# CS062

## DATA STRUCTURES AND ADVANCED PROGRAMMING

## 25: Summary

Alexandra Papoutsaki
she/her/hers

# Information

- Exam: Wednesday May 8, 2-5pm in the same room we always meet.
- You can bring two hand-written (ok *hand-written* on tablets and then printed) sheets of papers (i.e. four pages).
- Cumulative with a bigger emphasis on topics we covered since midterm 2.
- Review midterm 1 and 2, quizzes, worksheets, and practice problems in each presentation. Use the practice questions in this presentation.

# Java Basics

- Chapter 1.1 (Pages 8–35).
- Chapter 1.2 (Pages 64–77, 84—88, 96—99, 107).
- Quick overview of Java tutorials.
    - https://docs.oracle.com/javase/tutorial/java/
- In general, review the basics of OOP and of Java so that you are comfortable reading and writing code.

# Analysis of Algorithms

- Chapter 1.4 (Pages 172-205).
- Experimental analysis including doubling hypothesis. Pick two pairs of the largest input sizes and check that the T(n)/T(n/2) is consistently expressed as some power of 2.
- Mathematical analysis including reviewing (not memorizing) useful approximations of sums.
  - Use midterm review slides for practice.
- Order of growth classifications.
- Review of running time of operations on array lists, linked lists, stacks and queues.

# ArrayLists

- Chapter 1.3 (Pages 136-137).
- code
- Java Oracle API https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
- Amortized and worst-case time analysis.

# Singly Linked Lists

- Chapter 1.3 (Pages 142-146).
- code
- Worst-case time analysis for standard operations.

# Doubly Linked Lists

- Chapter 1.3 (Pages 126-157).
- code
- Java Oracle API.
  - https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
- Worst-case time analysis for standard operations.

# Stacks and Queues

- Chapter 1.3 (Pages 142-146).
- code for alternative implementations
- Worst-case time analysis for standard operations based on the underlying implementation

# Iterators and Comparators

- Comparable vs Comparator interface.
    - https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html
    - https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html
- Java Oracle Iterator and Iterable.
    - https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html
    - https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html
- code

# Sorting

- Chapter 2 (Pages 244-296).
- Selection sort and Insertion sort.
- Mergesort.
- Quicksort.
    - We have seen Lomuto's partition scheme
- Know how to apply them, best and worst case running times, in-place or not, stability

# Binary Trees, Binary Search, Heaps, and Priority Queues

- Chapter 2.4 (Pages 308-322)
- Definitions, basic properties of trees and traversals (pre-, in-, post-, level-order) for binary trees.
- Binary search implementation and complexity.
- Binary Heaps and operations.
  - Different implementations along with complexities.

# Heapsort

- Chapter 2.4 (323-327).
- Know how to apply, running time analysis.

# What you need to remember about sorting

| | In place | Stable | Best | Average | Worst | Remarks |
|---|---|---|---|---|---|---|
| Selection | X | | $n^2$ | $n^2$ | $n^2$ | $n$ exchanges |
| Insertion | X | X | $n$ | $n^2$ | $n^2$ | Use for small arrays or partially ordered |
| Merge | | X | $n \log n$ | $n \log n$ | $n \log n$ | Guaranteed performance; stable |
| Quick | X | | $n \log n$ | $n \log n$ | $n^2$ | $n \log n$ probabilistic guarantee; fastest in practice |
| Heap | X | | $n \log n$ | $n \log n$ | $n \log n$ | $n \log n$ guarantee; in place |

## Dictionaries and Binary Search Trees

- Chapter 3.1 (Pages 362-386).
- Different implementations along with complexities.
- Chapter 3.2 (Pages 396-414).
- Textbook code.
  - https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/BST.java.html
- Addition, Search, Hibbard's Deletion.

# 2-3 Search Trees

- Chapter 3.3 (Pages 424-291).
- Definitions, Search, Insertion, Construction.
- Performance.

# Left-leaning red-black trees

- Chapter 3.3 (Pages 292-447).
- Definitions, Operations, Insertion.
- Performance.

# Hash tables

- Chapter 3.3 (Pages 458-477).
- Hashing, separate chaining, open addressing.

# Summary for Dictionary operations

| | Worst case | | | Average case | | |
|---|---|---|---|---|---|---|
| | Search | Insert | Delete | Search | Insert | Delete |
| BST | $n$ | $n$ | $n$ | $\log n$ | $\log n$ | $\log n$ |
| 2-3 search tree | $\log n$ | $\log n$ | $\log n$ | $\log n$ | $\log n$ | $\log n$ |
| Red-black BSTs | $\log n$ | $\log n$ | $\log n$ | $\log n$ | $\log n$ | $\log n$ |
| Separate chaining | $n$ | $n$ | $n$ | $1$ | $1$ | $1$ |
| Linear probing | $n$ | $n$ | $n$ | $1$ | $1$ | $1$ |

# Undirected Graphs

- Chapter 4.1 (Pages 515-556).
- Definitions, representations, APIs.
- BFS.
- DFS.
- Textbook code.
    - https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Graph.java.html
    - https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/DepthFirstSearch.java.html
    - https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/BreadthFirstPaths.java.html

# Directed Graphs

- Chapter 4.2 (Pages 566-594).
- Definitions, representations, APIs.
- BFS.
- DFS.
- Textbook code.
  - https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Digraph.java.html
  - https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/DirectedDFS.java.html

# Shortest Paths

- Chapter 4.4 (Pages 638-676).
- Dijkstra's algorithm.
- https://visualgo.net/en/sssp

## Minimum Spanning Trees

- Chapter 4.3 (Pages 604-629).
- Know how to apply Kruskal's and Prim's algorithms.
    - https://algs4.cs.princeton.edu/43mst/
    - https://visualgo.net/en/mst

# Problem 1: Balanced Binary Search Trees

- a. Starting with an empty 2-3 search tree, what is the resulting 2-3 search tree after you insert the keys 2, 7, 17, 1, 90, 3, 36, 47?
- b. Starting with an empty LLRB search tree, what is the resulting LLRB search tree after you insert the keys 15, 6, 23, 4, 7, 5, 50, 71?

# Problem 2: Hashtables

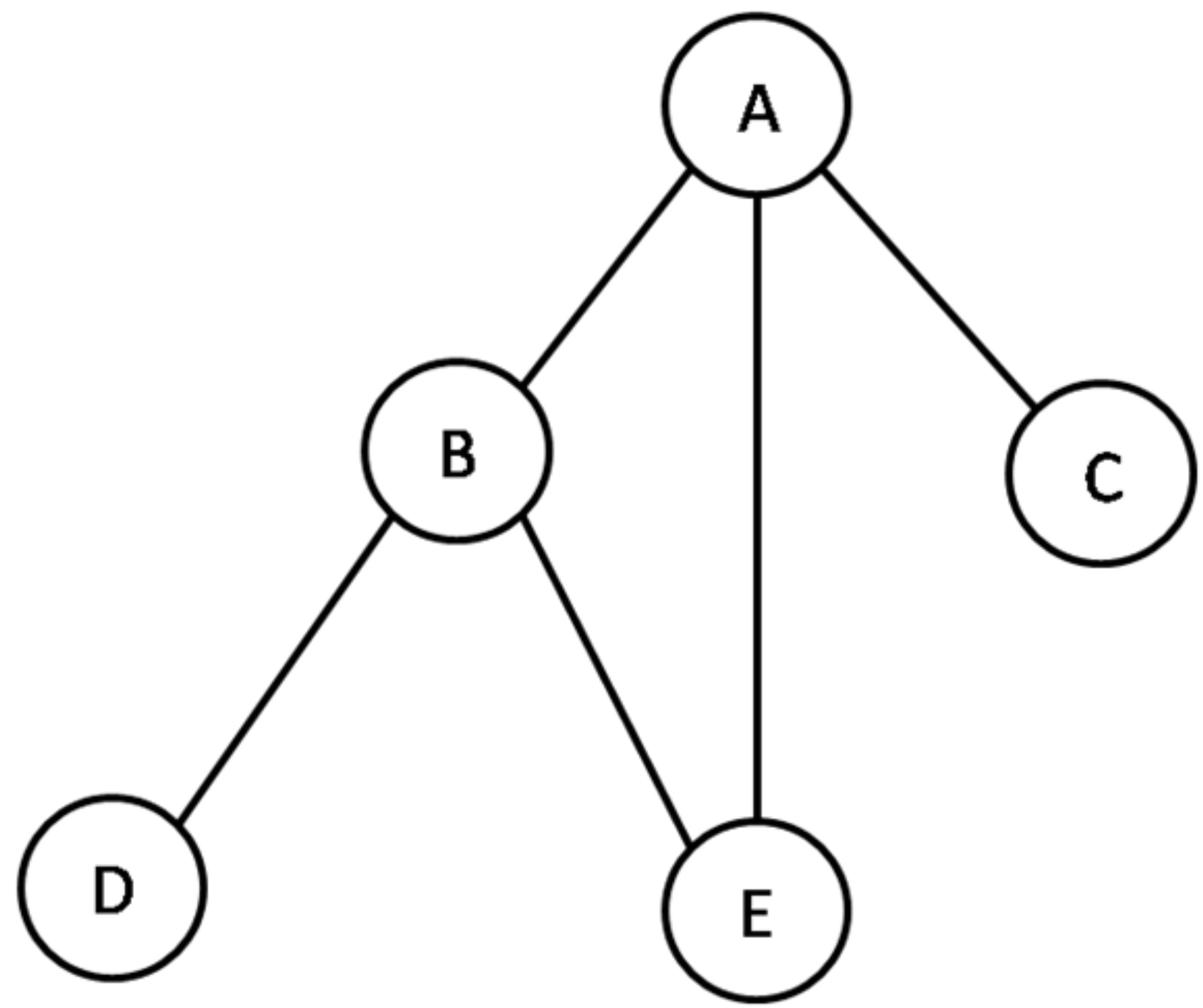
- Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \,\%\, m$. For each of the following questions, fill in the following hash table:
  - a. using open addressing and linear probing with $h(k, i) = (h(k) + i) \,\%\, m$.
  - b. using open addressing and quadratic probing with $h(k, i) = (h(k) + i^2) \,\%\, m$.
  - c. using external chaining.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Problem 3: Traversing Graphs

- Show the adjacency matrix and adjacency list representations of the undirected graph above.
- Run a.recursive and b. BFS starting at vertex A assuming adjacent vertices are returned in lexicographic order. Fill in the table below:

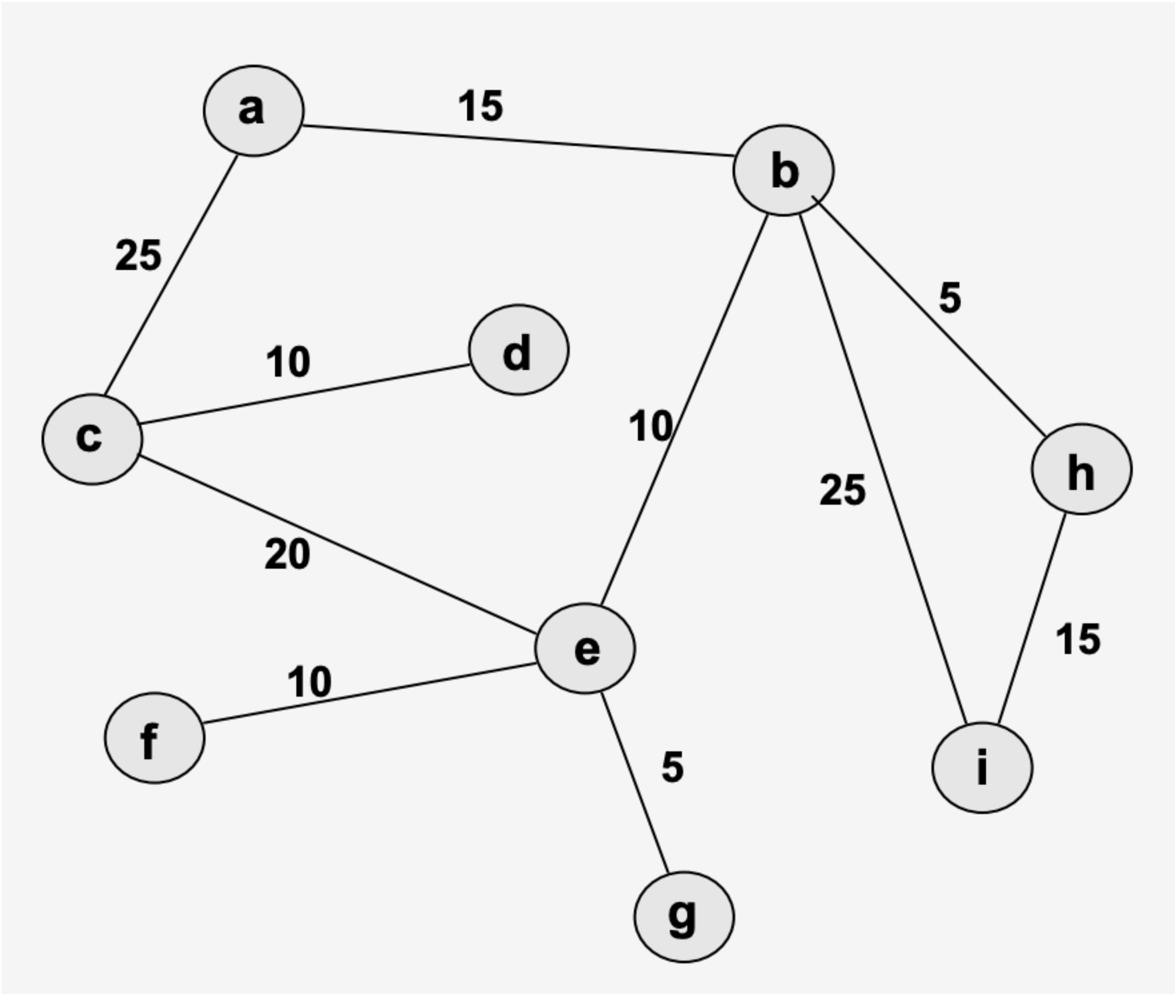
| v | marked | distTo | edgeTo |
|---|---|---|---|
| A | | | |
| B | | | |
| C | | | |
| D | | | |
| E | | | |



# Problem 4: Shortest Paths

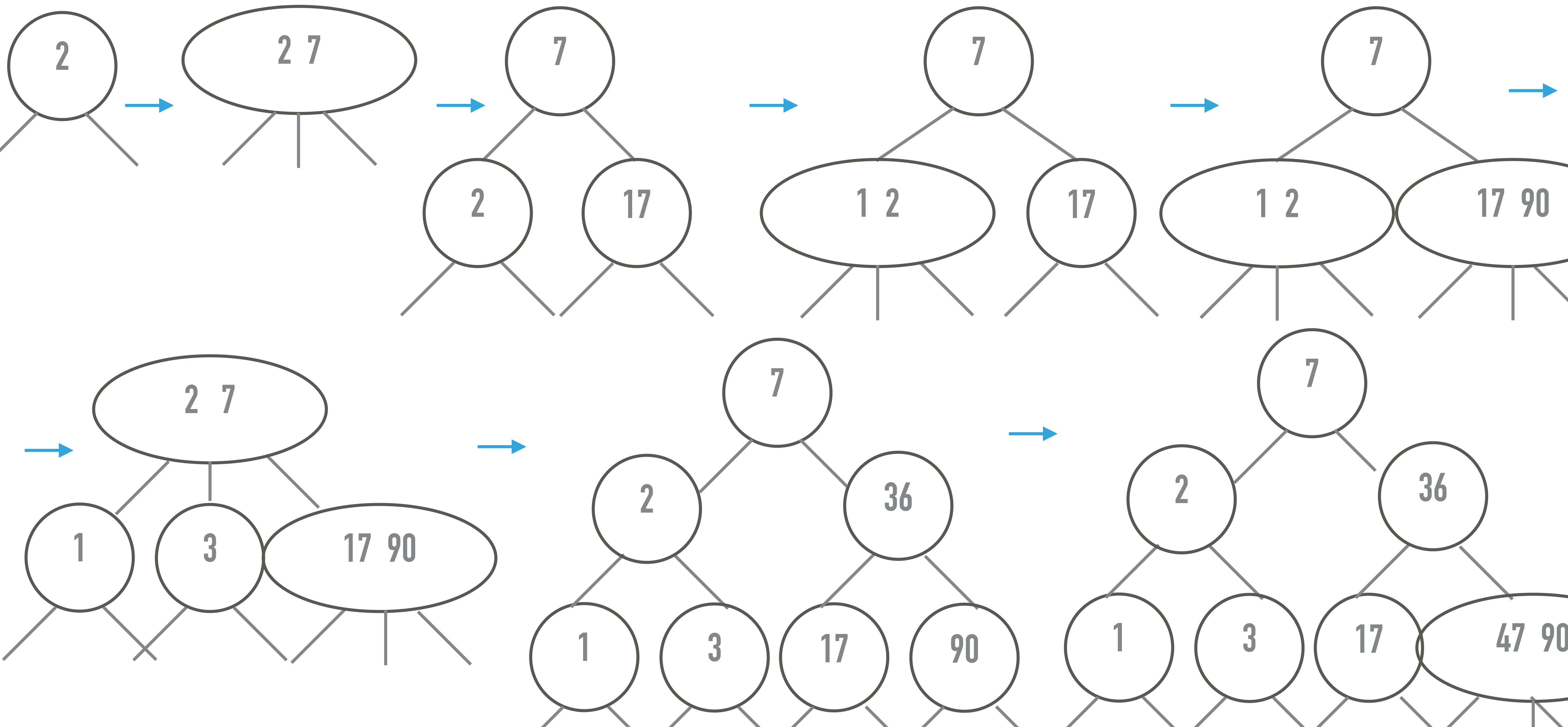
- Run Dijkstra's algorithm on this graph, starting at vertex a. Fill in the resulting distTo[] and edgeTo[] arrays below. In the edgeTo[] column, please indicate the last edge in the shortest path from a to every other vertex and mark the shortest path tree.

| v | distTo | edgeTo |
|---|---|---|
| a | | |
| b | | |
| c | | |
| d | | |
| e | | |
| f | | |
| g | | |
| h | | |
| i | | |



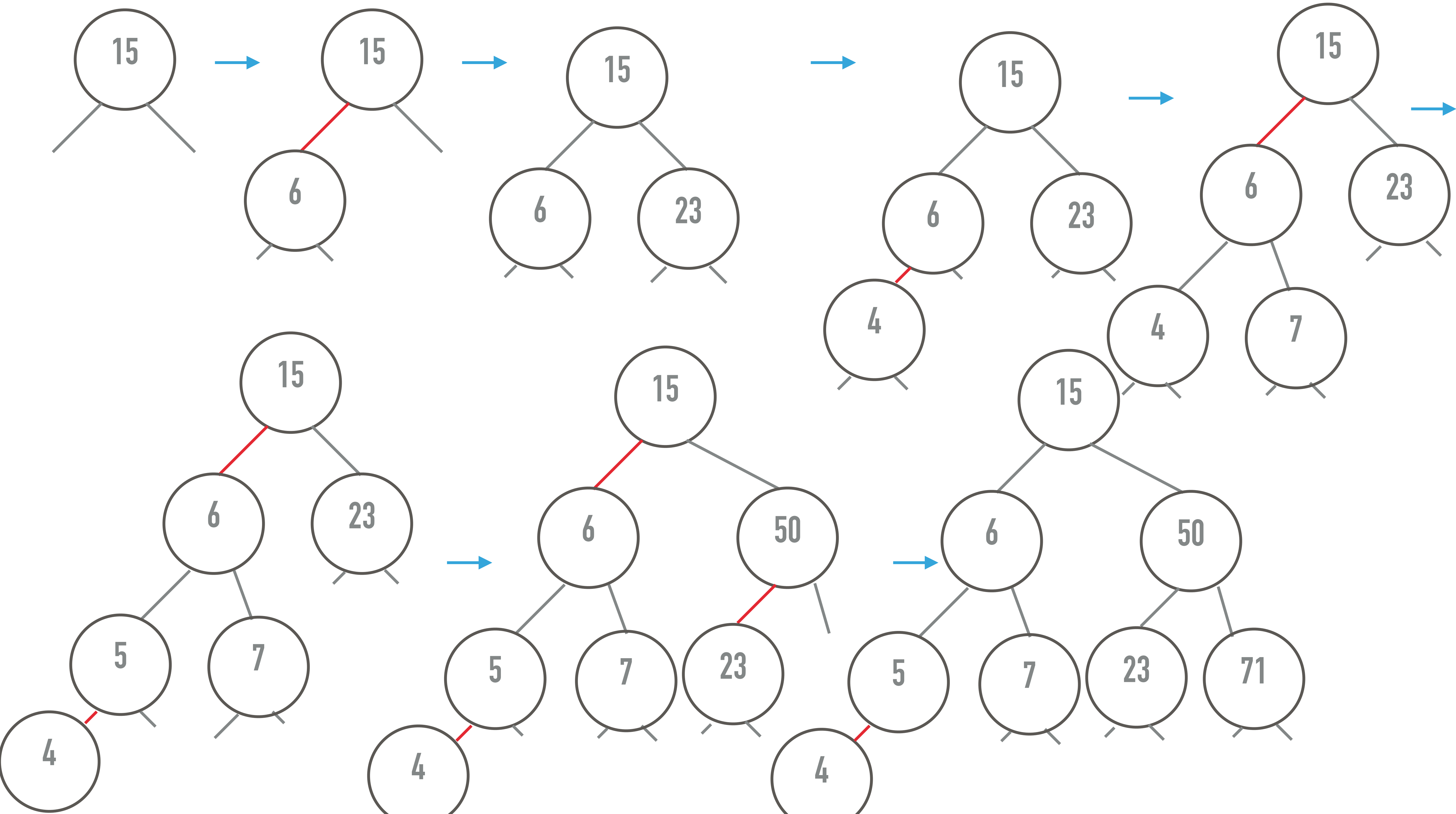
# Solution to Problem 1a: Balanced Binary Search Trees

- Starting with an empty 2-3 search tree, what is the resulting 2-3 search tree after you insert the keys 2, 7, 17, 1, 90, 3, 36, 47?



# Solution to Problem 1b: Balanced Binary Search Trees

- Starting with an empty LLRB search tree, what is the resulting LLRB search tree after you insert the keys 15, 6, 23, 4, 7, 5, 50, 71?



# Solution to Problem 2a: Hashtables

- Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \,\%\, m$. For each of the following questions, fill in the following hash table using open addressing and linear probing with $h(k, i) = (h(k) + i) \,\%\, m$.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 12 | 23 | 25 | 26 | 27 | 17 | 5 | 29 | 9 | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

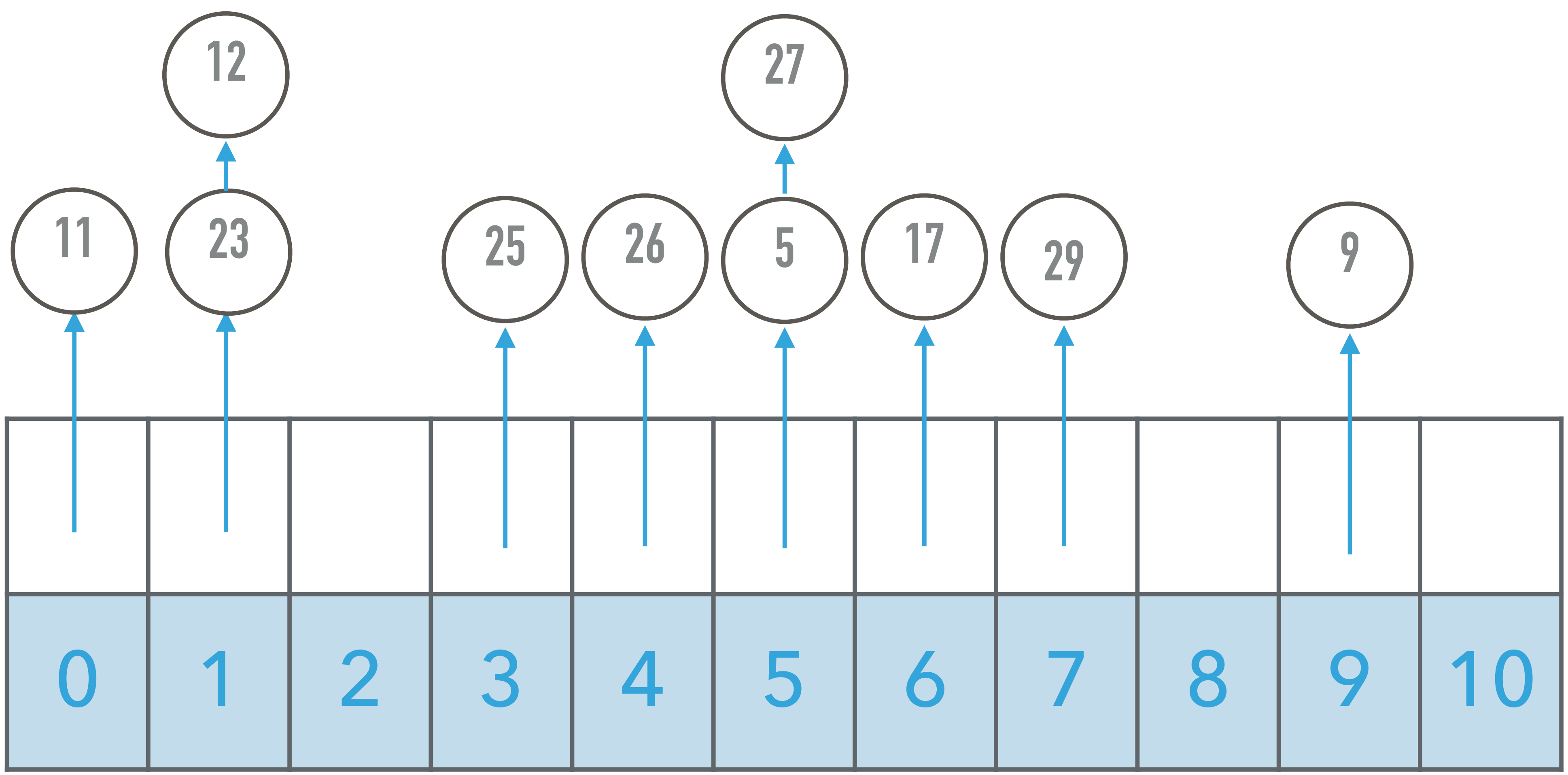
# Solution to Problem 2b: Hashtables

- Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \,\%\, m$. For each of the following questions, fill in the following hash table using open addressing and linear probing with $h(k, i) = (h(k) + i^2) \,\%\, m$.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 12 | 23 | 25 | 26 | 27 | 17 | 29 | | 5 | 9 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# Solution to Problem 2c: Hashtables

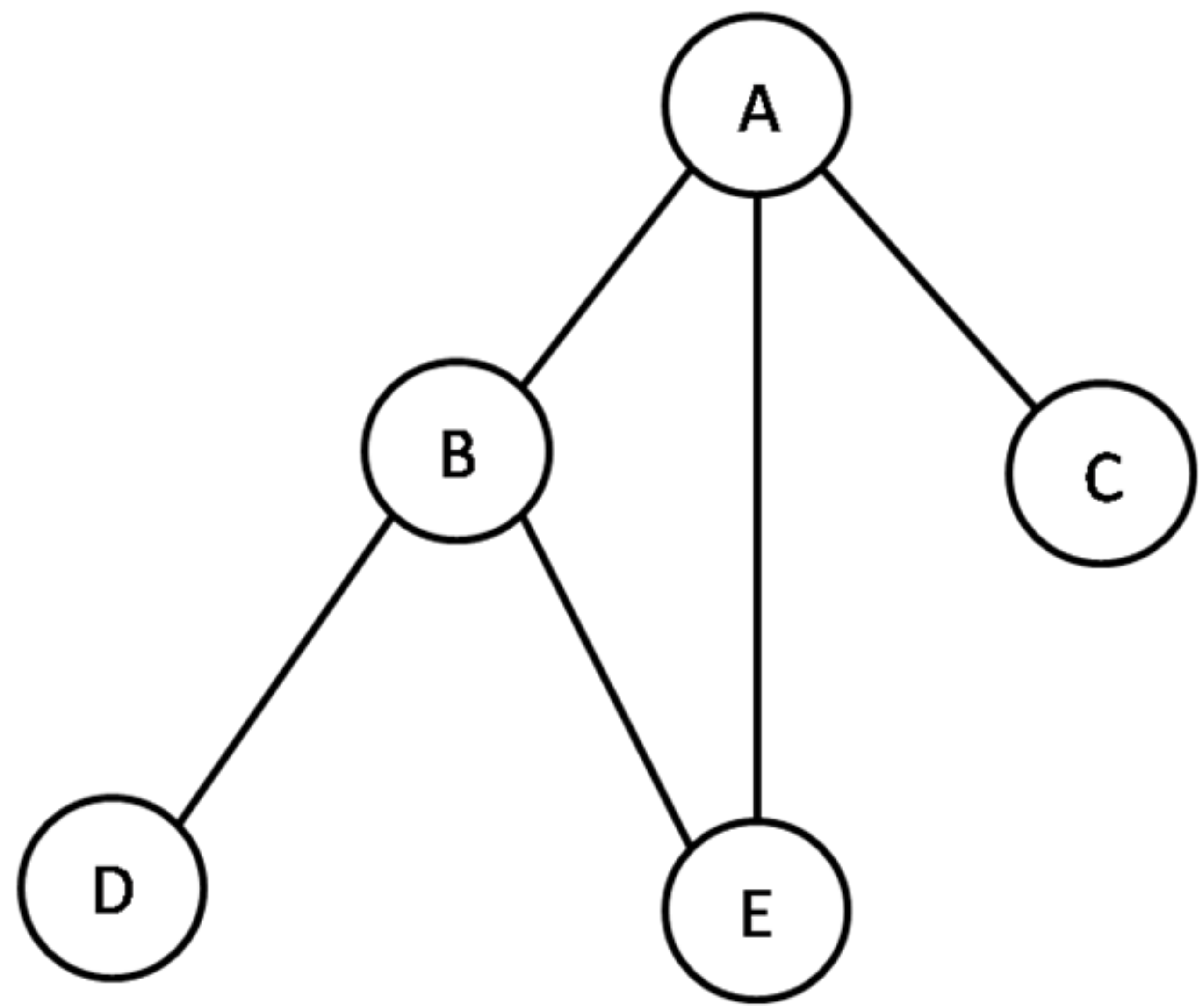
- Consider inserting the keys 25, 17, 12, 26, 27, 5, 9, 29, 11, 23 into a hash table of size $m = 11$ using the hash function $h(k) = k \,\%\, m$. For each of the following questions, fill in the following hash table using external chaining.



# Solution to Problem 3a: Traversing Graphs

- Show the adjacency matrix and adjacency list representations of the undirected graph above.
- Run a.recursive DFS assuming adjacent vertices are returned in lexicographic order.
- Order of visit: A, B, D, E, C

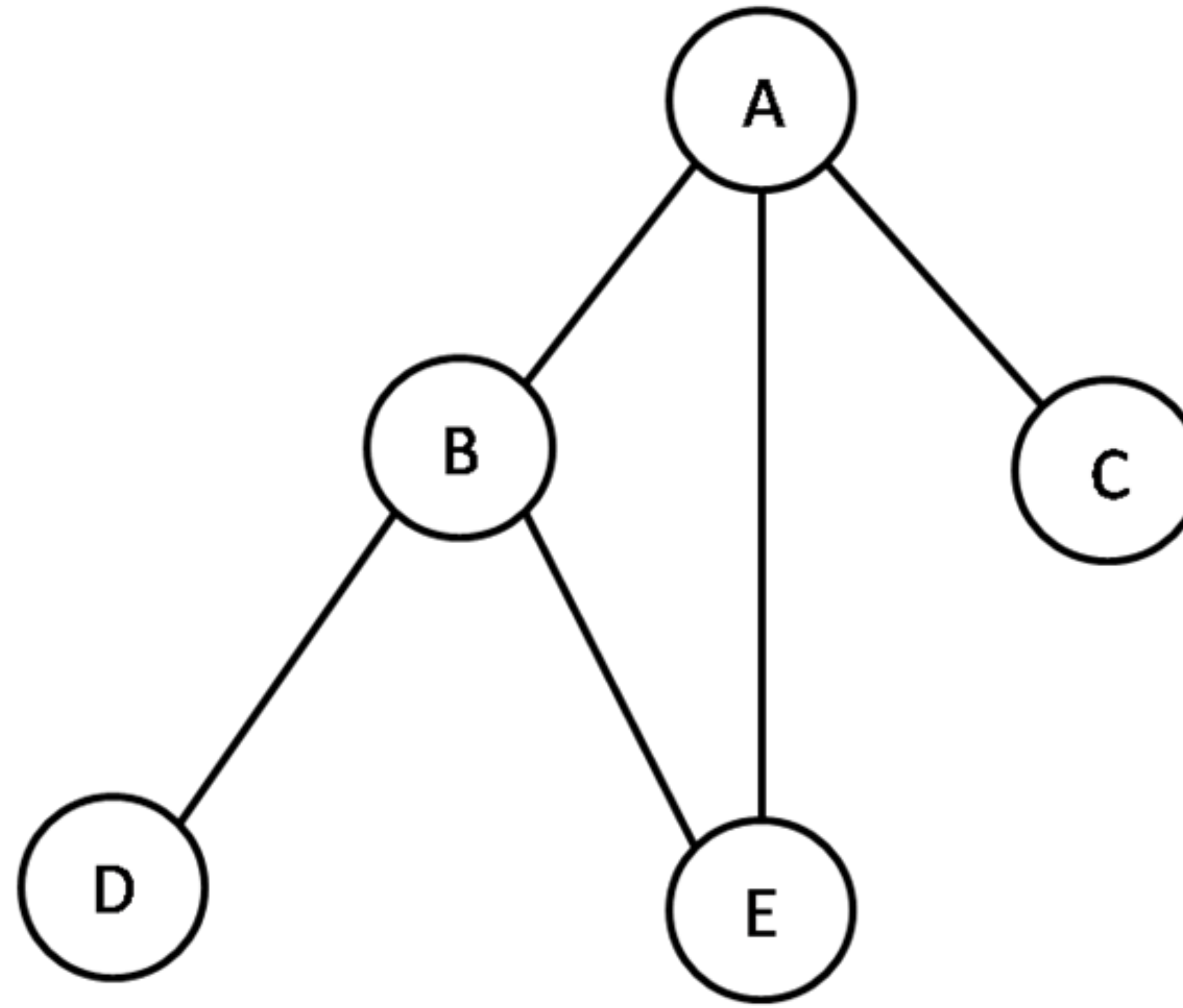
| v | marked | edgeTo |
|---|---|---|
| A | T | - |
| B | T | A |
| C | T | A |
| D | T | B |
| E | T | B |



# Solution to Problem 3b: Traversing Graphs

- Show the adjacency matrix and adjacency list representations of the undirected graph above.
- Run b. BFS assuming adjacent vertices are returned in lexicographic order.
- Order of visit: A, B, C, E, D

| v | marked | distTo | edgeTo |
|---|---|---|---|
| A | T | 0 | - |
| B | T | 1 | A |
| C | T | 1 | A |
| D | T | 2 | B |
| E | T | 1 | A |



# Solution to Problem 4: Shortest Paths

- Run Dijkstra's algorithm on this graph, starting at vertex a. Fill in the resulting distTo[] and edgeTo[] arrays below. In the edgeTo[] column, please indicate the last edge in the shortest path from a to every other vertex and mark the shortest path tree.

| v | distTo | edgeTo |
|---|---|---|
| a | 0 | - |
| b | 15 | a |
| c | 25 | a |
| d | 35 | c |
| e | 25 | b |
| f | 35 | e |
| g | 30 | e |
| h | 20 | b |
| i | 35 | h |