Official Course and Teaching Evaluation

- OCTE system: https://octe.cuhk.edu.cn/a/login

- Complete the survey before May 6, 2022

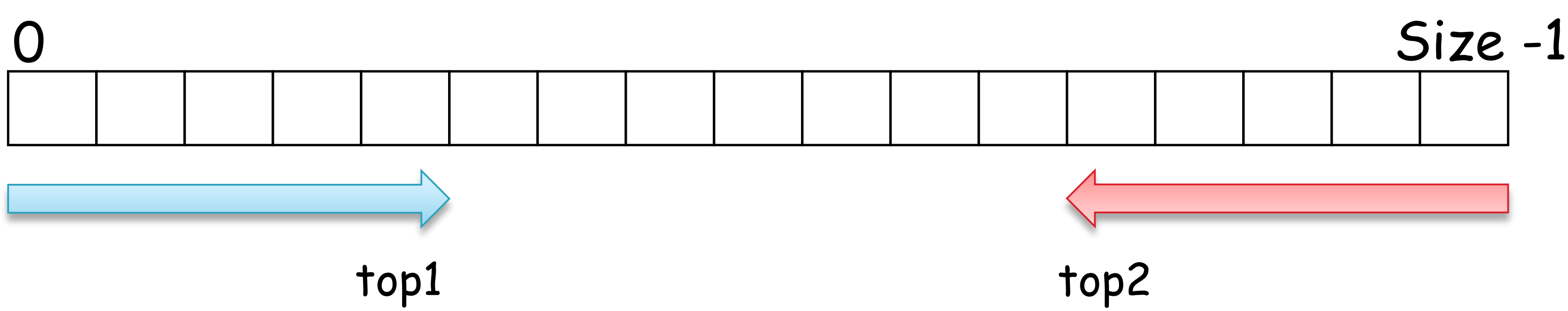
香港中文大學(深圳)
The Chinese University of Hong Kong

CSC3100 Data Structures
Lecture 26: Exercises

Yixiang Fang
School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

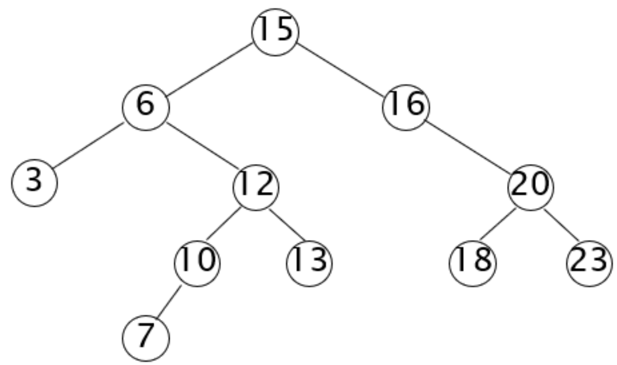
Exercise 1: Two stacks in one array

- Design two stacks in one array $A[1 \ldots n]$ ($n > 1$) in such that neither stack overflows unless the total number of elements in both stacks is n. The PUSH and POP operations should run in $O(1)$ time.
  - Please briefly explain how to implement the two stacks, and then use pseudocodes to explain the steps of PUSH and POP for them.



Exercise 2: Binary search tree

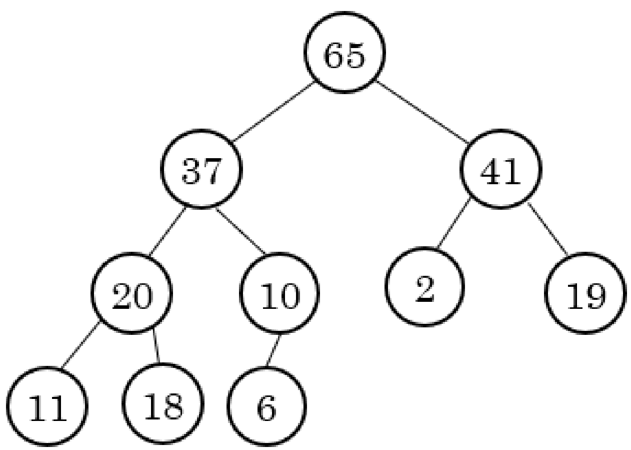
- Insertion and deletion on binary search tree:
  - Given an initial binary search tree as shown below, perform two <u>consecutive</u> operations: first insert a new key 14 into the tree, and then delete key 15 from the updated tree. Please draw these two updated trees after these operations respectively.



  - Analyze the time complexities of key insertion and key deletion on binary search tree with n nodes, in the best case and worst case respectively.

Exercise 3: Heap operations

- A priority queue is implemented as a max-heap. Use a list of tree graphs to show how the heap Q shown below would look like after a series of operations:
  - (1) *Q.Enqueue(12);*
  - (2) *Q.Enqueue(102);* (use the heap from step (1))
  - (3) *Q.Dequeue( ).* (use the heap from step (2))



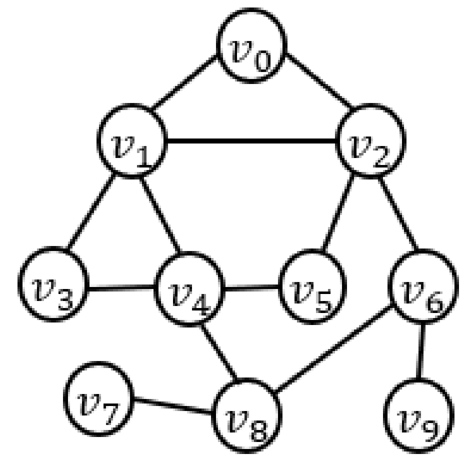
Exercise 4: Hashing

- Assume that we have a sequence of 9 keys: 6, 12, 34, 29, 18, 17, 1, 5, 4, and the size of the hash table is *m*=11. Show the filled hash tables after inserting these keys using the following collision resolution strategies and hashing functions respectively.
  - Use chaining with $h(k) = k \;\%\; 11$;
  - Use linear probing with $h(k) = k \;\%\; 11$;

0										10

Exercise 5: BFS & DFS

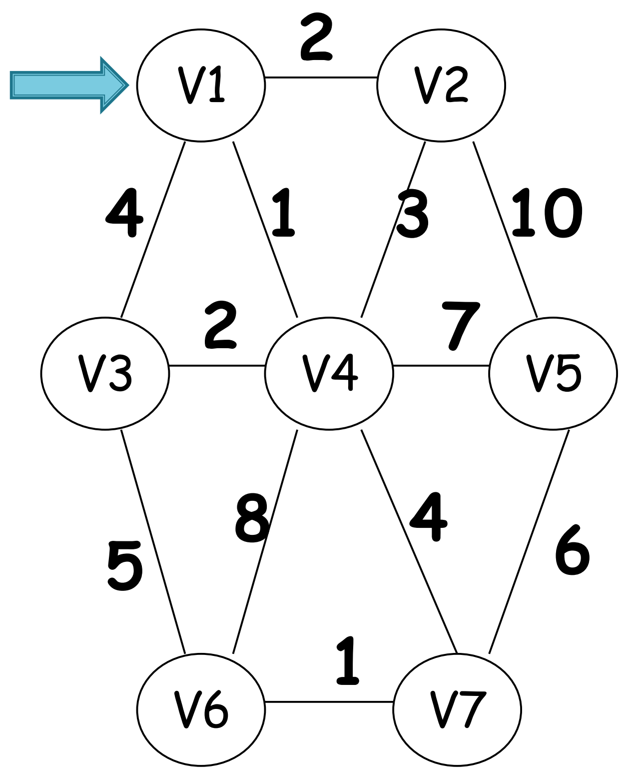
- Consider an undirected graph G_1:
  - Draw the BSF tree of G_1 by assuming that the starting node is set to v_1
  - Draw the DSF tree of G_1 by assuming that the starting node is set to v_2
  - Compare BFS with DFS in terms of time cost and extra space cost, if we use the adjacent list to represent the graph. Besides, for each of them, show a specific application that favors it and explain the reason briefly.



Graph G_1

Exercise 6: Find an MST

- Use Prim's algorithm and show its steps
- Use Kruskal's algorithm and show its steps



Exercise 7: "maximum" spanning tree

- Find an algorithm for the "maximum" spanning tree. That is, given an undirected weighted graph G, find a spanning tree of G of maximum cost, and prove the correctness of your algorithm

  - Consider choosing the "heaviest" edge (i.e., the edge associated with the largest weight) in a cut
  - The generic proof can be modified easily to show that this approach will work
  - Alternatively, multiply the weights by -1 and apply either Prim's or Kruskal's algorithms without any modification at all!

Exercise 8: shortest path with k edges

- Given a directed, weighted graph that might have negative weight edges, and we know that all the shortest paths use at most k edges. How to compute shortest paths from a source vertex s in O(k(V+E)) time?

  - Run the Bellman-Ford algorithm, but stop after k iterations of the outer loop.
  - After the i-th iteration of the outer loop, Bellman-Ford has found paths that are at least as good as the shortest paths from s that use at most i edges.
  - Thus, after k iterations, it has found paths that are at least as good as the shortest paths from s that use at most k edges, which under the given assumptions are the shortest paths overall.

Recommended reading

- Next lecture
  - Review of all course materials