

- 15 Questions
- 25 Minutes
- Best 1 out of 2 tries

Assessment Format

1) Open Chrome and ensure its fullscreened (F11)
2) No other window should be open
3) I'll be sharing the assessment link, copy-paste into the address bar but DO NOT START
4) Once I tell everyone to start, press enter on the address bar and immediately begin the assessment

Assessment Format

6) When you are done, DO NOT SUBMIT
   a) Inform the TAs by typing “done” in the zoom chat
   b) Wait for a TA to come to you, once the TA is ready he will tell you move the camera closer to the monitor, and press submit
   c) Display the result score screen until the TAs are done recording the score
   d) We will ask whether you want to go for another round

Assessment Format - Round 2

7) Those who want 2nd round will have to wait for everyone else to finish before second round will begin
8) 2nd Round will follow the same format
9) Those who are done can look at the next lab questions/take a break until the 2nd round is done

Visualgo Quiz

Single Source Shortest Paths

SSSP

- Many SSSP algorithms discussed in lecture
  - DFS
  - BFS
  - Bellman-Ford
  - Dijkstra’s (Original/Modified)
- Importance is knowing when to use each of them
  - Depends on the graph

SSSP Examples

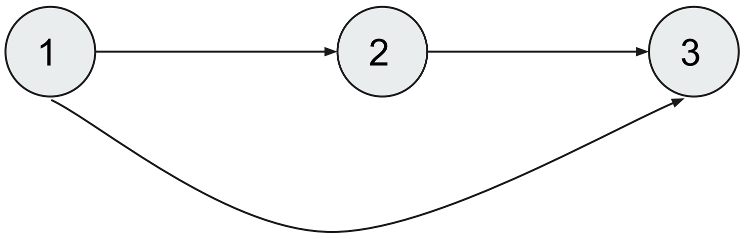
Graph is a Tree

- Use BFS/DFS
- Why?
    - There is only a single path from one vertex to any other vertex
    - Simple traversal is sufficient to retrieve this path length

SSSP Examples

Graph is a Unweighted

- Will DFS Work? (Graph is not a tree)
- No!
- Why?
    - Non-tree graphs may have multiple paths between two vertices



Output of DFS produced affected by order of traversal!

SSSP Examples

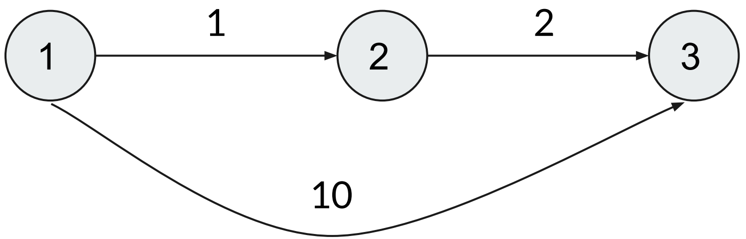
Graph is a Unweighted

- Use BFS
- Why?
  - In an unweighted graph, shortest distance between two vertices = minimum number of edges required to traverse between them
  - BFS visits all vertices in the order of increasing number of edges traversed
  - BFS will ensure destination vertex is visited at minimum distance possible

SSSP Examples

Graph is a DAG

- Will BFS work? (Graph is weighted now)
- No!
- Why?
    - In a weighted graph, minimum distance between two vertices is no longer equal to number of edges between them



SSSP Examples

Graph is a DAG

- Use One-Pass Bellman Ford
- Why?
  - Acyclic -> Topological order exists
  - Processing edges in this topological order ensures that by the time you reach vertex v, you have already processed all vertices that can be visited before v
  - Since we know the shortest distance to all vertices that precede v, calculating shortest path till vertex v is trivial

SSSP Examples

Graph has no negative weight edges

- Use Dijkstra
- Why?
    - Greedy property that works when all edge weights are positive
    - Faster
        - Bellman Ford runs in O(VE)
        - Dijkstra runs in O(V + E log(V))

SSSP Examples

Graph has no negative weight cycle

- Use Modified Dijkstra
- ..or just use Bellman Ford’s
    - Graph may cause modified dijkstra to become inefficient

SSSP Examples

Any other graph

- Use Bellman Ford

Dijkstra Implementation

- Java's PriorityQueue does NOT implement decreaseKey operation
- Make use of TreeSet data structure to simulate decreaseKey
  - Remove the old element
  - Add back the new element with different weight
- Alternatively, use Lazy Deletion
  - Track "best distance estimate" per-vertex in array
  - Compare de-queued value with current estimate
    - If not equal, reject the old value

One Day Assignment

One-Day Assignment 11 - Human Cannonball Run

- You are given an starting and ending coordinate
- Find the shortest time taken to reach the end
- Sounds easy right?

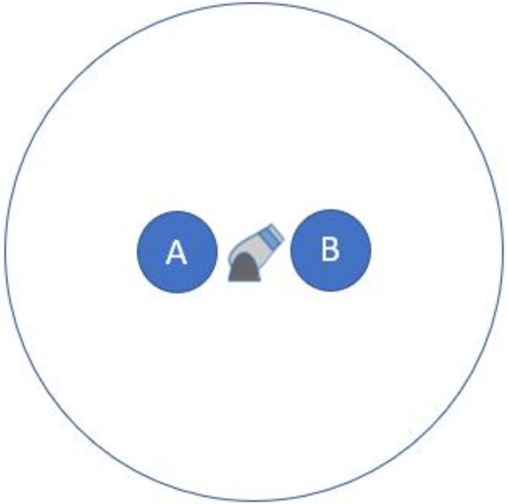
One-Day Assignment 11 - Human Cannonball Run

- Two methods of travelling
    - Running, 5m/s
    - Cannon, 50m in 2 seconds
        - Cannot stop halfway!
        - n cannons exist

One-Day Assignment 11 - Human Cannonball Run

- Idea 1: Always take cannons to travel
  - You can travel 5x faster using cannons
- Counterexample:

Clearly, running is faster!



One-Day Assignment 11 - Human Cannonball Run

- Instead, model this problem as an SSSP problem
- Main difficulty comes from modelling this question as a graph
    - Cannons can fire in any directions
    - How can we limit the number of edges?
        - Consider only the “essential” information

Human Cannonball Run

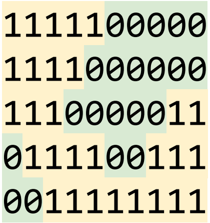
- What are my “points of interest”/vertices?
- How should I travel between points of interest?
  - Sometimes it’s faster to take a cannon jump and overshoot
  - Then run back a bit

Assignment Guidelines

- Include your **name** and **student number** in comments at the top of your code.
- You are allowed (and encouraged) to discuss algorithms
  - List down all your collaborators in your source code
- **You are NOT allowed to:**
  - **Copy another person's code**
  - **Look at another person's code**
  - **Use another person's code as a base for your own code**
- Plagiarism checks will be in place

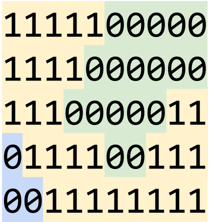
Take-Home Lab 4A: 10 Kinds of People

- Grid of RxC cells
  - Cells are either type 1 or type 0
- N queries
  - Positions of 2 cells A and B
  - Can I get from A to B?
    - Without changing types



```
1111100000
1111000000
1110000011
0111100111
0011111111
```

Take-Home Lab 4A: 10 Kinds of People



```
1111100000
1111000000
1110000011
0111100111
0011111111
```

- Can I get from A to B?
  - Without changing types
- May not always be able to go between any pair of 0s
  - Bottom left 0, top right 0
  - Cannot “jump” over the intervening 1s
  - Different “**islands**” of 0s

Islands = Connected components

- BFS/DFS over whole graph to find CCs (tracking 0/1ness of CC)
  - If both cells in same CC, possible, so print 0/1ness

Take-Home Lab 4B: Millionaire Madness

- RxC grid, of stacks of coins
    - Stacks have different (integer) height
- I can go 1 step up/down/left/right
    - If (next stack - current stack) <= ladder height
        - If next stack shorter, always possible
- Start at top left, end at bottom right
- Minimize ladder height needed

No ladder == “Height 0”

If you can play it slowly you can play it quickly

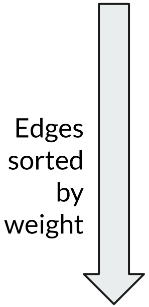
- Let’s make the question slightly more general
  - Start/end positions are no longer fixed

- **Can we get from any point A to any point B?**
- Minimize ladder height needed

Ladders

- Ladder heights are **non-negative integers**
- Can we get from any point A to any point B?

IsConnected(A,B)



Edges sorted by weight

  - ○ If my ladder is height 0?
  - ○ If my ladder is height 1?
  - ○ If my ladder is height 2?
  - ○ …

What changes, when I increase my ladder height (H-1) -> H?

- What is now usable?
  - ○ Height differences of +H can now be used
- Do I lose anything?
  - ○ Longer ladders can be used in place of shorter ones.

AddEdge(X,Y)

- Maybe we can track “usable so far” subgraph…
  - ○ What DS do we need here?

Dynamic Graph Connectivity

Effectively Kruskal’s!

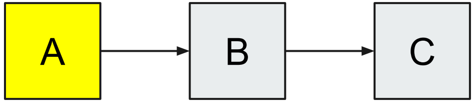
Slight Improvements

- We don’t actually need the **full** MST
    - Just enough to get from (A = top left) to (B = bottom right)
    - Run Kruskal’s until A,B are connected
    - Run Prim’s from A, stop early when we reach B

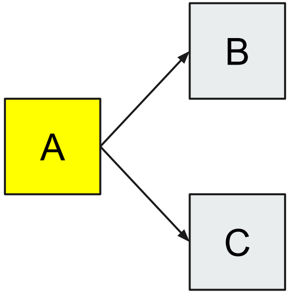
Take-Home Lab 4C: Dominos

- N dominos
- M relationships between dominoes
  - Domino A gets hit => Will knock down domino B
- Obvious graph model
  - Domino => vertex
  - Will-knockdown => directed edge
- How many dominoes need to be **manually** knocked over?
  - Corresponding concept in graph model?

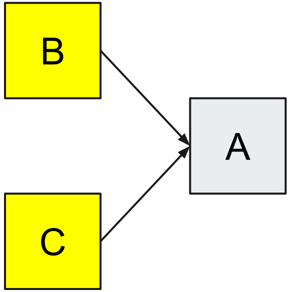
Examples



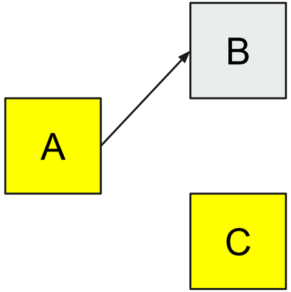
A knocks over B, which knocks over C.



A knocks over both B and C.



We need to knock down both B and C, even if A is hit by either.



A only knocks down B, but not C.

No Cycles

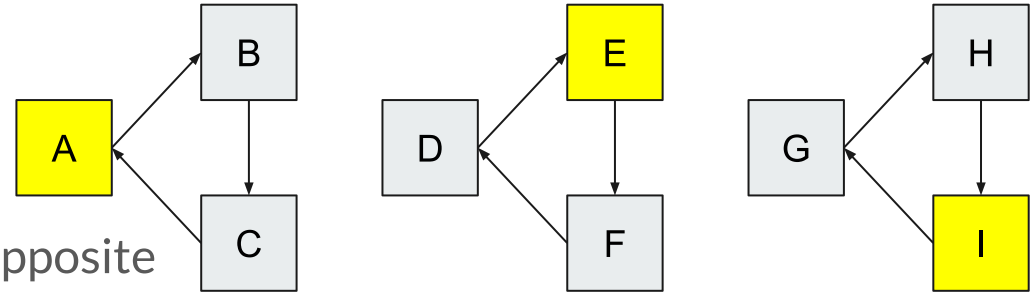
- If we don't have any domino "cycles"
  - Which dominoes are **necessarily** to manually hit?
    - Can they be knocked over by another domino?
  - Which dominoes are **sufficient** to manually hit?
    - For any domino, can we hit it by cascading instead?
- How does this idea carry over into our graph model?

Directed Acyclic Graph

Vertices with 0 incoming edges are **necessary**, and **sufficient**.

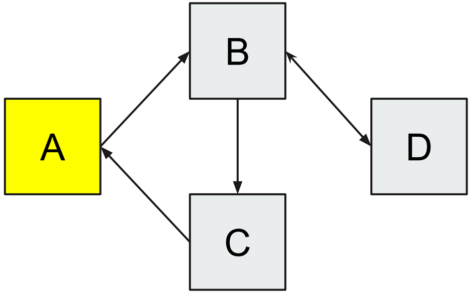
"Toposort" the DAG
Find vertices with in-degree 0

All Cycles



- Let’s try the complete opposite
  - Bunch of dominoes are arranged in a circle
- If we have a “cycle” of dominoes knocking down each other
  - Does it matter which we start with first?
- This still works if we have separate cycle
  - Pick 1 per cycle
  - “Shrink” cycle down to that single representative

Almost-Cycles



- For the previous to work, we don't actually need exact cycles
    - What exactly do we want?
    - Any choice in here, can "hit" all other dominoes in this "almost-cycle"
        - No matter what choice it is
- What does this correspond to in our graph model?

Strongly Connected Component

Some Cycles?

✓ No domino cycles

“Toposort” the DAG
Find vertices with in-degree 0

- ○ Find which ones are **necessary**/sufficient

❓ Some domino cycles

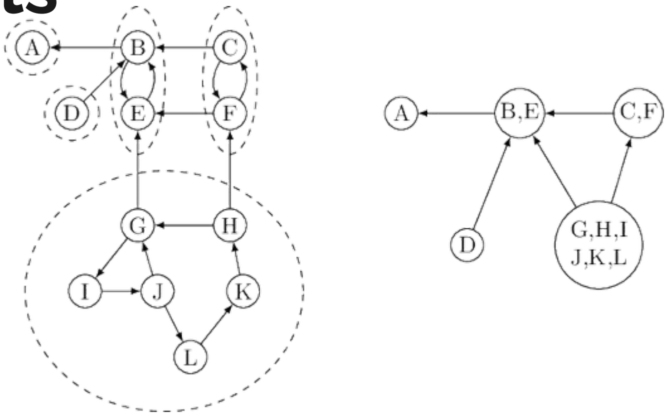
- ○ Knock down 1 representative per **necessary** “almost-cycle”?????

✓ All dominos in separate “almost-cycles”

Pick 1 representative per SCC

- ○ Knock down 1 representative per “almost-cycle”

Strongly Connected Components



- Running Kosaraju’s/Tarjan’s on a graph
    - Groups all vertices in an SCC together
    - Finds one-way “links” between SCCs
        - This is the part to **augment/add** to SCC code
- Effectively, we have “contracted” every SCC into a single vertex
    - Any cycle is part of exactly 1 SCC, but shrunk to 1 vertex
    - DAG!

Some Cycles?

✓ No domino cycles

- ○ Find which ones are **necessary**/sufficient

“Toposort” the DAG
Find vertices with in-degree 0

❓ Some domino cycles

- ○ Knock down 1 representative per **necessary** “almost-cycle”?????

Necessary SCCs have in-degree 0 in SCC graph.
Pick 1 representative per **necessary** SCC

✓ All dominos in separate “almost-cycles”

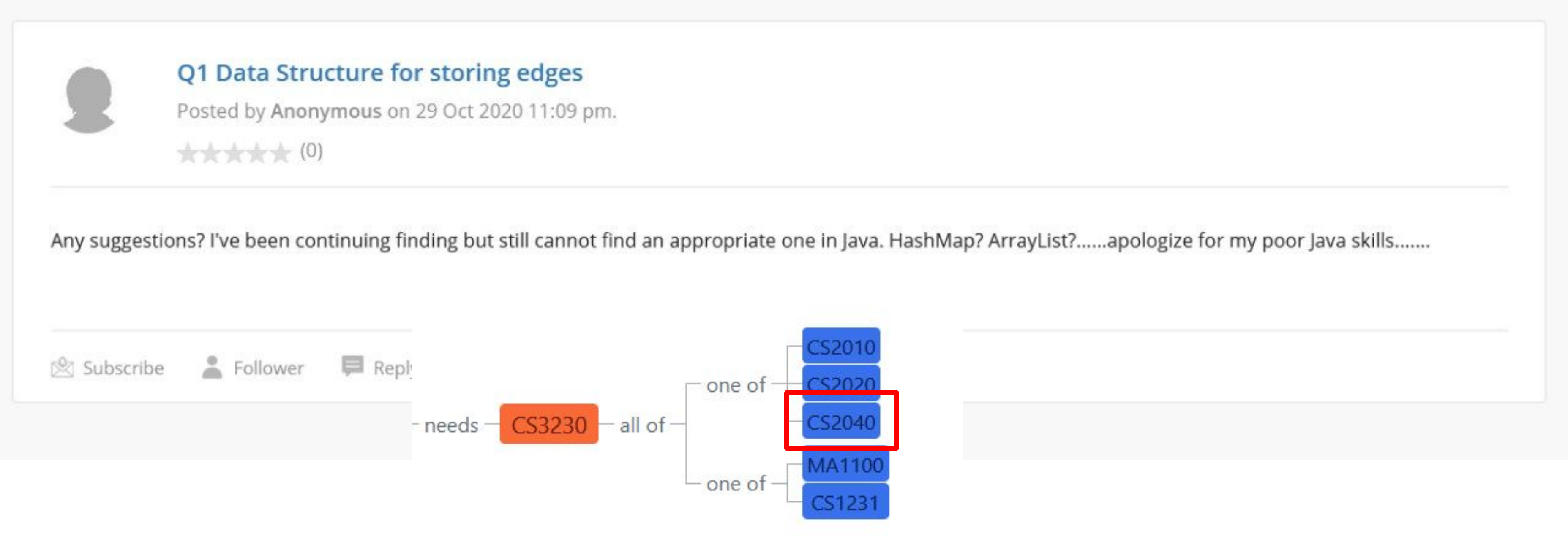
Pick 1 representative per SCC

- ○ Knock down 1 representative per “almost-cycle”

Before we end the semester...

- DS & Algorithm is a crucial skill
    - Needed for interviews
    - Needed for future mods
        - CS3230

Question posted on CS3230 forum



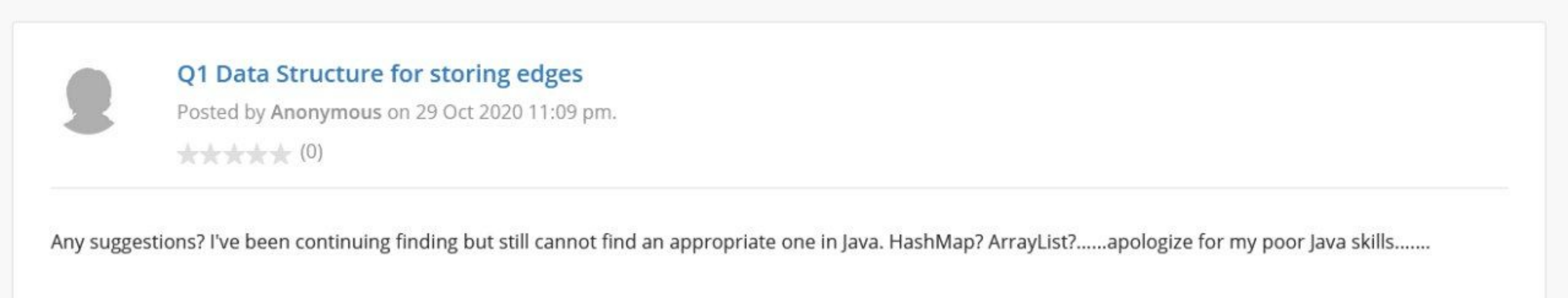
Q1 Data Structure for storing edges

Posted by Anonymous on 29 Oct 2020 11:09 pm.

(0)

Any suggestions? I've been continuing finding but still cannot find an appropriate one in Java. HashMap? ArrayList?......apologize for my poor Java skills.......

Question posted on CS3230 forum

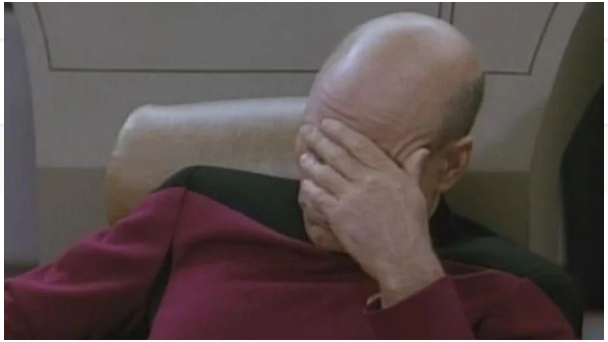


Q1 Data Structure for storing edges

Posted by **Anonymous** on 29 Oct 2020 11:09 pm.

★★★★★ (0)

Any suggestions? I've been continuing finding but still cannot find an appropriate one in Java. HashMap? ArrayList?......apologize for my poor Java skills.......



Before we end the semester...

- Try to revise/remember the content even after this module!