**LAPORAN TUGAS KECIL I**

**IF2211 STRATEGI ALGORITMA**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

A black background with white text

Description automatically generated

Disusun oleh:

Marvin Scifo Y. Hutahaean 13522110

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2024**

**Daftar Isi**

[**BAB I - PENJELASAN DASAR 3**](#_Toc125193769)

[**BAB II - SOURCE CODE 5**](#_Toc125193770)

[**BAB III TESTING 30**](#_Toc125193771)

[**LINK REPOSITORY 37**](#_Toc125193772)

[**CHECKLIST 37**](#_Toc125193773)

# BAB I PENJELASAN DASAR

1. **Algoritma *Brute Force***

Algoritma *Brute Force*, adalah salah satu pendekatan algoritma yang pendekatannya lempang (*straightforward*) dalam sebuah penyelesaian persoalan. Algoritma ini didasarkan kepada pernyataan pada persoalan (*problem statement*) dan definisi konsep yang dilibatkan. *Brute Force* memecahkan persoalan dengan sangat sederhana, langsung, dan jelas. Biasanya *Brute Force* melibatkan semua informasi yang ada pada sebuah data. Adapun contoh-contoh persoalan yang menggunakan algoritma *Brute Force* adalah

1. Mencari elemen terbesar atau terkecil dengan membandingkan suatu elemen dengan semua elemen yang ada
2. Pencarian beruntun dengan membandingkan nilai yang ingin kit acari dalam sebuah list dengan semua elemen yang ada dalam list tersebut
3. **Aplikasi Algoritma *Brute Force terhadap Cyberpunk 2077 Breach Protocol***

Pada tugas kecil ke-1 dari IF2211 Strategi Algoritma, penulis ditugaskan untuk membuat simulasi dari sebuah *minigame* dari Cyberpunk 2077 yang bernama Breach Protocol. Dari permainan ini, pemain diminta untuk mencari kombinasi alfanumerik yang identik dengan solusi alfanumerik yang ada untuk mendapatkan poin. Lebih banyak solusi yang didapat, lebih banyak poin yang didapatkan.

Cyberpunk 2077 Breach Protocol bisa dibuat simulasinya dengan menggunakan Algoritma *Brute Force*. Ini artinya program akan mencari semua kombinasi alfanumerik yang mungkin dari matriks yang disediakan dan dicocokan dengan solusi alfanumerik-nya untuk dicari nilainya. Program yang dibuat akan memberikan solusi yang paling optimal yaitu yang kombinasi alfanumeriknya paling pendek dan poinnya yang paling besar. Beginilah penjelasan singkat akan aplikasi algoritma *Brute Force* terhadap Cyberpunk 2077 Breach Protocol.

1. Program akan mulai dengan memproses elemen yang paling awal lalu ditambahkan ke list dalam bentuk koordinat. Lalu program akan memproses elemen selanjutnya secara vertikal. Elemen tersebut akan dimasukkan ke list berisi koordinat lalu program akan memproses elemen selanjutnya secara horizontal. Perlu diketahui bahwa jika terdapat koordinat yang sama dengan yang ada list, elemen tersebut akan dihindari dan program akan lanjut melakukan iterasi ke elemen yang lain.
2. Jika list sudah penuh dan elemen terakhir sudah mencapai ambang matriks (misalnya elemen 6,6), program akan menghapus elemen terakhir tersebut dan jika masih ada elemen terakhir yang mencapai ambang, program akan menghapusnya lagi. Setelah itu, program akan melakukan *backtrack* untuk mencari kombinasi lain dan iterasi yang baru akan dimulai.
3. Semua kemungkinan list yang ada akan dimasukkan ke dalam sebuah list (list berisi list). Setelah semuanya selesai, list berisi list tersebut akan berisi semua kemungkinan list koordinat yang akan dikonversi menjadi token di matriks (koordinatnya (1,1) maka akan dikonversi menjadi token elemen (0,0).
4. List akan diproses dan dicari solusi yang paling optimal untuk diberi informasinya ke terminal.

# BAB II SOURCE CODE

Tucil ini diimplementasikan dengan menggunakan Bahasa pemrograman Java dengan compiler versi 20+. Berikut adalah fungsi dan prosedur yang diimplementasikan dalam program ini.

1. first\_method()
2. second\_method()
3. main(String[] args)
4. Read(String path, Matrix M, Matrix N)
5. Write(ListString liststring, int prize, ListDyn listcoor)
6. ListDyn(int size)
7. getnEff()
8. getsize()
9. getprize()
10. setprize(int val)
11. getElmt\_Dyn(int index)
12. setElmt\_Dyn(int index, Point point)
13. InsertLast\_Dyn(Point point)
14. IsIn\_Dyn(Point point)
15. DeleteLast\_Dyn()
16. DisplayList\_Dyn()
17. DisplayList\_Dyn\_Resultado()
18. copyList\_Dyn(List\_Dyn list)
19. ListListDyn(int size)
20. getnEff\_list()
21. getsize\_list()
22. setElmt\_ListDyn(int index, ListDyn liste)
23. InsertLast\_ListDyn(ListDyn list)
24. IsIn\_ListDyn(ListDyn list)
25. DeleteLast\_ListDyn()
26. DisplayList\_ListDyn()
27. getElmt\_ListDyn(int n)
28. ListString(int size)
29. getnEff\_String()
30. getsize\_String()
31. getprize\_String()
32. setprize\_String(int val)
33. addprize\_String(int val)
34. getElmt\_String(int index)
35. setElmt\_String(int index, String string)
36. InsertLast\_String(String string)
37. IsIn\_String(String string)
38. DeleteLast\_String()
39. DisplayList\_String()
40. copyLIst\_String()
41. slice\_ListString(ListString list)
42. isEqual\_String\_Beginning(ListString list)
43. isSubset\_String(ListString list1, ListString list2)
44. compare\_ListString(ListString list2)
45. Matrix(int row, int col)
46. Row()
47. Col()
48. setRow(int newRow)
49. setCol(int newCol)
50. getElmt(int row, int col)
51. setElmt(int row, int col, String val)
52. getPoint(int row)
53. setPoint(int row, int rep)
54. setPointSize(int size)
55. fillEmpty()
56. displayMatrix()
57. displayCoordinate()
58. game\_vertical(int col, ListDyn coor\_list, ListListDyn token\_list)
59. game\_horizontal(int col, ListDyn coor\_list, ListListDyn token\_list)
60. toString(char[] array)
61. generateToken()
62. generateMatrix(int row, int col, String[] strings)
63. Point()
64. Point(int x, int y)
65. createPoint(int x, int y)
66. getX()
67. getY()
68. setX(int x)
69. setY(int y)
70. displayPoint()

Berikut ini adalah source code tucil ini.

|  |
| --- |
| Nama file: Driver.java  import java.util.**\***;  import java.io.**\***;  import java.lang.Math;  public class Driver {      public static final String black = "\u001B[30m";      public static final String red = "\u001B[31m";      public static final String green = "\u001B[32m";      public static final String yellow = "\u001B[33m";      public static final String blue = "\u001B[34m";      public static final String purple = "\u001B[35m";      public static final String cyan = "\u001B[36m";      public static final String white = "\u001B[37m";      public static final String reset = "\u001B[0m";      static int token\_count = 0;      static Matrix newMatrix = new Matrix(1,1);      public static void first\_method() {          System.out.println("WARNING: File yang diambil adalah dari folder test (test.txt, maka test/test.txt akan diproses)!");          System.out.print("Masukkan nama file untuk diproses (Format adalah \*.txt): ");          Scanner file\_name\_inpute = new Scanner(System.in);          String file\_name = file\_name\_inpute.nextLine();          Matrix M = new Matrix(1000,1000);          Matrix N = new Matrix(1000,1000);          FileProcess.Read(file\_name,M,N);          ListDyn coor\_liste = new ListDyn(N.Col());          ListListDyn token\_list = new ListListDyn(5);          long startTime = System.currentTimeMillis();            try {              for (int i = 0; i < 1; i++) {                  M.game\_horizontal(i, coor\_liste, token\_list);              }          } catch (ArrayIndexOutOfBoundsException *e*) {              System.out.print("");              return;          }          ListString[] possible\_tokens = new ListString[token\_list.getnEff\_list()];          for (int i = 0; i < possible\_tokens.length; i++) {              possible\_tokens[i] = new ListString(token\_list.getElmt\_ListDyn(i).getnEff());              for (int j = 0; j < possible\_tokens[i].getsize\_String(); j++) {                  possible\_tokens[i].InsertLast\_String(M.getElmt(token\_list.getElmt\_ListDyn(i).getElmt\_Dyn(j).getX()-1, token\_list.getElmt\_ListDyn(i).getElmt\_Dyn(j).getY()-1));              }          }          ListString[] possible\_tokens\_2 = new ListString[token\_list.getnEff\_list()];          for (int i = 0; i < possible\_tokens\_2.length; i++) {              possible\_tokens\_2[i] = new ListString(token\_list.getElmt\_ListDyn(i).getnEff());              possible\_tokens\_2[i].copyList\_String(possible\_tokens[i]);          }          ListString[] prize\_tokens = new ListString[N.Row()];          for (int i = 0; i < prize\_tokens.length; i++) {              prize\_tokens[i] = new ListString(N.Col());              for (int j = 0; j < N.Col(); j++) {                  if (N.getElmt(i, j) != null) {                      prize\_tokens[i].InsertLast\_String(N.getElmt(i, j));                  } else {                      break;                  }              }              prize\_tokens[i].setprize\_String(N.getPoint(i));          }          for (int i = 0; i < possible\_tokens.length; i++) {              for (int j = 0; j < prize\_tokens.length; j++) {                  if (possible\_tokens\_2[i].isSubset\_String(possible\_tokens\_2[i],prize\_tokens[j])) {                      possible\_tokens[i].addprize\_String(prize\_tokens[j].getprize\_String());                  }                  possible\_tokens\_2[i].copyList\_String(possible\_tokens[i]);              }          }          long endTime = System.currentTimeMillis();          ListString max\_ListString = new ListString(N.Col());          int max\_idx = 0;          max\_ListString.copyList\_String(possible\_tokens[0]);          for (int i = 0; i < possible\_tokens.length; i++) {              if(!max\_ListString.compare\_ListString(possible\_tokens[i])) {                  max\_ListString.copyList\_String(possible\_tokens[i]);                  max\_ListString.setprize\_String(possible\_tokens[i].getprize\_String());                  max\_idx = i;              }          }          if (possible\_tokens[max\_idx].getprize\_String() == 0) {              System.out.print("Tidak ada poin paling optimal. Maka hasilnya adalah 0: ");              System.out.print(possible\_tokens[max\_idx].getprize\_String());              System.out.println();          } else {              System.out.print("Sekuens Optimal: ");              possible\_tokens[max\_idx].DisplayList\_String();              System.out.println();              System.out.print("Poin Optimal: ");              System.out.print(possible\_tokens[max\_idx].getprize\_String());              System.out.println();              System.out.println("Koordinat Optimal: ");              token\_list.getElmt\_ListDyn(max\_idx).DisplayList\_Dyn\_Resultado();              System.out.println();          }          long duration = endTime-startTime;          System.out.print(duration);          System.out.println(" ms");          FileProcess.Write(max\_ListString, possible\_tokens[max\_idx].getprize\_String(), token\_list.getElmt\_ListDyn(max\_idx), duration);      }      public static void second\_method() {          Random r = new Random();          System.out.print("Masukkan jumlah token unik: ");          int unique\_token = (new Scanner(System.in)).nextInt();          String[] token = new String[unique\_token];          String[] token\_inpute;          while (true) {              System.out.print("Masukkan token-token buffer: ");              token\_inpute = (new Scanner(System.in)).nextLine().trim().split(" ");              int count = 0;              for (int i = 0; i < token\_inpute.length; i++) {                  if (token\_inpute[i].length() == 2) {                      count += 1;                  }              }              if (count == token\_inpute.length) {                  break;              } else {                  System.out.println("Jumlah digit setiap token harus 2!");              }          }          for (int i = 0; i < unique\_token; i++) {              token[i] = token\_inpute[i];          }          System.out.print("Masukkan ukuran buffer: ");          int buffer\_size = (new Scanner(System.in)).nextInt();          System.out.print("Masukkan ukuran matriks: ");          String[] matrix\_row\_col = (new Scanner(System.in)).nextLine().trim().split(" ");          int matrix\_row\_size = Integer.parseInt(matrix\_row\_col[1]);          int matrix\_col\_size = Integer.parseInt(matrix\_row\_col[0]);          System.out.print("Masukkan jumlah sekuens: ");          int jumlah\_sekuens = (new Scanner(System.in)).nextInt();          System.out.print("Masukkan ukuran maksimal sekuens: ");          int ukuran\_maksimal\_sekuens = (new Scanner(System.in)).nextInt();          Matrix matrix\_token = new Matrix(matrix\_row\_size, matrix\_col\_size);          matrix\_token.generateMatrix(matrix\_row\_size, matrix\_col\_size, token);          matrix\_token.displayMatrix();          System.out.println();          ListString[] prize\_token = new ListString[jumlah\_sekuens];          int rand;          for (int i = 0; i < jumlah\_sekuens; i++) {              rand = r.nextInt(ukuran\_maksimal\_sekuens-1)+2;              prize\_token[i] = new ListString(rand);              while (true) {                  for (int j = 0; j < rand; j++) {                      prize\_token[i].InsertLast\_String(token[r.nextInt(token.length)]);                  }                  int count = 0;                  for (int k = 0; k < prize\_token.length; k++) {                      if (prize\_token[k] != null) {                          if (prize\_token[i].isEqual\_String\_Beginning(prize\_token[i], prize\_token[k]) && i != k) {                              count -= 1;                          }                      }                  }                  if (count == 0) {                      break;                  } else {                      prize\_token[i] = new ListString(rand);                  }              }              prize\_token[i].setprize\_String(r.nextInt(1001));              int rando = r.nextInt(3);              if (rando == 2) {                  prize\_token[i].setprize\_String(-1\*prize\_token[i].getprize\_String());              }              prize\_token[i].DisplayList\_String();              System.out.println();              System.out.print(prize\_token[i].getprize\_String());              System.out.println();          }          ListDyn coor\_liste = new ListDyn(buffer\_size);          ListListDyn token\_list = new ListListDyn(5);          matrix\_token.setCol(matrix\_col\_size);          matrix\_token.setRow(matrix\_row\_size);            long startTime = System.currentTimeMillis();          for (int i = 0; i < 1; i++) {              matrix\_token.game\_horizontal(i, coor\_liste, token\_list);          }          ListString[] possible\_tokens = new ListString[token\_list.getnEff\_list()];          for (int i = 0; i < possible\_tokens.length; i++) {              possible\_tokens[i] = new ListString(token\_list.getElmt\_ListDyn(i).getnEff());              for (int j = 0; j < possible\_tokens[i].getsize\_String(); j++) {                  possible\_tokens[i].InsertLast\_String(matrix\_token.getElmt(token\_list.getElmt\_ListDyn(i).getElmt\_Dyn(j).getX()-1, token\_list.getElmt\_ListDyn(i).getElmt\_Dyn(j).getY()-1));              }          }          ListString[] possible\_tokens\_2 = new ListString[token\_list.getnEff\_list()];          for (int i = 0; i < possible\_tokens\_2.length; i++) {              possible\_tokens\_2[i] = new ListString(token\_list.getElmt\_ListDyn(i).getnEff());              possible\_tokens\_2[i].copyList\_String(possible\_tokens[i]);          }          for (int i = 0; i < possible\_tokens.length; i++) {              for (int j = 0; j < prize\_token.length; j++) {                  if (possible\_tokens\_2[i].isSubset\_String(possible\_tokens\_2[i],prize\_token[j])) {                      possible\_tokens[i].addprize\_String(prize\_token[j].getprize\_String());                  }                  possible\_tokens\_2[i].copyList\_String(possible\_tokens[i]);              }          }          long endTime = System.currentTimeMillis();          ListString max\_ListString = new ListString(buffer\_size);          int max\_idx = 0;          max\_ListString.copyList\_String(possible\_tokens[0]);          for (int i = 0; i < possible\_tokens.length; i++) {              if(!max\_ListString.compare\_ListString(possible\_tokens[i])) {                  max\_ListString.copyList\_String(possible\_tokens[i]);                  max\_ListString.setprize\_String(possible\_tokens[i].getprize\_String());                  max\_idx = i;              }          }          if (possible\_tokens[max\_idx].getprize\_String() == 0) {              System.out.print("Tidak ada poin paling optimal. Maka hasilnya adalah 0: ");              System.out.print(possible\_tokens[max\_idx].getprize\_String());              System.out.println();          } else {              System.out.print("Sekuens Optimal: ");              possible\_tokens[max\_idx].DisplayList\_String();              System.out.println();              System.out.print("Poin Optimal: ");              System.out.print(possible\_tokens[max\_idx].getprize\_String());              System.out.println();              System.out.println("Koordinat Optimal: ");              token\_list.getElmt\_ListDyn(max\_idx).DisplayList\_Dyn\_Resultado();              System.out.println();          }          long duration = endTime-startTime;          System.out.print(duration);          System.out.println(" ms");          FileProcess.Write(max\_ListString, possible\_tokens[max\_idx].getprize\_String(), token\_list.getElmt\_ListDyn(max\_idx), duration);      }    *// Main*      public static void main(String[] *args*) {          while (true) {              System. out. print("\033[H\033[2J");              System.out.flush();              System.out.println("=====================================");              System.out.println("|| Masukkan metode yang diinginkan ||");              System.out.println("=====================================");              System.out.println("|| 1. Matrix From File             ||");              System.out.println("|| 2. Random Generated Matrix      ||");              System.out.println("|| Other: Exit                     ||");              System.out.println("=====================================");              System.out.println();              System.out.print("Input: ");              int method;              try {                  method = (new Scanner(System.in)).nextInt();              } catch (InputMismatchException *e*) {                  System. out. print("\033[H\033[2J");                  System.out.flush();                  System.out.println("=====================================");                  System.out.println("||          Terima kasih!!         ||");                  System.out.println("=====================================");                  break;              }              if (method == 1) {                  first\_method();              } else if (method == 2) {                  second\_method();              } else {                  System. out. print("\033[H\033[2J");                  System.out.flush();                  System.out.println("=====================================");                  System.out.println("||          Terima kasih!!         ||");                  System.out.println("=====================================");                  break;              }          }      }  } |
| Nama file: FileProcess.java  import java.io.**\***;  import java.util.**\***;  public class FileProcess {      static Scanner scInt = new Scanner(System.in);      static Scanner scFlt = new Scanner(System.in);      static Scanner scStr = new Scanner(System.in);      public static void Read(String *path*, Matrix *M*, Matrix *N*){              try{                  String test = "../test/";                  File f = new File(test.concat(*path*));                  Scanner reader = new Scanner(f);                  if (reader.hasNextLine()) {                      String buffer\_size\_string = reader.nextLine();                      int buffer\_size = Integer.parseInt(buffer\_size\_string);  *N*.setCol(buffer\_size);                  }                  if (reader.hasNextLine()) {                      String[] matrix\_size\_string = reader.nextLine().trim().split(" ");                      int matrix\_row = Integer.parseInt(matrix\_size\_string[0]);                      int matrix\_col = Integer.parseInt(matrix\_size\_string[1]);  *M*.setCol(matrix\_row);  *M*.setRow(matrix\_col);                  }                  for (int i = 0; i < *M*.Row(); i++) {                      if (reader.hasNextLine()) {                          String[] alpha\_numeric\_string = reader.nextLine().trim().split(" ");                          for(int j = 0; j < *M*.Col(); j++) {  *M*.setElmt(i,j,alpha\_numeric\_string[j]);                          }                          for (int k = 0; k < *M*.Row(); k++) {                              for (int l = 0; l < *M*.Col(); l++) {                                  if (*M*.getElmt(k, l) != null) {                                      if (*M*.getElmt(k, l).length() != 2) {                                          System.out.println("Terdapat elemen berdigit tidak sama dengan 2! Menutup program...");                                          System.exit(0);                                      }                                  }                              }                          }                      }                  }                  if (reader.hasNextLine()) {                      String point\_size\_string = reader.nextLine();                      int point\_size = Integer.parseInt(point\_size\_string);  *N*.setRow(point\_size);  *N*.setPointSize(point\_size);                  }                  for (int i = 0; i < *N*.Row(); i++) {                      if (reader.hasNextLine()) {                          String[] buffer\_token\_string = reader.nextLine().trim().split(" ");                          for (int j = 0; j < buffer\_token\_string.length; j++) {  *N*.setElmt(i,j,buffer\_token\_string[j]);                          }                          if (reader.hasNextLine()) {  *N*.setPoint(i, Integer.parseInt(reader.nextLine()));                          }                          for (int k = 0; k < *N*.Row(); k++) {                              for (int l = 0; l < *N*.Col(); l++) {                                  if (*N*.getElmt(k, l) != null) {                                      if (*N*.getElmt(k, l).length() != 2) {                                          System.out.println("Terdapat elemen berdigit tidak sama dengan 2! Menutup program...");                                          System.exit(0);                                      }                                  }                              }                          }                      }                  }              } catch (FileNotFoundException *e*) {                  System.out.println("Nama file tidak ditemukan!");              }          }        public static void Write(ListString *liststring*, int *prize*, ListDyn *listcoor*, long *time*) {          System.out.println();          System.out.println("====================================");          System.out.println("||    Simpan output dalam file?   ||");          System.out.println("||--------------------------------||");          System.out.println("|| (Y/y) : YES                    ||");          System.out.println("|| Other Response : NO            ||");          System.out.println("====================================");          System.out.println();          try {              char input = (new Scanner(System.in)).next().charAt(0);              if (input == 'Y' || input == 'y') {                  System.out.print("Masukkan sebuah path: ");                  String path = scStr.nextLine();                  try {                      String test = "../test/";                      FileWriter writer = new FileWriter(test.concat(path));                      for (int i = 0; i < *liststring*.getnEff\_String(); i++) {                          writer.write(*liststring*.getElmt\_String(i));                          if (i != *liststring*.getnEff\_String()-1) {                              writer.write(" ");                          }                      }                      writer.write("\n");                      writer.write(Integer.toString(*prize*));                      writer.write("\n");                      for (int i = 0; i < *listcoor*.getnEff(); i++) {                          writer.write(Integer.toString(*listcoor*.getElmt\_Dyn(i).getY()));                          writer.write(",");                          writer.write(Integer.toString(*listcoor*.getElmt\_Dyn(i).getX()));                          if (i != *liststring*.getnEff\_String()-1) {                              writer.write("\n");                          }                      }                      writer.write("\n");                      writer.write(Integer.toString((int)*time*));                      writer.write(" ms");                      writer.close();                      System.out.println("Berhasil menulis ke file " + path);                  } catch (IOException *e*) {                      System.out.println("Nama file tidak ditemukan!");                  }              }          } catch (InputMismatchException *e*) {              char input = (new Scanner(System.in)).next().charAt(0);          }        }  } |
| Nama file: ListDyn.java  import java.util.**\***;  import java.io.File;  import java.lang.Math;  public class ListDyn {      public Point[] list;      public int nEff;      public int size;      public int prize;      public ListDyn(int *size*) {  **this**.list = new Point[*size*];  **this**.nEff = 0;  **this**.size = *size*;  **this**.prize = 0;      }      public int getnEff() {          return **this**.nEff;      }      public int getsize() {          return **this**.size;      }      public int getprize() {          return **this**.prize;      }      public void setprize(int *val*) {  **this**.prize = *val*;      }      public Point getElmt\_Dyn (int *index*) {          return **this**.list[*index*];      }      public void setElmt\_Dyn (int *index*, Point *point*) {  **this**.list[*index*] = *point*;      }      public void InsertLast\_Dyn(Point *point*) {          if (**this**.size > **this**.nEff) {  **this**.list[**this**.nEff] = *point*;  **this**.nEff += 1;          } else {              ListDyn temp = new ListDyn(**this**.list.length);              for (int i = 0; i < **this**.list.length; i++) {                  temp.list[i] = **this**.list[i];              }              temp.nEff = **this**.nEff;  **this**.list = new Point[temp.nEff\*2];  **this**.nEff = temp.nEff;  **this**.size = temp.nEff\*2;              for (int i = 0; i < temp.nEff; i++) {  **this**.list[i] = temp.list[i];              }  **this**.list[temp.nEff] = *point*;  **this**.nEff += 1;          }      }      public Boolean IsIn\_Dyn(Point *point*) {          Boolean bool = false;          for (int i = 0; i < **this**.nEff; i++) {              if (**this**.list[i] == *point*) {                  bool = true;                  break;              }          }          return bool;      }      public void DeleteLast\_Dyn() {  **this**.nEff -= 1;      }      public void DisplayList\_Dyn() {          for (int i = 0; i < **this**.nEff; i++) {              System.out.print("(");              System.out.print(**this**.list[i].getX());              System.out.print(",");              System.out.print(**this**.list[i].getY());              System.out.print(")");              if (i != **this**.nEff-1) {                  System.out.print(",");              }          }      }      public void DisplayList\_Dyn\_Resultado() {          for (int i = 0; i < **this**.nEff; i++) {              System.out.print(**this**.list[i].getY());              System.out.print(",");              System.out.print(**this**.list[i].getX());              if (i != **this**.nEff-1) {                  System.out.println();              }          }      }      public void copyList\_Dyn(ListDyn *list*) {  **this**.nEff = *list*.nEff;  **this**.size = *list*.size;          for (int i = 0; i < **this**.nEff; i++) {  **this**.list[i] = *list*.list[i];          }      }  } |
| Nama file: ListListDyn.java  import java.util.**\***;  import java.io.File;  import java.lang.Math;  public class ListListDyn {      private ListDyn[] buffer;      private int nEff\_list;      private int size\_list;      public ListListDyn(int *size*) {  **this**.buffer = new ListDyn[*size*];  **this**.nEff\_list = 0;  **this**.size\_list = *size*;      }      public int getnEff\_list() {          return **this**.nEff\_list;      }      public int getsize\_list() {          return **this**.size\_list;      }      public void setElmt\_ListDyn(int *index*, ListDyn *liste*) {  **this**.buffer[*index*] = new ListDyn(*liste*.getnEff());  *// liste.DisplayList\_Dyn();*  *// System.out.println();*  *// for (int i = 0; i < this.buffer[index].getnEff(); i++) {*  *//     this.buffer[index].list[i] = liste.list[i];*  *// }*  **this**.buffer[*index*] = *liste*;      }      public void InsertLast\_ListDyn(ListDyn *list*) {          if (**this**.size\_list > **this**.nEff\_list) {  **this**.setElmt\_ListDyn(**this**.nEff\_list, *list*);  **this**.nEff\_list += 1;          } else {              ListListDyn temp = new ListListDyn(**this**.buffer.length);              for (int i = 0; i < **this**.buffer.length; i++) {                  temp.buffer[i] = **this**.buffer[i];              }              temp.nEff\_list = **this**.nEff\_list;  **this**.buffer = new ListDyn[temp.nEff\_list\*2];  **this**.nEff\_list = temp.nEff\_list;  **this**.size\_list = temp.nEff\_list\*2;              for (int i = 0; i < temp.nEff\_list; i++) {  **this**.buffer[i] = temp.buffer[i];              }  **this**.setElmt\_ListDyn(**this**.nEff\_list, *list*);  **this**.nEff\_list += 1;          }      }      public Boolean IsIn\_ListDyn(ListDyn *list*) {          Boolean bool = false;          for (int i = 0; i < **this**.nEff\_list; i++) {              if (**this**.buffer[i] == *list*) {                  bool = true;                  break;              }          }          return bool;      }      public void DeleteLast\_ListDyn() {  **this**.nEff\_list -= 1;      }      public void DisplayList\_ListDyn() {          for (int i = 0; i < **this**.nEff\_list; i++) {  **this**.buffer[i].DisplayList\_Dyn();              if (i != **this**.nEff\_list-1) {                  System.out.print(",");              }              System.out.println();          }      }      public ListDyn getElmt\_ListDyn(int *n*) {          return **this**.buffer[*n*];      }  } |
| Nama file: ListString.java  import java.util.**\***;  import java.io.File;  import java.lang.Math;  public class ListString {      private String[] string\_content;      private int nEff\_string;      private int size\_string;      private int prize\_string;      public ListString(int *size*) {  **this**.string\_content = new String[*size*];  **this**.nEff\_string = 0;  **this**.size\_string = *size*;  **this**.prize\_string = 0;      }      public int getnEff\_String() {          return **this**.nEff\_string;      }      public int getsize\_String() {          return **this**.size\_string;      }      public int getprize\_String() {          return **this**.prize\_string;      }      public void setprize\_String(int *val*) {  **this**.prize\_string = *val*;      }      public void addprize\_String(int *val*) {  **this**.prize\_string += *val*;      }      public String getElmt\_String(int *index*) {          return **this**.string\_content[*index*];      }      public void setElmt\_String (int *index*, String *string*) {  **this**.string\_content[*index*] = *string*;      }      public void InsertLast\_String (String *string*) {          if (**this**.nEff\_string < **this**.size\_string) {  **this**.setElmt\_String(**this**.nEff\_string, *string*);  **this**.nEff\_string += 1;          } else {              ListString temp = new ListString(**this**.size\_string);              for (int i = 0; i < **this**.nEff\_string; i++) {                  temp.string\_content[i] = **this**.string\_content[i];              }              temp.nEff\_string = **this**.nEff\_string;  **this**.string\_content = new String[temp.size\_string\*2];  **this**.nEff\_string = temp.nEff\_string;  **this**.size\_string = temp.size\_string\*2;              for (int i = 0; i < temp.nEff\_string; i++) {  **this**.string\_content[i] = temp.string\_content[i];              }  **this**.string\_content[temp.nEff\_string] = *string*;  **this**.nEff\_string += 1;          }      }      public Boolean IsIn\_String(String *string*) {          Boolean bool = false;          for (int i = 0; i < **this**.nEff\_string; i++) {              if (**this**.string\_content[i] == *string*) {                  bool = true;                  break;              }          }          return bool;      }      public void DeleteLast\_String() {  **this**.nEff\_string -= 1;      }      public void DisplayList\_String() {          for (int i = 0; i < **this**.nEff\_string; i++) {              System.out.print(**this**.string\_content[i]);              System.out.print(" ");          }      }      public void copyList\_String(ListString *list*) {  **this**.nEff\_string = *list*.nEff\_string;  **this**.size\_string = *list*.size\_string;          for (int i = 0; i < **this**.nEff\_string; i++) {  **this**.string\_content[i] = *list*.string\_content[i];          }      }      public ListString slice\_ListString (ListString *list*) {          for (int i = 0; i < *list*.getnEff\_String()-1; i++) {  *list*.setElmt\_String(i, *list*.getElmt\_String(i+1));          }  *list*.DeleteLast\_String();          return *list*;      }      public Boolean isEqual\_String\_Beginning(ListString *list1*, ListString *list2*) {          Boolean bool = true;          if (*list1*.nEff\_string >= *list2*.nEff\_string) {              for (int i = 0; i < *list2*.nEff\_string; i++) {                  if (*list1*.getElmt\_String(i).equals(*list2*.getElmt\_String(i)) == false) {                      bool = false;                      break;                  }              }          } else {              bool = false;          }          return bool;      }      public Boolean isSubset\_String(ListString *list1*, ListString *list2*) {          if (*list1*.getnEff\_String() == 0) {              return false;          } else {              if (isEqual\_String\_Beginning(*list1*, *list2*)) {                  return true;              } else {                  return isSubset\_String(slice\_ListString(*list1*), *list2*);              }          }      }      public Boolean compare\_ListString(ListString *list2*) {  *// System.out.print(this.prize\_string);*  *// System.out.print(" ");*  *// System.out.print(list2.prize\_string);*  *// System.out.print(" ");*          if (**this**.prize\_string > *list2*.prize\_string) {  *// System.out.print("TRUE LENGTH");*              return true;          } else if (**this**.prize\_string == *list2*.prize\_string) {              if (**this**.nEff\_string <= *list2*.nEff\_string) {  *// System.out.print("TRUE LENGTH");*                  return true;              } else {  *// System.out.print("FALSE LENGTH");*                  return false;              }          } else {  *// System.out.print("FALSE VALUE");*              return false;          }      }  } |
| Nama file: Matrix.java  import java.util.**\***;  import java.io.File;  import java.lang.Math;  public class Matrix {      private String[][] content;      private Point[][] coordinate = new Point[1][1];      private int row;      private int col;      private int[] points;      public Matrix(int *row*, int *col*) {  **this**.row = *row*;  **this**.col = *col*;  **this**.content = new String[*row*][*col*];  **this**.coordinate = new Point[*row*][*col*];          for (int i = 0; i < **this**.row; i++) {              for (int j = 0; j < **this**.col; j++) {  **this**.coordinate[i][j] = new Point(i+1,j+1);              }          }      }      public int Row() {          return **this**.row;      }      public int Col() {          return **this**.col;      }      public void setRow(int *newRow*) {  **this**.row = *newRow*;      }      public void setCol(int *newCol*) {  **this**.col = *newCol*;      }      public String getElmt(int *row*, int *col*) {          return **this**.content[*row*][*col*];      }      public void setElmt(int *row*, int *col*, String *val*) {  **this**.content[*row*][*col*] = *val*;      }      public int getPoint(int *row*) {          return **this**.points[*row*];      }      public void setPoint(int *row*, int *rep*) {  **this**.points[*row*] = *rep*;      }      public void setPointSize(int *size*) {  **this**.points = new int[*size*];      }      public void fillEmpty() {          for (int i = 0; i < **this**.row; i++) {              for (int j = 0; j < **this**.col; j++) {  **this**.content[i][j] = "--";              }          }      }      public void displayMatrix() {          for (int i = 0; i < **this**.row; i++) {              for (int j = 0; j < **this**.col; j++) {                  System.out.print(**this**.content[i][j]);                  System.out.print(" ");              }              System.out.println();          }      }      public void displayCoordinate() {          for (int i = 0; i < **this**.row; i++) {              for (int j = 0; j < **this**.col; j++) {  **this**.coordinate[i][j].displayPoint();              }              System.out.println();          }      }      public void game\_vertical(int *col*, ListDyn *coor\_list*, ListListDyn *token\_list*) {          for (int i = 0; i < **this**.row; i++) {              if (!*coor\_list*.IsIn\_Dyn(**this**.coordinate[i][*col*]) && *coor\_list*.getnEff() < *coor\_list*.getsize()) {                  ListDyn new\_list = new ListDyn(*coor\_list*.getsize());                  ListDyn new\_list\_copy = new ListDyn(new\_list.getsize());                  new\_list.copyList\_Dyn(*coor\_list*);                  new\_list.InsertLast\_Dyn(**this**.coordinate[i][*col*]);                  new\_list\_copy.copyList\_Dyn(new\_list);  *token\_list*.InsertLast\_ListDyn(new\_list\_copy);                  game\_horizontal(i, new\_list, *token\_list*);                  new\_list.DeleteLast\_Dyn();              } else {                  try {                      if (!*coor\_list*.IsIn\_Dyn(**this**.coordinate[row][i]) && *coor\_list*.getnEff() == *coor\_list*.getsize()) {                          break;                      }                  } catch (ArrayIndexOutOfBoundsException *e*) {  *// Nothing*                  }              }          }      }      public void game\_horizontal(int *row*, ListDyn *coor\_list*, ListListDyn *token\_list*) {          for (int i = 0; i < **this**.col; i++) {              if (!*coor\_list*.IsIn\_Dyn(**this**.coordinate[*row*][i]) && *coor\_list*.getnEff() < *coor\_list*.getsize()) {                  ListDyn new\_list = new ListDyn(*coor\_list*.getsize());                  ListDyn new\_list\_copy = new ListDyn(new\_list.getsize());                  new\_list.copyList\_Dyn(*coor\_list*);                  new\_list.InsertLast\_Dyn(**this**.coordinate[*row*][i]);                  new\_list\_copy.copyList\_Dyn(new\_list);  *token\_list*.InsertLast\_ListDyn(new\_list\_copy);                  game\_vertical(i, new\_list, *token\_list*);                  new\_list.DeleteLast\_Dyn();              } else {                  try {                      if (!*coor\_list*.IsIn\_Dyn(**this**.coordinate[*row*][i]) && *coor\_list*.getnEff() == *coor\_list*.getsize()) {                          break;                      }                  } catch (ArrayIndexOutOfBoundsException *e*) {  *// Nothing*                  }              }          }      }      public String toString(char[] *array*) {          String string = new String(*array*);          return string;      }      public String generateToken() {          Random r = new Random();          char[] token = new char[2];          String upper = "AZB0CYD9EXF1GWV8UHT2SIR7QJP3OKL6M4N5";          for (int i = 0; i < token.length; i++) {              token[i] = upper.charAt(r.nextInt(upper.length()));          }          return toString(token);      }      public void generateMatrix(int *row*, int *col*, String[] *strings*) {          for (int i = 0; i < *row*; i++) {              for (int j = 0; j < *col*; j++) {  **this**.content[i][j] = *strings*[(new Random()).nextInt(*strings*.length)];              }          }      }  } |
| Nama file : Point.java  import java.util.**\***;  import java.io.File;  import java.lang.Math;  public class Point {      private int x;      private int y;      public Point() {  **this**.x = 0;  **this**.y = 0;      }      public Point(int *x*, int *y*) {  **this**.x = *x*;  **this**.y = *y*;      }      public void createPoint(int *x*, int *y*) {  **this**.x = *x*;  **this**.y = *y*;      }      public int getX() {          return **this**.x;      }      public int getY() {          return **this**.y;      }      public void setX(int *x*) {  **this**.x = *x*;      }      public void setY(int *y*) {  **this**.y = *y*;      }      public void displayPoint() {          System.out.print(**this**.x);          System.out.print(",");          System.out.print(**this**.y);          System.out.print(" ");      }  } |

# BAB III SCREENSHOT HASIL TEST

1. Input : 1\_in.png

A screenshot of a computer code

Description automatically generated

Output : 1\_out.png

A screenshot of a computer

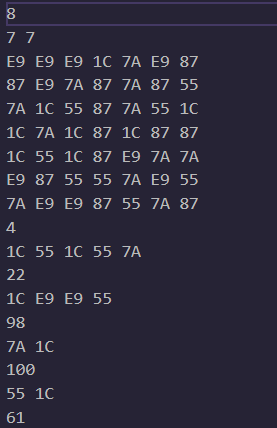
Description automatically generated

1. Input : 2\_in.png  
   A screenshot of a computer

   Description automatically generated

Output : 2\_out.png  
A screenshot of a computer

Description automatically generated

1. Input : 3\_in.png  
   

Output : 3\_out.png  
A screenshot of a computer

Description automatically generated

1. Input : 4\_in.png (Input ini juga digunakan untuk cek kesamaan 2 metode)  
   A screenshot of a computer

   Description automatically generated

Output : 4\_out\_1.png & 4\_out\_2.png  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Input : 5\_in.png (Input ini juga digunakan untuk cek kesamaan 2 metode)  
   A screenshot of a computer

   Description automatically generated

Output : 5\_out\_1.png & 5\_out\_2.png  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Input : 6\_in.png (Input ini juga digunakan untuk cek kesamaan 2 metode)  
   A screenshot of a computer screen

   Description automatically generated

Output : 6\_out\_1.png & 6\_out\_2.png  
A screenshot of a computer

Description automatically generated  
A screen shot of a computer

Description automatically generated

# LINK REPOSITORY

<https://github.com/scifo04/Tucil_Stima_IF2211_01>

# CHECKLIST

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program dapat dikompilasi tanpa kesalahan | ✓ |  |
| 1. Program berhasil dijalankan | ✓ |  |
| 1. Program dapat membaca berkas .txt | ✓ |  |
| 1. Program dapat menghasilkan masukan secara acak | ✓ |  |
| 1. Solusi yang diberikan program optimal | ✓ |  |
| 1. Program dapat menyimpan solusi dalam berkas .txt | ✓ |  |
| 1. Program memiliki GUI |  | ✓ |