Laporan Pengerjaan Tugas Pemrograman Kecil

Penyelesaian Permainan Kakurasu dengan Algoritma *Brute Force*

Oleh : Afrizal Fikri

NIM : 13513004

*Kakurasu* adalah sebuah permainan angka yang berasal dari Jepang dengan aturan bermain sebagai berikut :

* Diberikan sebuah petak persegi berukuran N x N sembarang
* Tiap petak memiliki nilai secara horizontal maupun vertikal
* Pilih sebagian (atau semua) petak sehingga jumlah nilai petak secara horizontal pada semua baris sesuai permintaan. Begitu pula dengan nilai vertikalnya.

Salah satu metode untuk menyelesaikan permainan ini yaitu dengan menggunakan pendekatan *brute force*, yaitu mencoba semua kemungkinan pemilihan petak.

Algoritma berbasis *brute force* dalam penyelesaian permainan *kakurasu* :

1. Cek apakah nilai pada petak-petak yang terisi sesuai dengan nilai yang diminta pada sisi bawah (untuk vertikal) dan kanan tabel (untuk horizontal)
2. Jika sudah tepat, algoritma berhenti
3. Jika belum dan belum mencapai petak terakhir, lanjut ke petak berikutnya dan lakukan dari langkah 1 lagi
4. Lanjutkan hingga mencapai petak terakhir. Lalu pilih petak sekarang lalu kembali ke petak sebelumnya dan lakukan dari langkah 1 lagi
5. Ulangi untuk kesemua petak

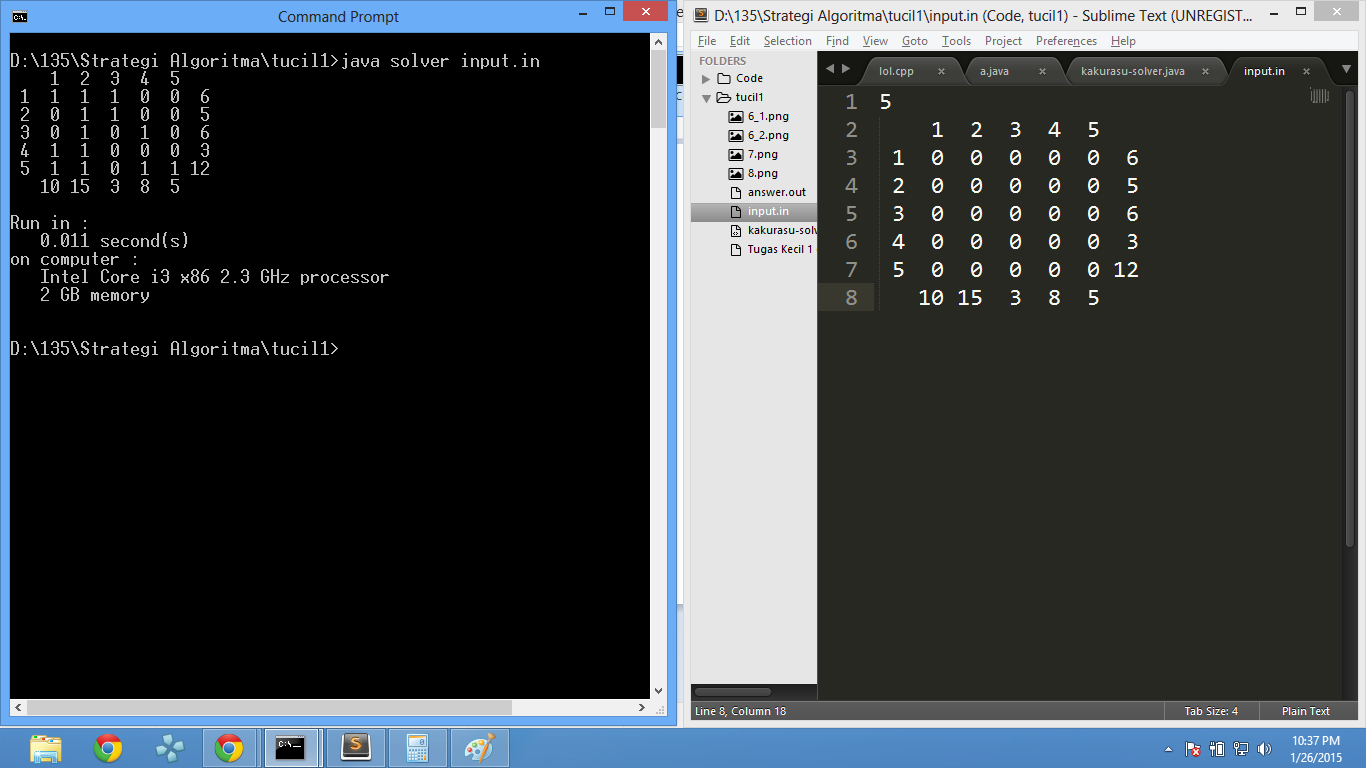
Saya menggunakan implementasi berbasis objek (*object oriented*) menggunakan bahasa Java. Berikut adalah kode sumber yang saya buat :

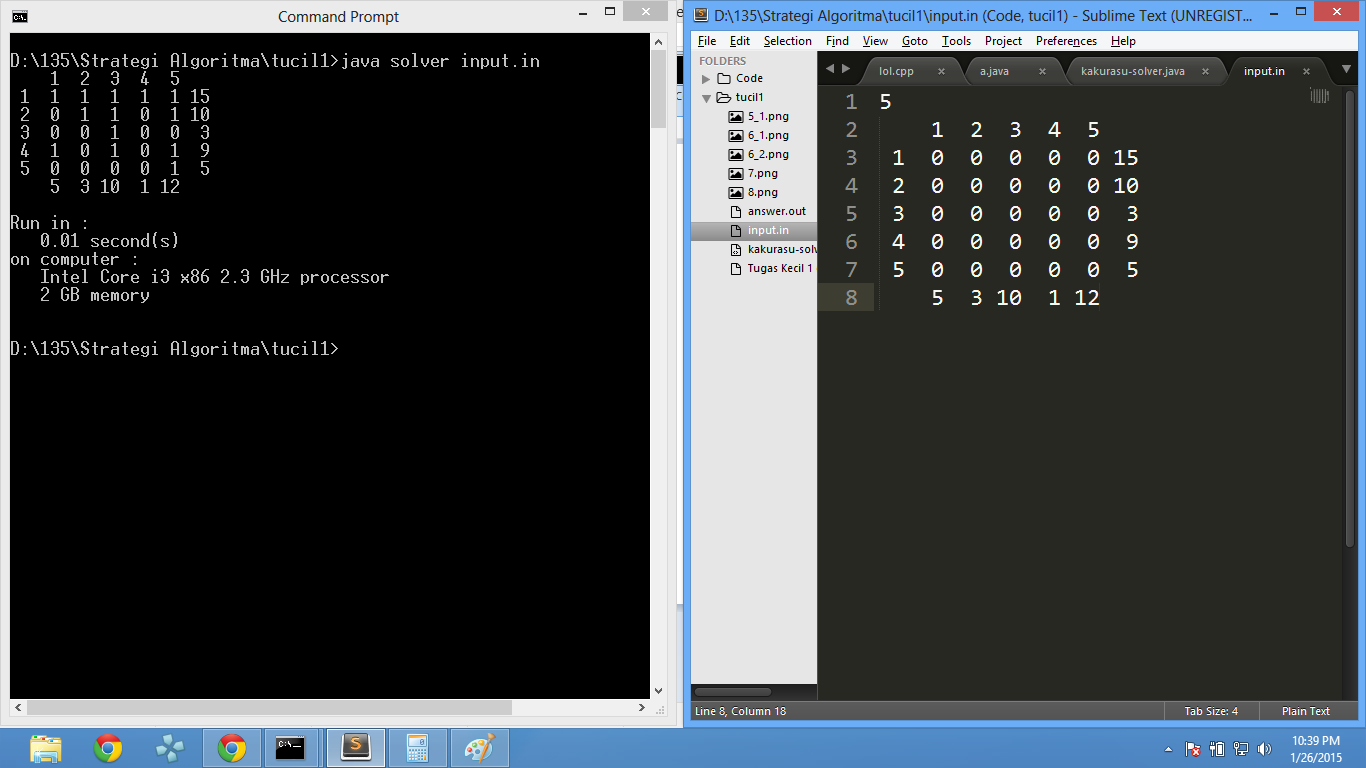
|  |
| --- |
| import java.lang.\*;  import java.io.\*;  import java.util.\*;  // Kelas solver utama  class solver {  public static void main(String[] args) throws Exception {  Reader in = new Reader();  try {  Reader.init(new FileInputStream(args[0]));  } catch (Exception e) {  Reader.init(System.in);  }  Board kaku = new Board();  kaku.init(in);  kaku.printBoard();  System.runFinalizersOnExit(true);  }  }  // Kelas untuk pencarian solusi per baris  class Helper {  protected int size;  protected int[] hval;  protected Map<Integer, Vector<Vector<Boolean>>> dict;  protected Vector<Vector<Boolean>> tmp;  protected Vector<Boolean> ans;  public void init(int n, int[] hval) {  size = n;  this.hval = hval;  dict = new HashMap<>();  ans = new Vector<>();  for(int i = 0; i < size; ++i) {  ans.add(false);  }  }  public void push(int n) {  try {  int tes = dict.get(n).size();  } catch (NullPointerException e) {  dict.put(n, new Vector<Vector<Boolean>>());  tmp = dict.get(n);  backtrack(n, 0);  }  }  protected void backtrack(int sum, int pos) {  try {  int k = hval[pos];  backtrack(sum, pos + 1);  if (sum >= k) {  ans.set(pos, true);  backtrack(sum - k, pos + 1);  ans.set(pos, false);  }  } catch (Exception e) {  if (sum == 0) {  tmp.add(new Vector());  for(Boolean b : ans) {  tmp.lastElement().add(b);  }  }  }  }  public Vector<Vector<Boolean>> get(int n) {  return dict.get(n);  }  }  // Kelas papan tempat solusi dijalankan  class Board {  protected boolean[][] brd;  protected int[] hsum, vval, vsum, hval;  protected int size;  protected Helper help;  protected Stopwatch timer;  public void init(Reader in) throws Exception {  size = in.nextInt();  hsum = new int[size];  vsum = new int[size];  vval = new int[size];  hval = new int[size];  brd = new boolean[size][size];  for(int i = 0; i < size; ++i) {  hval[i] = in.nextInt();  }  for(int i = 0; i < size; ++i) {  vval[i] = in.nextInt();  for(int j = 0; j < size; ++j) {  brd[i][j] = (in.nextInt() == 1);  }  hsum[i] = in.nextInt();  }  for(int i = 0; i < size; ++i) {  vsum[i] = in.nextInt();  }  timer = new Stopwatch();  help = new Helper();  help.init(size, hval);  for(int i = 0; i < size; ++i) {  help.push(hsum[i]);  }  backtrack(0);  }  protected boolean backtrack(int pos) {  boolean ans = false;  if (pos < size) {  for(Vector<Boolean> v : help.get(hsum[pos])) {  for(int i = 0; i < size; ++i) {  brd[pos][i] = v.get(i);  }  ans = backtrack(pos + 1);  if (ans) {  break;  }  }  } else {  ans = checkV();  }  return ans;  }  public void printBoard() {  printNumber(-1);  for (int i = 0; i < size; ++i) {  printNumber(hval[i]);  }  printNumber(-1);  System.out.println();  for (int i = 0; i < size; ++i) {  printNumber(vval[i]);  for (int j = 0; j < size; ++j) {  if (brd[i][j])  printNumber(1);  else  printNumber(0);  }  printNumber(hsum[i]);  System.out.println();  }  printNumber(-1);  for (int i = 0; i < size; ++i) {  printNumber(vsum[i]);  }  printNumber(-1);  System.out.println();  }  protected void printNumber(int n) {  if (n < 10) System.out.print(" ");  if (n < 0) System.out.print(" ");  else System.out.print(n);  System.out.print(" ");  }  protected boolean checkV() {  boolean ans = true;  int sum;  for(int j = 0; ans && j < size; ++j) {  sum = 0;  for(int i = 0; i < size; ++i) {  if (brd[i][j]) sum += vval[i];  }  if (sum != vsum[j]) ans = false;  }  return ans;  }  }  // Kelas pewaktu lama runtime program  class Stopwatch {  protected long starttime;  protected long endtime;  Stopwatch() {  starttime = System.currentTimeMillis();  }  protected void finalize() throws Throwable {  endtime = System.currentTimeMillis();  System.out.println("\nRun in :");  System.out.println(" " + (double)(endtime - starttime) / 1000.0 + " second(s)");  System.out.println("on computer :");  System.out.println(" Intel Core i3 x86 2.3 GHz processor");  System.out.println(" 2 GB memory\n");  super.finalize();  }  }  // Kelas untuk menangani metode input dengan atau tanpa file eksternal  class Reader {  static BufferedReader reader;  static StringTokenizer tokenizer;  static void init(InputStream input) {  reader = new BufferedReader(  new InputStreamReader(input),  32768);  tokenizer = null;  }  static String next() {  while (tokenizer == null || !tokenizer.hasMoreTokens()) {  try {  tokenizer = new StringTokenizer(reader.readLine());  } catch (Exception e) {  throw new RuntimeException(e);  }  }  return tokenizer.nextToken();  }  static int nextInt() {  return Integer.parseInt(next());  }  } |

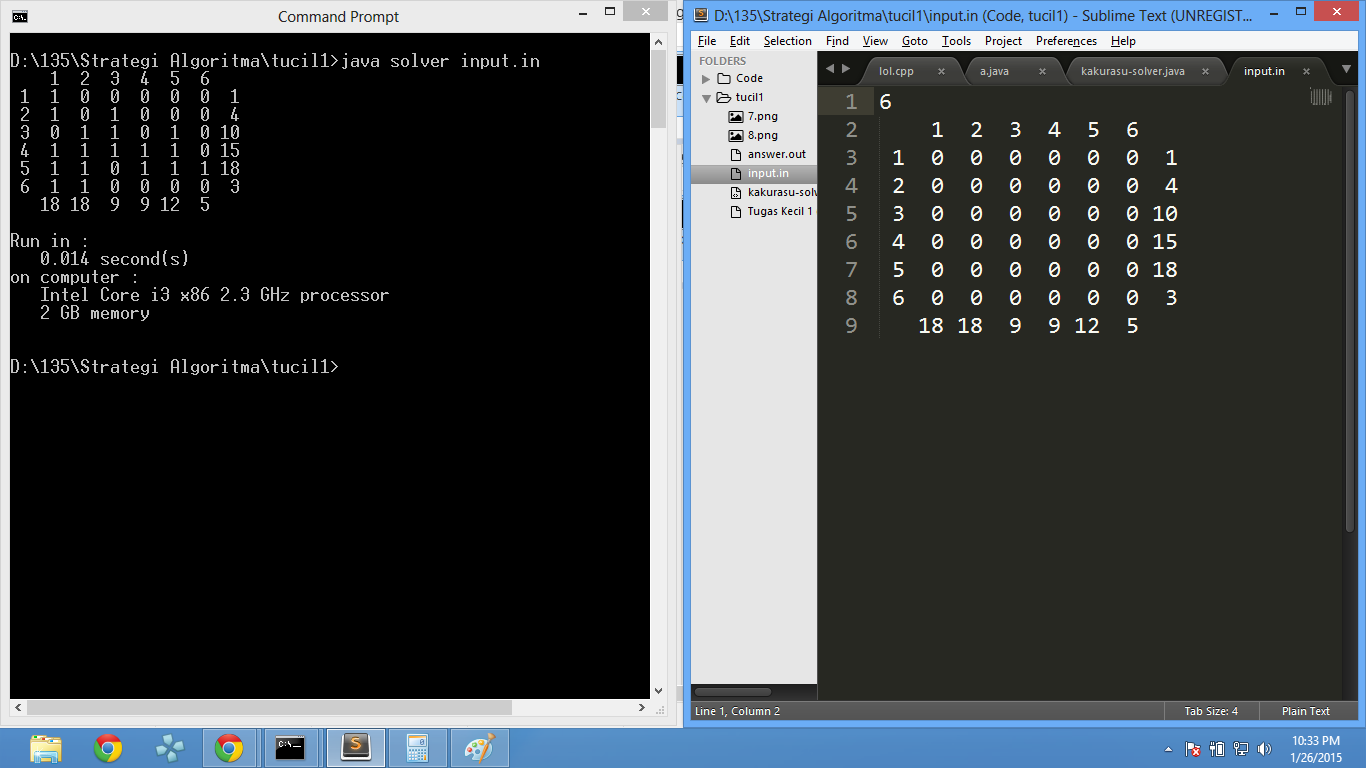
Seperti tertulis pada kode di atas ada 5 buah kelas :

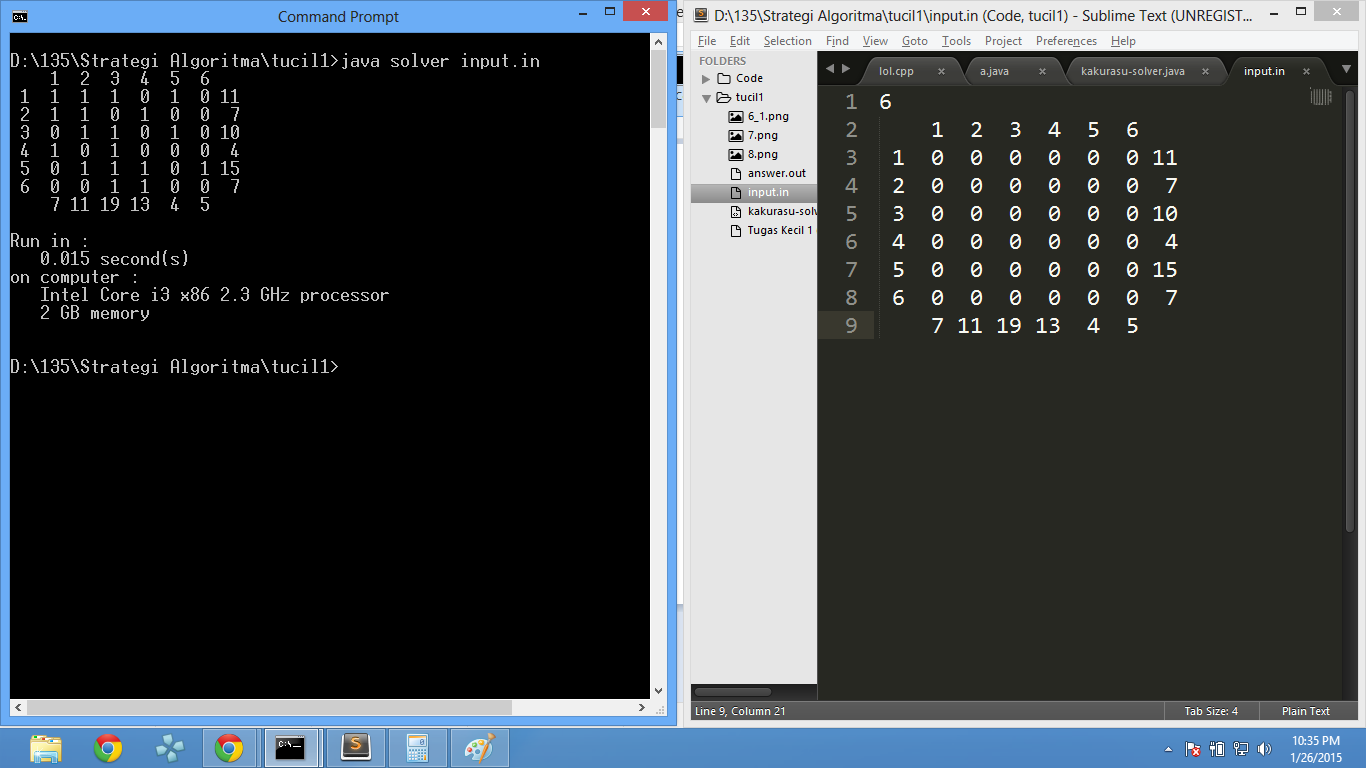
1. Kelas solver : tempat *main thread* dijalankan
2. Kelas Board : tempat board disimpan dan solusi dijalankan
3. Kelas Helper : tempat pencarian solusi tiap baris dijalankan dan disimpan
4. Kelas Stopwatch : penghitung waktu program berjalan
5. Kelas Reader : mengatasi masalah yang berhubungan dengan *file stream* dan pembacaan integer dengan *parsing string* (*BufferedReader* pada Java hanya dapat menerima input berupa *string*)

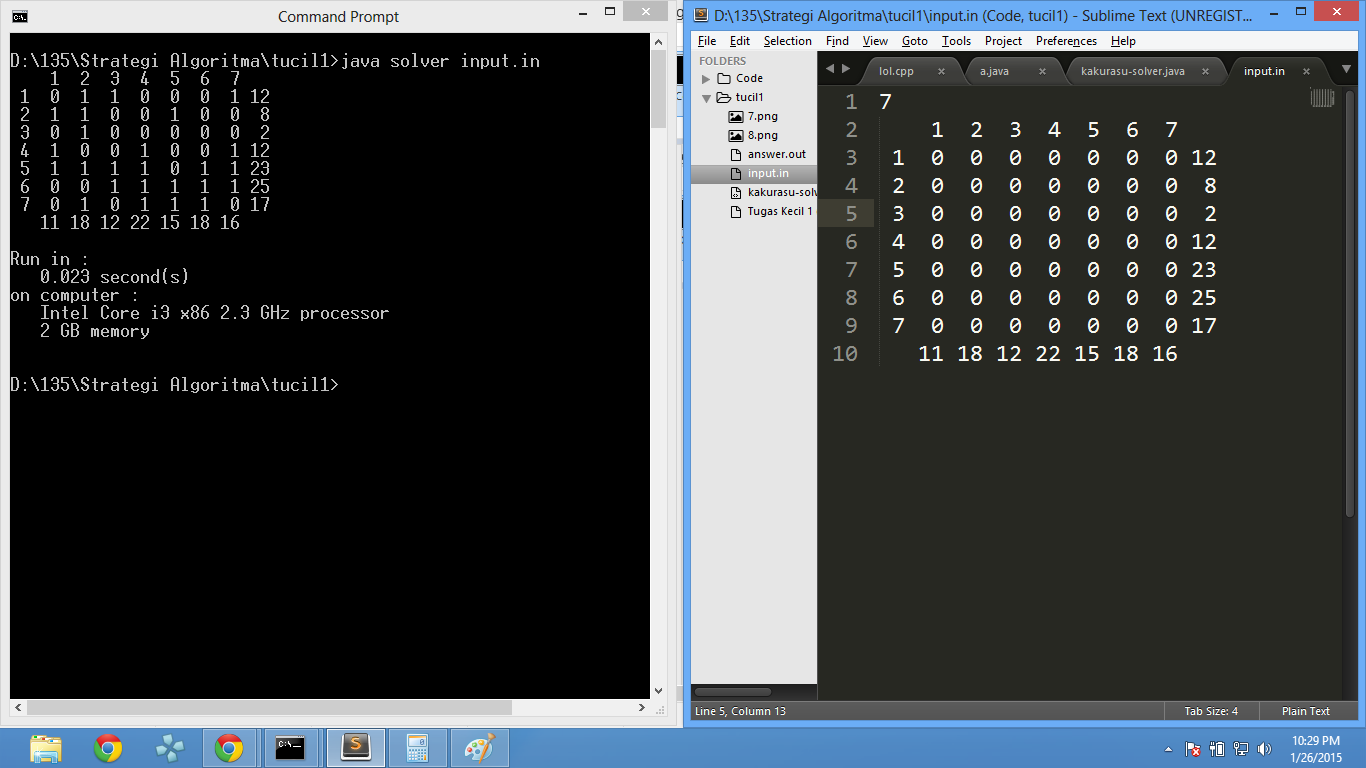
Berikut adalah contoh *screenshot* saat program berjalan pada masukan berbeda :

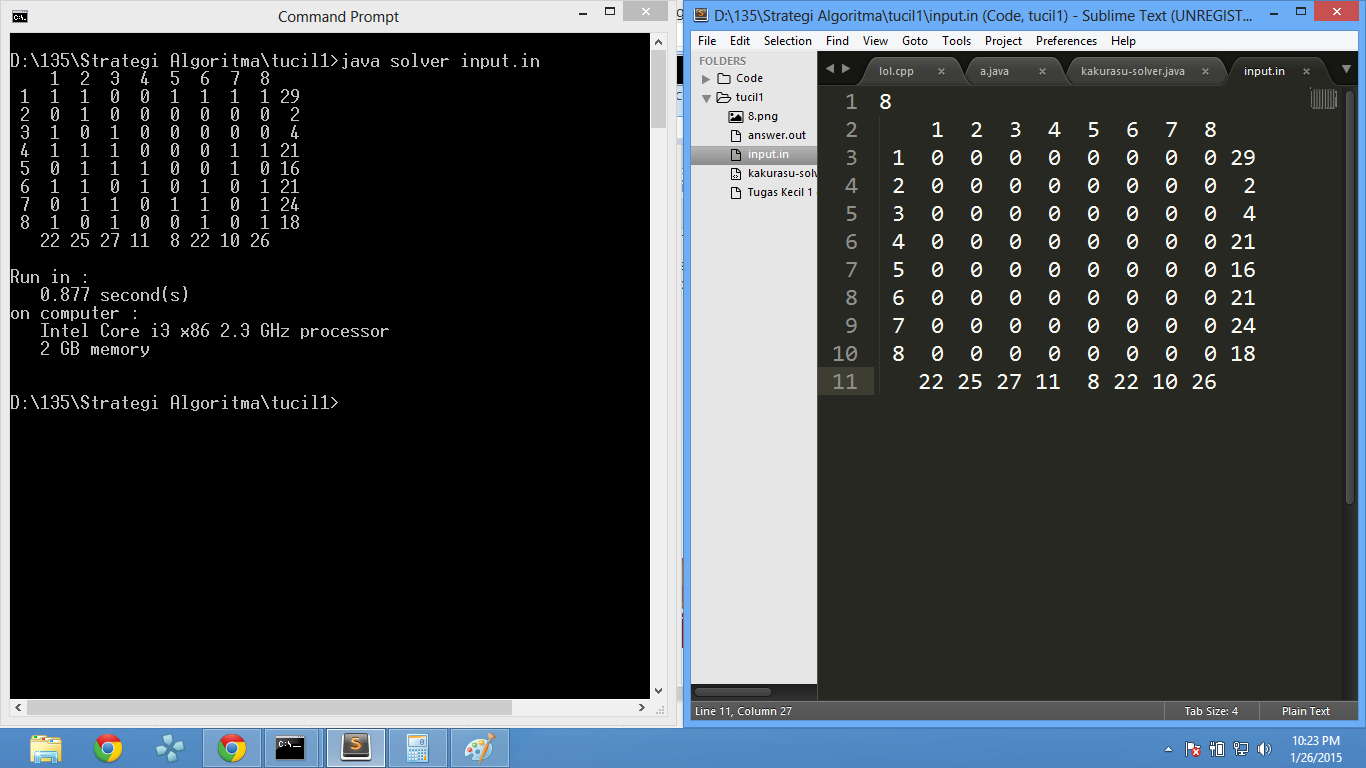












|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil dijalankan | V |  |
| 1. Program dapat membaca file input dan menuliskan keluaran | V |  |
| 1. Keluaran sudah benar | V |  |

Sekian laporan pengerjaan tugas pemrograman saya. Mohon maaf apabila ada kekurangan. Terima kasih