COS 226
Final Exam Review
Spring 2015

Ananda **Guna**wardena
(guna)
guna@cs.princeton.edu
guna@princeton.edu

Material covered

- The exam will *stress* material covered since the midterm, including the following components.
  - Lectures 13–23.
  - *Algorithms in Java, 4th edition*, Chapters 4–6.
  - Exercises 12–22.
  - Programming assignments 6–8
    - Wordnet, seam-carving, burrows-wheeler

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 note.
  - 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).

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?

Areas/Topics covered

Data Compression
LZW, Huffman, run Length Encoding

String Search
KMP, Boyer-Moore,
rabin-karp

String Sorts
MSD, LSD, 3-way radix quicksort

Graphs - Shortest Path
BFS, Dijkstra's, Bellman-Ford, DAGs

Graphs - Traversals/order
BFS, DFS, Topological sort
DFS - preorder, postorder

Graphs - MST
Kruskals, Prims

Areas/Topics covered

Maxflow / Mincut
Augmenting paths, Ford-Fulkerson

Reductions
X linear time reduces to Y

DFA / NFA
Regular Expressions

Tries
R-way, TST

Algorithm Analysis
Big O, order of growth, Tilde

Memory Analysis
primitive types, objects, arrays, nested classes

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 — False
  - There exist an algorithm where duplicity of elements in a set can be determined in sub-linear time — False
  - 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 — True
  - It is possible to insert n comparable keys into a BST in time proportional to n — False, cannot sort in < n lg n

Algorithm Analysis

Experimental to Predictive

$T = \frac{16}{(40000)^2} \times (200000)^2 = 400$ $\qquad a = \frac{16}{(40000)^2}$

Suppose that you observe the following running times for a program with an input of size N.

N	time
5,000	0.2 seconds
10,000	1.2 seconds
20,000	3.9 seconds
40,000	16.0 seconds
80,000	63.9 seconds

$T = aN^b$

$a = ?\ ,\ b = ?$

Estimate the running time of the program (in seconds) on an input of size $N = 200{,}000$.

$\frac{63.9}{16} = 2^b \qquad b = \lg_2\left(\frac{63.9}{16}\right) \approx 2 \qquad \frac{16}{63.9} = \frac{a(40000)^b}{a(80000)^b} = \left(\frac{1}{2}\right)^b$

Order of growth

```
public static int f3(int N) {
   if (N == 0) return 1;
   int x = 0;
   for (int i = 0; i < N; i++)
      x += f3(N-1);
   return x;
}
```

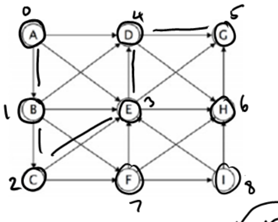
```
public static int f4(int N, int R) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 1; j <= R; j += j)
         x++;
   return x;
}
```

N, $\lg R$ → $N \lg R$

$f(n) = 1 + n \cdot f(n-1) \Rightarrow f(n) \sim n!$

$= 1 + n\left(1 + (n-1) f(n-2)\right)$

3. Graph Search

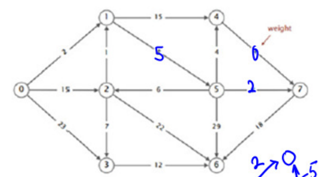


- Identify one situation where you would need to use BFS instead of DFS. — Shortest Path
- Identify one situation where you would need to use DFS instead of BFS. — Reachability
- Find a topological sort of the vertices (if possible)

G H D E I F C B A — Post order

8. Dijkstra's algorithm

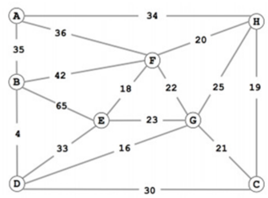
Complete



v	distTo[]	edgeTo[]
0	0	–
1	2	0
2	15	0
3	23	0
4	~~17~~ 12	~~1~~ 5
5	7	1
6	36	5
7	9	5

- Give an example where Dijkstra's fail when there is a negative edge.
- What algorithm can be applied to find the shortest path when there is a negative edge? — Bellman-Ford
- Is it always possible to find the shortest path when there are negative edges in the graph? — No if there is a negative cycle

5. MST



- How does Kruskals Differ from Prims? (edges / vertices)
- What data structure is useful when running kruskals on a graph? Union-Find
- What data structure is useful when running Prim's algorithm on a graph PQ
- Can minimum spanning tree algorithm be used to find the maximum spanning tree of a graph? yes negate weights
- How many edges does a MST contains (in terms of number of vertices) V-1

Challenge problems

- Answer each question as possible, impossible or unknown
  - Find the strong components in a digraph in linear time yes – Tarjan or Kosaraju
  - Construct a binary heap in linear time → yes
  - Find the maximum spanning tree in time proportional to E+V → Sure

6. MST Algorithm Design

Suppose you know the MST of a weighted graph G. Now, a new edge v-w of weight c is inserted into G to form a weighted graph G'. Design an $O(V)$ time algorithm to determine if the MST in G is also an MST in G'. You may assume all edge weights are distinct.

Your answer will be graded for correctness, clarity, and *conciseness*.

1. State the algorithm

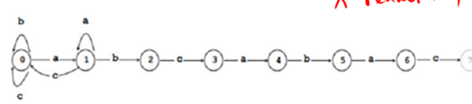
   Find a path from v→w if any edge has weight > c swap with new edge

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

   Since MST is a tree it has V-1 edges

13. KMP Table

X reduces to Y



Construct the KMP table for the search string

	a	b	c	a	b	a	c	
	0	1	2	3	4	5	6	7
a	1	1	1	4	1	6	-	
b	0	2	0	0	5			
c	0	0	3	0	0		7	

7. Match Algorithms

___ T9 texting in a cell phone
___ 1D range search
___ 2D range search
___ Document similarity
___ Traveling salesperson problem
___ Web crawler
___ Google map
___ PERT/CPM (Program Evaluation and Review Technique / Critical Path Method).

A. Trie
B. Hashing
C. 3-way radix quicksort
D. Binary search tree
E. Kd tree
F. Depth-first search
G. Breadth-first search
H. Dijkstra's algorithm
I. Topological sort
J. Bellman-Ford
K. Enumerate permutations

$a_0 + a_1 b + a_2 b^2 + \cdots + a_n b^n$

$a_0 + b(a_1 + a_2 b + \cdots + a_n b^{n-1})$

14. LZW compression

1. Compressing

A B A B A A B A A A A B B (A=41, B=42, next code= 81)

41 42 81 41 83 84 81 42

AB=81, BA=82, ABA=83, AA=84, ABAA=85, AAA=86, ABB=87

2. Expanding

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

43 41 42 42 82 43 81 41 87 82 80

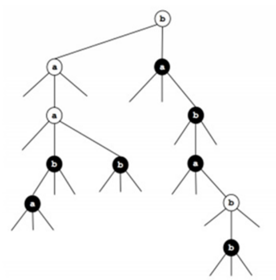
C A B B AB C CA A CAA AB (END)

Assume the original encoding table consists of all 7-bit ASCII characters and uses 8-bit codewords. Recall that codeword 80 is reserved to signify end of file.

CA=81, AB=82, BB=83, BA=84, ABC=85, CC=86, CAA=87, AC=88, CAAA=89

C	A	B	B												

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?

21. counting memory

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

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

   private class Node {
      private int count;                  // subtree count
      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?

10. String Sorting

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

	mergesort	LSD radix sort	MSD radix sort	3-way radix quicksort
stable	✓	✓	✓	X
in-place	X	X	X	✓
sublinear time (in best case)	X ✓?	X	✓	X ✓?
fixed-length strings only		✓		

22. String Sorting

KISS	ABBA	ENYA	ABBA	ENYA	ACDC	SOAD	SADE	ABBA
ENYA	ACDC	INXS	ACDC	ABBA	ABBA	WHAM	CAKE	ACDC
INXS	AQUA	DIDO	AQUA	AQUA	AQUA	ABBA	CARS	AQUA
STYX	BECK	CARS	BECK	ACDC	BUSH	MOBY	JAYZ	BECK
SOAD	BLUR	ACDC	BLUR	SOAD	BLUR	BECK	ABBA	BLUR
ACDC	BUSH	FUEL	BUSH	CAKE	BECK	ACDC	ACDC	BUSH
KORN	CAKE	BUSH	CAKE	MUSE	CAKE	SADE	BECK	CAKE
FUEL	CARS	ABBA	CARS	HOLE	CARS	DIDO	WHAM	CARS
BUSH	DIDO	AQUA	DIDO	SADE	DIDO	FUEL	DIDO	DIDO
ABBA	ENYA	CAKE	ENYA	BUSH	ENYA	CAKE	KISS	ENYA
WHAM	FUEL	BLUR	FUEL	RUSH	FUEL	HOLE	BLUR	FUEL
CAKE	HOLE	JAYZ	HOLE	BECK	HOLE	TSOL	INXS	HOLE
BLUR	INXS	BECK	INXS	FUEL	INXS	KORN	ENYA	INXS
MUSE	JAYZ	HOLE	JAYZ	TSOL	JAYZ	CARS	SOAD	JAYZ
BECK	KISS	KORN	KISS	WHAM	KISS	MUSE	MOBY	KISS
MOBY	KORN	KISS	KORN	KORN	KORN	BUSH	HOLE	KORN
HOLE	MUSE	TSOL	TSOL	DIDO	MUSE	RUSH	KORN	MOBY
TSOL	MOBY	MOBY	MOBY	BLUR	MOBY	KISS	AQUA	MUSE
JAYZ	RUSH	MUSE	MUSE	KISS	RUSH	AQUA	TSOL	RUSH
AQUA	STYX	SADE	SADE	INXS	STYX	BLUR	STYX	SADE
SADE	SOAD	WHAM	WHAM	CARS	SOAD	INXS	FUEL	SOAD
CARS	SADE	SOAD	SOAD	STYX	SADE	ENYA	MUSE	STYX
DIDO	TSOL	RUSH	RUSH	MOBY	TSOL	STYX	BUSH	TSOL
RUSH	WHAM	STYX	STYX	JAYZ	WHAM	JAYZ	RUSH	WHAM
----	----	----	----	----	----	----	----	----
0	3	4	4	2	3	2	2	1

(0) Original input
(1) Sorted
(2) LSD radix sort
(3) MSD radix sort
(4) 3-way string quicksort (no shuffle)

1-round 2-rounds 3 rounds

List key invariants for each algorithm

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

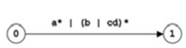
MSD

1	0	2
0	1	2
1	1	3

→ 012 / 102 / 113

12. Regular Expression to NFA

Convert the RE a* | (b | c d)* into an equivalent NFA using the algorithm described in lecture, showing the result after applying each transformation.



3

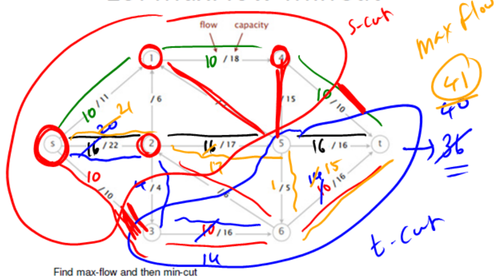
KMP Table

Identify the string using the partially completed DFA

	0	1	2	3	4	5	6	7	8	9	10
A	0	0	3	0			7			10	11
B	1	2	2	4	5		2				4
S	B	B	A	B	B	B	A	B	B	A	A

Complete the DFA

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, 0010, 1111} is prefix free, but {01, 10, 0010, 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

   Insert into a trie

1. 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?

   W

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

   WR (R=2)

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, 0010, 1111} is prefix free, but {01, 10, 0010, 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

1. 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?

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

19. Burrows-Wheeler

What is the Burrows-Wheeler transform of

b a b a a b a c

What is the Burrows-Wheeler inverse transform of

7
b b b b a a a a a

23. Reductions

Consider the following two problems:

- 3SUM. Given N integers $x_1, x_2, \ldots, x_N$, are there three distinct indices i, j, and k such that $x_i + x_j + x_k = 0$?
- 3SUMPLUS. Given N integers $x_1, x_2, \ldots, x_N$ and an integer b, are there three distinct indices i, j, and k such that $x_i + x_j + x_k = b$?

(a) Show that 3SUM linear-time reduces to 3SUMPLUS. To demonstrate your reduction, give the 3SUMPLUS instance that you would construct to solve the following 3SUM instance: $x_1, x_2, \ldots, x_N$.

(b) Show that 3SUMPLUS linear-time reduces to 3SUM. To demonstrate your reduction, give the 3SUM instance that you would construct to solve the following 3SUMPLUS instance: $b, x_1, x_2, \ldots, x_N$.

$(3x_i - b) + (3x_j - b) + (3x_k - b) = 0$

3Sum Reduce to 3Sumplus → b=0

3Sumplus Reduce to 3Sum →