# University of Waterloo
# CS240R Spring 2017
# Review Problems

**Reminder: Final on Thursday, August 3, 2017**

## True/False Problems

For each statement below, write true or false. Justify five of them.

(a) Open addressing hashing that uses linear probing will require two hash functions.

(b) Run length encoding may result in text expansion on some strings.

(c) When doing range search on a quad tree, if there is no point within the range specified, the worst case runtime complexity is $\Theta(1)$.

(d) Suffix trees for pattern matching require preprocessing the pattern.

(e) Deleting any element stored at the root of a 2-3 tree always decreases the height of the tree by 1.

(f) If the bubble-up version of $heapify$ is used in Heapsort, then the worst case runtime of Heapsort will be $\Omega(n^2)$.

(g) The runtime complexity of range query for KD-trees depends on the spread factor of points.

(h) A valid way to hash strings is to flatten the string into an integer using the ascii values of each letter and some radix base R.

(i) If an AVL tree node has balance 2 and its right child has balance 1, then a double left rotation is required.

(j) Move-to-front compression uses adaptive instead of fixed dictionaries.

# Multiple Choice

Pick the best answer for each question.

1. The last occurrence function for the pattern `MELSMEMES` would contain the following values for each character:

   (a) E=8, L=3, M=7, S=9

   (b) E=6, L=7, M=8, S=2

   (c) E=2, L=8, M=6, S=7

   (d) E=7, L=2, M=6, S=8

2. A 2-3 tree of height 1 in which every node contains two keys will have _____ NIL leaves:

   (a) 6

   (b) 8

   (c) 9

   (d) 12

3. Using LZW decoding, the last code 132 decodes to what?

   67 128 129 130 131 132

   (a) CCCCCC

   (b) CCCCCCC

   (c) CCCCCCCC

   (d) CCCCCCCCC

4. What one of these statements about hashing is false?

   (a) Constant time search and delete if Cuckoo Hashing is used

   (b) Hash tables may use more space than the number of elements

   (c) Two keys will never hash to the same index using chaining

   (d) Insert for Cuckoo Hashing can result in a loop

5. Suppose we have an array of $n$ numbers where each number is no larger than $n^3$, and assume $n$ is a perfect square. Consider running Heapsort, Quicksort, and Radix sort with radix base $R = \sqrt{n}$ on this array. The worst case asymptotic runtimes of each sorting algorithm, from best to worst, is:

   (a) Heapsort, Quicksort, Radix sort
   (b) Radix sort, Heapsort, Quicksort
   (c) Quicksort, Radix sort, Heapsort
   (d) Radix sort, Quicksort, Heapsort

6. Which one of these statements about compressed tries is false?

   (a) Every internal node stores an index indicating the bit to be tested on a search
   (b) The root of the compressed trie always tests the first bit
   (c) A compressed trie that stores $n$ keys has at most $n - 1$ internal nodes
   (d) The height of a compressed trie is at most the length of the longest string it stores

7. A quad tree with bounding box $[0, 8] \times [0, 8]$ over the following points has a height of _____.

$$(6, 2), (0, 1), (3, 4), (7, 5), (1, 0)$$

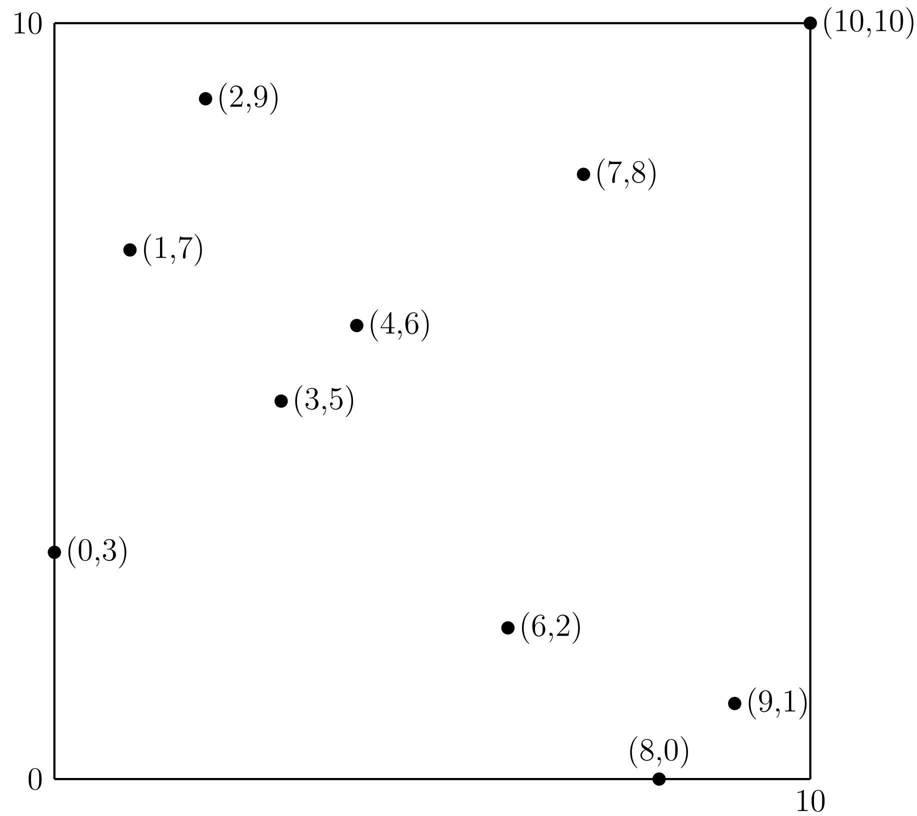
   (a) 2
   (b) 3
   (c) 4
   (d) 5

8. CS 240 is a course about:

   (a) Compilers
   (b) Software design patterns
   (c) Chinese history
   (d) Data structures and algorithms

# KD Trees

Consider the following set of points:




(a) Draw a plane partition diagram and the corresponding kd-tree.

(b) Show how a search for the points in the query rectangle $R = [2, 6.5] \times [0.5, 7]$ would proceed. More specifically, list the nodes of the kd-tree in the order that they are examined in the search, and underline the nodes that are reported.

# Knuth-Morris Pratt

(a) Compute the failure array for the pattern $P =$`JWJWJX`.

| J | W | X | J | J | W | J | J | W | J | W | J | W | J | X | J | W | X | J | J |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |

Table 1: Table for KMP problem.

(b) Show how to search for pattern $P =$`JWJWJX` using the KMP algorithm for the text in the table above. Place each character of $P$ in the column of the compared-to character of $T$. Put brackets around the character if an actual comparison was not performed. You may not need all the available space.

# Hashing

Using double hashing with the hash functions $h_1(n) = n \bmod 7$ and $h_2(n) = (3n \bmod 6) + 1$ and a table of size 7, answer the questions below:

(a) Fill the table with correctly hashed values such that a call to $search(6)$ succeeds at the end of a probe sequence of length four.

(b) Suppose the numbers written in your table above were inserted using linear probing instead with the hash function $h_2(n)$. Show the resulting table.

## Huffman Compression

(a) The following message was compressed using Huffman encoding and transmitted together with its dictionary:

00100000111010101110001011010010

| | | |
|---|---|---|
| ' ' | = | 100 (blank space) |
| : | = | 1011 |
| d | = | 1010 |
| $\ell$ | = | 010 |
| p | = | 001 |
| s | = | 000 |
| u | = | 11 |
| w | = | 011 |

Decompress the string using the dictionary and write the final message.

(b) Agent Bond doesn't know the password beforehand, but upon seeing the decoded string, she immediately realizes that the message has been tampered with. Explain how Jane determined this.

## Run Length Encoding

(a) Use run length encoding to compress the string shown below:

`01001000010000000011111110111010`

(b) State the compression ratio achieved.

(c) Use run length decoding to decompress the string shown below:

`11110001110001110001100011001001`

# Lempel-Ziv-Welch Encoding

Encode the following string using LZW compression:

DARK_DAN_BARKS_DANK

| Char | Ascii Value |
|---|---|
| A | 65 |
| B | 66 |
| D | 68 |
| K | 75 |
| N | 78 |
| R | 82 |
| S | 83 |
| _ | 95 |

Add new entries to the encoding dictionary starting at value 128.

# String Matching Automata

Dr. Taro invented a new string matching automata called NieR: Automata. His robot assistant Pascal discovers that it accepts three patterns:

```
2B9S
A2
B9A
```

The alphabet is $\sum = \{2, 9, A, B, S\}$.

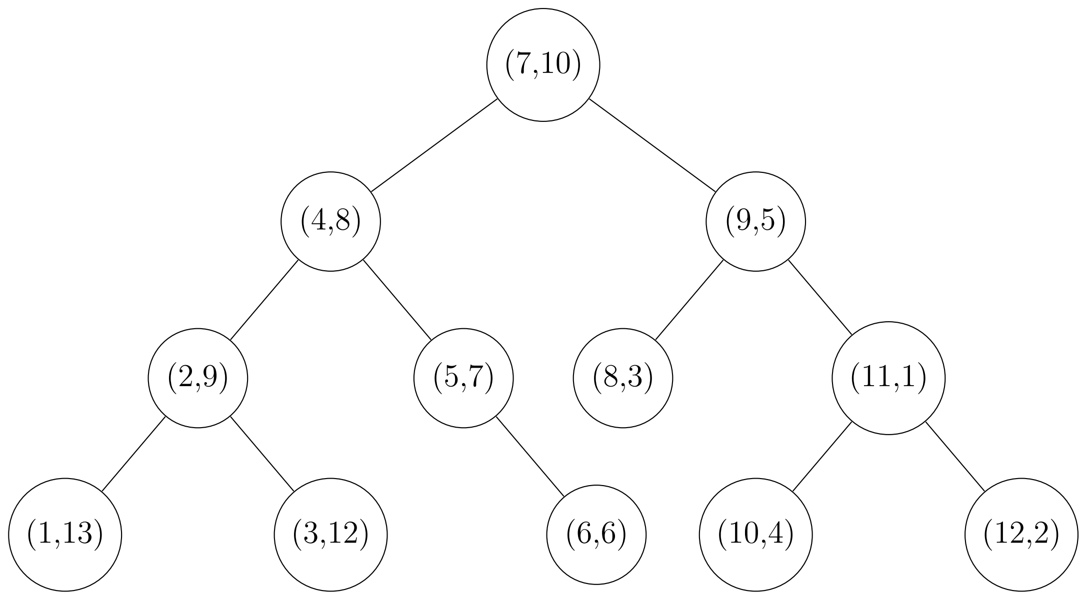
(a) Draw a deterministic finite automata (DFA )that accepts the strings as NieR: Automata. Handle all transitions. Assume there is no more input once an accepting state is reached.

(b) Using the DFA from part a), show the states traversed when the DFA reads the string 2B92SAB9A.

(c) Pascal accidentally let malware infect NieR: Automata. For some reason, it now does Boyer-Moore string matching for pattern 92BAS2B. Draw the suffix skip array.

## Range Trees

Consider the x-BST of a range tree shown below:




(a) Draw the y-BSTs at nodes (2,9), (5,7), and (9,5).

(b) For the query range $R = [0, 7.5] \times [9, 14]$, identify the boundary nodes, inside nodes, and outside nodes for just the x-dimension.

## Tries

Given a compressed trie T that stores a list of binary strings, write an algorithm *Consecutive*($T_1$, $T_2$) that takes as input two binary strings in T and outputs true if the strings are consecutive in an in-order traversal of the trie and false otherwise.

For example, suppose T stores: {000, 01, 0110, 101, 11}.

*Consecutive*(0110, 101) outputs true.
*Consecutive*(01, 000) outputs true.
*Consecutive*(11, 000) outputs false.

## Order Notation

Professor Thick has recently invented a new class of functions called $Onion(f)$. A function $g(n)$ is in $Onion(f(n))$ if there exists a constant $c > 0$ such that $g(n) \leq cf(n)$ for all $n \geq 0$. We assume that $f(n)$ and $g(n)$ are functions that map positive integers to non-negative reals.

(a) Give functions $g$ and $f$ such that $g(n) \in Onion(f(n))$.

(b) Professor Thick says: "If $g(n) \in O(f(n))$, then $g(n) \in Onion(f(n))$ because O is the first letter of Onion". Prove this claim by first principles or disprove with a counterexample.

## Burrows Wheeler Transform

The following key was encoded by the Burrows-Wheeler Transform.

`EPESLPP$ASEAR`

Decrypt it using the method outlined in the slides, showing the array of tuples A, sort(A), and each value of $j$.

## Suffix Trees

Given a string T of length $n$, suppose we have already constructed the suffix tree for T.

(a) Draw the suffix tree for the string `GCTAGCTAG`.

(b) The longest repeated substring is the longest substring of a string that occurs at least twice. For example, the longest repeated substring of `GCTAGCTAG` is `GCTAG`. Create an algorithm to find the longest repeated substring of T in $O(n)$ time.

## 2-3 Trees

(a) Insert the following keys, in the order given, into an initially empty $(2, 3)$-tree: 34, 4, 8, 5, 40, 11, 6, 12, 16, 21, 7, 9. Show the tree after every insertion.

(b) Remove the keys in this order: 5, 6, 4, 21, 9, 8, 40, 11, 7, 16, 34, 12. Show the tree after every removal.

## Range Query

Consider an array $A$ of $n$ integers. We want to implement a range query called MaxDiff$(i, j)$ which will find the maximal difference between two elements from $A[i]$ to $A[j]$, inclusive, for $i < j$. For example, if you run the query MaxDiff(3, 7) on the array below:

A = 5 0 2 8 9 4 6 7 6 1 3

Between indices 3 and 7 in the array above, the largest number is 9 and the smallest number is 4, so the maximal difference is $9 - 4 = 5$.

Using a data structure, implement MaxDiff$(i,j)$ so that its runtime is $O(\log n)$. The data structure must use $O(n)$ space. There are no limits on the time for preprocessing the array into the data structure, but it should not be a randomized algorithm.