COS 226
Final Exam Review
Spring 2015

Ananda Guna
awardena
(guna)
guna@cs.princeton.edu
guna@princeton.edu

Logistics

• The final exam time and location
  – The final exam is from 9am to 12noon on Saturday, May 16 in McCosh 28 or McCosh 50.
    • McCosh 28: Last name begins with A–F.
    • McCosh 50: Last name begins with G–Z.
  – The exam will start and end promptly, so please do arrive on time.
  – Alternate time and place
    • Monday May 18th at 1:30PM in Friend 008

• Exam Format
  – Closed book, closed notes.
  – You may bring one 8.5-by-11 sheet (both sides) with notes in your own handwriting to the exam.
  – No electronic devices (e.g., calculators, laptops, and cell phones).

Material covered

• The exam will stress material covered since the midterm, including the following components.
  – Lectures 13–23.
  – Exercises 12–22.
  – Programming assignments 6–8
    • Wordnet, seam-carving, burrows-wheeler

Areas/Topics covered

<table>
<thead>
<tr>
<th>Data Compression</th>
<th>String Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huffman, run Length Encoding</td>
<td>Boyer-Moore,</td>
</tr>
<tr>
<td>LZW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String Sorts</th>
<th>Graphs - Shortest Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSD, LSD, 3-way radix quicksort</td>
<td>BFS, Dijkstra’s, Bellman-Ford, DAgG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graphs - Traversal/order</th>
<th>Graphs - MST</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFS, DFS, Topological sort</td>
<td>Kruskal, Prim</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graphs - MST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kruskal, Prim</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big-O, order of growth, Title</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>primitive types, objects, arrays, nested classes</td>
</tr>
</tbody>
</table>

What to focus on

• focus on understanding basic issues, not memorizing details

• For each algorithm
  – understand how it works on typical input
  – Why do we care about this algorithm?
  – How is it different from other algorithms for the same problem?
  – When is it effective?

• For each data structure
  – invariants
  – Operations and complexity
  – applications
  – When is it effective to use a specific data structure?
Algorithm Analysis

Challenge Questions
• Consider each statement and state TRUE, FALSE, UNKNOWN
  – An algorithm for sorting n comparable keys in linear time or less has not been invented yet
  – There exist an algorithm where duplicity of elements in a set can be determined in sub-linear time
  – The convex hull problem (i.e. finding a set of points that encloses a given set of n points) can be solved in linearithmic time
  – It is possible to insert n comparable keys into a BST in time proportional to n

Experimental to Predictive

Graph Algorithms

Order of growth

3. Graph Search
• Identify one situation where you would need to use BFS instead of DFS.
• Identify one situation where you would need to use DFS instead of BFS.
• Find a topological sort of the vertices (if possible)
5. MST
- How does Kruskal's differ from Prim's?
- What data structure is useful when running Kruskal's on a graph?
- What data structure is useful when running Prim's algorithm on a graph?
- Can minimum spanning tree algorithms be used to find the maximum spanning tree of a graph?
- How many edges does a MST contain (in terms of number of vertices)?

6. MST Algorithm Design

1. State the algorithm

1. Explain why your algorithm takes $O(V)$ time

7. Match Algorithms

8. Dijkstra's algorithm
- Give an example where Dijkstra's fails when there is a negative edge.
- What algorithm can be applied to find the shortest path when there is a negative edge?
- Is it always possible to find the shortest path when there are negative edges in the graph?

Challenge problems
- Answer each question as possible, impossible or unknown
  - Find the strong components in a digraph in linear time
  - Construct a binary heap in linear time
  - Find the maximum spanning tree in time proportional to $E+V$

Strings
9. TST
1. List the words in alphabetical order (black nodes denote the end of a word)
2. Insert aaca to TST
3. Why and when would you use a TST instead of a R-way trie?

13. KMP Table
Construct the KMP table for the search string

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

10. String Sorting
Put an X in each box if the string sorting algorithm (the standard version considered in class) has the corresponding property.

<table>
<thead>
<tr>
<th></th>
<th>merge sort</th>
<th>LSD radix sort</th>
<th>MSD radix sort</th>
<th>3-way radix quicksort</th>
</tr>
</thead>
<tbody>
<tr>
<td>stable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inplace</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>substring time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fixed-length strings only</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compression

12. Regular Expression to NFA
Convert the RE \( b \cdot (a | b)^* \) into an equivalent NFA using the algorithm described in lecture, showing the result after applying each transformation.

```
1. Compressing
A B A A B A A A B B (A=41, B=42, next code= 81)

2. Expanding
What is the result of expanding the following LZW-encoded sequence of 11 hexadecimal integers?

43 41 42 42 80 43 81 41 87 83 80

Assume the original encoding table consists of all \( \text{ASCII} \) characters and uses 8-bit codewords. Recall that codeword 80 is reserved to signify end of file.

```

14. LZW compression

```

```
15. MaxFlow-MinCut

Find max-flow and then min-cut

17. Algorithm Design

In data compression, a set of binary code words is prefix-free if no code word is a prefix of another. For example, \(01, 10, 001, 111\) is prefix-free, but \(01, 10, 001, 10100\) is not because 10 is a prefix of 10100.

1. Design an efficient algorithm to determine if a set of binary code words is prefix-free

2. What is the order of growth of the worst-case running time of your algorithm as a function of \(N\) and \(W\), where \(N\) is the number of binary code words and \(W\) is the total number of bits in the input?

3. What is the order of growth of the memory usage of your algorithm?


What is the Burrows-Wheeler transform of:

\[ b a b a b a c \]

What is the Burrows-Wheeler inverse transform of:

\[ b b b b a a a a \]

21. Counting Memory

- standard data types
- object overhead – 16 bytes
- array overhead – 24 bytes
- references – 8 bytes
- inner class reference – 8 bytes

```java
public class TwoThreeTree extends Comparable<Key, Value> {
    private Node root;

    private class Node {
        private Key key1, key2; // the one or two keys
        private Value value1, value2; // the one or two values
        private Node left, middle, right; // the two or three subtrees
    }
}
```

- How much memory is needed for a 2-3 tree that holds \(N\) nodes?

22. String Sorting

List key invariants for each algorithm:

1. MSD
2. LSD
3. 3-way radix quicksort

KMP Table

Identify the string using the partially completed DFA

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>(B)</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complete the DFA
23. Reductions

Consider the following two problems:

- **Robots**. Given N integers \( x_1, x_2, \ldots, x_N \), are there three distinct indices \( i, j, k \) such that \( x_i + x_j = x_k \)?

- **3-Sat**. Given \( x_1, x_2, \ldots, x_n \) an integer \( k \), are there three distinct indices \( i, j, k \) such that \( x_i \lor x_j \lor x_k = \text{true} \)?

(a) Show that 3-Sat is NP-hard. To demonstrate your reduction, give the 3Sat/P3 instance that you would construct to solve the following 3Sat instance:

(b) Show that 3-Sat is NP-hard. To demonstrate your reduction, give the 3Sat/P3 instance that you would construct to solve the following 3Sat/P3 instance:

\( x_1, x_2, \ldots, x_N \).