Midterm 2 Overview

Fawzi Emad
Chau-Wen Tseng

Department of Computer Science
University of Maryland, College Park

Overview of Topics

1. Data structures
2. Graph algorithms
3. Searching
   - Search trees
   - Heaps
   - Hash tables
   - Tries
4. Sorting
5. Compression
Topic – Data Structures

- Why data structures are important
- Taxonomy of data structures
  - Linear
  - Hierarchical
  - Graph
  - Maps & sets
- Data structure examples
  - Relationship between elements
  - Operations supported
  - Impact on efficiency

Topic – Graph Algorithms

- Types of graphs
- Operations on graphs
  - Traversals
  - Spanning trees
  - Minimal spanning trees
  - Shortest paths
- Implementation methods
Topic – Searching

- Data structures for searching
  - Binary search tree
  - Heaps
  - Maps & hashing
  - Index search trees (tries)
  - Multi-way search trees
  - Balanced search trees

- Implementation & maintenance
  - Insert(), delete(), find()

- Comparing data structures
  - Advantages & disadvantages

---

Topic – Sorting

- Approaches to sorting
  - Comparison sorts
  - Linear sorts

- Properties of sorting

- Implementing sorting

- Comparing approaches
  - Advantages & disadvantages
Topic – Compression

- Approaches to compression
- Huffman encoding
  - Algorithm
  - Implementation
  - Properties

Midterm Question Formats

- Multiple choice questions
- Short 1-sentence answers
- Write code
- Apply & describe algorithms
Multiple Choice Question Example

- Sorting is different from searching because
  - Sorting is more complex
  - Searching is more complex
  - Sorting takes more space
  - Searching takes more space
  - Sorting compares two items by size

(circle all that apply)

Short 1-Sentence Answer Example

- Sorting is different from searching because

(provide short 1-sentence answer)
Write Code

Given the following Java code fragment, write code to insert into an unsorted linked list

class Node {
    Object value;
    Node next;
    void insert ( Node n ) {
        ... // place your code here
    }
}

void BubbleSort ( Comparable [] a ) {
    ... // place your code here
}

Apply & Describe Algorithms

Given the following binary search tree

- Draw tree after adding 9
- Draw tree after deleting 5
- Perform postorder traversal

```
      10
     /  \
   5     30
  /     /  \
2      25   45
```

Apply & Describe Algorithms

- Given the following graph
  - Apply depth-first search, starting at 1
  - Apply breadth-first search, starting at 1
  - Apply Dijkstra’s algorithm, starting at 1
  - Apply Kruskal’s algorithm

- Show steps