CPSC 457 T01/T04

Xining Chen

Deadlock Detector

5 -> 1

- Add 5 as a node into std::unordered_map adj_list
- Add 1 as a node into std::unordered_map adj_list
- Increment node 5 out count in out_counts list

- Missing a step?

Deadlock Detector

5 -> 1

- Add 5 as a node into std::unordered_map adj_list
- Add 1 as a node into std::unordered_map adj_list
- Increment node 5 out count in out_counts list

- out_counts is an unordered_map. Therefore node 1 is **not** recorded in out_counts.
- Need “dummy” / “filler” code – to add node 1 into out_counts, initialized with count to 0.

Deadlock Detector

- Q: Is it necessary to distinguish processes vs resources?
- A: Yes. When printing output, **only print processes**.

- Q: How do you distinguish them?
- A: Infinite number of ways. One option is:

  prepend the node name with string ‘p_’ and ‘r_’ for process and resources respectively. I.e. p_x is a process with name x, r_x is a process with name x.

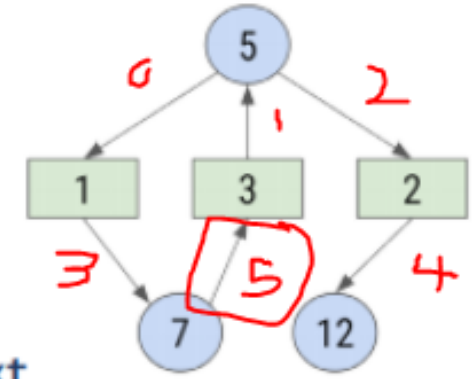
  p_x and r_x are clearly different strings. Hence hash values will be different.

Deadlock Detector

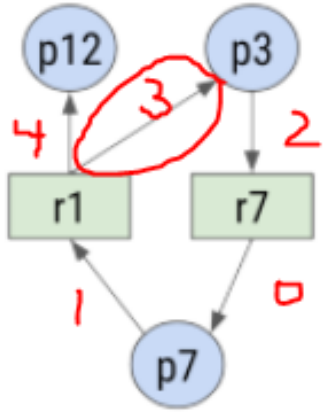
- Q: What's an edge index?
- A:

Few more examples:

```
$ cat test2.txt
5 -> 1
5 <- 3
5 -> 2
7 <- 1
12 <- 2
7 -> 3
$ ./deadlock < test2.txt
edge_index : 5
cycle      : [5,7]
```



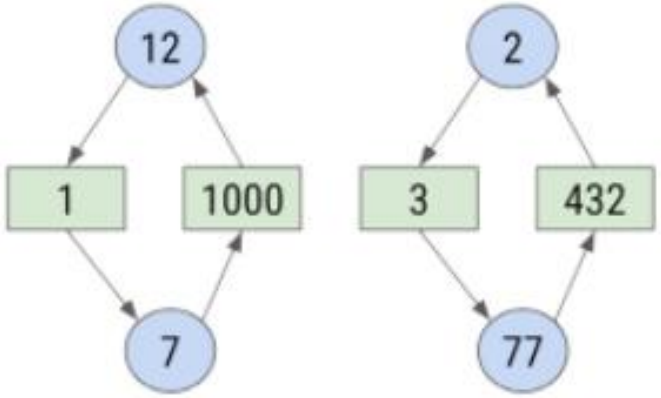

```
$ cat test3.txt
p7 <- r7
p7 -> r1
p3 -> r7
p3 <- r1
p12 <- r1
$ ./deadlock < test3.txt
edge_index : 3
cycle      : [p7,p3]
```



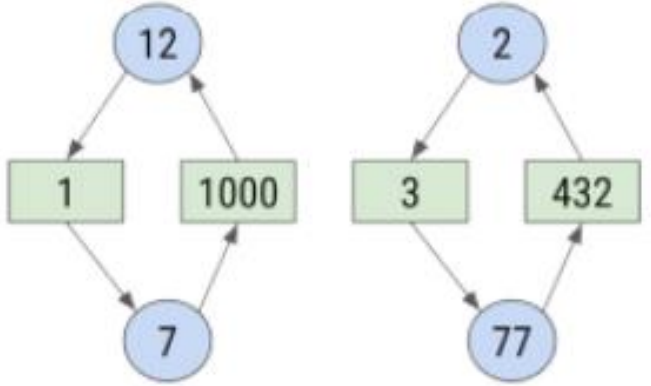
Deadlock Detector

- Exercise: Draw and figure out the edge_index. Does it match the computed edge_index?

```
$ cat test4.txt
12 -> 1
12 <- 1000
7 -> 1000
7 <- 1
2 -> 3
2 <- 432
77 -> 432
77 <- 3
$ ./deadlock < test4.txt
edge_index : 3
cycle      : [12,7]
```




```
$ cat test5.txt
12 -> 1
12 <- 1000
7 -> 1000
2 -> 3
2 <- 432
77 -> 432
77 <- 3
7 <- 1
$ ./deadlock < test5.txt
edge_index : 6
cycle      : [2,77]
```

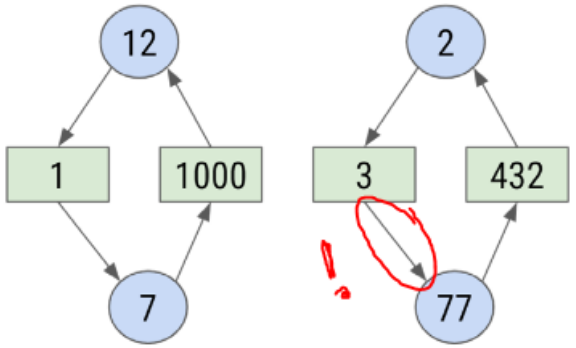


Simple “trivial” exercises are important. They are low time cost (1-3 mins) and ensures you truly do understand the concepts.

Deadlock Detector

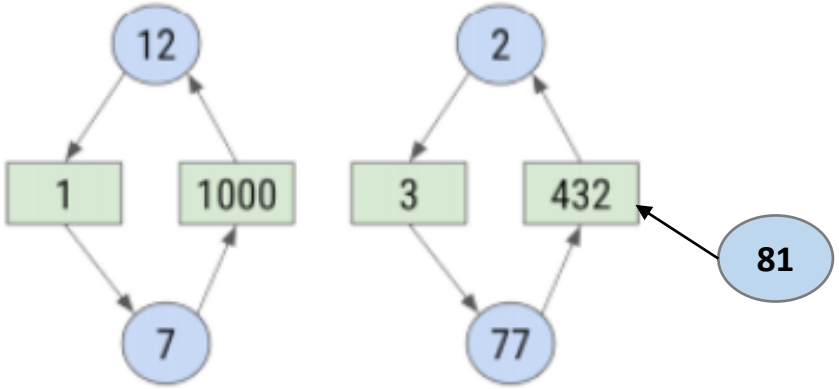
- Q: In test5.txt, can I report the cycle 12, 7 instead of 2, 77? Would that be correct?
- A: No. That is incorrect because it is not the first deadlock. The **first deadlock** occurs on the right cycle. You must report the right cycle.

```
$ cat test5.txt
12 -> 1
12 <- 1000
7 -> 1000
2 -> 3
2 <- 432
77 -> 432
77 <- 3
7 <- 1
$ ./deadlock < test5.txt
edge_index : 6
cycle      : [2,77]
```



Deadlock Detector

- Question: Does the following process count as being stuck in a deadlock (& therefore should be printed?)
- A:



Yes! Process 81 is stuck in a deadlock.

```
$ cat myTest5.txt
12 -> 1
12 <- 100
7 -> 1000
2 -> 3
2 <- 432
77 -> 432
81 -> 432
77 <- 3
7 <- 1
```

```
$ ./deadlock < myTest5.txt
edge index : 6
cycle      :  [2, 77, 81]
```

Deadlock Detector

- Q: My code is very slow on test6.txt. Why is that?
- A: Probably didn’t implement optimization #1 for the topological sort. See tutorial slides CPSC 457 – Assignment 4.pdf.

Deadlock Detector

- Checks:
  1. Before implementing topological sort, ensure your graph is processed correctly.

```
Result detect_deadlock(const std::vector<std::string> & edges);
```

```
class Graph {
    std::unordered_map<std::string, std::vector<std::string>> adj_list;
    std::unordered_map<std::string, int> out_counts;
    ...
} graph;
```

Are processes and nodes added to adj_list correctly?

Are the out_counts set correctly?

Can you traverse through both maps and print adj_list and out_counts?

Deadlock Detector

- Checks:
  2. Implement topological sort. Run it on the **final** graph. Easier to debug if you have 1 topological sort per call, instead of 1 per edge added. (But edge_index will be wrong, but that's ok).
     Goal here: ensure topological sort works.

```
out = out_counts # copy out_counts so that we can modify it
zeros[] = find all nodes in graph with outdegree == 0
while zeros is not empty:
 n = remove one entry from zeros[]
 for every n2 of adj_list[n]:
   out[n2] --
   if out[n2] == 0:
     append n2 to zeros[]
cycle[] = all nodes n that represent a process and out[n]>0
```

Deadlock Detector

- Checks:
  3. Move topological_sort() call to a different line in your code. It should be called after an edge is processed and added to Graph.

     Check that the edge_index output is correct.
     Output of cycle is correct (**print only processes in cycle**)

Deadlock Detector

- Checks:
  4. Convert into integer graph.

```
class FastGraph {
    std::vector<std::vector<int>> adj_list;
    std::vector<int> out_counts;
    ...
} graph;
```

Everything else stays the same. Might have to change some data types of functions.

1 Extra step: convert from integer (hash value) back to string for cycle[].

Scheduler simulation

- Talk about Appendix 2.
- Switch to ipad.

Next time:

- Open tutorial for 1-1 help.