Exam 2016, questions and answers

Data Structures and Algorithms (University of Pretoria)

56%

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Faculty of Engineering
Department of Computer Science
Data Structures and Algorithms
COS 212

Eksamengeleentheid 2 / Exam Opportunity 2
21/04/2016

39½

Surname, Initials:__________ ____________________

Student/Staff Nr:______________________________

Interne Eksaminatore / Internal Examiners:
Mr M Riekert, A Rakitianskaia, Mr M Nunes, Mr F Oberholzer, Ms E van Zyl

Eksterne Eksaminator / External Examiner:
Dr F Solms

Instruksies / *Instructions*

1. Die enigste additionele toelaatbare materiaal is die voorgeskrewe handboek en 'n nie-programeerbare sakrekenaar gebruik. / *The only additional allowed material is the prescribed textbook and a non-programmable calculator.*
2. Jy het 120 minute om die vraestel te voltooi. / *You have 120 minutes to complete this question paper.*
3. Beantwoord al die vrae. Gee antwoorde in die ruimte wat daarvoor verskaf word. / *Answer all the questions. Give answers in the space provided.*
4. Dui enige rofwerk duidelik aan. / *Indicate any rough work clearly.*
5. Alle betrokke figure verskyn op die laaste bladsy(e). Jy mag hierdie bladsy(e) losmaak van die res van die vraestel. / *All applicable figures appear on the last page(s) of this question paper. You may detach the aforementioned page(s) from the rest of your script.*

2.1: 0
2.2: 0
2.3: 0

Question:	1	2	3	Total
Points:	26	21	23	70
Score:				

Q3: 9½ + 5 + 3

1: 10 + 6

2.7: 1

2.8: 5

2.9: 0

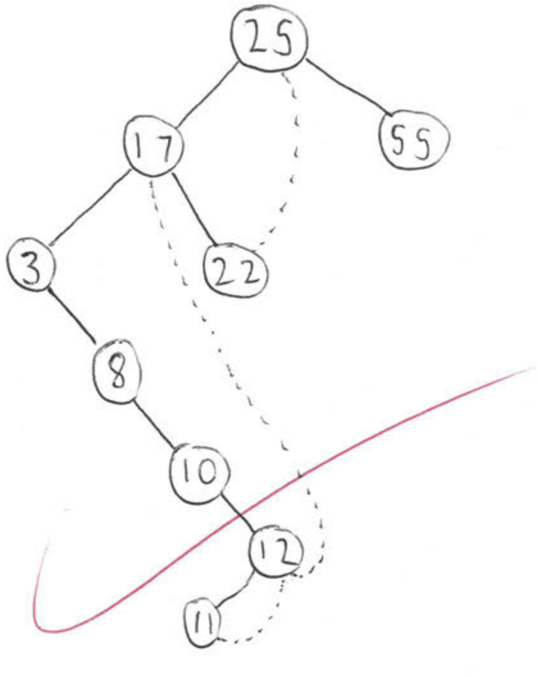
Question 1 1 .... (26 marks)

1.1 Heelgetalle word in 'n enkel-garing Binêre Soekboom ingevoeg in die volgende volgorde:
Integers are inserted into a single-threaded Binary Search Tree in the following order:

`25, 17, 3, 8, 10, 12, 11, 22, 55`

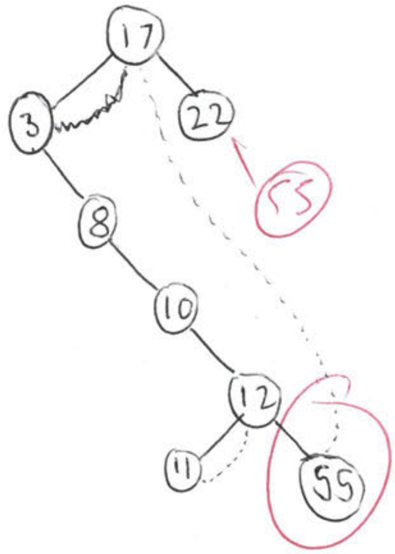
Die garings word gebruik om inorde deurstappings uit te voer. Die volgende vrae volg op mekaar in dat die veranderinge wat aan die boom gemaak word deur elke vraag duur voort na die volgende vraag. Antwoord die volgende:
The threads are used to perform inorder traversals. The following questions follow on each other in that the changes made to the BST by each question persists to the next question. Answer the following:

a) Teken die garingboom. (2)
Draw the threaded tree.



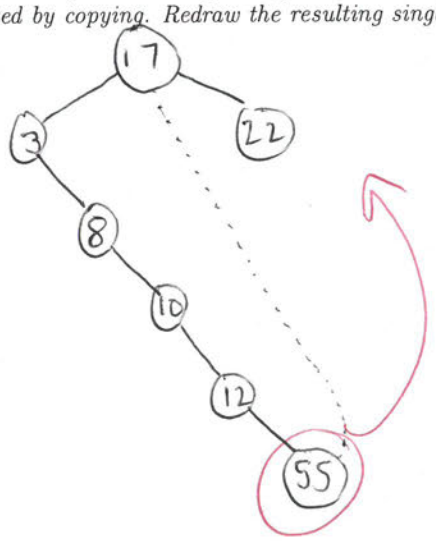
2

b) Die nodus wat 25 stoor word verwyder deur samevoeging. Hêrteken die resulterende garingboom. (2)
The node storing 25 is deleted by merging. Redraw the resulting single-threaded tree.



1

c) Die nodus wat 11 stoor word verwyder deur kopieëring. Hêrteken die resulterende garingboom. (2)
The node storing 11 is deleted by copying. Redraw the resulting single-threaded tree.



0

1.2 Aanvaar dat die volgende klas gebruik word om 'n nodus in 'n enkel-gegaringde binêre boom te implementeer: (5)
Assume the following class is used to implement a node in a single-threaded binary tree:

```
public class TNode<T> {
        protected T data;
        protected boolean hasThread;
        protected TNode<T> left, right;

        // constructors
}
```

Skryf 'n metode `int countThreads(TNode<T> root)` om die totale aantal garings in die boom te tel.
Write a method `int countThreads(TNode<T> root)` to count the total number of threads in the tree.

```
public int countThreads (TNode<T> root) {
        if (root == null)
                return 0;
        int count = (hasThread ? 1 : countThreads(root.right));
        count += countThreads(root.left);
        return count;

}
```

✓ 5

1.3 Die DSW algoritme word toegepas op 'n ongebalanseerde boom. Die rugraat is reeds geskep en het die volgende nodusse:
The DSW algorithm is being applied to an unbalanced tree. The backbone has already being created, and contains the following nodes:

A|B D|G K M P|Q R T U V W Z

a) Hoeveel vlakke sal die resulterende perfek gebalanseerde boom hê? (1)
How many levels will the resulting perfectly balanced tree have?

4 ✓ 1

b) Hoeveel blaarnodusse sal die resulterende perfek gebalanseerde boom hê? (1)
How many leaf nodes will the resulting perfectly balanced tree have?

7 ✓ 1

n=256

1.4 Aanvaar die DSW algoritme word huidiglik op 'n binêre soekboom toegepas. Die rugraat wat geskep is is van hoogte 256 en die `createPerfectTree()` prosedure moet volgende toegepas word. Hoeveel rotasies sal die stelling "**maak n-m rotasies vanaf die top van die rugraat;**" in die algoritme maak? (1)
*Assume that the DSW algorithm is currently being applied to a binary search tree. The backbone that was created is of height 256 and the `createPerfectTree` procedure should be applied next. How many rotations will the statement "**make n-m rotations starting from the top of the backbone;**" from the algorithm perform?* 255

X 0

1.5 Aanvaar die volgende beskrywing van 'n AVL boom: (4)

'n Nodus met die sleutel `K` het twee kinders met sleutels `P` en `B`. Nodus `B` het twee kinders met die sleutels `F` en `A`. Nodus `P` het twee kinders met die sleutels `R` en `M`. Die sleutel `C` en dan die sleutel `E` word in die boom gevoeg. Hêrbalanseer die boom deur die volgende sin te voltooi deur die romeinse syfers te vervang met die name van die nodusse:

Roteer (i) om (ii) en roteer dan (iii) om (iv).

Assume the following description of an AVL tree:

A node with the key `K` has two children with the keys `P` and `B`. Node `B` has two children with the keys `F` and `A`. Node `P` has two children with the keys `R` and `M`. The key `C` and then the key `E` are inserted into the tree. Rebalance the tree by completing the following sentence by substituting the roman numerals with the names of the nodes:

Rotate (i) about (ii) and then rotate (iii) about (iv)

i E ✓

ii C ✓

iii E ✓

iv F ✓

1.6 Oorweeg die AVL boom in `figuur 1`. Aanvaar dat die sleutel 10 uit die boom verwyder word.

Consider the AVL tree in `figure 1`. Assume that the key 10 is removed from the tree.

a) Gee twee redes waarom hierdie boom as 'n AVL boom geklassifiseer kon word voordat die sleutel 10 verwyder is? (2)

Give two reasons why this tree could have been classified as an AVL tree before the key 10 was removed?

1) All balance factors are >=-1 and <=1 ✓

2) It is a valid BST ✓

b) Watter rotasies sal nodig wees om die boom te hêrbalanseer na die verwydering? Dui duidelik die kind, die ouer en die rigting van die rotasie aan. (2)

What rotations need to be performed to re-balance the tree after the deletion? Indicate clearly by specifying the child, the parent, and the direction of rotation.

Rotate child node 3 right about parent node 8.

✗

1.7 Die volgende skikking word gegee: (4)

The following array is given:

[4, 8, 6, 1, 3, 5, 9, 2]

Gee die skikking nadat Floyd se "heapifying" algoritme toegepas is om die skikking in 'n **min-hoop** te omskep.

Show the array after applying Floyd's "heapifying" algorithm to convert the given array to a **min-heap**

✗

1.8 Oorweeg die *treap* in *figuur 2* waar die elemente met die kleinste gewigte die hoogste prioriteite het. Hoeveel rotasies is nodig om die die nodus `P/0` uit die boom te verwyder?
Consider the treap in figure 2, where the elements with the **lowest** *weights have the* **highest** *priority. How many rotations are necessary to delete `P/0` from the treap?*

4

✗

Question 2 2..........(21 marks)

Waar van toepassing kan jy aanvaar dat die volgende `BNode` klas bestaan wat 'n nodus in enige lid van die familie van `B-bome` kan voorstel:
Where applicable you may assume the following `BNode` class exists which is used to represent a node for any member of the family of `B-Trees`

```
class BNode
{
        protected BNode(int elem, int arity)
        {
                numKeys = 1;
                m = arity;

                keys = new int[m-1];
                links = new BNode[m];

                keys[0] = elem;
                for(int i = 0; i < m; ++i)
                        links[i] = null;

        }

        //Returns true if the node is a leaf and false otherwise
        protected boolean isLeaf()
        {
                \\implementation omitted.
        }

        protected int m; // order
        protected int [] keys; //The keys, size: m - 1
        protected BTNode [] links; //The children, size: m
        protected int numKeys; //Number of keys held by the node
}
```

Antwoord die volgende:
Answer the following:

2.1 Wat is die maksimum aantal sleutels wat 'n B-boom van hoogte 20 en m=9 kan huisves? (1)
What is the maximum number of keys that a B-tree of height 20 and m=9 can contain?

$n = -1 + 2(9)^{19}$ ✗

2.2 Wat is die minimum aantal sleutels wat 'n B*-boom van hoogte 20 en m=9 moet huisves? (1)
What is the minimum number of keys that a B*-tree of height 20 and m=9 must contain?

$n \geq -1 + 2(6)^{19}$

2.3 Wat is minimum hoogte van 'n B-boom met m=7 wat 400 sleutels bevat? (1)
What is the minimum height for a B-tree with m=7 which contains 400 keys?

$h \quad \log_4$

2.4 Sal die volgorde waarin sleutels in 'n B-boom geplaas die hoogte van die boom beinvloed? Motiveer jou antwoord. (2)
Will the order in which keys are inserted into a B-tree influence its height? Motivate your answer.

Yes. Sometimes siblings are far apart and the node splits rather than redistributing with a sibling further on the level (This split can propogate up and cause root to split.

2.5 Implementeer die **isLeaf** metode van die **BNode** klas. Die metode stuur waar terug indien die nodus 'n blaarnodus is en vals andersins. (1)
*Implement the **isLeaf** method of the **BNode** class. The method returns true if the node is a leaf and false otherwise.*

```
~~public~~ protected boolean isLeaf() {
        return (links[0] != null);

}
```

2.6 Aanvaar die gedeeltelik gegewe **BTree** klas hieronder wat 'n B-boom voorstel: (6)
Assume the partially given class below which represents a B-tree:

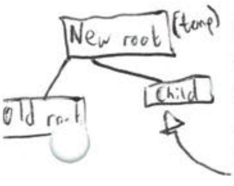
```
public class BTree
{
        public BTree(int arity)
        {
                m = arity;
                root = null;
        }

        //Methods to insert, delete and search omitted.

        private splitRoot(int key; BNode child)
        {
                //implementation omitted
        }

        private sort(int[] array)
        {
                //Sorts an array of elements
                //You may assume this method is correctly implemented.
        }

        private BNode root;//The root node.
        private int m;//Order of the tree.
}
```



Implementeer die **splitRoot** metode wat die vol wortel nodus van 'n **BTree** objek sal splits. Die metode ontvang as parameters die sleutel waarvoor daar nie plek is in die wortel nie (parameter **key**), asook die additionele kind nodus (parameter **child**) wat nie aan die huidige vol wortel nodus geheg kan word nie.
*Implement the **splitRoot** method which will split the full root node of a **BTree** object. The method accepts as parameters the additional key (parameter **key**) for which there is no space in the root, and the additional child (parameter **child**) that cannot be attached to the current full root.* While it's full? ~~Or is this the new root?~~

```
private void splitRoot(int key, BNode child) {
        BNode temp = new BNode(key, m);
        ~~for (int x = (m-1);~~
        while (root.numkeys > Math.ceil(m/2) -1) {
                child.keys[child.numkeys - 1] = root.keys[root.numkeys - 1];
                root.numkeys--;
                child.numkeys++;

        } //old keys now split equally between root and child
         sort(child.keys);
        temp.links[0] = root;
        temp.links[1] = child;
        if (temp.keys[0] < root.keys[root.numkeys -1]) {
                int swap = temp.keys[0];
                temp.keys[0] = root.keys[root.numkeys -1];
                root.keys[root.numkeys -1] = swap; }
         else if (temp.keys[0] > child.keys[0]) {
                int swap = temp.keys[0];
                temp.keys[0] = child.keys[0];
                child.keys[0] = swap;
                sort (child.keys);
          }
          root = temp;

}
```

2.7 'n Bit-boom wat heelgetalle stoor word deursoek vir die waarde 5 (0101). Die soektog lei tot die volgende blaarnodus: (1)
A bit-tree storing integers is searched for the value 5 ($\overset{0123}{0101}$). The search arrives at the following leaf:

n/a	2	3	1	2
ptr0	ptr1	ptr2	ptr3	ptr4

D-bits

Die eerste ry stoor D-bisse; die tweede ry stoor die ooreenstemmende leêrwysers. Watter wyser sal gebruik word om te kyk of die waarde 5 in die blaarnodus gestoor is?
First row stores D-bits; second row stores the corresponding file pointers. Which pointer will be used to check if value 5 is stored in the leaf?

ptr3 ✓

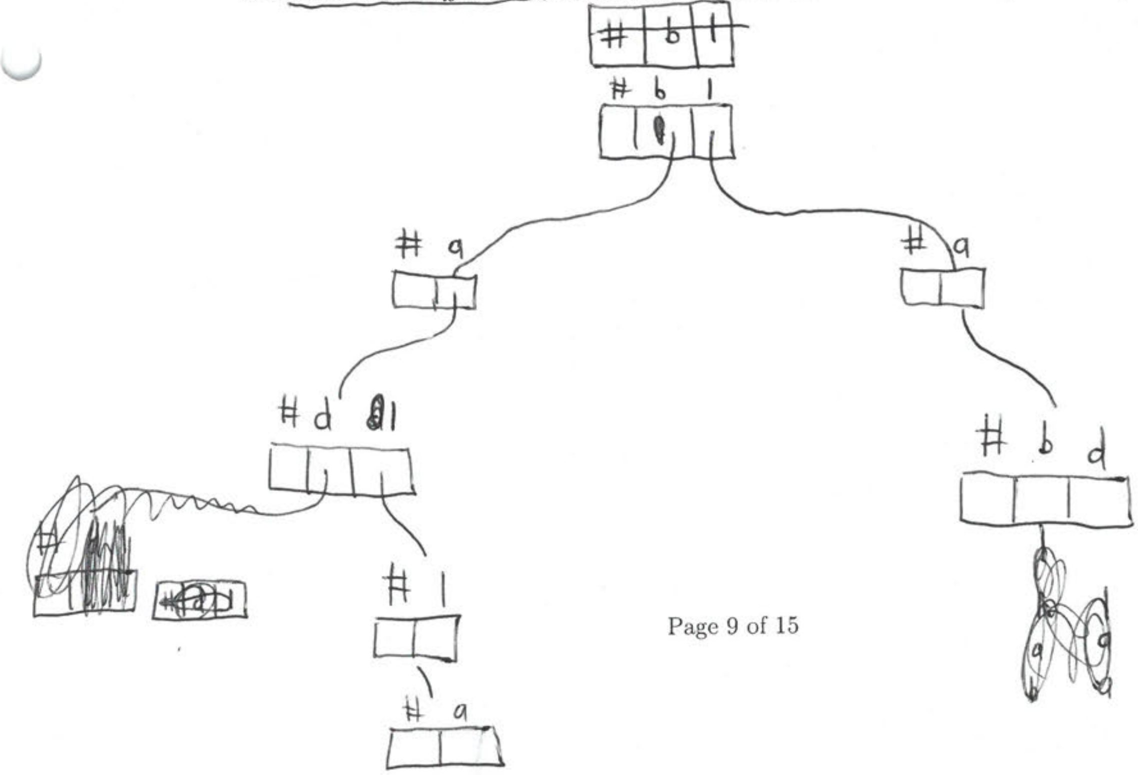
2.8 Die volgende versameling sleutels moet in 'n trie gestoor word: `ball, bald, bad, lab, lad, ballad`
The following set of strings must be stored in a trie: ball, bald, bad, lab, lad, ballad.

Antwoord die volgende:
Answer the following:

a) As die sleutels gestoor word in 'n skikking met 'n vaste grootte, watter skikking grootte moet gebruik word vir die gegewe versameling sleutels? # ● a b l d (1)
If keys are stored in an array of fixed size, what array size should be used for the given set of strings?

5 ✓

b) Teken die minimale trie vir die gegewe versameling sleutels. Aanvaar die sleutels word in datastrukture van aanpasbare groottes gestoor. Alle elemente in die nodusse moet alfabeties gesorteer word. 4/(5)
Draw the minimal trie for the given set of strings. Assume keys are stored in resizable data structures, and leaves store suffixes only. All elements within the nodes must be alphabetically sorted.



2.9 Aanvaar 'n trie a tergo struktuur wat die volgende woorde bevat:
Assume a trie a tergo structure containing the following words:

`no, non, nonsense, moose, meese, car, cat, cash, cashier, butter`

Elke interne nodus bevat die "einde-van-woord" karakter en slegs die karakters wat nodig is alfabeties gerangskik. Daar is geen ander optimerings nie. Antwoord die volgende:
Each internal node contains the "end-of-word" character and only the necessary characters arranged alphabetically. There are no other optimizations. Answer the following:

a) Wat is die hoogte van hierdie struktuur? (1) 0
What is the height of this structure?

6 ✗

b) Hoeveel interne nodusse is daar op die pad na die woord `cashier`? (1) 0
How may internal nodes are there on the path to the word `cashier`?

5 ✗

Question 3 3. . . . . (23 marks)

BELANGRIK: Waar daar 'n keuse is tussen punte in 'n grafiek wat volgende verwerk word, kies hulle **alfabeties**.
IMPORTANT: *Wherever there is a choice among vertices in a graph to be processed next, choose them* ***alphabetically***.

3.1 Die totale graad van 'n ongerigte grafiek $G = (V, E)$ is die som van die grade van al die punte in V. Bewys dat as G se totale graad ewe is dan bevat V 'n ewe aantal punte met onewe grade. Hanteer die punte met ewe grade en die punte met onewe grade as lede van twee disjunkte versamelings. ~~(3)~~ 1½.
The total degree of an undirected graph $G = (V, E)$ is the sum of the degrees of all of the vertices in V. Prove that if the total degree of G is even then V will contain an even number of vertices with uneven degrees. Handle the vertices with even degrees and the vertices with uneven degrees as members of two disjoint sets.

S_E = Set of vertices with even degrees

S_o = set of vertices with odd degrees ✓

S = Set of all vertices

$S_o \neq S_E$

actually, $S_o \cap S_E = \emptyset$
which is stronger.

The total degree is a sum.

Sum of even numbers is even.

The total degree is even, meaning there is an even amount of vertices.

3.2 Hoeveel punte sal 'n komplete grafiek met 6561 lyne hê? (1)
How many vertices will a complete graph with 6561 edges have?

3.3 'n Boom is 'n gerigte asikliese grafiek. Wat sal $|E|$ wees vir 'n boom? (1)
A tree is a directed acyclic graph. What will $|E|$ be for tree?

$|V| - 1$ ✓

3.4 Die pseudokode vir 'n breedte eerste deurstappings algoritme kan gegee word as: (1)
The pseudo code for the breadth first search graph traversal algorithm could be given as:

```
BFS()
{
        for all vertices v
                v.visited=false

        while there is a v such that v.visited==false
        {
                v.visited = true
                q.enqueue(v)
                while !q.isEmpty()
                {
                        v = q.dequeue
                        output(v)
                        for all vertices u adjacent to v
                        {
                                if !u.visited
                                u.visited = true
                                q.enqueue(u)
                        }
                }
        }
}
```

In hierdie algoritme, instede daarvan om vir punte nommers te gee om aan te dui dat hulle besoek was, word 'n veld `visited` dienooreenkomstig na waar of vals gestel. In hierdie algoritme word 'n punt slegs in die ry geplaas nadat die punt se `visited` veld na waar gestel is. Wat sal die implikasie wees indien die `visited` veld na waar gestel word eers nadat 'n punt wat verwerk moet word uit die ry verwyder word?
In this algorithm, instead of giving vertices numbers to indicate that they have been processed, the field `visited` *is used and set to true or false accordingly. In this algorithm a vertex is only placed in the queue after its* `visited` *field has been set to true. What would the implication be when the* `visited` *field is set to true only after a vertex to be processed has been dequeued?*

The vertex would be seen as unvisited when dequeueing and be processed again. ✗

3.5 Dijkstra se kortste pad algoritme hoef nie uit te voer totdat die struktuur `toBeChecked` leeg is nie. Wanneer sal dit geldig wees om die algoritme te staak voordat `toBeChecked` leeg is? (1)
Dijkstra's shortest path algorithm does not need to execute until the structure `toBeChecked` *is empty. When will it be valid to halt the algorithm before* `toBeChecked` *is empty?*

When the destination is found to be isolated ((cannot be reached) or in a different subgraph. ✓

3.6 Hoe moet Dijkstra se kortste pad algoritme aangepas word om die kortste paaie vanaf enige punt tot enige ander punt te bepaal? ~~(1)~~
How should Dijksta's shortest path algorithm be modified so that the shortest paths from any vertex to any other vertex may be calculated?

Change the contents of the ~~to be checked~~. "toBeChecked" data structure ✗

3.7 Hoe moet Dijkstra se kortste pad algoritme aangepas word om die kortste pad te vind vanaf 'n punt v na 'n punt w? ~~(1)~~ 1/2
How should Dijkstra's shortest path be adapted to find the shortest path from a vertex *v* *to a vertex w?*

Check when current vertex being processed is w. ✓

3.8 Beskou die grafiek in *figuur 3* en antwoord die volgende vrae:
Consider the graph in figure 3 and answer the following questions:

a) Gee die volgorde waarin punte besoek sal word indien 'n breedte-eerste soek op die grafiek uitgevoer word. (4)
Give the order in which vertices will be visited if a breadth first search was applied to the graph.

A, G, H, D, F, B, C, J, I, E, X, Z
✓ ✓ ✓ ✓

b) Gestel die grafiek was ongerig, gee die volgorde waarin punte besoek sal word indien 'n diepte eerste soek op die grafiek uitgevoer word. (4)
Suppose the graph was undirected, give the order in which vertices will be visited if a depth first search was applied to the graph.

A, G, H, D, B, C, I, J, F, E, X, Z
✗ ✓ ✗

c) Voer Dijkstra se kortste pad algoritme op die grafiek uit en vul die afstande vir al die punte in die tabel hieronder. Punt A se afstand is na 0 geinitialiseer. ~~(6)~~ 5 1/2.
Apply Dijkstra's shortest path algorithm to the graph and fill in the distances for all of the vertices in the table below. Vertex A's distance is initialized to 0.

Vertex	**Cost**
A	0
B	15
C	35
D	10
E	∞
F	0
G	20
H	50
I	25
J	23
X	∞
Z	∞

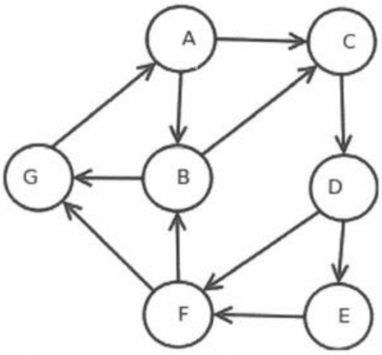


Figure 1: Directed Graph

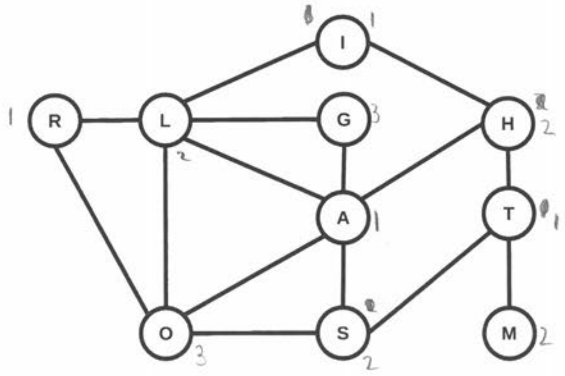


Figure 2: Undirected Graph

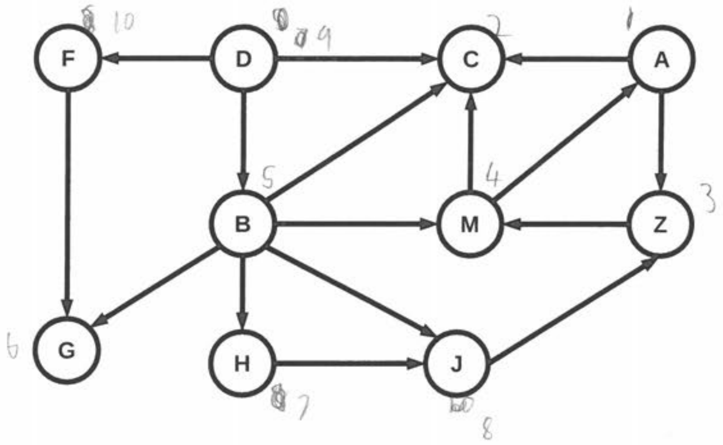


Figure 3: Directed Graph

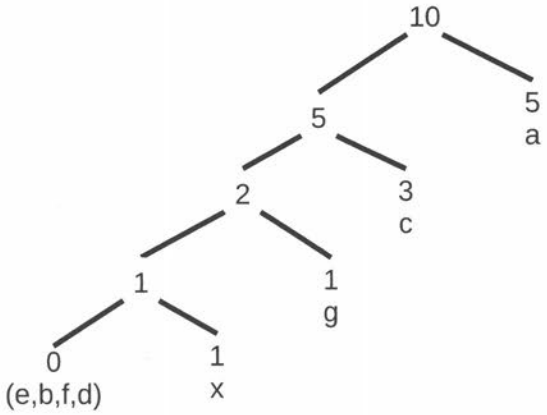


Figure 4: Adaptive Huffman Tree