Lab5 Solutions

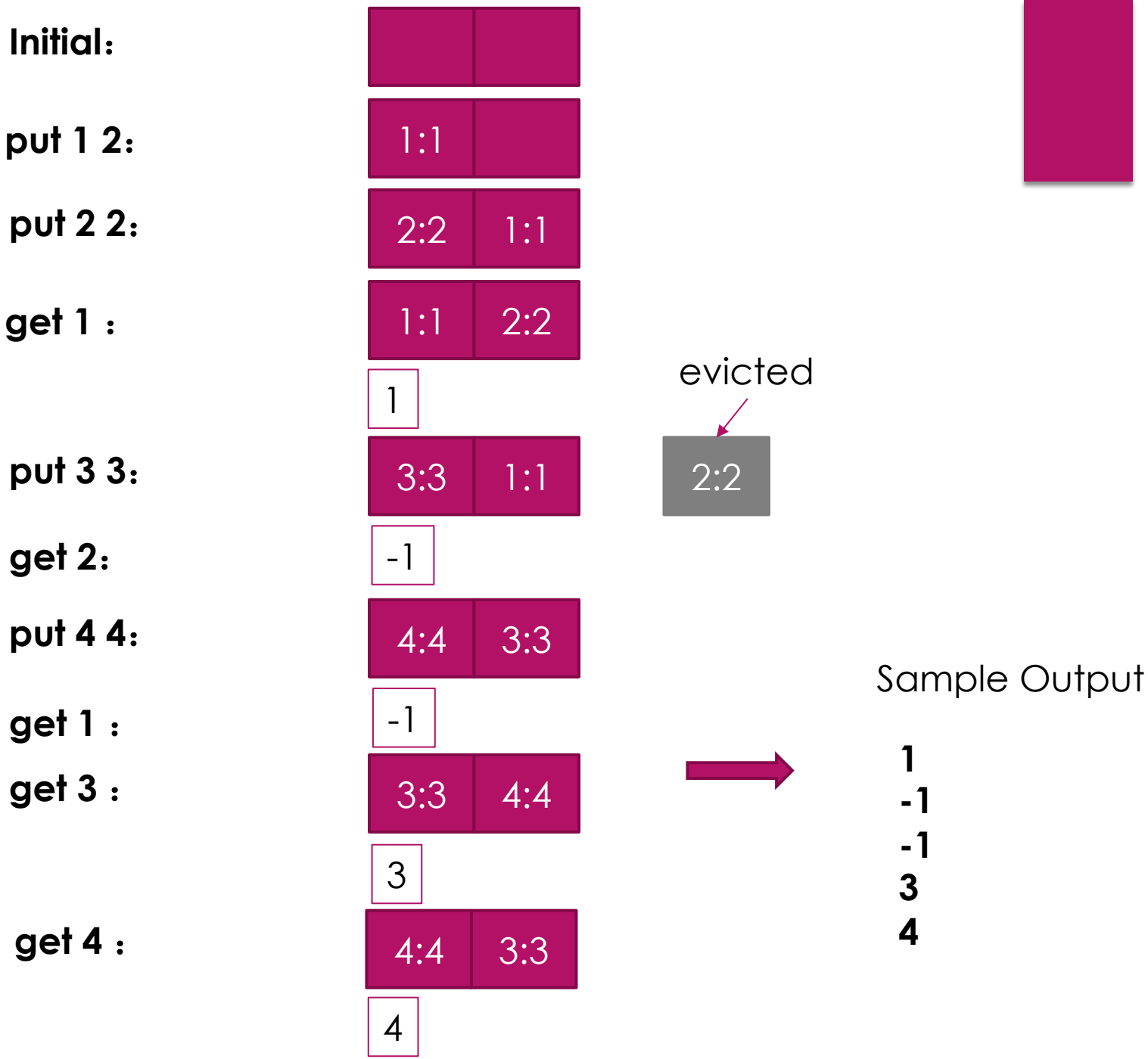
YAO ZHAO

Lab5.A: LRU Cache

- Design a data structure that follows the constraints of a **Least Recently Used (LRU)** cache.
- Implement a LRUCache with capacity N.
- There are M operations including two type:
  - **get key :** Print the value of the key if the key exists, otherwise print **-1**.
  - **put key value** : Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, evict the least recently used key.
- The operations get and put must each run in O(1) average time complexity.
- **Please do not use LinkedHashMap in java.**

Sample Input

2 9
put 1 1
put 2 2
get 1
put 3 3
get 2
put 4 4
get 1
get 3
get 4



Sample Output

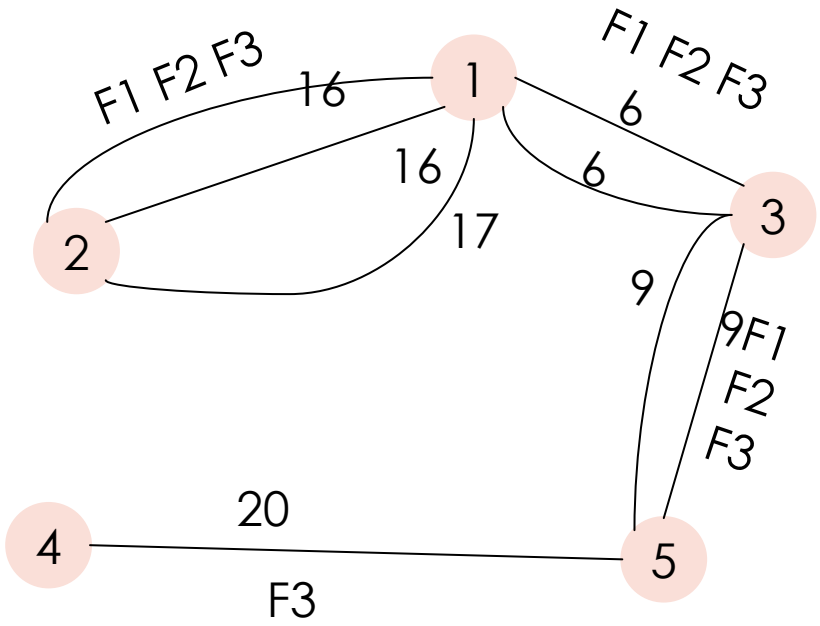
1
-1
-1
3
4

Lab5.B: CHAO MAN

- One day **CHAO MAN** decides to do an experiment. He takes his P followers to a maze. The maze is a connected, undirected, and weighted graph with N nodes and M edges, where the i^{th} follower of **CHAO MAN** is initially at node s_i and wants to reach node t_i.
- Then **CHAO MAN** orders: "RUN!" And all his followers rush to their destination just like arrows off the string. As **CHAO MAN**'s fans, they are also super smart, so each of them would definitely choose the shortest route. If there are multiple shortest route for a fan, he or she can choose anyone of them.
- Now **CHAO MAN** wants to know, for each edge, how many people would visit it at most for all possible situations?

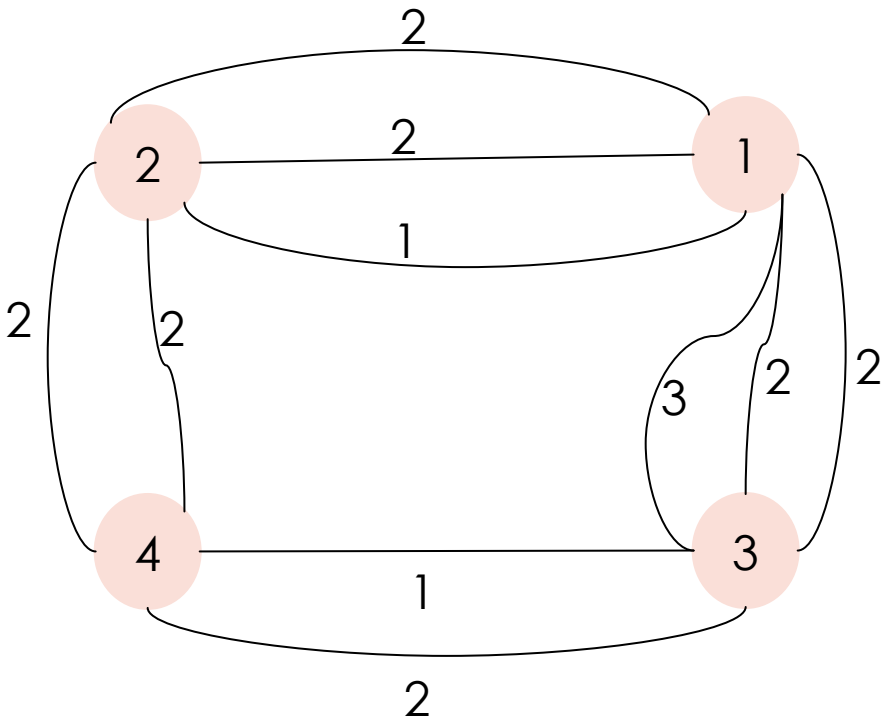
Sample Input

```
5 8
3 5 9
4 5 20
1 3 6
2 1 16
2 1 16
3 5 9
2 1 17
1 3 6
3
5 2
5 2
2 4
```



F1: 5 → 2: 5 > 3 > 1 > 2
F2: 5 → 2: 5 > 3 > 1 > 2
F3: 2 → 4: 2>1> 3 >5> 4

Edge	Follows	number
E1:3 5 9	F1, F2, F3	3
E2:4 5 20	F3	1
E3:1 3 6	F1, F2, F3	3
E4:2 1 16	F1, F2, F3	3
E5:2 1 16	F1, F2, F3	3
E6:3 5 9	F1, F2, F3	3
E7:2 1 17		0
E8:1 3 6	F1, F2, F3	3



Dijkstra：4->1

index	1	2	3	4
loop1	∞	**2**	**1**	**0**
loop2	**∞→3**	**2**	**1**	**0**
loop3	**3**	**2**	**1**	**0**
loop4	**3**	**2**	**1**	**0**

Explored 4→ update dij[3] to 2 update dij[2] to 2

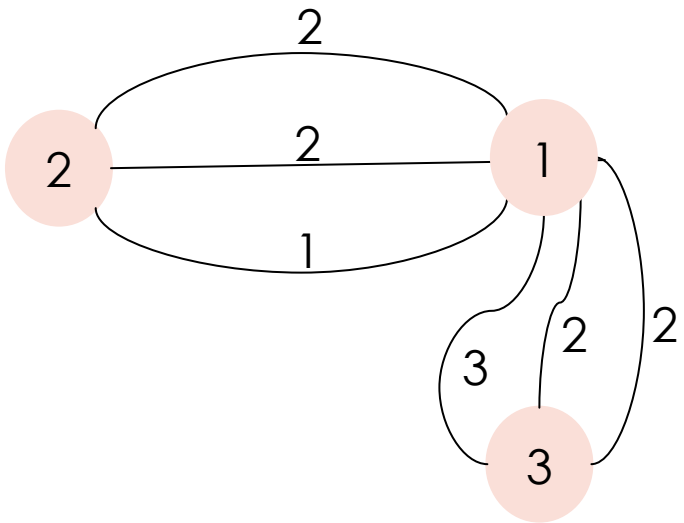
explored 3 → update dij[1] to 3 select(4,3) → next node 1

explored 2

explored1→ pq is empty → end

dij[i] records the shortest distance from node i to start node 4.
The shortest distance from 4 to 1 is 3.

Then search from 1



Then search from 1