UNIVERSITAS INDONESIA
FAKULTAS ILMU KOMPUTER
Veritas, Probitas, Justitia

CSGE602040 – Struktur Data & Algoritma
UAS Semester Gasal 2019/2020
21 Desember 2019, 13.00 – 16.00

Pengajar: Suryana Setiawan, Rahmad Mahendra,
Iis Solichah, Dipta Tanaya

PERNYATAAN KESANGGUPAN MENAATI TATA TERTIB UJIAN

"Saya telah membaca dan memahami ketentuan tata tertib berikut ini, serta menyatakan bahwa jawaban ujian ini adalah hasil pekerjaan saya sendiri. Saya menyetujui jika melakukan pelanggaran atas ketentuan tersebut, saya bersedia diproses sesuai ketentuan yang berlaku (SK DGB UI No.1 Tahun 2014) dengan sanksi maksimal **nilai akhir E**."

Nama & Tanda-tangan:	Kelas:	Nomor Pokok Mahasiswa:

TATA TERTIB UJIAN

- Semua alat komunikasi elektronik dalam kondisi non-aktif (dimatikan), dimasukkan ke dalam tas dan diletakkan pada tempat yang telah disediakan.
- Peralatan ujian yang boleh dibawa adalah alat tulis dan yang diperbolehkan sesuai sifat ujian.
- Peserta ujian menempati tempat duduk yang telah ditentukan.
- Peserta ujian menuliskan nama dan NPM pada setiap lembar jawaban ujian.
- Peserta mulai membuka soal dan mengerjakan ketika pengawas mengatakan ujian dimulai dan berhenti bekerja (meletakkan alat tulis) ketika pengawas mengatakan waktu habis.
- Peserta tidak berkomunikasi dalam bentuk apapun dengan peserta lain selama berada di ruang ujian, termasuk pinjam meminjam alat tulis, serta tidak memberi atau menerima bantuan dari siapapun selama ujian berlangsung.
- Peserta yang meninggalkan ruang ujian dianggap selesai mengerjakan. Jika karena kondisi medis khusus tidak bisa memenuhi ketentuan ini, peserta wajib melaporkan kepada pengawas sebelum ujian dimulai.
- Setelah selesai mengerjakan atau setelah waktu habis, peserta segera meninggalkan berkas soal dan lembar jawaban ujian di meja masing-masing, mengambil tas dan segera keluar tanpa mengganggu peserta lain serta tanpa berkomunikasi dengan peserta lain.
- Jawaban ujian ini tidak akan dinilai jika pernyataan di atas ini tidak ditandatangani.

Informasi Tambahan

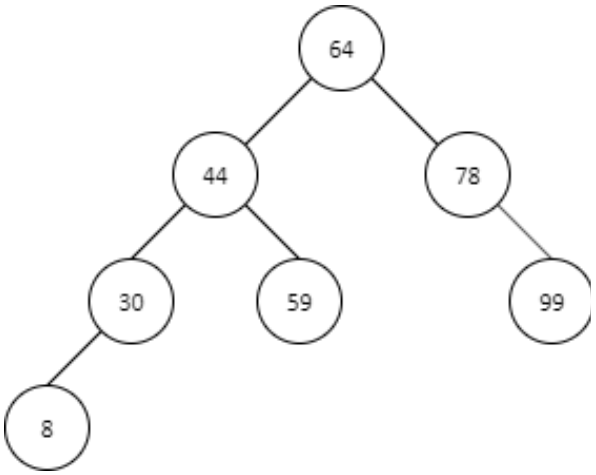
- Sifat ujian: open notes (1 lembar A4)

Nama	Steven	Kelas	A
NPM	2206030855	No. Meja	13

Bagian A (40 poin)

Bagian A terdiri atas soal isian pendek dan pilihan ganda. Untuk isian pendek, jawablah di tempat yang telah disediakan. Untuk pilihan ganda, lingkari jawaban yang benar.

1. Perhatikan *tree* berikut ini.



Jika elemen dari *binary tree* dicetak dengan menggunakan algoritma *level order traversal* dengan urutan pencetakan dari kiri ke kanan, maka *tree* di samping akan dicetak sebagai berikut:

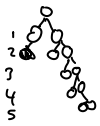
64, 44, 78, 30, 59, 99, 8

2. Diketahui sebuah BST yang jika dicetak dengan *level-order traversal* menghasilkan barisan elemen yang terurut terbalik (dari besar ke kecil). BST tersebut juga <u>pasti</u> tercetak dengan pola ...
   a. Terurut (dari kecil ke besar) dengan *pre-order traversal* dan terurut terbalik (dari besar ke kecil) dengan *post-order traversal*
   b. Terurut terbalik (dari besar ke kecil) dengan *pre-order traversal* dan terurut (dari kecil ke besar) dengan *post-order traversal*
   c. Terurut (dari kecil ke besar) dengan *pre-order traversal* maupun *post-order traversal*
   d. Terurut terbalik (dari besar ke kecil) dengan *pre-order traversal* maupun *post-order traversal*
   e. Tidak dapat ditentukan urutannya (acak) dengan *pre-order traversal* maupun *post-order traversal*

3. Dari pernyataan berikut, manakah yang salah mengenai Binary Search Tree?
   a. Elemen pada *left child* dari sebuah *node* pasti lebih kecil dari elemen pada *node* tersebut
   b. Elemen pada *right child* dari sebuah *node* pasti lebih besar dari elemen pada *node* tersebut
   c. *Left subtree* dari sembarang *node* pada BST pasti merupakan BST juga
   d. *In-order traversal* dapat mencetak elemen pada BST secara terurut dari besar ke kecil

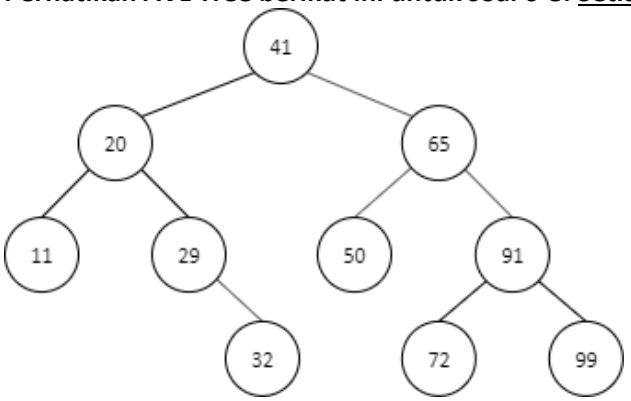
4. Kompleksitas algoritma menghitung tinggi sebuah sembarang *binary tree* yang memiliki N buah *node* adalah _____

```
public static int height (BinaryNode<E> t){
    if(t == null){
        return -1;
    }
    else{
        return max(height (t.left) + 1, height (t.right) + 1);
    }
}
```

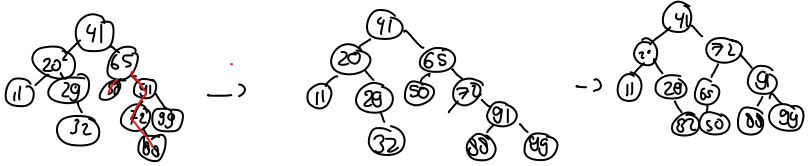


5. Banyak *node* minimal pada AVL Tree dengan tinggi 5 adalah 11

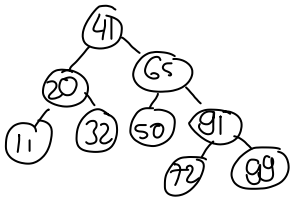
Perhatikan AVL Tree berikut ini untuk soal 6-8. <u>Setiap soal mengacu pada kondisi awal *tree.*</u>



6. Dari AVL Tree **awal**, lakukan operasi *insert* elemen 88, gambarkan hasil akhir AVL Tree tersebut.



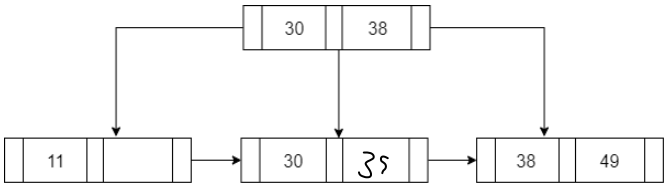
7. Dari AVL Tree **awal**, lakukan operasi *delete* 29, gambarkan hasil akhir AVL Tree tersebut.



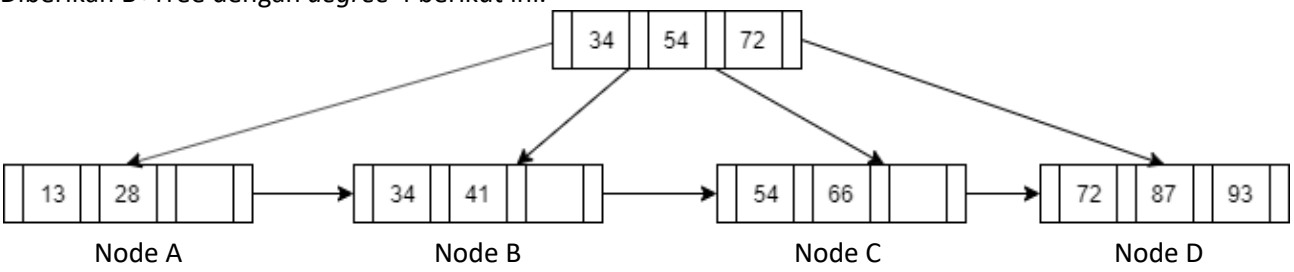
8. Dari AVL Tree **awal**, dilakukan operasi *delete root*. Elemen apakah yang menjadi *root* baru dari AVL Tree jika menggunakan kaidah *successor inorder*? 50

Nama	Steven	Kelas	A
NPM	2206030855	No. Meja	13

9. Perhatikan B+Tree dengan *degree* 3 berikut ini. Lakukan *insert* data 35. Gambarkan hasil akhir B+Tree.



10. Diberikan B+Tree dengan *degree* 4 berikut ini.



Jika data 34 dihapus, maka akan terjadi ...

a. Node A dan B akan di-*merge*
b. Node B dan C akan di-*merge*
c. Data dari node A di-*redistribute* ke node B
d. Data dari node C di-*redistribute* ke node B
e. Tidak terjadi *merge* atau *redistribute*

11. Diketahui sebuah struktur data B+Tree dengan degree 5 yang menyimpan sejumlah data. Berapa ukuran data minimal sehingga **tinggi** B+Tree tersebut tidak kurang dari 5? _____

12. Kompleksitas pencarian elemen paling kecil (**findMin**) di Minimum Binary Heap adalah O(1)

13. Diberikan sebuah Minimum Binary Heap, dengan representasi *array*-nya adalah sebagai berikut:

Indeks:	0	1	2	3	4	5	6	7	8	9
Data	5	10	15	17	20	30	18			

5
10 15
17 20 30 18

Lakukan *insert* 11 pada *binary heap* tersebut. Gambarkan isi *array*-nya setelah operasi tersebut selesai:

Indeks:	0	1	2	3	4	5	6	7	8	9
Data	5	10	15	11	20	30	18	17		

14. Berapakah tinggi dari *binary tree* yang merepresentasikan sebuah Binary Heap yang memiliki 100 *node*?

_____

15. Jika tidak pernah terjadi *collision* dalam proses *insertion* elemen, berapa kompleksitas operasi pencarian sebuah elemen pada *hash table*? O(1)

16. Diberikan sebuah *hash table* H diimplementasikan dengan *linear probing* dan menggunakan *hash function* H(x) . Manakah di antara pernyataan berikut ini yang benar?
    a. Tidak mungkin terjadi *primary clustering* pada *hash table* H.
    b. Dua buah data yang berbeda namun memiliki nilai H(x) yang sama, jika akan dimasukkan ke dalam H namun terjadi *collision*, pasti akan diarahkan (*probed*) ke *alternative cell* yang berbeda.
    c. Jika suatu elemen di sebuah *cell* pada H harus dihapus, yang dilakukan adalah memberi tanda bahwa *cell* tersebut elemennya tidak aktif, tetapi elemen tersebut tidak benar-benar dihapus.
    d. Pada hash table dengan *load factor* kurang dari 0.5, tidak mungkin terjadi collision.

17. Diberikan sebuah *hash table* dengan *linear probing* sebagai *collision resolution* berikut ini. Jika suatu *cell* belum pernah diisi data, maka *flag*-nya *inactive*. Jika suatu *cell* berisi data dan *flag*-nya *active*, artinya data tersebut statusnya tidak dihapus. Namun, jika suatu *cell* berisi data namun *flag*-nya *inactive*, artinya data tersebut statusnya dihapus.

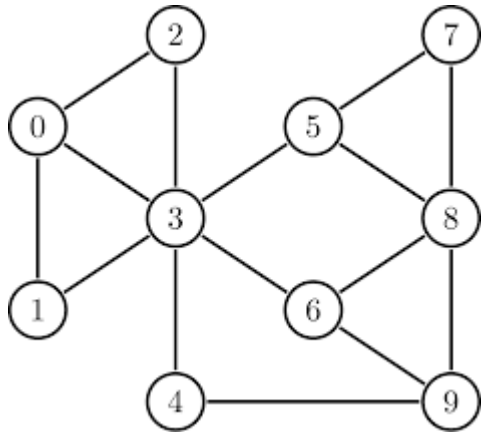
Indeks	Cell	
	Data	Flag
0	apel	active
1		inactive
2	kiwi	active
3	nanas	active
4	mangga	active
5	pisang	inactive
6	durian	active
7		inactive
8		inactive
9	pir	active
10		active

Jika data “nangka” akan dimasukkan ke *hashtable* dan diketahui nilai *hash function* dari data “nangka” adalah 3. Menggunakan linear probing, maka data “nangka” akan diletakkan pada cell dengan indeks 5

Nama	Steven	Kelas	A
NPM	2206030855	No. Meja	13

18. Salah satu permasalahan pada graf yang dapat diselesaikan dengan mengaplikasikan algoritma Dijkstra adalah mencari shortest _____

19. Verteks apa yang terakhir dikunjungi jika dilakukan Depth-First Search pada graf berikut dimulai dari verteks 6 (ketika terdapat lebih dari satu *adjacent* yang belum dikunjungi, verteks dengan <u>angka lebih kecil</u> akan dikunjungi terlebih dahulu)? g _____



6, 3, 0, 1, 2, 4, 5, 7, 8, 9

20. Berapa tinggi BFS *tree* jika dilakukan *traversal* pada sebuah **complete graph** yang memiliki 100 verteks?

_____

Halaman ini tidak berisi soal

Nama	Steven	Kelas	A
NPM	2206030955	No. Meja	13

Bagian B (45 poin)

1. (8 poin) Perhatikan program berikut.

```
class Node {
    int data;
    Node kiri, kanan;

    public Node(int data){
       this.data = data;
       kiri = kanan = null;
    }
}

public class BinaryTree{
    Node root;

    int doSomething() {
       return doSomething(root);
    }

    int doSomething(Node node){
       if(node == null) return 0;
       int result = 0;
       if(node.kiri == null && node.kanan != null ||
              node.kiri != null && node.kanan == null ||
              node.kiri == null && node.kanan == null) result++;
       result = result + doSomething(node.kiri) + doSomething(node.kanan);
       return result;

    }

    public static void main(String args[]){
       Node satu = new Node(1);
       Node dua = new Node(2);
       Node tiga = new Node(3);
       Node empat = new Node(4);
       Node lima = new Node(5);
       BinaryTree tree = new BinaryTree();
       tree.root = satu;
       satu.kiri = dua; satu.kanan = tiga;
       tiga.kiri = empat;
       empat.kiri = lima;

       System.out.println(tree.doSomething());
    }
}
```

result : nyari null

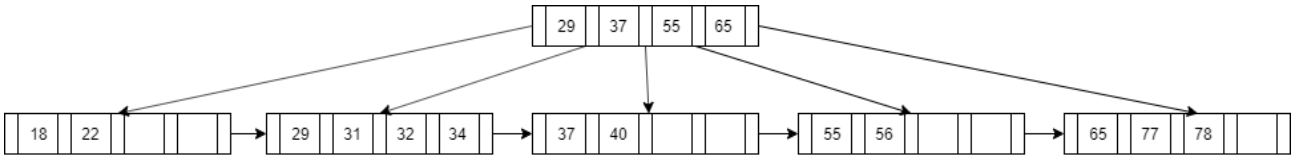
1 2 3 4 5

0 2 3 4

a. *Output* dari program di atas adalah 4

b. Apa yang dilakukan *method* doSomething(Node node)? cari jumlah null

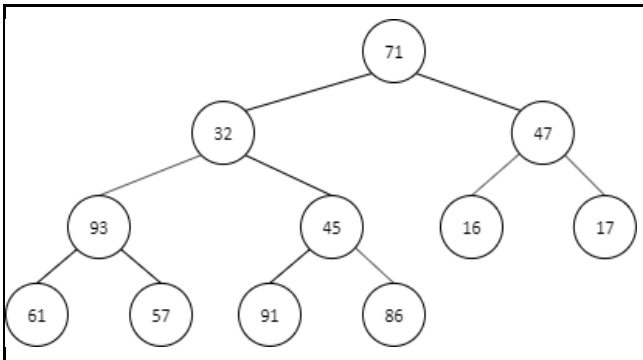
2. (8 poin) Perhatikan B+Tree dengan *degree* 5 berikut ini.



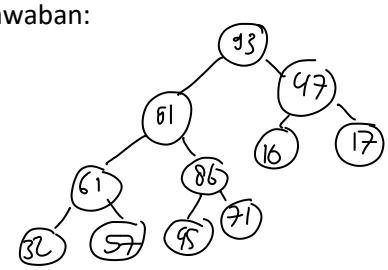
a. Dari B+Tree **awal**, lakukan *insert* data 35. Gambarkan hasil akhir B+Tree.

b. Dari B+Tree **awal**, lakukan *delete* data 22. Gambarkan hasil akhir B+Tree.

3. (7 poin) Diberikan sebuah *complete binary tree be*rikut ini. Lakukan operasi *fix heap / heapify* agar *complete binary tree* tersebut menjadi **Maximum Binary Heap**. Gambarkan hasil akhir Maximum Binary Heap.

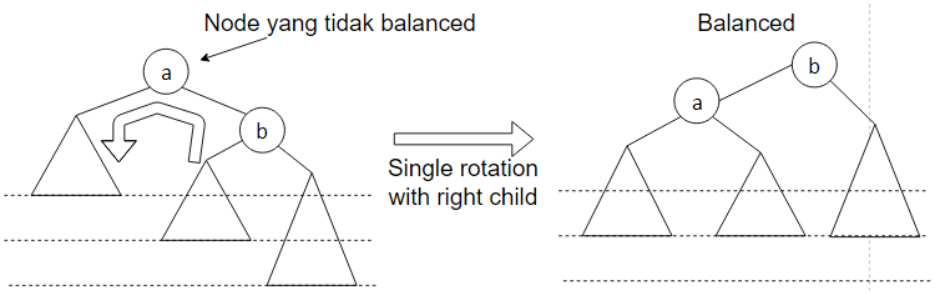


Jawaban:



Nama		Kelas	
NPM		No. Meja	

4. (5 poin) Diberikan ilustrasi mengenai keadaan sebuah subtree yang tidak balanced sebagai berikut (gambar sebelah kiri), lalu dilakukan single rotation terhadap anak kanan sehingga menghasilkan subtree yang balanced (gambar sebelah kanan):



Lengkapilah implementasi *class* AvlNode dengan *method* `singleRotateWithRightChild`. *Method* singleRotateWithRightChild menerima sebuah **`AvlNode` a** yang merupakan *root* dari *subtree* yang tidak *balanced*, lalu melakukan *balancing* dengan *single rotation* sesuai gambar di atas, dan mengembalikan AvlNode b yang menempati posisi *root* dari *subtree* yang sudah *balanced*.

```
class AvlNode{

       int elemen;
       AvlNode kiri, kanan;

       public AvlNode(int data){
              this(data, null, null );
       }

       public AvlNode(int data, AvlNode nodeKiri, AvlNode nodeKanan ){
              elemen = data;
              kiri = nodeKiri;
              kanan = nodeKanan;
       }
      AvlNode singleRotateWithRightChild(AvlNode a){
            AvlNode b= a.right;
            AvlNode c = b.left
            b.left = a ;
            a.right = c;
            return b;









       }
}
```

5. (5 poin) Jelaskan dengan singkat kelebihan AVL Tree dibandingkan dengan BST (maksimal 50 kata).

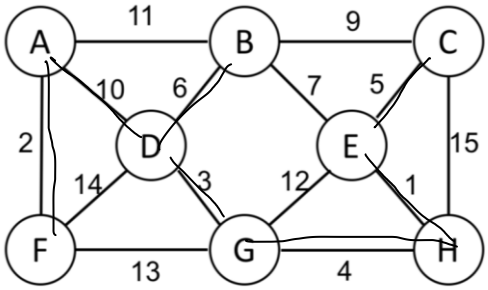
6. (6 poin) Diberikan sebuah *hash table* berukuran 13 dengan *hash function* sebagai berikut:
   H(x) = x mod 13, x adalah *key* dari data masukan.
   Gunakan *quadratic probing* untuk operasi *insertion* data berikut ini, dengan urutan *insertion*:
   16, 20, 55, 42, 3
   Tuliskan hasil akhir setelah seluruh data selesai dimasukkan ke *hash table.*

   Jawab:

Indeks	Data
0	
1	
2	3
3	16
4	55
5	
6	
7	20
8	
9	
10	
11	
12	42

Nama		Kelas	
NPM		No. Meja	

7. (6 poin) Perhatikan graf berikut.



Gunakan algoritma Kruskal untuk menentukan Minimum Spanning Tree (MST) dari graf di samping.

Edge apa saja yang <u>tidak</u> ditambahkan ke dalam himpunan *edge* MST karena membentuk *cycle*? Tuliskan sesuai urutan *edge* tersebut diketahui membentuk *cycle.* ER

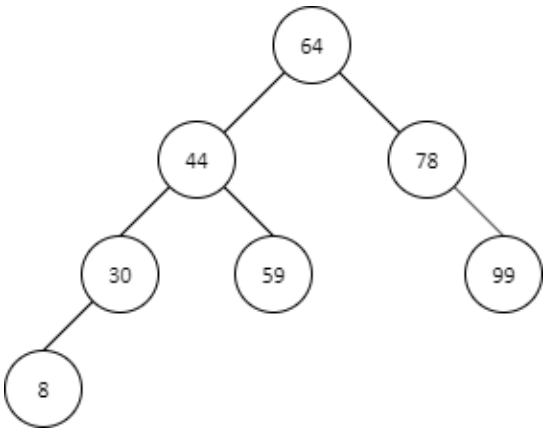
Untuk masing-masing *edge* tersebut, *cycle* apa yang terbentuk?

EH, DG, GH, CE, DB, AF

Jawaban:

Bagian C (20 poin)

1. (10 poin) Implementasikan *method* untuk mencari *sibling* dari sebuah *node* pada sebuah BST. Pada soal ini, diasumsikan bahwa Binary Node yang diberikan sebagai parameter input pasti terdapat dalam BST. Sebagai contoh, perhatikan *binary tree* berikut ini. *Sibling* dari 30 adalah 59.




```
class BinaryNode{
   int data;
   BinaryNode left, right;

   BinaryNode(int data){
          this.data = data;
          left = null;
          right = null;
   }
```

```
}

class BinaryTree{
   BinaryNode root;

   BinaryTree(BinaryNode root){
          this.root = root;
   }

   // TO-DO
   BinaryNode sibling(BinaryNode iniNode){









   }
}
```

Nama		Kelas	
NPM		No. Meja	

2. Diberikan sebuah graf tidak berarah dan tidak berbobot. Buatlah sebuah *method* untuk mencari panjang lintasan terpendek dari dua buah verteks.

Format Masukan Baris pertama berisi dua buah bilangan **M** dan **N**. **M** adalah jumlah verteks (label setiap verteks adalah 0, 1, 2, … M-1). **N** adalah jumlah edge. N baris berikutnya berisi informasi edge. Baris terakhir berisi dua buah bilangan bulat, yaitu **verteks awal dan verteks akhir**	**Contoh Masukan** 5 5 0 3 0 4 1 2 1 3 2 3 4 2
Format Keluaran Sebuah angka yang menyatakan panjang lintasan terpendek dari dua buah verteks	**Contoh Keluaran** 3

```
class Graph{
    private static int JUMLAH_VERTEX;
    private static ArrayList<Integer> adjList[];

    Graph(int v){
        JUMLAH_VERTEX = v;
        adjList = new ArrayList[JUMLAH_VERTEX];
        for (int i=0; i<JUMLAH_VERTEX; i++) adjList[i] = new ArrayList();
    }

    void addEdge(int v, int w){
        adjList[v].add(w);  adjList[w].add(v);
    }

    public static void main(String args[]) throws IOException{
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        String[] in = reader.readLine().split(" ");
        int jmlVertex = Integer.parseInt(in[0]);
        int jmlEdge = Integer.parseInt(in[1]);

        Graph graf = new Graph(jmlVertex);
        for(int i=0; i<jmlEdge; i++){
            in = reader.readLine().split(" ");
            graf.addEdge(Integer.parseInt(in[0]), Integer.parseInt(in[1]));
        }

        in = reader.readLine().split(" ");
        int vertexAwal = Integer.parseInt(in[0]);
        int vertexAkhir = Integer.parseInt(in[1]);
        int res = graf.solve(vertexAwal, vertexAkhir);
        System.out.print(res);
    }

    int solve(int awal, int akhir){
```

```
    }
}
```

Selamat mengerjakan \^_^/