Thursday, 12 October 2017

LAB DEMO 06

PS3 Debrief – Common Mistakes

Typical common mistakes in PS3:

- WA/RTE in B/C/D: Trapped in corner cases
- TLE in C: Forgot to convert Adjacency Matrix to Adjacency List
  - UFDS solution (yes, it is possible ☺) should not get TLE as it has similar time complexity
  - UFDS solution may be slightly slower due to constant factor overhead of UFDS object creation and deletion ☹
    - Solution (Software Engineering purist may complain): Avoid using OOP…
      - Yeah, this is a bad software engineering practice
- TLE in D: Expected, this requires a totally different algorithm
  - See the next slide

PS3 Debrief – Our Answer

The ultimate solution for PS3 Subtask D:

- Tarjan's algorithm for finding articulation points (cut vertices) of an undirected graph in O(V+E)
  - Hopcroft, J.; Tarjan, R. (1973). "Efficient algorithms for graph manipulation". *Communications of the ACM* **16** (6): 372–378.

The standard solution for PS3 Subtask A+B+C:

- Convert the graph from Adjacency Matrix to Adjacency List
- For each vertex, try (virtually) blocking it, then runs O(**V**+**E**) DFS/BFS to see if the number of connected components increase (not necessarily to two, but can be more than two)
  - You can also use UFDS for this, but be careful of high constant factor

PS4 Overview

PS4 is already out since last Friday, 6 Oct 2017,

- It is about Minimum Spanning Tree++
- The last subtask is a mini challenge ☺

Let's review: https://visualgo.net/en/mst

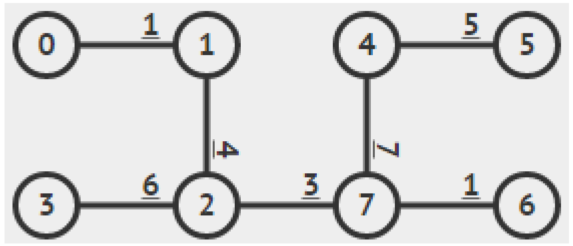
PS4 Status (as of 12 Oct, 2am)

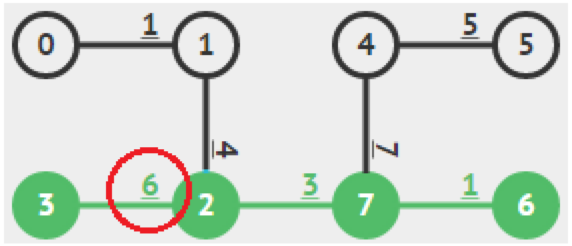
Name	A	B	C	D
Passed D	-	-	-	1
Passed C	-	-	5	
Passed B	-	-		
Passed A	-			
The rest				

Time to return to CS2010… This is week08 already

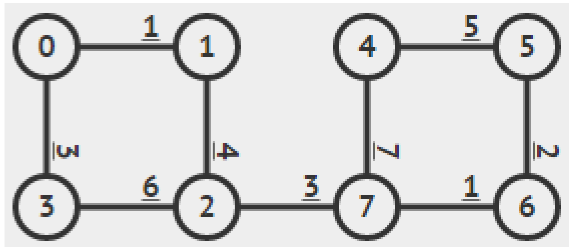
PS4 Subtask A+C Discussion

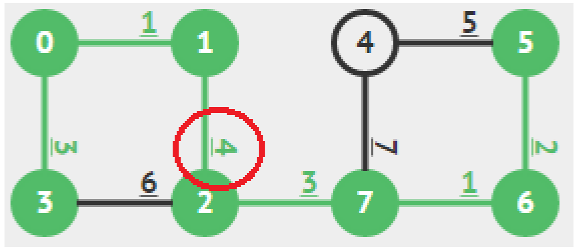
Review solution for Subtask A (from tutorial)





What kind of tree can be used to solve Subtask C (+B)?





- Answer: MST (you can use either PrimDemo or KruskalDemo with proper—but minor—modifications), then reuse Subtask A solution

Review of Prim’s & Kruskal’s Code

Now let’s review **PrimDemo.java** and **KruskalDemo.java** that were briefly presented in Lecture on Week 07

You will implement at least one of them for PS4 (your choice)

Discussion: How to **improve** PrimDemo.java and KruskalDemo.java to better forms?

- Hint: Consider a complete graph with $\mathbf{E} = O(\mathbf{V}^2)$
- Can you make especially Prim’s algorithm run faster than $O(\mathbf{V}^2 \log \mathbf{V})$? How about Kruskal’s algorithm?
- Especially when the edge weights are random (not uniform)

PS4 Subtask B Simple Solution

Discussion of Subtask B →
anyone know a "four-lines solution" (from CP3)?

- Answer: Floyd Warshall's (modified), see CP3 page 159
- We will revisit this 4 lines wonder on Week 12…
- But you can use this now to get extra 10 marks ☺

```
for (k = 0 to V-1)
  for (i = 0 to V-1)
    for (j = 0 to V-1)
      M[i][j] = min(M[i][j], max(M[i][k], M[k][j]));
```

PS4 Subtask D Discussion

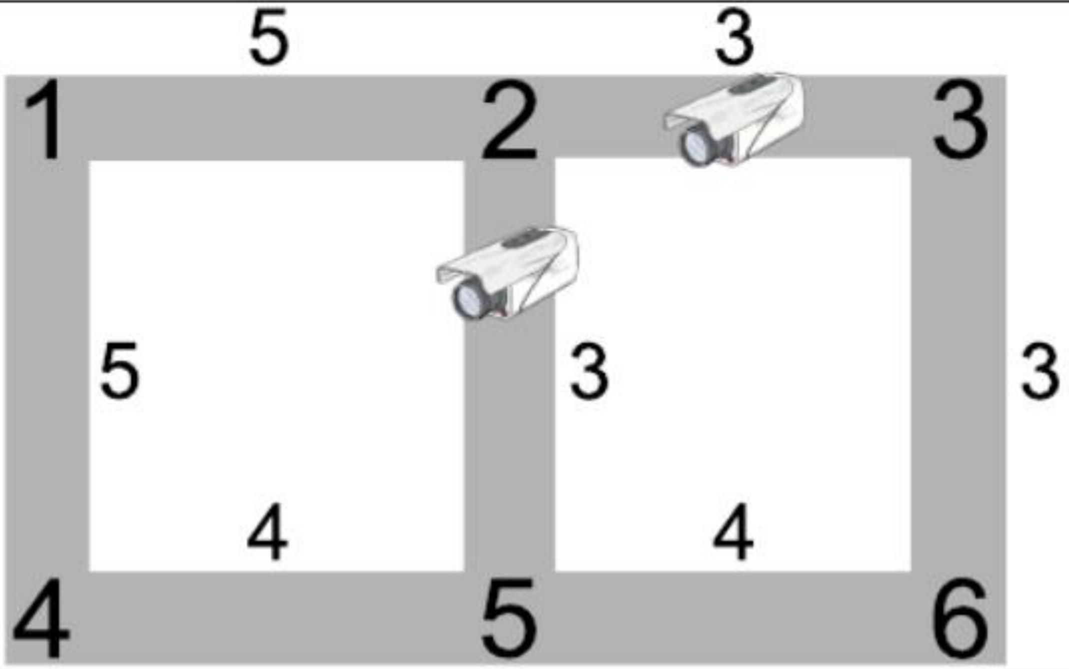
What is so difficult about subtask D?

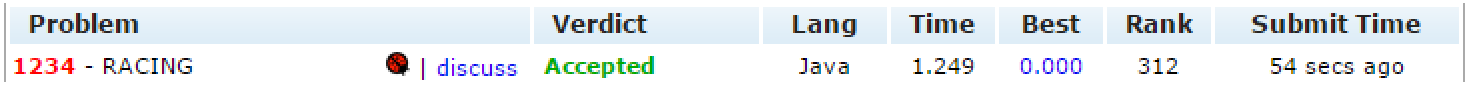
- A: #Q goes up from 5 to 100 000 (a gigantic increase)
- Hint: You need **O(1)** per query… but how?
- Think about it first

Live Demo to Solve UVa 1234

The same problem as "Vehicle Monitoring System" discussed in tutorial this week

Figure 5 Illustration of sample input and one possible optimal placement of cameras





Problem		Verdict	Lang	Time	Best	Rank	Submit Time
1234 - RACING	| discuss	Accepted	Java	1.249	0.000	312	54 secs ago