Chapter 1

Hashing

1.1 Exercise

The goal of hashing is to produce a search that takes

a) $O(1)$ time

b) $O(n^2)$ time

c) $O(\log n)$ time

d) $O(n \log n)$ time

1.2 Exercise

Consider a hash table of size seven, with starting index zero, and a hash function $(3x + 4) \mod 7$. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

a) $8, \_, \_, \_, \_, \_, 10$

b) $1, 8, 10, \_, \_, \_, 3$

c) $1, \_, \_, \_, \_, \_, 3$

d) $1, 10, 8, \_, \_, \_, 3$

1.3 Exercise

A hash table can store a maximum of 10 records, currently there are records in location $1, 3, 4, 7, 8, 9, 10$. The probability of a new record going into location 2, with hash functions resolving collisions by linear probing is

a) 0,1

b

c

d

1

Which of the following scenarios leads to linear running time for a random search hit in a linear-probing hash table?

a) All keys hash to different indices

b) All keys hash to an even-numbered index

c) All keys hash to same index

d) All keys hash to different even-numbered indices

1.5 Exercise

Given the following input $(4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199)$ and the hash function $x \mod 10$, which of the following statements are true? (GATE CS 2004)

i. $9679, 1989, 4199$ hash to the same value

ii. $1471, 6171$ has to the same value

iii. All elements hash to the same value

iv. Each element hashes to a different value

a) i only

b) ii only

c) i and ii only

d) iii and iv

1.6 Exercise

A hash table of length 10 uses open addressing with hash function $h(k) = k \mod 10$, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
[illegible]	
[illegible]	42
[illegible]	23
[illegible]	34
[illegible]	52
[illegible]	46
[illegible]	33
[illegible]	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

a) 46, 42, 34, 52, 23, 33

b) 34, 42, 23, 52, 33, 46

c) 46, 34, 42, 23, 52, 33

d) 42, 46, 33, 23, 34, 52

1.7 Exercise

How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table given in Exercise 1.6 above?

a) 10

b) 20

c) 30

d) 40

Chapter 2

Disjoint Sets

2.1 Exercise

A relation R on a set S, defined as $x \in y$ if and only if $y \in x$. This is an example of?

a) reflexive relation

b) symmetric relation

c) transitive relation

d) invalid relation

2.2 Exercise

What is the definition for Ackermann's function?

a) $A(1, i) = i + 1$ for $i \geq 1$

b) $A(i, j) = i + j$ for $i \geq j$

c) $A(i, j) = i + j$ for $i = j$

d) $A(1, i) = i + 1$ for $i < 1$

2.3 Exercise

____________ is one of the earliest forms of a self-adjustment strategy used in splay trees, skew heaps.

a) Union by rank

b) Equivalence function

c) Dynamic function

d

2

Wh number of nodes of rank r?

a

b) N/2

c) $N/2^r$

d) N^r

2.5 Exercise

In the Union/Find algorithm, the ranks of the nodes on a path will increase monotonically from?

a) leaf to root

b) root to node

c) root to leaf

d) left subtree to right subtree

Chapte

Heaps & HeapSort

3.1 Exercise

On which algorithm is heap sort based on?

a) Fibonacci heap

b) Binary tree

c) Priority queue

d) FIFO

3.2 Exercise

In what time can a binary heap be built?

a) $O(N)$

b) $O(N \log N)$

c) $O(\log N)$

d) $O(N^2)$

3.3 Exercise

In a binary max heap containing n numbers, the smallest element can be found in time

a) $O(n)$

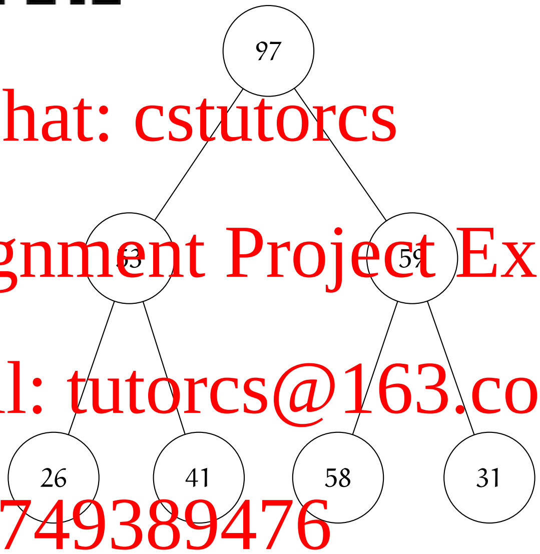
b) $O(\log n)$

c) O(log log n)

d

3

Co p after buildheap phase. What will be its corresponding-
ing



a) 26, 53, 41, 97, 58, 59, 31

b) 26, 31, 41, 53, 58, 59, 97

c) 26, 41, 53, 97, 31, 58, 59

d) 97, 53, 59, 26, 41, 58, 31

3.5 Exercise

Consider a binary max-heap implemented using an array. Which one of the following array represents a binary max-heap?

a) 25, 12, 16, 13, 10, 8, 14

b) 25, 12, 16, 13, 10, 8, 14

c) 25, 14, 16, 13, 10, 8, 12

d) 25, 14, 12, 13, 10, 8, 16

3.6 Exerc[illegible]

Suppose we are [illegible]ht integers using heapsort, and we have just finished so[illegible]heapify or minheapify) operations. The array now looks [illegible] 12, 27, 28. How many heapify operations have been perf[illegible]

a) 1

b) 2

c) 3 or 4

d) 5 or 6

3.7 Exercise

Consider a binary min heap containing n elements and every node is having degree 2 (i.e. full binary min heap tree). What is the probability of finding the largest element at the last level?

a) $1/2$

b) 1

c) $1/n$

d) $1/2^n$

Chapte

Binary search tree & AVL trees

4.1 Exercise

What is the maximum height of an AVL tree with p nodes?

a) p

b) log p

c) log p/2

d) p/2

4.2 Exercise

Given an empty AVL tree, how would you construct AVL tree when a set of numbers are given without performing any rotations?

a) just build the tree with the given input

b) find the median of the set of elements given, make it as root and construct the tree

c) use trial and error

d) use dynamic programming to build the tree

4.3 Exercise

Which of the following is TRUE?

a) The cost of searching an AVL tree is $\Theta(\log n)$ but that of a binary search

b g an AVL tree is $\Theta(\log n)$ but that of a complete g $n)$

c a binary search tree is $O(\log n)$ but that of an AVL

d an AVL tree is $\Theta(n \log n)$ but that of a binary search tree is $O(n)$

4.4 Exercise

The worst case running time to search for an element in a balanced in a binary search tree with $n \cdot 2^n$ elements is

a) $\Theta(n \log n)$

b) $\Theta(n \cdot 2^n)$

c) $\Theta(n)$

d) $\Theta(\log n)$

4.5 Exercise

Consider the below left-left rotation pseudo code where the node contains value pointers to left, right child nodes and a height value and Height() function returns height value stored at a particular node.

```
avltree leftrotation(avltreenode z):
    avltreenode w = x-left
    x-left = w-right
    w-right = x
    x-height = max(Height(x-left),Height(x-right))+1
    w-height = max(missing)+1
return w
```

What is missing?

a) Height(w-left), x-height

b) Height(w-right), x-height

c) Height(w-left), x

d) Height(w-left)

Chapte

Red Black trees

5.1 Exercise

What are the operations that could be performed in $O(\log n)$ time complexity by red-black tree?

a) insertion, deletion, finding predecessor, successor

b) only insertion

c) only finding predecessor, successor

d) for sorting

5.2 Exercise

Why do we impose restrictions like

- root property is black
- every leaf is black
- children of red node are black
- all leaves have same black

a) to get logarithm time complexity

b) to get linear time complexity

c) to get exponential time complexity

d) to get constant time complexity

5.3 E

Wh an application of Red-black trees and why?

a efficiently

b s efficiently

c ss schedulers, maps, sets

d

5.4 Exercise

When it would be optimal to prefer Red-black trees over AVL trees?

a) when there are more insertions or deletions

b) when more search is needed

c) when tree must be balanced

d) when log(nodes) time complexity is needed

5.5 Exercise

What is the below pseudo code trying to do, where pt is a node pointer and root pointer?

```
redblack(Node root, Node pt) :
    if (root == NULL)
       return pt

    if (pt.data < root.data)
    {
        root.left  =  redblack(root.left, pt);
        root.left.parent = root
    }
    else if (pt.data > root.data)
    {
        root.right = redblackt(root.right, pt)
        root.right.parent = root
    }
    return root
```

a) insert a new node

b) delete a node

c) search a node

d) count the number of nodes

Chapter 6

Algorithm Paradigms

6.1 Exercise

If a problem can be solved by combining optimal solutions to non-overlapping problems, the strategy is called

a) Dynamic programming

b) Greedy

c) Divide and conquer

d) Recursion

6.2 Exercise

What problem is not example of paradigm **Divide and Conquer**?

a) Mergesort

b) Prim's algorithm

c) Strassen's algorithm

d) FFT

6.3 Exercise

Match the following with respect to algorithm paradigms:

- Merge sort

angulation

i. Dynamic Programming

ii. Greedy approach

iii. Divide and conquer

iv. Back tracking

a

b) ii i iv iii

c) ii i iii iv

d) iii ii i iv

6.4 Exercise

Select the wrong statement from the following given options

a) Dynamic programming is applicable when subproblems are not independent.

b) Divide and conquer algorithm does more work than necessary repeatedly solving the common subproblems

c) Dynamic programming solves each problem exactly once and saves the result into a table.

d) Longest path problem has optimal substructure property.

6.5 Exercise

Predict output of following pseudo code:

```
function fun(int n)
{
    if (n == 4)
        return n;
    else return 2*fun(n+1);
}

print(fun(2))
```

a) 4

b) 8

c) 16

d) Runtime Error

Chapter 7

Divide and Conquer

7.1 Exercise

What is complexity of $T(n)$, if

$$T(n) = 2 \cdot T(n/2) + c \cdot n, T(1) = a = const$$

a) $T(n) = O(n^2)$

b) $T(n) = O(n \log n)$

c) $T(n) = O(n \log \log n)$

d) $T(n) = O(n)$

7.2 Exercise

What does the below pseudo compute?

```
int x, y, m, n;
input: x, y;
/* x > 0 and y > 0 */
m = x; n = y;
while (m != n)
{
    if(m > n):
        m = m - n;
    else:
        n = n - m;
}
print(n);
```

a) $x + y$ using repeated subtraction

b) $x \mod y$ using repeated subtraction

c) the greatest common divisor of x and y

d) [illegible] ultiple of x and y

7 [illegible]

Co [illegible] $(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, where $a_i \neq 0$, for all i. [illegible] of multiplications needed to evaluate p on an input x is:

a) 3

b) [illegible]

c) 6

d) 9

7.4 Exercise

Consider a situation where you don't have function to calculate power (pow() function in Java) and you need to calculate x^n where x can be any number and n is a positive integer. What can be the best possible time complexity of your power function?

a) $O(n)$

b) $O(n \log n)$

c) $O(\log \log n)$

d) $O(\log n)$

7.5 Exercise

Consider the problem of computing min-max in an unsorted array where min and max are minimum and maximum elements of array. Algorithm A_1 can compute min-max in a_1 comparisons without divide and conquer. Algorithm A_2 can compute min-max in a_2 comparisons by scanning the array linearly. What could be the relation between a_1 and a_2 considering the worst case scenarios?

a) $a_1 < a_2$

b) $a_1 > a_2$

c) $a_1 = a_2$

d) Depends on input

Chapter 8

Master Theorem

8.1 Exercise

What is the result of the recurrences which fall under first case of Master's theorem (let the recurrence be given by $T(n) = aT(n/b) + f(n)$ and $f(n) = n^c$)?

a) $T(n) = O\left(n^{\log_b a}\right)$

b) $T(n) = O\left(n^c \log n\right)$

c) $T(n) = O(f(n))$

d) $T(n) = O(n^2)$

8.2 Exercise

What is the result of the recurrences which fall under third case of Master's theorem (let the recurrence be given by $T(n) = aT(n/b) + f(n)$ and $f(n) = n^c$)?

a) $T(n) = O\left(n^{\log_b a}\right)$

b) $T(n) = O\left(n^c \log n\right)$

c) $T(n) = O(f(n))$

d) $T(n) = O(n^2)$

8.3 Exercise

Consider the following recurrence:

$$T(n) = 2T\left(\lceil\sqrt{n}\rceil\right) + 1, \quad T(1) = 1$$

Which one of the following is true?

a) $T(n) = \Theta(\log\log n)$

b

c

c

8

When $n = 2^{2k}$ for some $k \geq 0$, the recurrence relation

$$T(n) = \sqrt{2}T(n/2) + \sqrt{n}, \quad T(1) = 1$$

evaluates to:

a) $\sqrt{n}(\log n + 1)$

b) $\sqrt{n}(\log n)$

c) $\sqrt{n}\log\sqrt{n}$

d) $n\log\sqrt{n}$

8.5 Exercise

The running time of an algorithm is represented by the following recurrence relation:

```
if  n <= 3  then  T(n) = n
else T(n) = T(n/3) + cn
```

Which one of the following represents the time complexity of the algorithm?

a) $\Theta(n)$

b) $\Theta(n \log n)$

c) $\Theta(n^2)$

d) $\Theta(n^2 \log n)$

8.6 Exercise

Consider the following recurrence:

$$T(n) = 8T\left((n - \sqrt{n})/4\right) + n^2$$

Which one of the following represents the time complexity of the algorithm?

a) $\Theta(n)$

b) $\Theta(n \log n$

c) $\Theta(n^2)$

d) $\Theta(n^2 \log$

Chapte

The Loop Invariant

9.1 Exercise

Consider the following procedure:

```
1 procedure NotEfficient(c)
2     x = c
3     y = c
4     while x > 0 do
5         y = y + 1
6         x = x − 1
7     return y
```

State the loop invariant for the while loop on line 3.

a) x

b) y

c) $x + y$

d) c

9.2 Exercise

What is the loop invariant for the **while** loop on line 4 in function gcd?

```
1  int gcd(int K, int M) {
2      int k = K;
3      int m = M;
4      while (k != m) {
5          if (k > m)
6              k = k − m;
7          else
8              m = m − k;
9      }
10     return k;
11 }
```

a) K

b

c

d

9

Consider the **bubbleSort** algorithm.

```
void bubbleSort(int arr[])
{
    int n = arr.length;
    for (int i = 0; i < n-1; i++)
        for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
            {
                // swap arr[j+1] and arr[j]
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
}
```

At the end of i iteration right most $n - i$ elements are sorted and in place

a) True

b) False

9.4 Exercise

Consider the following program fragment for reversing the digits in a given integer to obtain a new integer. Let $n = \overline{D_1 D_2 \dots D_m}$.

```
int n, rev;
rev = 0;
while (n > 0)
{
    rev = rev*10 + n%10;
    n = n/10;
}
```

The loop invariant condition at the end of the i-th iteration is:

a) $n = \overline{D_1 D_2 \dots D_{m-i}}$ and $rev = \overline{D_m D_{m-1} \dots D_{m-i+1}}$

b) $n = \overline{D_{m-i+1} \dots D_{m-1} D_m}$ and $rev = \overline{D_{m-1} \dots D_2 D_1}$

c) $n \neq rev$

d) $n = \overline{D_1 D_2 \dots D_m}$ and $rev = \overline{D_m D_{m-1} \dots D_2 D_1}$

9.5 Exerc

Consider the fo

```
    int j = 9
    for(int i
        j = j
```

State the lo s cycle.

a) $j > 0$

b) $i < 10$

c) $i + j = 10$

d) $i + j = 9$

Chapter 10

Dynamic Programming

10.1 Exercise

Which of the following is (are) property (properties) of a dynamic programming problem?

a) Optimal substructure

b) Overlapping subproblems

c) Greedy approach

d) Both optimal substructure and overlapping subproblems

10.2 Exercise

If a problem can be broken into subproblems which are reused several times, the problem possesses ___________ property.

a) Overlapping subproblems

b) Optimal substructure

c) Memoization

d) Greedy

10.3 Exercise

When dynamic programming is applied to a problem, it takes far less time as compared to other methods that don't take advantage of overlapping subproblems.

a) True

b)

1

In the technique of storing the previously calculated values

a) Saving value property

b) Storing value property

c) Memoization

d) Mapping

10.5 Exercise

Which of the following problems is NOT solved using dynamic programming?

a) 0/1 knapsack problem

b) Matrix chain multiplication problem

c) Edit distance problem

d) Fractional knapsack problem

10.6 Exercise

When a top-down approach of dynamic programming is applied to a problem, it usually

a) Decreases both, the time complexity and the space complexity

b) Decreases the time complexity and increases the space complexity

c) Increases the time complexity and decreases the space complexity

d) Increases both, the time complexity and the space complexity

Chapte

Greedy Algorithms

11.1 Exercise

A greedy algorithm can be used to solve all the dynamic programming problems.

a) True

b) False

11.2 Exercise

What is the objective of the knapsack problem?

a) To get maximum total value in the knapsack

b) To get minimum total value in the knapsack

c) To get maximum weight in the knapsack

d) To get minimum weight in the knapsack

11.3 Exercise

Given items as value,weight pairs $\{\{40, 20\}, \{30, 10\}, \{20, 5\}\}$. The capacity of knapsack is 20. Find the maximum value output assuming items to be divisible.

a) 60

b) 80

c) 100

d) 40

11.4 E

Wh he code if character c_i is at depth d_i and occurs at fre

d) $f_i d_i$

11.5 Exercise

What is the running time of the Huffman encoding algorithm?

a) $O(C)$

b) $O(\log C)$

c) $O(C \log C)$

d) $O(N \log C)$

Chapte

Answer to all problems

CHAPTER 1

Exercises 1.1, page 1

Answer: a)
Explanation: Time complexity is given by the big oh notation.

Exercises 1.2, page 1

Answer: b)
Explanation: A formula generates the hash, which helps to protect the security of the transmission from unauthorized users.

Exercises 1.3, page 1

Answer: b)
Explanation: Hashing is used to index and retrieve items in a database because it is easier to find the item using the shortened hashed key than using the original value.

Exercises 1.4, page 2

Answer: c)
Explanation: If all keys hash to the same location then the i-th inserted key would need i lookups to be found. The probability of looking up i-th key is $1/n$ (since it's random). If you know some probability it's trivial to show that such lookups have linear time.

Exercises 1.5, page 2

Answer: c)
Explanation: Using given hash function $h(x) = x \mod 10$

$$h(9679) = 9679\%10 = 9$$

$$h(1989) = 1989\%10 = 9$$
$$h(4199) = 4199\%10 = 9$$
$$h(1471) = 1471\%10 = 1$$
$$h(6171) = 6171\%10 = 1$$

As and 4199 hash to same value 9. Also, 1471 and 6171 has refore, statement (i) and (ii) are correct which match wit

Exercises 1.6, page 2

Answer: **c)**
Explanation: We will check whether sequence given in option a) can lead to hash table given in question. Option a) inserts 46, 42, 34, 52, 23, 33 as:

- For key 46, h(46) is 46%10 = 6. Therefore, 46 is placed at 6th index in the hash table.
- For key 42, h(42) is 42%10 = 2. Therefore, 42 is placed at 2nd index in the hash table.
- For key 34, h(34) is 34%10 = 4. Therefore, 34 is placed at 4th index in the hash table.
- For key 52, h(52) is 52%10 = 2. However, index 2 is occupied with 42. Therefore, 52 is placed at 3rd index in the hash table. But in given hash table, 52 is placed at 5th index. Therefore, sequence in option A can't generate hash table given in question.

In the similar way, we can check for other options as well which leads to answer as **c)**.

Exercises 1.7, page 3

Answer: **c)**
Explanation: The first key which is not at the index computed by hash function is 52. It means index 2, 3 and 4 were already occupied and therefore, key 52 is placed at index 5.

The keys 42, 23 and 34 are present at index 2, 4, and 4 respectively. As these keys are at their correct position, their order of insertion does not matter. These 3 keys can be inserted in 3! = 6 ways. Therefore, the sequence will be any order of (42, 23, 34) followed by 52.

The next key which is not at the index computed by hash function is 33. It means indexes 3 to 6 were already occupied and key 33 is placed at index 7. Therefore, it is the last key to be inserted into hash table.

The key 46 is present at its correct position computed by hash function. Therefore, it can be inserted at any place in the sequence before 33. The sequence excluding 33 has 4 elements 42, 23, 34, 52 which create 5 positions for 46 (3 in-between and 2 corners). Total number of ways is: $6 \cdot 5 = 30$.

CHAPTER 2

Exercises 2.1

Answer: b)
Explanation: in an equivalence relation is defined as $x \in y$ if and onl

Exercises 2.2

Answer: a
Explanation: The Ackermann's function is defined as $A(1, i) = i + 1$ for $i \geq 1$. This form in text grows faster and the inverse is slower.

Exercises 2.3, page 5

Answer: d)
Explanation: Path compression is one of the earliest forms of self-adjustment used in extremely important strategies using theoretical explanations.

Exercises 2.4, page 6

Answer: c)
Explanation: Each node of a rank r is the root of a subtree of at least 2^r. Therefore, there are at most $N/2^r$ disjoint subtrees.

Exercises 2.5, page 6

Answer: a)
Explanation: One of the lemmas state that, in the Union/Find algorithm, the ranks of the nodes on a path will increase monotonically from leaf to root.

CHAPTER 3

Exercises 3.1, page 7

Answer: c)
Explanation: Heap sort is based on the algorithm of priority queue and it gives the best sorting time.

Exercises 3.2, page 7

Answer: a)
Explanation: The basic strategy is to build a binary heap of N elements which takes $O(N)$ time.

Exercises 3.3, page 7

Answer: a)
Explanation: In a max heap, the smallest element is always present at a leaf node. So we need to check for all leaf nodes for the minimum value. Worst case complexity will be $O(n)$.

Exercises 3.4, page 8

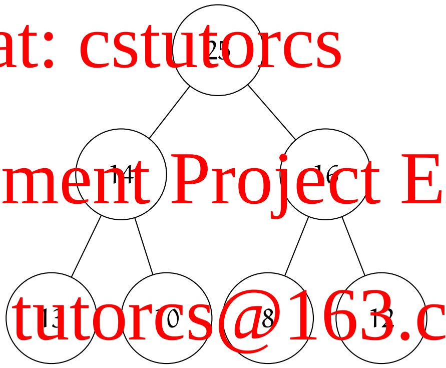
Ans

Exp ng a max heap using the elements 97, 53, 59, 26, 41, 58, 31 wil like that.

Ex

Ans

Exp ax-heap if data at every node in the tree is greater tha ren' s data. In array representation of heap tree, a node at index i has its left child at index $2i + 1$ and right child at index $2i + 2$.



Exercises 3.6, page 9

Answer: b)
Explanation: In Heapsort, we first build a heap, then we do following operations till the heap size becomes 1. a) Swap the root with last element b) Call heapify for root c) reduce the heap size by 1. In this question, it is given that heapify has been called few times and we see that last two elements in given array are the 2 maximum elements in array. So situation is clear, it is maxheapify which has been called 2 times.

Exercises 3.7, page 9

Answer: b)
Explanation: Always 1 as maximum element will be present in the leaf nodes in case of binary min heap.

CHAPTER 4

Exercises 4.1, page 11

Answer: b)
Explanation: Consider height of tree to be 'he', then number of nodes which totals to p can be written in terms of height as $N(he) = N(he-1)+1+N(he-2)$.

since N(he) which is n can be written in terms of height as the beside recurrence relation which = $O(\log p)$ as worst case height.

Exercises 4.2

Answer: b)
Explanation: find the median element among them, make it as root right subtrees with elements lesser and greater than th rsively. this ensures the subtrees differ only by height

Exercises 4.3, page 11

Answer: a)
Explanation: AVL tree's time complexity of searching, insertion and deletion are equal $\Theta(\log n)$. But a binary search tree, may be skewed tree, so in worst case BST searching, insertion and deletion complexity are equal $O(n)$.

Exercises 4.4, page 12

Answer: c)
Explanation: Time taken to search an element is $\Theta(\log n)$ where n is number of elements in AVL tree. As number of elements given is $n \cdot 2^n$, the searching complexity will be $\Theta(\log(n \cdot 2^n))$ which can be written as

$$\Theta(\log(n \cdot 2^n)) = \Theta(\log n) + \Theta(\log 2^n) = \Theta(\log n) + \Theta(n \cdot \log 2) = \Theta(\log n) + \Theta(n)$$

As $\log n$ is asymptotically smaller than n, $\Theta(\log n) + \Theta(n)$ can be written as $\Theta(n)$ which matches option c).

Exercises 4.5, page 12

Answer: a)
Explanation: In the code we are trying to make the left rotation and so we need to find maximum of those two values.

CHAPTER 5

Exercises 5.1, page 13

Answer: a)
Explanation: We impose restrictions (refer Ex. **5.2**) to achieve logarithm time complexities.

Exercises 5.2, page 13

Answer: a)
Explanation: We impose such restrictions to achieve self balancing trees with logarithmic complexities for insertions, deletions, search.

Exercises 5.3, page 14

Ans

Exp used for Linux kernel in the form of completely fair sch g algorithm. It is used for faster insertions, retrievals.

Ex

Ans

Exp th trees are balanced, when there are more insertions and tree balanced, AVL trees should have more rotations, it would be better to use red-black. but if more search is required AVL trees should be used.

Exercises 5.5, page 14

Answer: a)
Explanation: The code is taking the root node and to be inserted node and is performing insertion operation.

CHAPTER 6

Exercises 6.1, page 15

Answer: c)
Explanation: In divide and conquer, the problem is divided into smaller non-overlapping subproblems and an optimal solution for each of the subproblems is found. The optimal solutions are then combined to get a global optimal solution. For example, mergesort uses divide and conquer strategy.

Exercises 6.2, page 15

Answer: **b)**
Explanation: Prim's algorithm is **Greedy Algorithm**.

Exercises 6.3, page 15

Answer: **d)**

Exercises 6.4, page 16

Answer:
Explanation:

Exercises 6.5, page 16

Answer: **c)**
Explanation: $\mathtt{Fun}(2) = 2 \cdot \mathtt{Fun}(3)$ and $\mathtt{Fun}(3) = 2 \cdot \mathtt{Fun}(4) \ldots$ (i) $\mathtt{Fun}(4) = 4 \ldots$ (ii) From equation (i) and (ii), $\mathtt{Fun}(2) = 2 \cdot 2 \cdot \mathtt{Fun}(4)$ $\mathtt{Fun}(2) = 2 \cdot 2 \cdot 4$ $\mathtt{Fun}(2) = 16$. So, **c)** is the correct answer

CHAPTER 7

Exercises 7.1, page 17

Answer: **b)**

Explanation: number of number of inversions, merge sort (and many

$O(n \log n)$

Exercises 7.2

Answer: **c)**

Explanation: This is an implementation of Euclid's algorithm to find GCD.

Exercises 7.3, page 18

Answer: **a)**

Explanation: Multiplications can be minimized using following order for evaluation of the given expression:

$$p(x) = a_0 + x(a_1 + x(a_2 + a_3 x))$$

Exercises 7.4, page 18

Answer: **d)**

Explanation: We can calculate power using divide and conquer in $O(\log n)$ time.

Exercises 7.5, page 18

Answer: **b)**

Explanation: When Divide and Conquer is used to find the minimum-maximum element in an array, Recurrence relation for the number of comparisons is

$$T(n) = 2T(n/2) + 2,$$

where 2 is for comparing the minimums as well the maximums of the left and right subarrays On solving, $T(n) = 1.5n - 2$. While doing linear scan, it would take $2 * (n - 1)$ comparisons in the worst case to find both minimum as well maximum in one pass.

CHAPTER 8

Exercises 8.1, page 19

Answer: **a)**

Explanation: In first case of master theorem the necessary condition is that $c < \log_b a$. If this condition is true then $T(n) = O\left(n \log_b a\right)$.

Exercises 8.2, page 19

Ans

Exp e of master's theorem the necessary condition is that
c > n is true then $T(n) = O(f(n))$.

Ex

Ans

Exp on can be solved by first change of variable and then
Ma

Let $n = 2^m$

$$T(2^m) = T(2^{m/2}) + 1$$

Let $T(2^m) = S(m)$

$$S(m) = 2S(m/2) + 1$$

Above expression is a binary tree traversal recursion whose time complexity is $\Theta(m)$. You can also prove using Master theorem.

$$\begin{aligned} S(m) &= \Theta(m) \\ &= \Theta(\log n) \end{aligned}$$

since $n = 2^m$.

Now, let us go back to the original recursive function $T(n)$:

$$\begin{aligned} T(n) = T(2^m) &= S(m) \\ &= \Theta(\log n) \end{aligned}$$

Exercises 8.4, page 20

Answer: a)

Explanation: Please note that the question is asking about exact solution. Master theorem provides results in the form of asymptotic notations. So we can't apply Master theorem here. We can solve this recurrence using simple expansion or recurrence tree method.

$$\begin{aligned} T(n) &= \sqrt{2}T(n/2) + \sqrt{n} \\ &= \sqrt{2}\left[\sqrt{2}T(n/4) + \sqrt{n/2}\right] + \sqrt{n} \\ &= 2T(n/4) + \sqrt{2}\sqrt{n/2} + \sqrt{n} \\ &= 2\left[\sqrt{2}T(n/8) + \sqrt{n/4}\right] + \sqrt{2}\sqrt{n/2} + \sqrt{n} \\ &= \sqrt{2^3}T(n/8) + 2\sqrt{n/4}) + \sqrt{2}\sqrt{n/2} + \sqrt{n} \\ &= \sqrt{2^3}T(n/8) + \sqrt{n} + \sqrt{n} + \sqrt{n} \\ &= \sqrt{2^3}T\left(n/(2^3)\right) + 3\sqrt{n} \\ &\ldots \\ &= \sqrt{2^k}T(n/(2^k)) + k\sqrt{n} \\ &= \sqrt{2^{\log n}} + \log n\sqrt{n} \\ &= \sqrt{n} + \log n\sqrt{n} \\ &= \sqrt{n}(\log n + 1) \end{aligned}$$

Alternate Sol

This question c bstitution method look:

; GIVEN.

Now use $n = 2$ e relation which gives $2 \cdot (1.414)$ (since value of root o looking at the options use $n = 2$ which satisfies option)

Exercises 8.5, page 20

Answer: **a)**

Explanation:

$$
\begin{aligned}
T(n) &= cn + T(n/3) \\
&= cn + cn/3 + T(n/9) \\
&= cn + cn/3 + cn/9 + T(n/27)
\end{aligned}
$$

Taking the sum of infinite GP series. The value of $T(n)$ will be less than this sum.

$$
\begin{aligned}
T(n) &\leq cn\left(\frac{1}{1 - 1/3}\right) \\
&\leq 3cn/2
\end{aligned}
$$

or we can say

$$cn \leq T(n) \leq 3cn/2$$

Therefore $T(n) = \Theta(n)$

Exercises 8.6, page 20

Answer: **c)**

Explanation: Master Theorem doesn't apply.

Using Akra-Bazzi can ignore $\sqrt{n}/4$, which gives $\Theta(n^2)$. Could also use Master Theorem to get an upper bound of $O(n^2)$ by removing the $\sqrt{n}/4$ term and a lower bound of $\Omega(n^2)$ by replacing the $(n - \sqrt{n})/4$ term by $0.24n$.

CHAPTER 9

Exercises 9.1, page 23

Answer: **c)**

Explanation: As $x + y = 2c$ before the loop and after every iteration of cycle $(x - 1 + y + 1 = 2c)$, we get answer **c)**.

Exercises 9.2, page 23

Answer: **d)**

Explanation: It can be clearly seen that $\gcd(K, M) = \gcd(k, m)$, so it is loop invariant, as it follows from properties of Euclid's algorithm.

Exercises 9.3, page 24

Ans

Exp ort algorithm, after each iteration of the loop largest ele ays placed at right most position. Therefore, the loop inv at the end of i iteration right most i elements are sor

Ex

Ans)

Explanation: Loop invariant must hold at the end of the iteration. In the given code, the least significant digit is taken from n and added to *rev*. So, at the end of i-th iteration, n will have its least significant bits removed and they will be seen in *rev*. So, answer is a).

Exercises 9.5, page 25

Answer: d)
Explanation: It is true (for every iteration) that i + j = 9.

CHAPTER 10

Exercises 10.1, page 27

Answer: d)
Explanation: A problem that can be solved using dynamic programming possesses overlapping subproblems as well as optimal substructure properties.

Exercises 10.2, page 27

Answer: a)
Explanation: Overlapping subproblems is the property in which value of a subproblem is used several times.

Exercises 10.3, page 27

Answer: a)
Explanation: Dynamic programming calculates the value of a subproblem only once, while other methods that don't take advantage of the overlapping subproblems property may calculate the value of the same subproblem several times. So, dynamic programming saves the time of recalculation and takes far less time as compared to other methods that don't take advantage of the overlapping subproblems property.

Exercises 10.4, page 28

Answer: c)
Explanation: Memorization is the technique in which previously calculated values are stored, so that, these values can be used to solve other subproblems.

Exercises 10.5, page 28

Answer: d)
Explanation: [illegible] k problem is solved using a greedy algorithm.

Exercises 10. [illegible]

Answer: b)
Explanation: [illegible] uses the memorization technique which stores the prev [illegible] s. Due to this, the time complexity is decreased but the space complexity is increased.

CHAPTER 11

Exercises 11.1, page 29

Answer: b)
Explanation: A greedy algorithm gives optimal solution for all subproblems, but when these locally optimal solutions are combined it may NOT result into a globally optimal solution. Hence, a greedy algorithm CANNOT be used to solve all the dynamic programming problems.

Exercises 11.2, page 29

Answer: a)
Explanation: The objective is to fill the knapsack of some given volume with different materials such that the value of selected items is maximized.

Exercises 11.3, page 29

Answer: a)
Explanation: The value/weight ratio are: $\{2, 3, 4\}$. So we include the second and third items wholly into the knapsack. This leaves only 5 units of volume for the first item. So we include the first item partially. Final value is $20 + 30 + (40/4) = 60$.

Exercises 11.4, page 30

Answer: c)
Explanation: If character c_i is at depth d_i and occurs at frequency f_i, the cost of the codeword obtained is $\sum f_i d_i$.

Exercises 11.5, page 30

Answer: c)
Explanation: If we maintain the trees in a priority queue, ordered by weight, then the running time is given by $O(C \log C)$.