



Ujian Akhir Semester

Sabtu, 16 Desember 2017
150 menit, open notes (1 lembar A4 tertulis)

CSE602040
Struktur Data & Algoritma
Fakultas Ilmu Komputer
Universitas Indonesia

NPM / Nama / Kelas: _____ / _____ / _____

Peraturan UAS mengikuti aturan standar ujian Fasilkom UI. Tidak boleh menggunakan Internet ataupun alat bantu elektronik lainnya. Peserta ujian hanya boleh menggunakan catatan yang ditulis sendiri. Harap junjung kejujuran akademis.

Ujian ini terdiri dari 3 bagian:

1. Bagian A: Pilihan Ganda terdiri dari 20 soal.
2. Bagian B: Isian Singkat terdiri dari 10 soal.
3. Bagian C: Pemrograman terdiri dari 3 soal.

Ujian mencakup topik-topik dari pertengahan semester hingga akhir semester. Tuliskan jawaban dari masing-masing bagian di lembar jawaban terspisah yang tersedia!

SELAMAT MENGERJAKAN!

CSE602040 – Struktur Data & Algoritma – Ujian Akhir Semester

NPM / Nama / Kelas: _____ / _____ / _____

Bagian A: Pilihan Ganda

1. Berapakah kompleksitas proses initialisasi graph yang diimplementasikan menggunakan *adjacency matrix* dimana V adalah himpunan vertex dan E adalah himpunan edge?
 - a. $O(|V|^2)$
 - b. $O(|E|^2)$
 - c. $O(\log(|V|))$
2. Manakah pernyataan yang benar mengenai B+Tree yang mempunyai m degree:
 - a. Setiap non-leaf (internal) nodes (kecuali root) jumlah anaknya (m tidak null) antara $[m/2]$ dan $m-1$
 - b. Sebuah non-leaf (internal) node yang memiliki m cabang memiliki sejumlah m keys
 - c. Setiap leaves berada pada level yang sama, dengan kata lain, memiliki depth yang sama dari root
 - d. Semua key value hanya muncul satu kali di dalam tree
 - e. Kompleksitas pencarian data pada B+Tree adalah konstan
3. Manakah dari pernyataan berikut yang benar terkait dengan *hash table*:
 - a. *Hash table* memiliki kompleksitas linier pada proses pencarian elemen
 - b. *Quadratic probing* pada *hash table* bisa menyebabkan *primary clustering*
 - c. *Linear probing* diterapkan dengan membuat fungsi hash kedua
 - d. Salah satu penerapan *open hashing* adalah dengan menggunakan *linked list*
 - e. *Linear probing* disarankan untuk *hash table* dengan λ (*load factor*) lebih dari 0.5
4. Manakah dari pernyataan mengenai fungsi hash di bawah ini yang salah:
 - a. *Collision* dapat diminimalkan dengan membuat fungsi hash yang baik
 - b. Fungsi hash yang ideal adalah fungsi yang mudah dihitung, dan dapat membagi key secara merata ke seluruh sel
 - c. Pemilihan fungsi hash yang kurang baik dapat mengakibatkan *primary* dan *secondary clustering*
 - d. Fungsi hash dapat memproses key berupa angka
 - e. Fungsi hash pasti selalu bisa memetakan elemen tepat ke dalam satu cell yang unik
5. Anda sudah mengetahui bahwa tree sebenarnya adalah graph yang spesial. Ketika mempelajari graph, diketahui bahwa terdapat ada 2 skema traversal pada graph, yaitu *breadth-first search* (BFS) dan *depth-first search* (DFS). Manakah dari jenis-jenis traversal tree berikut yang sebenarnya tidak masuk ke dalam kategori DFS?
 - a. Inorder traversal
 - b. Postorder traversal
 - c. Preorder traversal
 - d. Level-order traversal
 - e. Tidak ada jawaban yang benar
6. Manakah pernyataan berikut ini yang benar mengenai *binary heap*:
 - a. *Binary heap* merupakan satu-satunya struktur data yang bisa digunakan untuk implementasi *priority queue*
 - b. *Binary heap* selalu dijaga dalam struktur *full* dan *complete binary tree*
 - c. Worst case operasi *find minimum* pada *maximum binary heap* adalah $O(1)$
 - d. Kompleksitas running time *merge sort* dan *heap sort* adalah berbeda
 - e. Worst case running time operasi *insert* selalu dalam orde $O(\log n)$
7. Struktur data yang paling cocok digunakan dalam algoritma BFS adalah:
 - a. Queue
 - b. ListNode
 - c. Priority Queue
 - d. Stack
 - e. Tree
8. Dalam mengatasi adanya *collision* dalam *hash table*, terdapat kemungkinan menimbulkan sebuah masalah. Ilustrasi masalah yang ditimbulkan dapat dilihat pada contoh gambar ilustrasi di bawah ini:

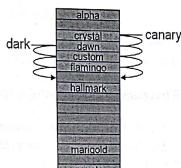
Page 1 of 12

Scanned by CamScanner

Scanned by CamScanner

CSE602040 – Struktur Data & Algoritma – Ujian Akhir Semester

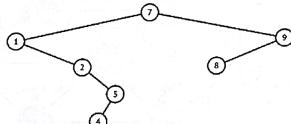
NPM / Nama / Kelas: _____ / _____ / _____



- Masalah tersebut adalah:
- a. Primary clustering
 - b. Linear clustering
 - c. Quadratic clustering
 - d. Secondary clustering
 - e. Double clustering

9. Struktur data Tree paling tepat dalam memecahkan masalah dalam domain permasalahan berikut:
 - a. Penjadwalan tugas-tugas yang dapat saling terkait dalam sebuah perencanaan proyek
 - b. Memodelkan sistem berkas (*file system*), dimana sebuah folder dapat berisi banyak folder dan/atau banyak berkas
 - c. Memodelkan hubungan pertemanan dalam sebuah media sosial
 - d. Menghitung frekuensi kemunculan huruf dalam sebuah dokumen teks
 - e. Memodelkan antrian pencetakan dokumen pada printer
10. Operasi *rangeQuery()*, yakni operasi mencari himpunan nilai yang berada dalam rentang nilai tertentu, paling tepat didukung dalam struktur data berikut:
 - a. B-Tree
 - b. B+Tree
 - c. AVL Tree
 - d. Binary Tree
 - e. Binary Heap

11. Perhatikan BST berikut:



Urutan pemasukan (*insertion*) yang tidak mungkin ialah:

- a. 7, 1, 9, 2, 4, 5, 8
 - b. 7, 1, 2, 5, 4, 9, 8
 - c. 7, 1, 9, 8, 2, 5, 4
 - d. 7, 1, 2, 9, 8, 5, 4
 - e. 7, 1, 9, 4, 5, 8, 2
- a. 64 s.d. 128
 - b. 32 s.d. 64
 - c. 21 s.d. 128
 - d. 21 s.d. 64
 - e. 34 s.d. 128

12. Jika tinggi suatu AVL Tree adalah 7, maka jumlah leaf yang mungkin dari AVL Tree tersebut adalah:
 - a. 64 s.d. 128
 - b. 32 s.d. 64
 - c. 21 s.d. 128
 - d. 21 s.d. 64
 - e. 34 s.d. 128

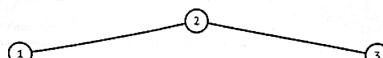
Page 3 of 12

Scanned by CamScanner

CSE602040 – Struktur Data & Algoritma – Ujian Akhir Semester

NPM / Nama / Kelas: _____ / _____ / _____

13. Perhatikan AVL Tree berikut:



Jika pemasukan selanjutnya adalah 5, dan kemudian 4, maka hasil traversal yang benar setelah terjadi pemasukan kedua elemen tersebut adalah:

- a. 2, 1, 3, 4, 5 (level-order traversal)
- b. 2, 1, 3, 4, 5 (in-order traversal)
- c. 2, 1, 4, 3, 5 (level-order traversal)
- d. 2, 1, 4, 3, 5 (post-order traversal)
- e. 1, 2, 3, 4, 5 (pre-order traversal)

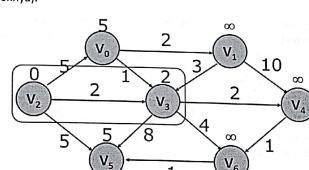
14. Bagaimakah kompleksitas dari pengaksesan seluruh elemen pada suatu *binary heap* melalui *removeMin()* sampai tidak ada elemen lagi yang tersisa (asumsi n adalah jumlah elemen total di *binary heap* tersebut)?

- a. $O(n \log n)$
- b. $O(\log \log n)$
- c. $O(n)$
- d. $O(n \log n)$
- e. $O(1)$

15. Diketahui suatu fungsi hash pada *hash table* $h(k) = 1$, dengan teknik *open hashing* (*separate chaining*).

- Berapakah *average-case complexity* untuk pengaksesan elemen pada *hash table* tersebut?
- a. $O(1)$
 - b. $O(\log n)$
 - c. $O(n)$
 - d. $O(n \log n)$
 - e. $O(n)$

16. Perhatikan gambar berikut: (awan hijau mencakup V_2 dan V_3 , yang menandakan untuk vertex tersebut telah dihitung jarak terpendeknya).



Manakah vertex selanjutnya yang masuk ke awan hijau sesuai algoritma Dijkstra?

- a. V_0
- b. V_1
- c. V_4
- d. V_5
- e. V_2

17. Struktur data yang cocok digunakan pada penentuan vertex yang akan masuk ke himpunan vertex dengan jarak terpendek dalam algoritma Dijkstra adalah:

- a. Priority queue
- b. Queue
- c. Stack
- d. AVL tree
- e. Circular linked list

18. (Mengacu ke graph pada soal no. 16) Jika dilakukan *topological sort*, vertex pertama yang dipilih adalah:

- a. V_0
- b. V_1
- c. V_2
- d. V_3
- e. V_4

Page 4 of 12

Scanned by CamScanner



Ujian Akhir Semester

Sabtu, 16 Desember 2017
150 menit, open notes (1 lembar A4 tertulis)

CSGE602040
Struktur Data & Algoritma
Fakultas Ilmu Komputer
Universitas Indonesia

NPM / Nama / Kelas: _____ / _____

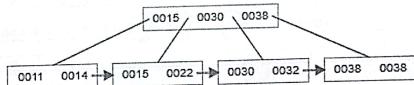
19. Struktur data yang cocok digunakan untuk menyimpan himpunan vertex yang akan dikunjungi dalam algoritma BFS adalah:
- Circular linked list
 - AVL tree
 - Stack
 - Queue
 - Priority queue

20. Implementasi operasi mencari tetangga-tetangga sebuah vertex pada graph dalam bentuk adjacency matrix akan memiliki kompleksitas waktu: (E = himpunan edge, V = himpunan vertex)
- $O(|E|)$
 - $O(|E|^2)$
 - $O(|E| \cdot |V|)$
 - $O(|E|/|V|)$
 - $O(1)$

Bagian B: Isian Singkat

1. Diketahui sebuah graph terdiri dari m buah vertex dan n buah edge. Berapakah kompleksitas proses pencarian edge dengan bobot terbesar pada graph yang direpresentasikan dengan edge list?

2. Diketahui sebuah B+Tree dengan degree $m = 4$ sebagai berikut:



Tuliskan hasil level-order traversal dari B+Tree diatas setelah terjadi pemasukan nilai 0031 dan 0034.

3. Terdapat sebuah pesan yang perlu dikirimkan melalui jaringan. Untuk mempercepat proses pengiriman, pesan tersebut perlu dikomprimi menggunakan Huffman Code. Pesan tersebut memiliki tabel frekuensi sebagai berikut:

Simbol	Frekuensi	Simbol	Frekuensi
c	4	a	2
i	5	k	2
d	3	g	1

Berapakah ukuran data (dalam bit) dari pesan tersebut setelah dikomprimi menggunakan Huffman code?

Page 5 of 12

Scanned by CamScanner

CSGE602040 – Struktur Data & Algoritma – Ujian Akhir Semester

NPM / Nama / Kelas: _____ / _____

- Sebutkan urutan vertex yang dimasukkan ke dalam kelompok hijau/putih dengan algoritma Dijkstra pada graph di atas bila dimulai dari vertex A

7. Suatu B+Tree memiliki 256 search key di dalamnya. Jika diketahui bahwa setiap node memiliki maksimal 8 anak (child), maka berapakah tinggi maksimum dari B+Tree tersebut?

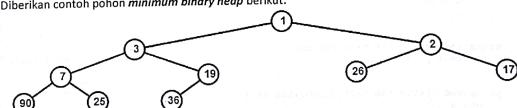
8. Terdapat sebuah operasi pada graph berbasiskan adjacency matrix yang memeriksa apakah sebuah pasangan node v dan w memiliki bidirectional edge (memiliki hubungan timbal-balik). Berapa kompleksitas running time tercepat dari operasi tersebut?

9. Diberikan sebuah maximum binary heap dengan representasi bentuk array sebagai berikut:

value	41	39	21	29	39	7	8	9	6	17
index	0	1	2	3	4	5	6	7	8	9

value	0	1	2	3	4	5	6	7	8	9
index	0	1	2	3	4	5	6	7	8	9

10. Diberikan contoh pohon minimum binary heap berikut:



Lengkapi tabel berikut yang menggambarkan pemetaan posisi elemen-elemen data dalam binary heap ketika direpresentasikan dalam array.

value	0	1	2	3	4	5	6	7	8	9
index	0	1	2	3	4	5	6	7	8	9

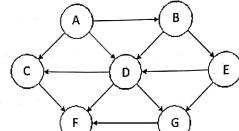
Page 7 of 12

Scanned by CamScanner

CSGE602040 – Struktur Data & Algoritma – Ujian Akhir Semester

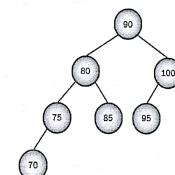
NPM / Nama / Kelas: _____ / _____

4. Perhatikan gambar graph di bawah ini!



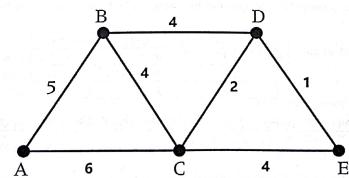
Bila dilakukan topological sorting, pada urutan kunjungan berapa vertex F akan dikunjungi?
Jika ada vertex dengan prioritas sama, urutkan berdasarkan leksikografi label vertex.

5. Perhatikan struktur AVL Tree di bawah ini:



Tuliskan hasil tree traversal berdasarkan dalam urutan preorder setelah dilakukan operasi remove root menggunakan successor inorder!

6. Perhatikan graph di bawah ini:



Page 6 of 12

Scanned by CamScanner

CSGE602040 – Struktur Data & Algoritma – Ujian Akhir Semester

NPM / Nama / Kelas: _____ / _____

Bagian C: Pemrograman

1. Anda sedang membuat implementasi binary heap dengan constructor yang menerima array of integer sebagai input. Jika array tersebut memenuhi bentuk binary heap, maka data bisa dibuat berdasarkan array. Jika tidak memenuhi bentuk binary heap maka akan dibuat data kosong. Untuk melengkapi constructor tersebut diperlukan fungsi untuk mengecek apakah sebuah array of integer memenuhi bentuk binary heap atau tidak.

Tugas Anda adalah buatlah sebuah method yang berfungsi untuk mengecek apakah sebuah array berisi integer memenuhi bentuk binary heap atau tidak. Method tersebut menerima sebuah input berupa Array of Integer dan mengembalikan nilai boolean true/false. Nilai true dikembalikan jika array tersebut memenuhi syarat binary heap (artinya array tersebut berupa binary heap) dan mengembalikan nilai false jika tidak memenuhi syarat binary heap. Berkait disediakan implementasi kelas ArrayListHeap yang sudah dibuat. Anda boleh menggunakan method yang sudah ada dalam mengimplementasikan method isHeap(). Note: binary heap yang dimaksud adalah minimum binary heap.

```

import java.util.Arrays;
import java.util.ArrayList;
import java.util.List;

public class ArrayListHeap {
    private List<Integer> data;

    public ArrayListHeap() {
        data = new ArrayList<Integer>();
    }

    public ArrayListHeap(Integer[] v) {
        if (isHeap(v)) {
            data = Arrays.asList(v);
        } else {
            data = new ArrayList<Integer>();
        }
    }

    private boolean isHeap(Integer[] arr) {
        // TODO Implement me!
    }

    protected static int parentOf(int i) {
        return (i - 1) / 2;
    }

    protected static int leftChildOf(int i) {
        return 2 * i + 1;
    }

    protected static int rightChildOf(int i) {
        return 2 * i + 2;
    }
}
  
```

Page 8 of 12

Scanned by CamScanner

NPM / Nama / Kelas: _____ / _____ / _____

2. Implementasikan method `getReciprocalVertices()` yang akan mengembalikan kumpulan vertex yang memiliki hubungan dua arah. Yang dimaksud dengan hubungan dua arah adalah apabila ada vertex v memiliki edge ke vertex w, maka vertex w juga harus memiliki sebuah edge menuju vertex v. Note: Graph direpresentasikan dalam edge list.

```
import java.util.*;

/**
 * Directed graph dengan bentuk representasi edge list.
 */
public class Graph {
    // Asumsikan kumpulan vertex dan edge tidak kosong
    private List<Vertex> vertices;
    private List<Edge> edges;

    public List<Vertex> getReciprocalVertices() {
        // TODO Implement sel! (Soal no. 2)
    }

    public boolean isReachable(Vertex source, Vertex target) {
        // TODO Implement sel! (Soal no. 3)
    }

    /**
     * Representasi node/vertex pada graph.
     */
    class Vertex {
        String label;
        Object value;

        public Vertex(String label, Object value) {
            this.label = label;
            this.value = value;
        }
    }

    /**
     * Representasi edge pada graph.
     */
    class Edge {
        String label;
        Vertex from, to;

        public Edge(String label, Vertex from, Vertex to) {
            this.label = label;
            this.from = from;
            this.to = to;
        }
    }
}
```

3. Implementasikan method `isReachable(Vertex from, Vertex target)` yang akan mengembalikan nilai Boolean. Method akan mengembalikan boolean true apabila terdapat path (`reachable`) dari vertex `from` menuju vertex `target`. Sebaliknya, apabila tidak terdapat path dari vertex `from` menuju vertex `to`, maka kembalikan nilai Boolean false. Kode dasar implementasi graph mengacu pada kode dasar di soal nomor 2.

SELESAI – JANGAN LUPA MENULISKAN JAWABAN ANDA DI LEMBAR JAWABAN TERPISAH!

Page 9 of 12

NPM / Nama / Kelas: _____ / _____ / _____

Lembar Jawaban Bagian C

1. Lengkapi implementasi `method isHeap()`! Gunakan kode dasar di soal bagian C nomor 1 sebagai acuan.

```
private boolean isHeap(Integer[] arr) {
```

2. Lengkapi implementasi `method getReciprocalVertices()`! Gunakan kode dasar di soal bagian C nomor 2 sebagai acuan.

```
public List<Vertex> getReciprocalVertices() {
```

Page 11 of 12

NPM / Nama / Kelas: _____ / _____ / _____

Jangan lupa mengisi identitas Anda di lembar jawaban ini!

Lembar Jawaban Bagian A

No.	Jawaban	No.	Jawaban	No.	Jawaban	No.	Jawaban
1.		6.		11.		16.	
2.		7.		12.		17.	
3.		8.		13.		18.	
4.		9.		14.		19.	
5.		10.		15.		20.	

Lembar Jawaban Bagian B

No.	Jawaban
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	

9. Lengkapi kotak kosong berikut sesuai dengan instruksi di soal!

value	0	1	2	3	4	5	6	7	8	9
index	0	1	2	3	4	5	6	7	8	9

10. Lengkapi kotak kosong berikut sesuai dengan instruksi di soal!

value	0	1	2	3	4	5	6	7	8	9
index	0	1	2	3	4	5	6	7	8	9

Page 10 of 12

NPM / Nama / Kelas: _____ / _____ / _____

NPM / Nama / Kelas: _____ / _____ / _____

3. Lengkapi implementasi `method isReachable()`! Gunakan kode dasar di soal bagian C nomor 2 sebagai acuan.

```
public boolean isReachable(Vertex source, Vertex target) {
```

SELESAI – PASTIKAN ANDA TELAH MENULISKAN IDENTITAS ANDA DI LEMBAR JAWABAN INI!

Page 12 of 12

2. [8 poin]

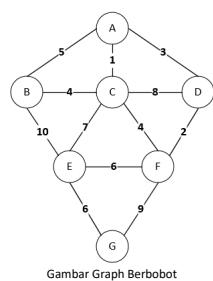
Diberikan sebuah potongan code sebagai berikut. Method fun1 menerima parameter input `BinaryNode t` yang merupakan root dari sebuah Binary Search Tree (BST).

```
BinaryNode fun1 (BinaryNode t){
    if (t == null) throw exception;
    while (t.right != null) {
        t = t.right;
    }
    return t;
}
```

a. Jelaskan dengan singkat apa yang dilakukan method `fun1` tersebut?

b. Diberikan data String sebagai berikut: "Sumatera", "Jawa", "Kalimantan", "Bali", "Sulawesi", "Madura", "Lombok", "Sumbawa".
Gambaran visualisasi tree BST yang menyimpan seluruh data tersebut di mana eksekusi method `fun1` merupakan kasus worst case.

c. Tentukan urutan *insertion* data String tersebut ke dalam BST sehingga menghasilkan tree seperti jawaban no b.



Gambar Graph Berbobot

3. [6 poin]

Terapkan algoritma Dijkstra pada graf berbobot di atas untuk mencari jarak terpendek dari vertex A ke vertex E. Tuliskan isi tabel Dijkstra (lihat lembar jawaban) untuk masing-masing vertex ketika vertex E sudah diketahui jarak minimumnya dari A (saat E masuk ke dalam "white-cloud").

13

4. [10 poin]

Pelajari code Java berikut

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.util.Iterator;
import java.util.List;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Map;
import java.util.HashMap;
import java.util.Queue;
import java.util.PriorityQueue;

class Vertex {
    String label;

    Vertex(String name) {
        label = name;
    }
    public boolean sameVertex(Vertex anotherVertex) {
        return this.label.equals(anotherVertex.label);
    }
}

class Edge implements Comparable<Edge> {
    Vertex v1;
    Vertex v2;
    int weight;

    Edge(Vertex v1, Vertex v2, int weight) {
        this.v1 = v1;
        this.v2 = v2;
        this.weight = weight;
    }

    public void printEdge() {
        System.out.println(v1.label + " " + v2.label);
    }

    public int compareTo(Edge anotherEdge) {
        return Integer.valueOf(this.weight).compareTo(
            Integer.valueOf(anotherEdge.weight));
    }
}
```

14

```
class GraphAnalysis{
    public static void main(String args[]) throws IOException{
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String[] in = reader.readLine().split(" ");
        int numOfVertices = Integer.parseInt(in[0]);
        int numOfEdges = Integer.parseInt(in[1]);

        List<Vertex> allVertices = new ArrayList<Vertex>();
        List<Edge> allEdges = new ArrayList<Edge>();

        for(int i=0; i<numOfVertices; i++){
            String vertexLabel = reader.readLine();
            allVertices.add(new Vertex(vertexLabel));
        }

        for(int j=0; j<numOfEdges; j++){
            in = reader.readLine().split(" ");
            Vertex firstVertex = findVertexByLabel(in[0], allVertices);
            Vertex secondVertex = findVertexByLabel(in[1], allVertices);
            int thisIsWeight = Integer.parseInt(in[2]);

            Edge aNewEdge = new Edge(firstVertex, secondVertex, thisIsWeight);
            allEdges.add(aNewEdge);
        }

        Graph thisIsGraph = new Graph(allVertices, allEdges);

        Graph findThisGraphForYourMark = thisIsGraph.anotherGraph(true);
        System.out.println("Graph TRUE: ");

        for(int k=0; k<findThisGraphForYourMark.edges.size(); k++) {
            findThisGraphForYourMark.edges.get(k).printEdge();
        }

        Graph thisGraphForAnotherMark = thisIsGraph.anotherGraph(false);
        System.out.println("\nGraph FALSE: ");

        for(int k=0; k<thisGraphForAnotherMark.edges.size(); k++) {
            thisGraphForAnotherMark.edges.get(k).printEdge();
        }

        int luckyNumber = thisGraphForAnotherMark.countUsingBFS();
        System.out.println("\n" + luckyNumber);
    }

    public static Vertex findVertexByLabel(String name, List<Vertex> vertices) {
        for(int i=0; i<vertices.size(); i++) {
            Vertex thisIsVertex = vertices.get(i);
            if(thisIsVertex.label.equals(name))
                return thisIsVertex;
        }
        return new Vertex(name);
    }
}
```

15

```
class Graph {
    List<Vertex> vertices;
    List<Edge> edges;

    Graph(List<Vertex> listOfVertices, List<Edge> listOfEdges) {
        vertices = listOfVertices;
        edges = listOfEdges;
        adjacencyList = adjacents();
    }

    Map<Vertex, List<Vertex>> adjacents(List<Edge> listOfEdges) {
        Map<Vertex, List<Vertex>> newAdjacents = new HashMap< > ();
        List<Vertex> adj;

        for(int b = 0; b < listOfEdges.size(); b++) {
            Edge connection = listOfEdges.get(b);

            if(!newAdjacents.containsKey(connection.v1))
                adj = newAdjacents.get(connection.v1);
            else
                adj = new ArrayList<Vertex>();

            adj.add(connection.v2);
            newAdjacents.put(connection.v1, adj);

            if(!newAdjacents.containsKey(connection.v2))
                adj = newAdjacents.get(connection.v2);
            else
                adj = new ArrayList<Vertex>();

            adj.add(connection.v1);
            newAdjacents.put(connection.v2, adj);
        }
        return newAdjacents;
    }

    Graph anotherGraph(boolean typeOfGraph){
        Queue<Edge> myQueue = new PriorityQueue<Edge>();

        for(int i = 0; i < edges.size(); i++){
            myQueue.add(edges.get(i));
        }

        List<Edge> edgeMainGroup = new ArrayList<Edge>();
        List<Edge> edgeEliminatedGroup = new ArrayList<Edge>();

        while(!myQueue.isEmpty()){
            Edge thisIsAnEdge = myQueue.poll();
            if(edgeMainGroup.size() < this.vertices.size() - 1) {
                edgeMainGroup.add(thisIsAnEdge);
                if(containsDonut(thisIsAnEdge, edgeMainGroup)) {
                    edgeMainGroup.remove(thisIsAnEdge);
                }
            } else
                edgeEliminatedGroup.add(thisIsAnEdge);
        }

        if(typeOfGraph)
            return new Graph(vertices, edgeMainGroup);
        return new Graph(vertices, edgeEliminatedGroup);
    }

    int countUsingBFS(){
    }
```

16

```

Map<Vertex, List<Vertex>> adjacencyList = adjacents(this.edges);
Map<Vertex, Boolean> visited = new HashMap<>();

for(int a = 0; a < vertices.size(); a++) {
    visited.put(vertices.get(a), new Boolean(false));
}

//LinkedList "visitQueue" adalah implementasi sebuah queue
LinkedList<Vertex> visitQueue = new LinkedList();
Iterator<Vertex> itrSequence = visitQueue.listIterator();
int counter = 0;

for(int i = 0; i < vertices.size(); i++){
    Vertex oneNode = vertices.get(i);

    if(visited.get(oneNode).booleanValue() == false) {
        counter++;
        visitQueue.add(oneNode);
        visited.put(oneNode, new Boolean(true));

        while(itrSequence.hasNext()){
            Vertex dequeuedNode = visitQueue.poll();
            if(adjacencyList.containsKey(dequeuedNode)) {
                List<Vertex> ourNeighbors = adjacencyList.get(dequeuedNode);
                Iterator<Vertex> itr = ourNeighbors.listIterator();

                while(itr.hasNext()){
                    Vertex oneNeighbor = itr.next();
                    if(visited.get(oneNeighbor).booleanValue() == false) {
                        visited.put(oneNeighbor, new Boolean(true));
                        visitQueue.add(oneNeighbor);
                    }
                }
            }
        }
    }
}

return counter;
}

boolean checkDonutWithDFS(Vertex current, Vertex previous, Map<Vertex, List<Vertex>> adjacencies, Map<Vertex, Boolean> visited) {
    boolean res = false;
    visited.put(current, new Boolean(true));
    Iterator<Vertex> vertexIter = adjacencies.get(current).listIterator();

    while(vertexIter.hasNext()){
        Vertex next = vertexIter.next();
        if(visited.get(next).booleanValue() == false){
            if(checkDonutWithDFS(next, current, adjacencies, visited))
                return true;
        } else if(!next.sameVertex(previous)){
            return true;
        }
    }
    return false;
}

boolean containsDonut(Edge singleEdge, List<Edge> edgeGroup) {

```

17

```

Map<Vertex, List<Vertex>> listOfAdjacencies = adjacents(edgeGroup);
Map<Vertex, Boolean> visited = new HashMap<>();
for(int a = 0; a < vertices.size(); a++) {
    visited.put(vertices.get(a), new Boolean(false));
}

//parameter kedua adalah vertex yang dijamin bukan merupakan vertex dari Graph
if(checkDonutWithDFS(singleEdge.v1, new Vertex("#"), listOfAdjacencies, visited))
    return true;
}

} //akhir dari class Graph

```

a. Method **anotherGraph** pada code di atas (halaman 16) adalah implementasi dari modifikasi algoritma

b. Tuliskan output program jika method **main** pada class **GraphAnalysis** (halaman 15) menerima input sebagai berikut

```

7 12
A
B
C
D
E
F
G
A B 5
A C 1
A D 3
B C 4
B E 10
C D 8
C E 7
C F 4
D F 2
E F 6
E G 6
F G 9

```

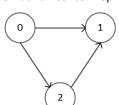
Keterangan: input yang diberikan soal 4b ini adalah graf berbobot yang sama dengan Gambar soal 3 (halaman 13)

18

Bagian C (3 soal @10 poin. Nilai total: 30 poin)

1. Diberikan *directed graph* dalam representasi *adjacency matrix* dengan nilai elemen matriks $[i,j]$ *true* jika vertex i memiliki *adjacent* ke vertex j . Untuk menjalankan algoritma Topological Sort, diperlukan indegree dari setiap vertex.
Buatlah sebuah *method* untuk menghitung *in-degree* dari setiap vertex.

Perhatikan contoh representasi matriks graf berikut



	Vertex j		
Vertex i	0	1	2
0	false	true	true
1	false	false	false
2	false	true	false

2. Buatlah sebuah *method* yang menerima parameter *input* sebuah *Tree* dan mengembalikan nilai *boolean* apakah Tree tersebut merupakan *AVL Tree* atau bukan.

Untuk soal no 2 ini, *input tree* yang akan diujii diasumsikan sudah terdefinisi sebagai object dari Class *BinaryTree* (lihat template jawaban).
Anda tidak diminta mengimplementasikan method *main* untuk memroses input.

3. Diberikan sebuah *undirected graph*. Buatlah sebuah *method* (menerima parameter *input* dua vertex A dan B) yang mencetak seluruh lintasan *elementer* dari vertex A ke vertex B.
Elementary path adalah path yang **TIDAK** mengulang *vertex*

Format input:

Terdapat $(e + 2)$ baris input

- Baris pertama berisi dua bilangan v dan e . v menyatakan jumlah vertex (dalam soal ini, himpunan vertex adalah bilangan bulat dari 0 sampai $v - 1$). e menyatakan jumlah edge
- e baris berikutnya yang mendefinisikan himpunan edge
- Baris terakhir berisi dua bilangan m dan n . Soal yang ditanyakan berarti: cetaklah seluruh *elementary path* dari vertex m ke vertex n

Format output:

Elementary path dicetak per baris (tidak ada ketentuan urutan pencetakan)

Contoh input output:
Misalkan diberikan *complete graph* yang terdiri dari 4 vertex

Contoh Format Input:

```

4 6
0 1
0 2
0 3
1 2
1 3
2 3
1 3

```

Yang ditanyakan adalah cetak seluruh *elementary path* dari vertex 1 ke vertex 3.

Salah satu Kemungkinan Output yang Benar:

(karena urutan pencetakan tidak ada ketentuan khusus, Anda cukup mengimplementasikan solusi untuk salah satu kemungkinan saja)

```

1 3
1 0 3
1 2 3
1 0 2 3
1 2 0 3

```

19

20



Kuis Tryout UAS SDA

100 menit, open notes

CSGE602040
Struktur Data & Algoritma
Fakultas Ilmu Komputer
Universitas Indonesia

NPM / Nama: _____ / _____

Ujian ini terdiri dari 3 bagian:

1. Bagian A: Pilihan Ganda

Terdapat 5 Soal Pilihan Ganda mencakup topik-topik perkuliahan SDA dari pertengahan semester hingga akhir semester.

Lingkari/silang jawaban yang dianggap benar

2. Bagian B: Isian

Terdapat 15 soal mencakup topik-topik perkuliahan SDA dari pertengahan semester hingga akhir semester.

Tuliskan jawaban dari soal-soal Isian di kotak jawaban yang tersedia setelah setiap pertanyaan/soal!

3. Bagian C: Pemrograman

Terdapat 1 soal mengenai Graph.

Tuliskan jawaban dari soal-soal Pemrograman di kertas folio jawaban yang disediakan!

SELAMAT MENGERJAKAN!

Page 1 of 9

Page 2 of 9

CSGE602040 – Struktur Data & Algoritma – Tryout Ujian Akhir Semester

NPM / Nama: _____ / _____

Bagian A: Pilihan Ganda

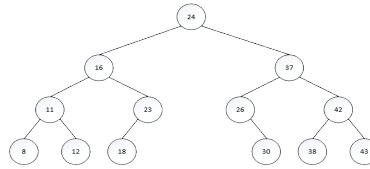
1. Berapakah kompleksitas pencarian pada Binary Search Tree pada kasus worst case?

- a. O(N)
- b. O(log N)
- c. O(N²)
- d. O(N³)
- e. O(1)

2. Manakah pernyataan yang benar mengenai Binary Tree?

- a. Jumlah maksimum node dalam binary tree dengan tinggi k adalah $2^{k+1} - 1$
 - b. Sebuah complete binary tree dengan tinggi k adalah sebuah binary tree yang memiliki 2^k nodes
 - c. Sebuah complete binary dengan tinggi k adalah binary tree yang memiliki jumlah maximum nodes di levels 0 sampai k - 1 (semua level terisi kecuali pada level terakhir, yang terisi dari sisi kanan ke kiri)
 - d. Jumlah leave nodes selalu lebih banyak daripada jumlah internal nodes
 - e. Jumlah nodes pada level ke i (dengan root adalah level 0) pada complete binary tree adalah 2^i
3. Diketahui frekuensi kemunculan karakter: { a/20, b/15, c/5, d/15, e/45 }. Berapa rata-rata besaran bits per karakter dalam Huffman encoding?
- a. 1 bit
 - b. 2.1 bits
 - c. 3.3 bits
 - d. 3.5 bits
 - e. 8 bits

4. Perhatikan Binary Search Tree berikut



Manakah urutan operasi yang tidak menghasilkan BST seperti diatas?

- a. Insert 24, 16, 37, 23, 18, 11, 42, 26, 30, 43, 38, 8, 12
- b. Insert 24, 16, 37, 11, 23, 12, 42, 26, 30, 43, 38, 8, 18
- c. Insert 24, 16, 37, 23, 12, 11, 42, 26, 30, 43, 38, 8, 18
- d. Insert 24, 37, 16, 11, 23, 12, 26, 30, 42, 43, 38, 8, 18

Page 2 of 9

CSGE602040 – Struktur Data & Algoritma – Tryout Ujian Akhir Semester

NPM / Nama: _____ / _____

- e. Insert 24, 37, 16, 23, 18, 11, 8, 26, 30, 42, 43, 38, 12

5. Space complexity dari representasi directed graph menggunakan adjacency list dimana V adalah total vertex dan E adalah total edge ialah ...

- a. O(log V)
- b. O(log E)
- c. O(V)
- d. O(V + E)
- e. O(1)

Bagian B: Short Essay

1. Diberikan sebuah array yang merupakan **minimum binary heap** X berisi data acak sebagai berikut (indeks dimulai dari 0):

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]
15	20	100	33	40	101	135	63	50	51	49

Gambarkan isi dari array tersebut setelah terjadi **satu kali** pengambilan elemen terkecil!

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]
20	33	100	49	40	101	135	63	50	51	49

2. Diberikan sebuah **hashtable** yang bisa menampung 13 elemen (posisi 0 sd. 12). **Hashtable** ini menggunakan **linear probing** untuk penanganan collision. Diketahui **k** adalah key, **i** menunjukkan sudah berapa **collision** yang terjadi, **m** adalah ukuran **hashtable**.

$$H(\text{key}, i) = (F(\text{key}) + i) \bmod m$$

Diketahui juga fungsi F(key) sudah diperoleh sebagai dalam tabel berikut:

Key	F(Key)	Key	F(Key)	Key	F(Key)
Apel	17	Durian	56	Salak	99
Jambu	10	Jeruk	49	Markisa	17
Duku	49	Mangga	105	Nangka	25

Jika dimulai dari kosong, berturut-turut diisikan **Apel**, **Jambu**, **Duku**, **Durian**, dan **Jeruk**, sebutkanlah (posisi, key) dalam **hashtable** tersebut. Sesuai urutan yang terjadi dalam tabel (Misalnya "(4, Apel), (x, Jambu),").

(4, Apel), (5, Durian), (10, Jambu), (11, Duku), (12, Jeruk) ATAU
(4, Apel), (10, Jambu), (11, Duku), (5, Durian), (12, Jeruk), jawaban sampai jeruk saja

(3, Mangga), (9, Salak), (6, Markisa), (1, Nangka)

Page 3 of 9

CSGE602040 – Struktur Data & Algoritma – Tryout Ujian Akhir Semester

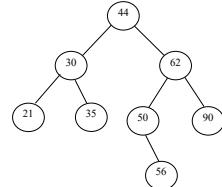
NPM / Nama: _____ / _____

3. Mengacu pada soal sebelumnya, berapa kali **collision** yang terjadi pada urutan key masuk Apel, Jambu, Duku, Durian, Jeruk, Mangga, Salak, Markisa, Nangka?

2

Soal-soal nomor 4 – 8 berikut ini mengacu pada deskripsi ini dan setiap soal (6 – 9) tidak mengacu pada soal lainnya.

Diberikan suatu **binary search tree** yang merupakan AVL Tree sebagai berikut:



Operasi penghapusan key akan mengikuti kaidah **predecessor inorder** (mencari maksimum di **subtree kiri**). Singkatkan operasi rotasi: SRR (rotasi tunggal ke kanan), SLR (rotasi tunggal ke kiri), DRR (rotasi ganda ke kanan), DLR (rotasi ganda ke kiri), TT (tidak terjadi apa-apa).

4. Pada AVL tree awal di atas jika **disisipkan 19** maka akan terjadi operasi apakah?

TT

5. Pada AVL tree awal di atas jika **disisipkan 59** maka akan terjadi operasi apakah?

SLR

6. Pada AVL tree awal di atas jika **dihapuskan 35** maka akan terjadi operasi apakah?

TT

7. Pada AVL tree awal di atas jika **dihapuskan 90** maka akan terjadi operasi apakah?

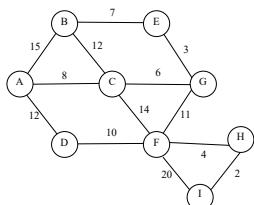
DRR

8. Pada AVL tree awal di atas jika **dihapuskan 30** maka akan terjadi operasi apakah?

TT

Page 4 of 9

NPM / Nama: _____ / _____

Soal-soal nomor 9 – 13 mengacu pada graph berikut:

- Bobot pada setiap sisi menunjukkan jarak antar kedua vertex terkait.
 - "White cloud" adalah vertex yang sudah diketahui dengan pasti *shortest path*-nya dari A menurut algoritma Dijkstra.
9. Jika MST dicari dengan algoritma Kruskal, tuliskan urutan sisi yang didapatkan!

H-I, E-G, F-H, C-G, B-E, A-C, D-F, F-G

10. Jika MST dicari dengan algoritma Prim yang dimulai dari C, tuliskan urutan sisi yang didapatkan!

C-G, E-G, B-E, A-C, F-G, F-H, H-I, F

11. Jika *shortest path* dari A ke setiap vertex dicari dengan algoritma Dijkstra, tuliskan urutan vertex yang didapatkan dari iterasi pertama sampai terakhir. Note: pada iterasi pertama "white cloud" hanya berisi {A}

Iterasi 1: {A}

Iterasi 2: {A, ...}

(dst)

```
{A}
{A, C}
{A, C, D}
{A, C, D, G}
{A, C, D, G, B}
{A, C, D, G, B, E}
{A, C, D, G, B, E, F}
```

Page 5 of 9

NPM / Nama: _____ / _____

{A, C, D, G, B, E, F, H}
{A, C, D, G, B, E, F, H, I}

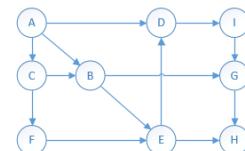
12. Jika DFS dilakukan mulai dari A (tentu saja DFS tidak perlu memperhatikan weight!), tuliskan urutan vertex yang dikunjungi (apabila ada beberapa pilihan jalur, pilih vertex sesuai urutan abjad)

A, B, C, F, D, G, E, H, I

13. Jika BFS dilakukan mulai dari A (tentu saja BFS tidak perlu memperhatikan weight!), tuliskan urutan vertex yang dikunjungi (apabila ada beberapa pilihan, pilih vertex sesuai urutan abjad)

A, B, C, D, E, F, G, H, I

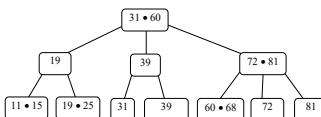
14. Perhatikan graph berikut



Tuliskan urutan vertex jika dilakukan topological sort pada Graph tersebut! (Bila ada vertex dengan indegree 0 lebih dari satu, pilih sesuai urutan abjad)

A, C, B, F, E, D, I, G, H

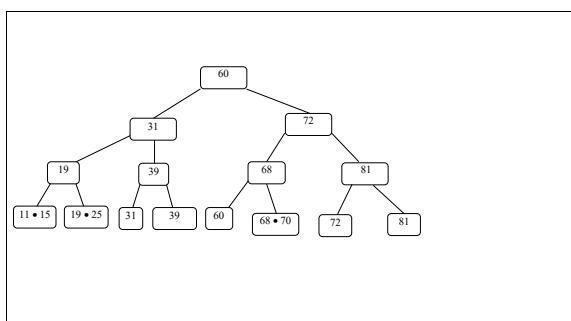
15. Jika Perhatikan contoh B+Tree dengan max degree m = 3 berikut:



Gambarkan kondisi B+Tree setelah dilakukan penyisipan data 70?

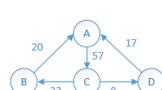
Page 6 of 9

NPM / Nama: _____ / _____



Bagian B: Pemrograman Singkat

1. Directed Graph direpresentasikan dengan Adjacency Matrix dengan dimensi *from - to*. Jika tidak terdapat jalur antar dua vertex, maka nilai weight yang dimasukkan adalah -1. Berikut adalah contoh graph dan konsep representasi matrixnya:



from\to	A	B	C	D
A	0	-1	57	-1
B	20	0	-1	-1
C	-1	33	0	8
D	17	-1	-1	0

Implementasikan method untuk mencari vertex-vertex dengan *out-degree* terbanyak (*getMDV* = *getMaxDegreeVertices*). Method tersebut akan mengembalikan List of Vertex yang bisa berisi satu atau lebih vertex. Note: bisa terdapat lebih dari satu vertex dengan degree sama. Gunakan asumsi jika diperlukan.

```
/** 
 * Representasi dari keterkaitan objek-objek dalam bentuk graph.
 */
public class ReferenceGraph {
    // Asumsikan adjacencyList tidak kosong
    private List<Vertex> nodeList;
    private Edge[][] adjacencyMatrix;
```

Page 7 of 9

NPM / Nama: _____ / _____

```
public List<Vertex> getMDV() {
    List<Vertex> maxDegVert = new ArrayList();
    int max_now = 0;
    int counter = 0;
    for(int i = 0; i < adjacencyMatrix[0].length; i++) {
        for(int j = 0; j < adjacencyMatrix[i].length; j++) {
            if(adjacencyMatrix[i][j].weight > 0) {
                counter++;
            }
        }
        if(counter == max_now) {
            maxDegVert.add(nodeList.get(i));
        } else if(counter > max_now) {
            max_now = counter;
            maxDegVert.removeAll();
            maxDegVert.add(nodeList.get(i));
        } else{
        }
        counter = 0;
    }
    return maxDegVert;
}

/**
 * Representasi objek dalam sebuah bahasa pemrograman.
 */
class Vertex {
    String label;
    Object value; // Menyimpan rujukan ke objek dalam program yang sedang berjalan
    public Vertex(String label, Object value) {
        this.label = label;
        this.value = value;
    }
}

/**
 * Representasi rujukan sebuah objek ke objek Lain, e.g. objek A
 * memiliki sebuah instance variable ke objek Lain B.
 */
class Edge {
    String label;
    int weight;
    public Edge(String label, Vertex weight) {
        this.label = label;
        this.weight = weight;
    }
}
```

Page 8 of 9

NPM / Nama: _____ / _____

SELESAI

Contoh-contoh Soal Pilihan Ganda Struktur Data Algoritma

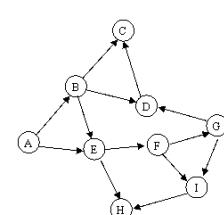
Prepared by: Suryana Setiawan (Des. 2017)

Deskripsi 1 (HASH TABLE)

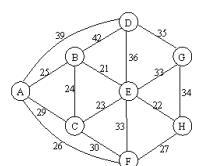
Berikut ini berurut-urut sejumlah data key dimasukan ke dalam tabel hash yang semula kosong:
REZA, AKHMAD, JUDHA, KARIM, RAHMAT,
WIDODO, ANWAR, JAMES, JOJO, PENDI
Tabel adalah berukuran $T = 13$.
Fungsi hash $H(x)$ menunjukkan ordinal dari 3 karakter pertama key, i.e., jika $x = "x_1x_2x_3\dots"$ maka $H(x) = (\text{ord}(x_1)+\text{ord}(x_2)+\text{ord}(x_3)) \% T$

- kolisi:** situasi ketika suatu data X akan ditempatkan pada posisi p dalam tabel tetapi ternyata pada posisi tersebut sudah ditempatkan data lainnya.
- probing:** adalah mencari posisi lain saat terjadi kolisi.
- jumlah probing:** berapa kali mencari posisi berikutnya setelah terjadi kolisi; jika tidak terjadi kolisi berarti jumlah probing = 0.
- Ordinal:** $\text{ord}(A) = 1$, $\text{ord}(B)=2$, $\text{ord}(Z) = 26$.

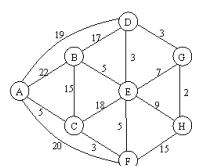
Deskripsi 2 (DIRECTED GRAPH)



Deskripsi 3 (WEIGHTED GRAPH A)



Deskripsi 4 (WEIGHTED GRAPH B)



1. (Soal ini mengacu pada **Deskripsi 1 (HASH TABLE)** di atas) Jika kolisi terjadi maka dilakukan kuadratik probing $(H(x)+i^2) \% 13$ dengan i jumlah probing setelah terjadi kolisi. Maka berapa kaliakah data JAMES mengalami probing?

- A. 0 B. 2 C. 1 D. 4 E. 3

2. (Soal ini mengacu pada **Deskripsi 4 (WEIGHTED GRAPH B)** di atas) verteks yang tidak dilakui oleh lintasan terpendek antara A dan G adalah?

- A. B B. C C. E D. F E. D

3. Jika tinggi suatu suatu AVL tree adalah 7 maka jumlah leaf yang mungkin dari AVL tree tersebut

- A. 64 s.d. 128 B. 32 s.d. 64 C. 21 s.d. 128

- D. 21 s.d. 64 E. 34 s.d. 128

A. D

B. H

C. F

D. I

E. G

13. Penghapusan node 100 pada AVL tree berikut menyebabkan



- A. Tidak terjadi rotasi
B. SRR
C. DLR
D. DRR
E. SLR

14. Jika dikatakan suatu AVL tree memiliki ketinggian 5 maka maka jumlah node dari avl tree tersebut berkisar antara:

- A. 32 s.d. 63 B. 54 s.d. 63 C. 20 s.d. 53

- D. 32 s.d. 53

- E. 20 s.d. 63

15. (Soal ini mengacu pada **Deskripsi 3 (WEIGHTED GRAPH A)** di atas) Jika algoritma Kruskal dijalankan untuk mendapatkan minimum spanning tree dari graph tersebut, urutan sisi yang ditemukan membentuk MTS berturut-turut adalah sbb:

- A. BE, EH, EC, AB, AF, EG, DG
B. AB, EH, CE, AF, EG, DG
C. BE, EH, EC, BC, AB, FG

- D. AB, AF, BE, CE, EH, EG, DG
E. BE, EH, EF, AB, AF, EG, DG

16. (Soal ini mengacu pada **Deskripsi 1 (HASH TABLE)** di atas) Jika kolisi terjadi maka dilakukan linear probing $(H(x)+i) \% 13$ dengan i jumlah probing setelah terjadi kolisi untuk data tersebut. Maka pada index berapakah data PENDI ditempatkan?

- A. 5 B. 3 C. 0 D. 9 E. 12

17. (Soal ini mengacu pada **Deskripsi 2 (DIRECTED GRAPH)** di atas) Berdasarkan graph tersebut, urutan verteks yang tidak mungkin merupakan hasil dari traversal menurut **DFS** dari A adalah

- A. A - B - C - D - E - F - I - G - H
B. A - E - F - I - G - D - H - B - C
C. A - E - H - F - G - D - C - I - B

- D. A - E - F - G - D - C - I - H - B
E. A - B - D - C - E - H - F - G - I

18. Jika dikatakan suatu AVL tree memiliki node sebanyak 53 maka maka tinggi dari avl tree tersebut yang mungkin adalah

- A. 6 B. 3 C. 4 D. 8 E. 10

19. (Soal ini mengacu pada **Deskripsi 4 (WEIGHTED GRAPH B)** di atas) Mengikuti algoritma Dijkstra dengan verteks asal A, jarak terpendek ke B mengalami update salah satunya ketika:

- A. saat D masuk ke dalam "white cloud"
B. saat H masuk ke dalam "white cloud"
C. saat G masuk ke dalam "white cloud"

- D. saat F masuk ke dalam "white cloud"
E. saat E masuk ke dalam "white cloud"

20. Penghapusan node 85 pada AVL tree berikut menyebabkan

4. Struktur data yang berpotensi lebih mengefisienkan algoritma Dijkstra dengan implementasi adjacency list pada bagian penentuan verteks berikut untuk bergabung dalam "white cloud" (yaitu yang terpendek jaraknya) adalah:

- A. priority queue
B. circular linked list
C. queue

- D. stack
E. AVL tree

5. (Soal ini mengacu pada **Deskripsi 2 (DIRECTED GRAPH)** di atas) Jika dilakukan topological sorting maka verteks urutan pertama adalah?

- A. D B. C C. A D. B E. E

6. (Soal ini mengacu pada **Deskripsi 3 (WEIGHTED GRAPH A)** di atas) Minimum spanning tree yang diperoleh dari graph di atas tidak termasuk sisi:

- A. antara A dan B
B. antara E dan G
C. antara C dan E

- D. antara E dan F
E. antara A dan F

7. (Soal ini mengacu pada **Deskripsi 1 (HASH TABLE)** di atas) Jika kolisi terjadi maka dilakukan double hashing $(H(x)+H_2(x,i)) \% 13$ dengan i jumlah probing setelah terjadi kolisi dan $H_2(x,i) = (i^{\text{ordinal huruf ke } 4 \text{ dari } x}) \% 13$. Maka salah satu posisi dalam tabel yang kosong adalah?

- A. 10 B. 8 C. 5 D. 7 E. 0

8. Jika dikatakan suatu AVL tree memiliki node sebanyak 88 maka kemungkinan tinggi dari avl tree tersebut berkisar antara:

- A. 5 s.d. 7 B. 7 s.d. 9 C. 6 s.d. 8 D. 8 s.d. 10
E. 4 s.d. 10

9. (Soal ini mengacu pada **Deskripsi 1 (HASH TABLE)** di atas) Jika kolisi terjadi maka dilakukan double hashing $(H(x)+H_2(x,i)) \% 13$ dengan i jumlah probing setelah terjadi kolisi dan $H_2(x,i) = (i^{\text{ordinal huruf ke } 4 \text{ dari } x}) \% 13$. Maka berapa kali kah data JOJO mengalami probing?

- A. 1 B. 3 C. 4 D. 2 E. 0

10. Penghapusan node 45 pada AVL tree berikut menyebabkan:



- A. DLR
B. Tidak terjadi rotasi
C. SLR
D. DRR
E. SRR

11. (Soal ini mengacu pada **Deskripsi 4 (WEIGHTED GRAPH B)** di atas) Jika algoritma Dijkstra dijalankan pada graph tersebut dengan verteks asal adalah A maka urutan verteks yang berurut-urut masuk ke dalam "white cloud" (verteks jarak terpendeknya sudah dipastikan) adalah:

- A. A, C, F, E, B, D, G, H
B. A, C, B, F, E, D, H, G
C. A, B, C, D, E, F, G, H

- D. A, C, F, E, D, B, H, G
E. A, C, F, E, D, B, G, H

12. (Soal ini mengacu pada **Deskripsi 2 (DIRECTED GRAPH)** di atas) Jika dilakukan topological sorting maka verteks urutan kelima adalah?

- A. DRR
B. Tidak terjadi rotasi
C. SRR
D. DRR
E. SLR



- A. DRR
B. Tidak terjadi rotasi
C. SRR
D. DLR
E. SLR

21. (Soal ini mengacu pada **Deskripsi 3 (WEIGHTED GRAPH A)** di atas) Panjang total semua sisi dalam minimum spanning tree dari graph tersebut adalah?

- A. 185 B. 155 C. 192 D. 188 E. 161

22. (Soal ini mengacu pada **Deskripsi 4 (WEIGHTED GRAPH B)** di atas) Berapakah panjang lintasan terpendek antara A dan H?

- A. 21 B. 25 C. 15 D. 23 E. 30

23. (Soal ini mengacu pada **Deskripsi 3 (WEIGHTED GRAPH A)** di atas) Jika algoritma Prim dijalankan untuk mendapatkan minimum spanning tree dari graph tersebut dan pencarian dimulai dari verteks A, urutan sisi yang ditemukan membentuk MTS berturut-turut adalah sbb:

- A. AB, BC, CE, FD, DG, GH, FH
B. BE, EH, EC, AB, AF, EG, DG
C. AB, BE, EH, CE, AF, EG, DG

- D. AB, AF, BE, CE, EH, EG, AF, DG
E. AB, BE, CE, EH, EG, AF, DG

24. (Soal ini mengacu pada **Deskripsi 2 (DIRECTED GRAPH)** di atas) Berdasarkan graph tersebut, urutan verteks yang tidak mungkin merupakan hasil dari traversal menurut **BFS** dari B adalah

- A. B - E - C - D - G - F - I
B. B - C - E - D - F - H - I - G
C. B - D - E - C - F - H - I - B

- D. B - E - C - D - F - H - I - G
E. B - D - C - E - H - F - G - I

25. (Soal ini mengacu pada **Deskripsi 2 (DIRECTED GRAPH)** di atas) Berdasarkan graph tersebut, urutan verteks yang tidak mungkin merupakan hasil dari traversal menurut **BFS** dari A adalah

- A. A - E - B - H - F - C - D - G - I
B. A - E - B - F - H - C - D - I - G
C. A - B - E - C - D - F - H - I - G

- D. A - B - E - D - C - F - H - G - I
E. A - E - B - C - D - F - H - I - G

26. (Soal ini mengacu pada **Deskripsi 2 (DIRECTED GRAPH)** di atas) Jika graph tersebut dilakukan BFS mulai dari B, manakah verteks yang tidak didatangi?

- A. C B. D C. C D. E E. A

27. Dalam hashing, metoda yang paling cenderung menimbulkan primary clustering adalah?

- A. open hashing
B. close hashing dengan quadratic probing
C. close hashing dengan linear probing
D. separate chaining
E. close hashing dengan double hashing

28. (Soal ini mengacu pada **Deskripsi 1 (HASH TABLE)** di atas) Jika kolisi terjadi maka dilakukan kuadratik probing $(H(x)+i^2) \% 13$ dengan i jumlah probing setelah terjadi kolisi. Maka pada indeks berapakah PENDI ditempatkan?

- A. 12 B. 0 C. 9 D. 8 E. 7

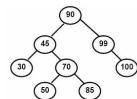
D. 5

E. 3

29. (Soal ini mengacu pada Deskripsi 2 (DIRECTED GRAPH) di atas) Berdasarkan graph tersebut, urutan verteks yang tidak mungkin merupakan hasil dari traversal menurut **DFS** dari E adalah
- E - F - G - D - C - I - H
 - E - F - H - G - D - C - I
 - E - F - I - G - D - C - H
 - E - H - F - I - G - D - C
 - E - H - F - G - D - C - I

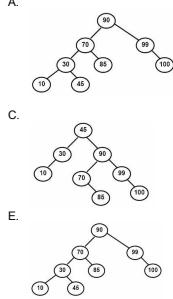
30. (Soal ini mengacu pada Deskripsi 1 (HASH TABLE) di atas) Jika kolisi terjadi maka dilakukan double hashing $(H(x)+H_2(x,i)) \% 13$ dengan jumlah probing setelah terjadi kolisi dan $H_2(x,i) = (l^i \text{ordinal huruf ke } 4 \text{ dari } x) \% 13$. Maka pada indeks berapakah data PENDI dilewatkan?
- 8
 - 12
 - 11
 - 9
 - 10

31. Penghapusan node 100 pada AVL tree berikut menyebabkan
- DRR
 - Tidak terjadi rotasi
 - DLR
 - SLR
 - SRR



32. (Soal ini mengacu pada Deskripsi 1 (HASH TABLE) di atas) Jika kolisi terjadi maka dilakukan kuadratik probing $(H(x)+H_2(x,i))^2 \% 13$ dengan jumlah probing setelah terjadi kolisi. Maka salah satu posisi dalam tabel yang kosong adalah?
- 7
 - 0
 - 5
 - 8
 - 10

33. Jika dilakukan penyisipan dalam AVL Tree yang kosong dengan urutan data: 30, 70, 45, 10, 90, 99, 85, 100. AVL Tree yang terbentuk berdasarkan algoritme yang dijelaskan di kelas adalah:



34. (Soal ini mengacu pada Deskripsi 1 (HASH TABLE) di atas) Jika kolisi terjadi maka dilakukan linear probing $(H(x)+l)\%13$ dengan jumlah probing setelah terjadi kolisi untuk data tersebut. Maka berapa banyak probing yang terjadi pada data JOJO?

- 3
- 0
- 2
- 7

E. 6

35. (Soal ini mengacu pada Deskripsi 2 (DIRECTED GRAPH) di atas) Jika dilakukan topological sorting maka verteks urutan yang tidak dapat terjadi adalah?

- A - B - E - F - G - D - C - I - H
- A - B - E - F - G - I - H - D - C
- A - B - E - F - G - I - C - D - H
- D - A - B - E - F - G - D - I - H - C
- E - A - B - E - F - G - D - I - C - H

36. Perhatikan AVL Tree berikut ini. Operasi mana yang paling tepat jika terjadi penyisipan data 78 ke dalam AVL Tree di bawah ini?

- rotasi ke kanan pada node 86 (SRR)
- rotasi ke kiri pada node 75 (SLR)
- rotasi ke kanan pada node 86, lalu rotasi ke kiri pada node 55 (DLR)
- rotasi ke kiri pada node 55 (SLR)
- rotasi ke kiri pada node 75, lalu rotasi ke kanan pada node 86 (DRR)



37. Di antara kondisi graph berikut ini manakah yang paling menjadikan masalah untuk algoritma BFS?

- | | |
|--|---|
| A. dense graph dan jumlah verteks besar sekali | C. sparse graph dan jumlah verteks besar sekali |
| B. dense graph dan jumlah sisi besar sekali | D. sparse graph dan jumlah sisi besar sekali |
| E. unweighted graph | |

38. (Soal ini mengacu pada Deskripsi 4 (WEIGHTED GRAPH B) di atas) Jarak lintasan terpendek dari A ke B adalah?

- 20
- 18
- 22
- 28
- 19

39. (Soal ini mengacu pada Deskripsi 1 (HASH TABLE) di atas) Jika kolisi terjadi maka dilakukan linear probing $(H(x)+l)\%13$ dengan jumlah probing setelah terjadi kolisi. Maka salah satu posisi dalam tabel yang kosong adalah?

- 0
- 7
- 10
- 6
- 3

40. Dalam hashing, metoda yang paling cenderung menimbulkan secondary clustering adalah?

- close hashing dengan quadratic probing
- close hashing dengan double hashing
- close hashing dengan linear probing
- open hashing
- separate chaining

41. (Soal ini mengacu pada Deskripsi 4 (WEIGHTED GRAPH B) di atas) verteks yang dilakukan lintasan terpendek antara A dan E adalah?

- D
- G
- B
- F
- H

42. (Soal ini mengacu pada Deskripsi 1 (HASH TABLE) di atas) Jika kolisi terjadi maka dilakukan double hashing $(H(x)+H_2(x,i)) \% 13$ dengan jumlah probing setelah terjadi kolisi dan $H_2(x,i) = (l^i \text{ordinal huruf ke } 4 \text{ dari } x) \% 13$. Maka pada indeks berapakah data WIDODO dilewatkan?

- 11
- 10
- 12
- 12
- 9

Soal Isian Singkat SDA 2017

Prepared by: Suryana Setiawan

Petunjuk: Jawablah sesingkat-singkatnya (tanpa penjelasan).

AVL

- Masukkan ke dalam AVL-tree nilai-nilai berikut ini: 21,3,34,23,56,45,18. Keluarannya jika dicetak secara *level order* adalah
- Masukkan ke dalam AVL-tree nilai-nilai berikut ini: 21,3,34,23,56,45,18. Kemudian, rotasi apa yang terjadi jika 45 dihapus?
- Jika kompleksitas waktu *insert* pada AVL-tree adalah $O(\log N)$, nama kompleksitas waktu proses *balancing*-nya adalah Penercian *shortest path* pada graf dengan bobot yang sama pada setiap edge-nya dapat menggunakan algoritma Dijkstra. Benar/Salah?
- Masukkan ke dalam AVL-tree nilai-nilai berikut ini: 21,3,34,23,56,45,18. Berapa banyak perbandingan yang terjadi ketika menghapus 45?
- Terdapat 4 operasi pada AVL-tree yang mungkin terjadi ketika proses *balancing*, yaitu
- Berapa banyak *node minimum* yang bisa terdapat pada AVL-tree dengan tinggi 2?
- Berapa banyak *node minimum* yang bisa terdapat pada AVL-tree dengan tinggi 10?
- Masukkan ke dalam AVL-tree nilai-nilai berikut ini: 21,3,34,23,56,45,18. Keluarannya jika dicetak secara *inorder* adalah....

B+Tree

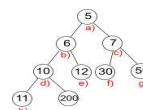
- Apakah terjadi jika banyak data dalam sebuah *node* pada B+ tree melebihi setengah dari kapasitasnya?
- Yang membedakan B-tree dengan B+ tree dalam hal penyimpanan data adalah
- Masukkan ke dalam B+ tree nilai-nilai berikut dengan *degree* 3: 21,3,34,23,56,45,18. Berapa tinggi B+ tree yang terjadi?
- Jika sebuah B+ tree memiliki *branching factor* 100 dan data yang terdapat pada B+ tree tersebut ada 500000, berapa tinggi maksimum dari B+ tree tersebut?
- Aplikasi struktur data B+ tree biasa ditemukan pada
- Jika ukuran satu blok data adalah 5 KB dan satu blok pointer adalah 1 KB, maka *spacemaksimal* yang diperlukan untuk menyimpan 50000 data pada B+ tree dengan *degree* 10 adalah ... KB.
- Masukkan ke dalam B+ tree nilai-nilai berikut dengan *degree* 3: 21,3,34,23,56,45,18. Ketika memasukkan data di atas, pada *insert* nilai apa terjadi *due key split* pada penyisipan tsb.
- Jika *branching factor* dari B+ tree adalah 500 dan tinggi dari B+ tree adalah 20. Seburuk-buruknya, berapa perbandingan data yang akan terjadi pada B+ tree tersebut?

HEAP

- Tentukan kompleksitas dari operasi melakukan operasi *heapify* pada array acak berisi n elemen.

- Tentukan kompleksitas dari operasi memasukkan n elemen pada binary heap.

- Diberikan sebuah *minheap* seperti pada Gambar di bawah dengan key-key yang unik. Jika dilakukan operasi *deleteMin*, selanjutnya node yang berisi key 200 akan dipindahkan pada posisi yang ditunjuk oleh huruf berapakah?



- Ketika elemen x dimasukkan ke minheap pada Soal sebelumnya, kemudian berakibat key 12 akan diletakkan pada posisi j . Jika x adalah sebuah bilangan bulat, berapakah rentang nilai x ?

- Banyak *edge* yang arahnya masuk ke suatu *vertex* disebut sebuah binary heap disimpan dalam array seperti yang diberikan di bawah ini. Jika dimasukkan elemen bernilai 4, tuliskan indeks berapa sajakah yang diperlukan untuk menentukan posisi yang tepat bagi nilai 4!

1	3	5	18	25	36	7	25	100
---	---	---	----	----	----	---	----	-----

Hash Table

- Jika sebuah hashtable mempunyai λ (load factor) lebih dari 1, maka teknik untuk mengatasi *collision* yang memungkinkan adalah:

- Jika M adalah ukuran hashtable dan λ adalah load factor hashtable tersebut, hitunglah jumlah posisi array yang kosong!

- Diberikan sebuah fungsi *hash* sebagai berikut:

```
public static int hash(String key, int tableSize){  
    return ((key.charAt(0) - '0') + (key.charAt(2) - '0')) % tableSize;  
}
```

Diajukan bahwa *key* mempunyai panjang paling sedikit 4 digit. Jika kondisi *hashtable* saat ini digambarkan oleh tabel di samping ini. Pada indeks berapakah *key* 5907 dimasukkan menggunakan *quadratic probing*?

0	3121
1	4321
2	
3	
4	

- Gاءcu pada soal sebelumnya. Pada indeks berapakah *key* 1829 dimasukkan menggunakan *linear probing*?

- Gاءcu pada soal sebelumnya. Pada indeks berapakah *key* 3626 dimasukkan menggunakan *linear probing*?

Graph: Travesal (BFS/DFS)

- Strategi menjelajahi graf di mana semua *vertex* yang mempunyai jarak yang sama dengan *vertex* asal yang ditelusuri terlebih dahulu disebut
- Kompleksitas waktu dalam notasi Theta untuk algoritma DFS jika menggunakan *adjacency list* sebagai representasi grafnya adalah (M = banyak *edge*, N = banyak *vertex*)
- Jika sebuah graf berbentuk ..., maka *shortest path* antara dua buah *vertex* dapat dicari dengan BFS/DFS.
- Struktur data yang mendukung algoritma DFS adalah
- Penggunaan struktur data stack pada program untuk implementasi DFS bisa diganti dengan membuat fungsi DFS menjadi bersifat
- Struktur data yang mendukung algoritma BFS adalah
- Strategi menjelajahi graf di mana *vertex* ditelusuri terus sampai tidak punya tetangga lagi baru kemudian *backtrack* disebut
- Sebuah graf berarah yang tidak memiliki *cycles* di dalamnya biasa disebut
- Kompleksitas waktu dalam notasi Theta untuk algoritma BFS jika menggunakan *adjacent matrix* adalah (M = banyak *edge*, N = banyak *vertex*)

Graph: Sortest Path

- Kompleksitas waktu terbaik dalam notasi big-Oh dari algoritma Dijkstra dengan *priorit queue* adalah (M = banyak *edge*, N = banyak *vertex*)
- BFS dapat digunakan untuk mencari *shortest path* pada graf khusus di mana setiap *edge*-nya tidak berbobot atau bobotnya sama? [Pilihan jawaban: Benar/Salah]

Graph:MST

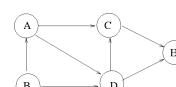
- MST merupakan singkatan dari
- Banyak *edge* yang arahnya keluar dari suatu *vertex* disebut
- Algoritma untuk mencari MST hanya dapat dijalankan jika dan hanya jika graf tersebut merupakan graf asiklik, Benar/Salah?
- Jika sebuah graf merupakan ..., maka MST dari graf itu adalah graf itu sendiri.
- Implementasi algoritma Kruskal membutuhkan kompleksitas waktu $O(M \cdot \log N)$, Benar/Salah? (M = banyak *edge*, N = banyak *vertex*)
- Implementasi algoritma Prim dengan representasi graf dalam bentuk *adjacency matrix* membutuhkan kompleksitas waktu dalam notasi big-Oh (M = banyak *edge*, N = banyak *vertex*)
- Implementasi algoritma Kruskal membutuhkan kompleksitas waktu $O(M \cdot \log M)$, Benar/Salah? (M = banyak *edge*, N = banyak *vertex*)
- Implementasi algoritma Prim dengan representasi graf dalam bentuk *adjacency list* dan penggunaan *binary heap* membutuhkan kompleksitas waktu dalam notasi big-Oh (M = banyak *edge*, N = banyak *vertex*)

- Jika *edge-edge* dalam sebuah graf telah terurut berdasarkan bobotnya, maka algoritma Kruskal dapat berjalan dengan kompleksitas waktu yang lebih baik dibandingkan dengan jika *edge-edge*-nya tidak terurut berdasarkan bobotnya, Benar/Salah?

- Representasi graf yang dapat mencapai kompleksitas waktu yang paling baik pada algoritma Kruskal adalah dalam bentuk

Graph: Topological Sort

- Jika dilakukan *topological sort*, *vertex* pertama yang dipilih adalah



- Dari soal *Topological sort* sebelumnya, *vertex* terakhir yang dipilih adalah

- Dari soal *Topological sort* sebelumnya, *vertex* kedua yang dipilih adalah

- Setiap Directed Acyclic Graph (DAG) mempunyai minimal ... *topological sort*.

- Cara menyelesaikan *topological sort* adalah dengan mencari *vertex* yang mempunyai *in-degree* bernilai

- Jumlah *in-degree* suatu graf pasti dari jumlah *out-degree*-nya. [pilihan jawaban: lebih besar atau lebih kecil atau sama dengan]

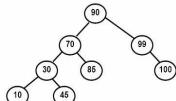
Ujian Akhir Semester Kuliah Struktur Data dan Algoritma
Fakultas Ilmu Komputer Universitas Indonesia
17 Desember 2016

Bagian A. Isian Singkat

Penjelasan umum: setiap pertanyaan berdiri sendiri (tidak ada soal yang tergantung atau menjadi turusan dari soal lainnya).

Topik AVL Tree

AVL Tree berikut menggunakan kaidah “memilih successor inorder” saat terjadi penghapusan suatu key pada internal node.



1. Jika key berharga 99 dihapus dari AVL Tree di atas, maka rotasi apakah (SRR, DRR, SLR, DLR, atau tidak terjadi) yang terjadi?
2. Jika dimasukkan ke dalam AVL Tree di atas key-key ini: 21,3,34,23, urutkan key-key yang berada pada leaf-leafnya dari terkecil hingga terkemau.
3. Berapa kedalaman (depth) leaf terbesar yang mungkin pada AVL tree dengan jumlah node 120?
4. Berapa banyaknya *node minimum* pada AVL Tree dengan ketinggian 10?
5. Suatu operasi penghapusan key AVL Tree dapat berakibat dilakukan lebih dari satu kali rebalancing (dengan asumsi double rotation dihitung satu kali rebalancing). Benar/salah kah itu?

Topik Binary Heap

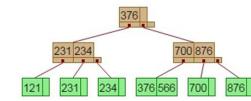
Isi suatu Array berikut mendefinisikan suatu binary heap (min-heap) dalam struktur complete binary tree.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Data	5	8	9	10	17	11	22	25	40	18	19	21	42	23	50	31	26	47		

6. Jika satu kali elemen terkecil **dambil** (dequeued) dari heap, lalu kemudian **dimasukkan kembali** (enqueue) maka berada pada posisi array (index) berapakah kemudian elemen berharga 10?
7. Jika **dua kali elemen** terkecil dequeued (dikeluarkan) dari heap, maka berada pada index berapakah kemudian elemen berharga 17 dalam array tsb?
8. Jika dimasukkan elemen baru bernilai 6, tuliskan indeks elemen berapa sajakah yang diperlukan (dibandingkan dengan 6) untuk menentukan posisi akhir yang tepat bagi elemen baru tersebut!
9. Jika Binary Heap digunakan untuk sorting dengan mengambil elemen-elemen pada root, maka apakah kompleksitas waktunya pengurutan ini?
10. Jika sejumlah data disisipkan pada binary heap (seperti di atas) yang **semula kosong**, kondisi data manakah yang menyebabkan waktu **kompotasi paling cepat**: (a) data acak; (b) data terurut menaik; (c) data terurut menurun; (d) sama saja ketiga kondisi tersebut; (e) tidak mungkin diperkirakan.

1

Topik B Tree / B+ Tree



Gambar 1. B+Tree dengan degree 3 (kondisi awal).

11. Tambahkan ke dalam B+Tree di atas 5 buah data berikutnya: 100, 213, 343, 564, dan 518 Berapa tinggi tree akhir yang dihasilkan?
12. Jika ada 700 dihapuskan pada B+Tree pada di atas, maka manakah yang akan terjadi: (a) merging node; (b) perpindahan key dari sibling node; atau (c) tidak terjadi apa-apa.
13. Jika 234 dan 121 berturut-turut dihapuskan pada B+Tree di atas, maka menjadi berapakah tinggi tree tersebut kemudian?
14. Pada suatu B+tree dengan degree 100, berapakah jumlah minimum data yang bisa dimasukkan ke dalamnya jika diketahui tingginya 3? [Ingin: untuk degree 100, satu node leaf harus minimum 49, dan maksimum 99 data aktual.]
15. Jika suatu blok disk didefinisikan sebagai suatu node B+tree dengan degree 200, berapa jumlah blok maksimum yang perlu dibaca dari disk jika ukuran tabel data adalah 10^3 record, sebelum pembacaan blok yang berisi record data yang bersangkutan.

Topik Tabel hash

Suatu tabel hash berukuran 13 terdefinisi dengan fungsi hash $H(X)$ sebagai berikut. $H(X)$ adalah menjumlahkan ordinal dari 3 karakter pertama data, yaitu untuk key $X = x_1x_2x_3\dots$, maka

$$H(X) = (\text{ord}(x_1)+\text{ord}(x_2)+\text{ord}(x_3)) \% T, \text{ dengan } \text{ord}(A)=1, \text{ ord}(B)=2, \dots, \text{ ord}(Z)=26.$$

Berurut-turut sejumlah data dimasukkan ke dalam tabel hash yang semula kosong:

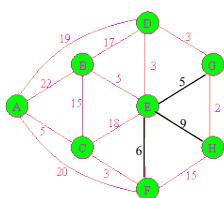
REZA, AKHMAD, JUDHA, KARIM, RAHMAT, WIDODO, ANWAR, JAMES, JOJO, PENDI.

Catatan mengenai peristiwa:

- **kolisi:** situasi ketika suatu data X akan ditempatkan pada posisi p dalam tabel tetapi ternyata pada posisi tersebut sudah ditempatkan data lainnya.
- **probing:** adalah mencari posisi lain saat terjadi kolisi.
- **jumlah probing:** berapa kali mencari posisi berikutnya setelah terjadi kolisi; jika tidak terjadi kolisi berarti jumlah probing = 0.
- 16. Berdasarkan metoda **open hashing (chaining)**, pada posisi berapakah rantai terpanjang?
- 17. Berdasarkan metoda closed hashing dan kolisi ditangani dengan **linear probing**, maka berapa banyak probing yang terjadi untuk penyisipan data JOJO?
- 18. Berdasarkan metoda closed hashing dan kolisi ditangani dengan **kuadratik probing**, maka berapa banyak probing yang terjadi untuk penyisipan data PENDI (banyaknya probing menunjukkan berapa kali mencari posisi lain setelah terjadi kolisi)?
- 19. Berdasarkan metoda closed hashing dan kolisi ditangani **double hashing**, dan fungsi hashig kedua $H_2(X) = (\text{ord}(x_4) \% 12 + 1)$, dimana PENDI akan ditempatkan?
- 20. Berdasarkan metoda closed hashing dan kolisi ditangani dengan **linear probing**, maka setelah semua data masuk, berapa panjang primary clustering yang terbentuk?

2

Topik Graph



Gambar 3

21. **DFS** dilakukan mulai dari verteks A, kemudian B, E, dst. Verteks-verteks manakah yang mungkin menjadi yang terakhir dikunjungi?
22. **BFS** dilakukan mulai dari verteks C. Verteks-verteks manakah yang mungkin menjadi yang terakhir dikunjungi?
23. **Algoritma Kruskal** digunakan untuk mendapatkan MST. Tuliskan pada urutan berapa sisi EF teridentifikasi masuk dalam MST (yang membentuk siklik tidak termasuk)?
24. **Algoritma Prim** digunakan untuk mendapatkan MST. Jika algoritma dimulai dari A, pada urutan keberapa D dipertimbangkan dimasukkan dalam tree (catatan: A pada urutan 1)?
25. Dapatkan panjang **shortest path** (lintasan terpendek) antara A dan H, dan tuliskan deretan verteks-verteks dalam lintasannya.
26. **Algoritma Dijkstra** akan digunakan untuk mengetahui shortest path verteks A ke E. Verteks-verteks manakah yang shortest pathnya juga diketahui lebih dulu sebelum shortest path ke verteks E di ketahui?
27. Struktur data pendukung yaitu untuk menyimpan tabel D (untuk mendapatkan verteks berikutnya yang diketahui shortestpath-nya) demi mengetahui efisiensi algoritma Shortest Path adalah?
28. Jika tabel D (untuk mendapatkan verteks berikutnya yang diketahui shortestpath-nya) diimplementasikan sebagai array linear biasa, dan graph direpresentasikan dengan **adjacency matrix** maka kompleksitas waktu algoritma shortest path Dijstra adalah (diketahui N adalah jumlah verteks dan M adalah jumlah edge)?
29. Struktur data apakah untuk topological ordering yang paling efisien dan berapa kompleksitas waktunya jika N jumlah verteks adalah M jumlah edge.
30. Dalam algoritma Topological sorting, kesimpulan apakah yang didapat ketika tidak ada lagi verteks dengan in-degree 0 sementara masih ada verteks yang belum diurutkan?

Bagian B. Short Coding

Penjelasan umum: Dalam menjawabnya, sebaiknya anda menuliskan ide algoritma terlebih dulu serta penggunaan variabel-variabel agar isi algoritm dapat dibaca dengan lebih baik saat diperiksa. Kecuali, jika anda anggap algoritma anda sudah cukup jelas.

Suatu **unweighted, directed** graph direpresentasikan dengan adjacency-matrix bernama mGraph. Dengan pernyataan dalam Bahasa Java , mGraph dideklarasikan dalam baris-baris berikut.

```
int nV = ... ; //jumlah verteks
boolean myGraph[][][] = new boolean[nV][nV]; //graph
```

31. [10 point] Pada Graph tersebut akan dilakukan topological sorting, tuliskan private method bernama **getAllInDegree()** untuk menghitung dan mencetak harga indegree setiap verteks dan me-return ke suatu array integer indegree (misalkan dipanggil dengan:

```
int indegree[] = getAllInDegree();
```

Tuliskan isi method **getAllInDegree()** ini.
32. [15 point] Dengan graph yang isimpang dalam myGraph serta array pendukung indegree[] di atas, buatlah method **topologicalSorting()** untuk mencetak verteks-verteks yang terurut secara topologis (dengan topological algoritma sorting).
33. [Bonus 10 point] Graph tersebut hendak diperiksa apakah semua verteks terhubungkan ke dalam satu connected component dengan suatu method bernama **isConnected** yang akan mengembalikan harga boolean true jika seluruh verteks terhubungkan, dan false jika tidak. [Hint: anda boleh menggunakan BFS atau DFS untuk mengimplementasikan method ini, sehingga anda juga perlu menambahkan array boolean **visited[nV]** untuk mencatat sudah didatangi atau belum oleh algoritma.

(Bagian B di halaman berikutnya)

3

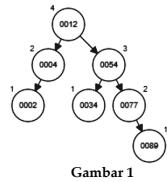
4

BAGIAN A (Bobot 24x2.5=60)

Petunjuk: Isilah pada lembar jawaban yang disediakan di bagian belakang berkas soal ini. Dalam menjawab setiap pertanyaan, ingat bahwa setiap pertanyaan tidak saling berkaitan (tidak meneruskan pertanyaan sebelumnya, melainkan selalu mengacu bentuk awalnya di Gambar terkait!).

Binary tree di **Gambar 1** di samping ini dipandang sebagai suatu BST, maka:

1. Data 0000 dan 0001 hendak disisipkan pada BST di **Gambar 1**, menjadi berapakah tinggi tree setelah itu? [Def: tinggi tree adalah jumlah sisi terpanjang yang ada pada tree]
2. Penghapusan 0012 akan dilakukan (mengikuti kaidah *successor inorder*) pada BST di **Gambar 1**, data apakah yang menempati root setelah itu?
3. Penghapusan 0054 akan dilakukan (mengikuti kaidah *predecessor inorder*) pada BST di **Gambar 1**, apakah yang menjadi anak kanan dari 0012?



Binary tree di **Gambar 1** di atas dipandang sebagai AVL tree. Di bagian ini anda akan ditanyakan operasi apa ygngn terjadi. Jawaban anda hanya berupa nama rotasi (SKRR=kasus 1; DDR=kasus 2; DLR=kasus 3; SLR=kasus 4; atau "Tidak ada rotasi" jika memang tidak terjadi rotasi).

4. Apa yang terjadi saat penyisipan data 0003 ke dalam AVL Tree di **Gambar 1**?
5. Apa yang terjadi saat penyisipan data 0035 ke dalam AVL Tree di **Gambar 1**?
6. Apa yang terjadi saat penghapusan data 0002 pada AVL Tree di **Gambar 1**?
7. Jika 100.000 data unik disusun membentuk suatu AVL tree, berapa range tinggi yang mungkin terjadi (minimum, maksimum).

Suatu B+Tree dengan **Max Degree 4** sudah berisi sejumlah data sebagai berikut. Setiap pertanyaan memerlukan jawaban salah satu saja: split; merge; berpindah; atau struktur tetap.



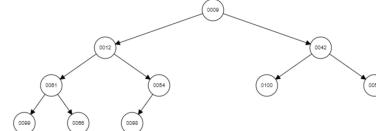
8. Apa yang terjadi pada B+Tree di atas saat disisipkan data 24?
9. Apa yang terjadi pada B+Tree di atas saat disisipkan data 12?

2

10. Apa yang terjadi pada B+Tree di atas saat data 14 dihapus?
11. Apa saja yang terjadi pada B+Tree di atas saat data 21 dan kemudian 22 dihapus? (bisa lebih dari satu kejadian)

12. Pada suatu B+tree dengan max degree 200 dan tinggi 4, berapa minimum banyaknya *actual data key* yang mungkin tersimpan di dalamnya? Ingat bahwa root satu-satunya node yang dapat memiliki dua cabang saja.
13. Pada suatu B+tree dengan max degree 200 dan tinggi 4, berapa maksimum banyaknya *actual data key* yang mungkin tersimpan di dalamnya? Ingat bahwa root satu-satunya node yang dapat memiliki dua cabang saja.

Berikut ini sejumlah data sudah berada dalam heaptree dan untuk menyimpannya digunakan suatu array DATA berindeks dari 0, 1, 2, 3, dst. Jadi 0009 berada pada DATA[0], 0012 berada pada DATA[1], 0042 berada pada DATA[2], dst.



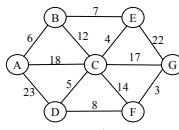
Gambar 3

14. Data 10 akan disisipkan dalam heaptree, pada indeks berapakah dalam array DATA data 10 itu kemudian berada?

15. Dua data terceluk berturut-turut hendak dihapuskan dari heaptree, tuliskan isi array DATA mulai dari indeks 0, 1, 2 ... dst setelah penghapusan itu dilakukan.

Gambar 4 berikut ini berisi weighted graph dengan weight merupakan data jarak.

16. Algoritma BFS hendak dijalankan dimulai dari A dengan tambahan aturan: *verteks dengan weight lebih kecil yang dipush ke queue lebih dulu* maka bagaimanakah urutan verteks yang dikunjungi algoritma?

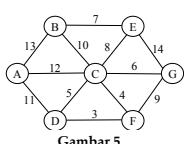


17. jika algoritma pencarian Shortest Path Dijkstra dijalankan pada Graph di **Gambar 4** dimulai dari verteks E, dapatkan urutan verteks-verteks berikutnya yang diketahui shortest pathnya oleh algoritma tsb.

3

18. Mengacu pada **Gambar 4** jika pencarian shortest path dimulai dari D ke B, pada interasi ke berapa B akan diketahui shortest pathnya (catatan: D yang pertama)?

Gambar 5 disamping ini berisi wighted graph dengan weight merupakan data jarak.



19. algoritma Kruskal digunakan untuk pencarian MST pada grah di **Gambar 5**, sisi mana yang pertama kali ditolak karena diketahui membentuk siklik. [Tulis sebagai (X,Y)].

20. Mengacu pada **Gambar 5** algoritma Prim digunakan untuk pencarian MST dimulai dari verteks F, sebutkan urutan masuknya verteks-verteks ke dalam MST oleh algoritma Prim ini.

21. Topological sorting hendak dijalankan pada graph berikut ini. Graph merepresentasikan keterkaitan *prerequisite* (sis-i sis-j berarti) antara task-task (sebagai verteks-verteks) suatu kegiatan. Tuliskan urutannya, jika ada beberapa task yang dapat dijalankan bersamaan (urutannya tidak penting), tandai task-task itu dengan tanda kurung.

22. Jika graph memiliki in/out-degree average adalah suatu konstanta D dan jumlah verteks adalah N berapa kompleksitas algoritma topological sorting ini (dinyatakan dengan variabel D dan N)?

Beberapa soal mengenai hashing berikutnya mengacu pada tabel disamping ini. Tabel ini menunjukkan berurut-urut duabelas data (pada kolom kedua) dimasukan ke dalam tabel hash yang berukuran 17 dengan tabel semula adalah kosong.

Urutan	X	H(x)	H2(x)
1	REZA	110	15
2	AKHMAD	337	6
3	JUDHA	294	8
4	KARIM	244	9
5	RAHMAT	171	3
6	WIDODO	105	10
7	ANWAR	112	2
8	JAMES	131	1
9	JOJO	129	14
10	PENDI	89	3
11	PARTO	320	5
12	JOKO	150	16

Suatu fungsi hash H(x) digunakan untuk menempatkan data dalam tabel hash tersebut yang harganya sebagai pada kolom ketiga. Kolom keempat berisi Fungsi hash kedua H2(x) yang akan digunakan untuk soal double hashing.

23. Jika *collision* ditangani dengan *linear probing* $(H(x)+I)\%17$ dengan I jumlah probing setelah terjadi kolisi. Tuliskan penempatan akhir dari setiap data pada tabel (mengikuti urutan data masuk di atas) format (data, posisi) dan dipisahkan spasi. Misalnya: (REZA, 10), (AKHMAD, 4), (JUDHA, 9), ... dan seterusnya.

24. Jika *collision* ditangani dengan *double hashing* $(H(x)+H2(x)*I)\% 17$ dengan I jumlah probing setelah terjadi kolisi dan $H2(x)$ sebagai pada kolom ketiga di tabel di atas. Tuliskan penempatan akhir dari setiap data pada tabel (mengikuti urutan data masuk di atas) dengan format seperti soal sebelumnya.

BAGIAN B (2x20=40)

Ada isian pada lembar jawaban halaman kedua yang bisa anda isi untuk menuliskan ide penyelesaian dari algoritma tersebut. Isian itu akan diperiksa manakala coding anda bernilai < 40 per soal. Dari ide tersebut akan dinilai apakah anda bisa mendapat nilai maksimum 40. Anda baru dapat melakukan coding dengan komputer setelah menyerahkan lembar jawaban anda. Oleh sebab itu anda boleh mencatat ide algoritma anda dikertas buram agar dapat diimplementasikan saat coding.

1. Suatu array X berisi N data bilangan integer acak. Tuliskan algoritma *fixheap* (atau disebut juga *heapsify*) dengan **method heapsify()** untuk menyusunnya menjadi **maxheap** secara $O(N \log N)$. Syarat: algoritma hanya bekerja pada array X, tanpa menggunakan array bantuan lainnya.

2. Suatu matrix boolean bernama *myGraph* berukuran $N \times N$ merepresentasikan suatu *undirected graph* dengan N verteks. Tuliskan algoritma untuk memeriksa berapa banyak *connected component* terdapat di dalam graph tersebut dengan nama **method connComp**
3. Suatu *connected component* adalah subgraph sebesar-besarnya yang setiap verteksnya masih saling berhubungan.

4

5

Bagian A: Isian

1. Terdapat dua jenis sifat yang harus dipenuhi dalam sebuah implementasi *binary heap*, yaitu **structural property** dan **order property**. Jelaskan secara singkat masing-masing sifat tersebut!

Structural property: complete binary tree, dapat direpresentasikan sebagai array

Order property: parent node lebih prioritas daripada anak-anaknya

2. Gambarkan sebuah **binary search tree** dengan kedalaman (tinggi atau *height*) 3 yang juga merupakan sebuah **binary heap**!

Anulir. Tidak ada solusinya.

3. Diberikan sebuah array yang merupakan **minimum binary heap** X berisi data acak sebagai berikut (indeks dimulai dari 0):

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]
1	2	99	10	5	100	200	77	50	21	8

Gambarkan isi dari array tersebut setelah terjadi dua kali pengambilan elemen terkecil!

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]	X[10]
5	8	99	10	21	100	200	77	50		

4. Diberikan sebuah **hashtable** yang bisa menampung 13 elemen (posisi 0 sd. 12). **Hashtable** ini menggunakan prinsip *double hashing*, dimana jika *k* adalah *key*, *i* menunjukkan sudah berapa *collision* yang terjadi, *m* adalah ukuran **hashtable**, dan fungsi lengkap untuk mencari posisi sebuah *key* di dalam **hashtable** tersebut adalah (*m* = ukuran tabel dan i urutan terjadi probing akibat collision pada *key* tersebut):

$$H(\text{key}, i) = (F(\text{key}) + i * h_2(\text{key})) \bmod m$$

$$h_2(\text{key}) = 5 - (F(\text{key}) \bmod 5)$$

Diketahui juga fungsi F(*key*) sudah diperoleh sebagai dalam tabel berikut:

Key	F(Key)	Key	F(Key)	Key	F(Key)
Apel	17	Durian	56	Salak	99

Page 2 of 13

Jambu	10	Jeruk	49	Markisa	17
Duku	49	Mangga	105	Nangka	25

Jika dimulai dari kosong, berturut-turut diisikan Apel, Jambu, Duku, Durian, dan Jeruk, sebutkanlah (posisi, key) dalam **hashtable** tersebut. Sesuai urutan yang terjadi dalam tabel (Misalnya "(4, Apel), (8, Durian),").

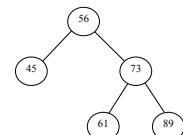
(4, Apel), (8, Durian), (10, Jambu), (11, Duku), (12, Jeruk)

5. Mengacu pada soal sebelumnya, berapa kali *collision* yang terjadi pada urutan *key* masuk Apel, Jambu, Duku, Durian, Jeruk, Mangga, Salak, Markisa, Nangka?

12

Soal-soal nomor 6 – 9 berikut ini mengacu pada deskripsi ini dan setiap soal (6 – 9) tidak mengacu pada soal lainnya.

Diberikan suatu **binary search tree** yang merupakan AVL Tree sebagai berikut:



Operasi penghapusan *key* akan mengikuti kaidah *predecessor inorder* (mencari maksimum di *subtree kiri*). Singkatan operasi rotasi: SRR (rotasi tunggal ke kanan), SLR (rotasi tunggal ke kiri), DRR (rotasi ganda ke kanan), DLR (rotasi ganda ke kiri), TT (tidak terjadi apa-apa).

6. Pada AVL tree awal di atas jika **disisipkan 20** maka akan terjadi operasi apakah?

TT

7. Pada AVL tree awal di atas jika **disisipkan 65** maka akan terjadi operasi apakah?

DRR

8. Pada AVL tree awal di atas jika **dihapuskan 56** maka akan terjadi operasi apakah?

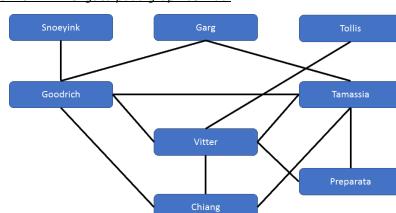
TT

9. Pada AVL tree awal di atas jika **dihapuskan 73** maka akan terjadi operasi apakah?

Page 3 of 13

TT

Soal-soal nomor 10 – 12 mengacu pada graph berikut:



10. Isilah tabel berikut yang mendeskripsikan beberapa atribut dari graph di atas!

Atribut	Jawaban
Graph Undirected/Directed? (pilih salah satu)	Undirected
Jumlah Vertex	8
Jumlah Edge	13
Degree Maksimum	5

11. Buatlah representasi dari graph di atas dalam bentuk **adjacency list** dengan melengkapi tabel berikut!

Indeks	Vertex	Adjacents (daftar edges yang berangkat dari vertex tsb.)
0	Chiang	Goodrich, Tamassia, Vitter
1	Garg	Goodrich, Tamassia
2	Goodrich	Chiang, Garg, Snoeyink, Tamassia, Vitter
3	Preparata	Tamassia, Vitter
4	Snoeyink	Goodrich

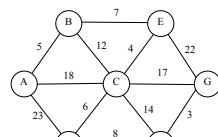
Page 4 of 13

5	Tamassia	Chiang, Garg, Goodrich, Preparata, Vitter
6	Tollis	Vitter
7	Vitter	Chiang, Goodrich, Preparata, Tamassia, Tollis

12. Buatlah representasi dari graph di atas dalam bentuk **adjacency matrix** dengan melengkapi matriks berikut! Gunakan simbol T untuk menandakan adanya *edge* antara 2 vertex atau simbol F untuk menandakan tidak adanya *edge* antara 2 vertex. (Note: pemetaan nomor indeks dengan vertex mengikuti tabel pada pertanyaan sebelumnya)

	0	1	2	3	4	5	6	7
0	F		T		T		T	
1		F	T			T		
2	T	T	F		T	T		T
3				F		T		T
4				T		F		
5	T	T	T	T		F		T
6							F	T
7	T		T	T		T	T	F

Soal-soal nomor 13 – 18 mengacu pada graph berikut:



- Bobot pada setiap sisi menunjukkan jarak antar kedua vertex terkait.

Page 5 of 13

NPM / Nama: _____ / KUNCI

- "White cloud" adalah vertex yang sudah diketahui dengan pasti *shortest path*-nya dari A menurut algoritma Dijkstra.
13. Jika MST dicari dengan algoritma Kruskal, dalam iterasinya pemeriksaan sisi manakah yang pertama kali diketahui membentuk siklik (*cycle*)?

Iterasi ke-7 pada edge BC

14. Jika MST dicari dengan algoritma Prim yang dimulai dari C, jika pada iterasi pertama sisi (*edge*) CE bergabung dalam MST, maka sisi DF akan bergabung pada iterasi ke berapakah?

Iterasi ke-2

15. Jika *shortest path* dari A ke setiap vertex dicari dengan algoritma Dijkstra dan pada iterasi pertama "white cloud" hanya berisi {A}, tuliskan isi dari "white cloud", pada iterasi ke empat!

A, B, C

16. Jika *shortest path* dari A ke setiap vertex dicari dengan algoritma Dijkstra, pada iterasi keberapakah F diketahui *shortest path*-nya? (Note: A diketahui pada iterasi pertama)

6

17. Jika *shortest path* dari A ke setiap vertex dicari dengan algoritma Dijkstra, dalam iterasi dimana "white cloud" sudah berisi {A,B} dan akan menentukan vertex berikutnya yang akan dipilih untuk bergabung ke dalam "white cloud" tersebut, sebutkan vertex-vertex dengan harga-harga jarak terpendek sementaranya (hanya yang berharga berhingga).

(E, 12), (C, 17), (D, 23)

18. Jika DFS dilakukan mulai dari A (tentu saja DFS tidak perlu memperhatikan weight!), jadi A adalah node pertama yang dikunjungi, kemudian diketahui pula bahwa C dan B adalah berturut-turut node kedua dan keempat yang dikunjungi, tuliskan urutan node-node itu dikunjungi selengkapnya.

A, C, E, B

19. Berapakah tinggi maksimum suatu **AVL Tree** yang berisi 10.000 (sepuluh ribu) key?

17

20. Berapa jumlah node minimum pada suatu **AVL Tree** dengan tinggi 12?

NPM / Nama: _____ / KUNCI

986

21. Berapakah tinggi maksimum suatu **B+Tree** yang berisi 10.000.000 (sepuluh juta) data key dengan *degree* maksimum = 200? (Note: Tinggi B+Tree dihitung dari root ke *leaf* node yang berisi pointer ke data)

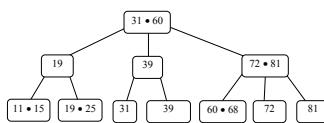
3

22. Jika suatu graph berisi node 1.000 (seribu) dan degree dari graph tidak lebih dari 10. Algoritma *shortest path* Dijkstra diimplementasikan tanpa *priority queue* pada graph dan ternyata diperlukan waktu pencarian *shortest path* keseluruhan adalah t detik. Ketika jumlah node dalam graph dinaikkan menjadi 1.000.000 (satu juta), berapakah waktu yang diperlukan dalam menjalankan hal yang sama tersebut?

23. Jika tiga buah array (A, B, dan C) akan disusun ke dalam *binary heap*-nya masing-masing mulai dari keadaan kosong, diketahui jika isi array A adalah acak, B turut menurun, dan C turut menaik. Data satu demi satu disisipkan ke dalam masing-masing binary heap ybs mulai dari data pertama hingga terakhir. Dari perkiraan waktu eksekusinya, tuliskan urutan array dari yang paling cepat hingga terakhir selesainya.

C, A, B

24. Perhatikan contoh **B+Tree** berikut:



Berapakah jumlah key minimum dan jumlah key maksimum pada setiap node B+Tree di atas?

[1, 2]

25. Mengacu pada **B+Tree** di nomor sebelumnya, berapa banyak operasi *split* yang terjadi apabila B+Tree di atas disisipkan key 70?

3 kali

NPM / Nama: _____ / KUNCI

Bagian B: Pemrograman Singkat

1. Perhatikan contoh kode berikut:

```

public class BinaryHeap {
    // Asumsikan books tidak kosong
    private List<Book> books;

    public void add(Book book) {
        books.add(book);
        percolateUp(books.size() - 1);
    }

    private void percolateUp(int pos) {
        // TODO Implement me!
    }
}

class Book implements Comparable<Book> {

    String title;
    int pages;

    public Book(String title, int pages) {
        this.title = title;
        this.pages = pages;
    }

    @Override
    public int compareTo(Book other) {
        if(this.pages == other.pages) {
            return this.title.compareTo(other.title);
        } else {
            return this.pages - other.pages;
        }
    }
}
  
```

Lengkapi implementasi method *percolateUp()* dalam sebuah **minimum binary heap** yang menyimpan kumpulan objek bertype Book.

2. Beberapa bahasa pemrograman tingkat tinggi (*high-level*) menerapkan mekanisme *garbage collection* secara otomatis yang bertujuan untuk membersihkan ruang memori dari objek-objek yang sudah tidak terpakai. Salah satu teknik yang diterapkan untuk menentukan objek yang dapat dihapus adalah dengan menghitung jumlah rujukan (*reference counting*) pada masing-masing objek ketika program sedang berjalan.

Untuk dapat menerapkan *garbage collection* dengan teknik *reference counting*, objek-objek yang aktif berserta keterkaitan diantara objek-objek direpresentasikan sebagai sebuah **directed graph**. Objek dalam bahasa pemrograman direpresentasikan sebagai sebuah vertex dan rujukan dari sebuah objek ke objek lain direpresentasikan sebagai sebuah **directed edge**. Perhitungan jumlah rujukan dilakukan dengan menghitung *in-degree* setiap vertex di dalam graph. Proses *garbage collection* akan

NPM / Nama: _____ / KUNCI

dilakukan pada vertex-vertex dengan nilai *in-degree* sama dengan 0. Dengan kata lain, *garbage collection* akan menghapus objek-objek yang sudah tidak dirujuk oleh objek-objek lainnya.

Implementasikan method **getGCCandidates()** (kepanjangan *getGarbageCollectionCandidates()*) di bawah ini yang bertujuan untuk memperoleh sebuah kumpulan vertex-vertex yang memiliki nilai *in-degree* sama dengan 0 dari dalam graph.

```

/**
 * Representasi dari keterkaitan objek-objek dalam bentuk graph.
 */
public class ReferenceGraph {

    // Asumsikan adjacencyList tidak kosong
    private Map<Vertex, List<Edge>> adjacencyList;

    public List<Vertex> getGCcandidates() {
        // TODO Implement me!
    }
}

/**
 * Representasi objek dalam sebuah bahasa pemrograman.
 */
class Vertex {

    String label;
    Object value; // Menyimpan rujukan ke objek dalam program yang sedang berjalan

    public Vertex(String label, Object value) {
        this.label = label;
        this.value = value;
    }

    @Override
    public boolean equals(Object other) {
        if(other instanceof Vertex) {
            return this.label.equals(other.label) &&
                   this.value.equals(other.value);
        } else
            return false;
    }

    @Override
    public int hashCode() {
        return (label.hashCode() + value.hashCode()) * 13;
    }
}

/**
 * Representasi rujukan sebuah objek ke objek Lain, e.g. objek A
 * memiliki sebuah instance variable ke objek Lain B.
 */
class Edge {
    String label;
    Vertex from;
}
  
```

NPM / Nama: _____ / KUNCI

```

Vertex to;

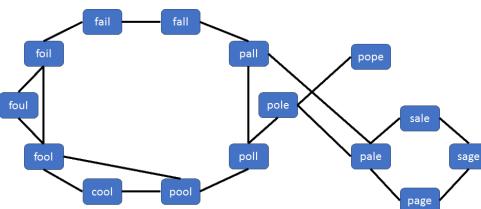
public Edge(String label, Vertex from, Vertex to) {
    this.label = label;
    this.from = from;
    this.to = to;
}
)
}

```

3. Teka-teki (*puzzle*) Word Ladder adalah sebuah permainan dimana sebuah kata harus dapat diubah menjadi kata lain dengan mengubah satu per satu huruf pada kata. Di setiap perubahan huruf, kata sementara yang dihasilkan harus berupa sebuah kata yang sah (i.e. sebuah kata yang ada di kamus).

Sebagai contoh, misalkan kata **FOOL** ingin diubah menjadi kata **SAGE**. Maka salah satu urutan pengubahan yang mungkin adalah sebagai berikut: **FOOL** → **POOL** → **POLL** → **POLE** → **PALE** → **SALE** → **SAGE** (garis bawah menandakan huruf yang berubah dari kata sebelumnya)

Pencarian solusi dari teka-teki Word Ladder dapat dilakukan dengan merepresentasikan kata-kata beserta perubahannya yang sah diantara kata-kata tersebut sebagai suatu **undirected graph**. Setiap kata dapat direpresentasikan sebagai vertex dan perubahannya sebuah kata ke kata lain direpresentasikan sebagai edge. Contoh visualisasi graph yang dimaksud dapat dilihat pada gambar berikut:



Dengan menggunakan kode dasar berikut, implementasikan method `List<String> getShortestTransformation(String from, String to)` yang mengembalikan urutan perubahan tersingkat dari kata `from` menjadi kata `to` dengan cara melakukan BFS (breadth-first search) pada graph.

```

public class WordLadderGraph {
    // Asumsikan adjacencyList tidak kosong
    private Map<Vertex, List<Edge>> adjacencyList;
}

```

Page 10 of 13

NPM / Nama: _____ / KUNCI

```

public List<String> getShortestTransformation(String from, String to) {
    // TODO Implement me!
}

class Vertex {
    String word;
    Vertex prev;

    public Vertex(String word) {
        this.word = word;
        this.prev = null;
    }

    @Override
    public boolean equals(Object other) {
        if(other instanceof Vertex) {
            return this.word.equals(other.word);
        }
        else
            return false;
    }

    @Override
    public int hashCode() {
        return word.hashCode() * 13;
    }
}

class Edge {
    Vertex from;
    Vertex to;

    public Edge(Vertex from, Vertex to) {
        this.from = from;
        this.to = to;
    }
}

```

Sebagai contoh, jika mengacu pada contoh urutan transformasi **POOL** menjadi **SAGE** di atas, maka isi dari `List` yang dikembalikan oleh pemanggilan method `getShortestTransformation()` adalah: **[FOOL, POOL, POLL, POLE, PALE, SALE, SAGE]**

4. Diberikan sebuah string hasil pengkodean Huffman beserta representasi pohon Huffman berikut:

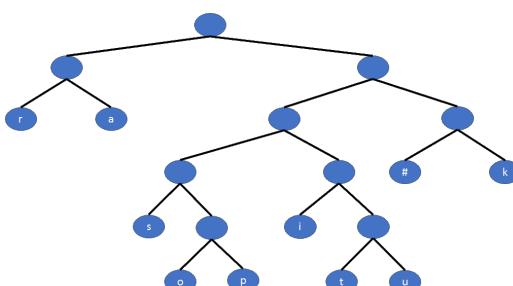
```

111 1010 00 01 110 111 1010 00 01 110 111 01 00 1000 01 110 111 01 00 10010 110 111 01
010110 01 110 111 10111 00 10111 110 1000 01 10011

```

Page 11 of 13

NPM / Nama: _____ / KUNCI



Dengan menggunakan kode dasar berikut, implementasikan method `String decode()` yang mengembalikan string asli dari bentuk kode Huffman.

Bantuan 1: Proses *decoding* setiap simbol dilakukan dengan cara membaca kode Huffman yang diberikan untuk mengarahkan traversal pohon Huffman dari root hingga mencapai leaf yang mengandung simbol dari string asli. Setiap kali traversal telah mencapai leaf, maka pembacaan simbol selanjutnya dari string kode Huffman akan dimulai kembali dari root.

Bantuan 2: Simbol **0** mengarahkan traversal ke anak subtree kiri, **1** mengarahkan traversal ke anak subtree kanan

```

class HuffmanNode implements Comparable<HuffmanNode> {
    // Mengandung sebuah karakter jika node berada di Leaf
    Character ch;

    // Mengandung frekuensi kemunculan karakter pada sebuah subtree
    // pohon Huffman
    int count;

    // Masing-masing merupakan rujukan ke subtree kiri dan kanan
    HuffmanNode left;
    HuffmanNode right;

    public HuffmanNode(Character ch, int count, HuffmanNode left,
                       HuffmanNode right) {
        this.ch = ch;
        this.count = count;
        this.left = left;
        this.right = right;
    }

    public HuffmanNode(Character ch, int count) {
        this(ch, count, null, null);
    }
}

```

Page 12 of 13

NPM / Nama: _____ / KUNCI

```

@Override
public int compareTo(HuffmanNode other) {
    return this.count - other.count;
}

class HuffmanCode {
    // Mengandung string biner hasil pengkodean Huffman
    // Simbol 0 mengarahkan traversal ke subtree kiri, 1 ke subtree kanan
    String code;

    // Merujuk ke root pohon Huffman
    HuffmanNode root;

    public String decode() {
        // TODO Implement me!
    }
}

```

5. Mengacu pada kode dasar dari soal sebelumnya (nomor 4), implementasikan static method baru, `TreeMap<Character, Int> countFrequency(String document)` pada class `HuffmanCode`, yang menghitung frekuensi kemunculan setiap simbol unik dalam string `document` yang diberikan dan mengembalikan tabulasi frekuensinya sebagai objek `TreeMap`. Tipe parameter kunci (`key`, i.e. `Character`) pada `TreeMap` mengacu pada simbol unik dari dalam dokumen, sedangkan tipe parameter nilai (`value`, i.e. `Integer`) dari suatu `key` mengacu pada nilai frekuensi kemunculan simbol tersebut.

SELESAI

Page 13 of 13