**LAPORAN RESMI 15.2**

**PRAKTIKUM ALGORITMA DAN STRUKTUR DATA**

GRAPH DJIKSTRA



Nama : Aqilah Akmalia Dewi

Kelas : 1 D4 IT B

NRP : 3120600046

PROGRAM STUDI D4 TEKNIK INFORMATIKA

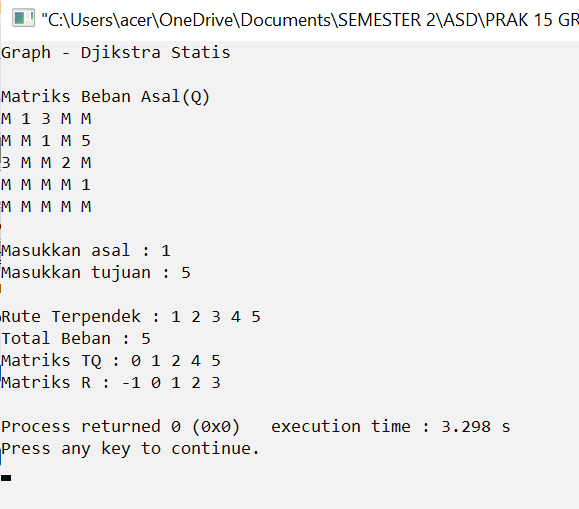
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

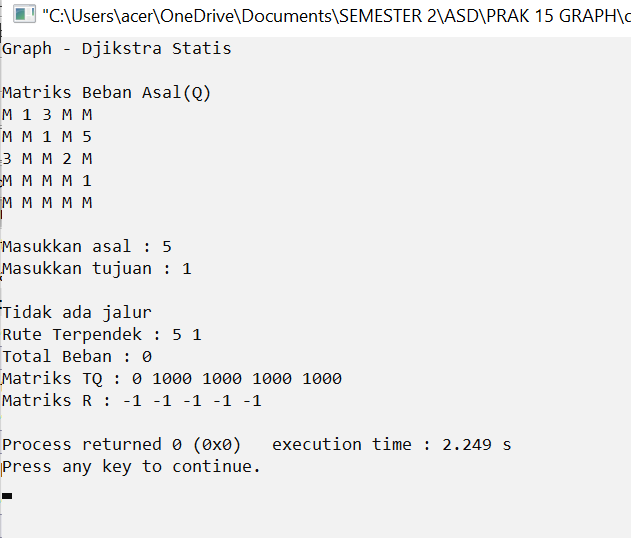
2021

# 1. DJIKSTRA STATIS

1. Listing Program

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define N 5  #define M 1000  #define MAX 30  typedef struct{  int item[MAX];  int front;  int rear;  int count;  }queue;  queue antrian;  typedef int itemType;  typedef struct{  itemType data[MAX];  int count;  }stack;  stack tumpukan;  int asal, tujuan;  int main(){  char jawab;  int Q[N][N] = {M,1,3,M,M,  M,M,1,M,5,  3,M,M,2,M,  M,M,M,M,1,  M,M,M,M,M};  int TQ[N] = {0,M,M,M,M};  int R[N] = {-1,-1,-1,-1,-1};  puts("Graph - Djikstra Statis\n");  tampil(Q, "Matriks Beban Asal(Q)");  printf("Masukkan asal : ");  scanf("%d", &asal);  printf("Masukkan tujuan : ");  scanf("%d", &tujuan);  fflush(stdin);  puts("");  djikstra(Q, TQ, R);  baca(TQ, R);  }  void djikstra(int Q[N][N], int TQ[N], int R[N]){  int currentNode, i;  inisialisasiQ(&antrian);  enqueue(asal, &antrian);  if(TQ[tujuan]==M){  printf("Tidak ada jalur\n");  }else{  while(!kosongQ(&antrian)){  currentNode = dequeue(&antrian)-1;  i = 0;  while(i<N){  if(Q[currentNode][i] != M){  if((Q[currentNode][i] + TQ[currentNode]) < TQ[i]){  TQ[i] = Q[currentNode][i] + TQ[currentNode];  R[i] = currentNode;  if(i != asal-1 && i != tujuan-1 && (cari(i+1,&antrian)==0))  enqueue(i+1,&antrian);  }  }  i++;  }  }  }  }  void cari(int a, queue \*q){  int x, y, data;  x = q->front;  y = q->count;  while(y!=0){  data = q->item[x];  x = (x+1) % MAX;  y--;  if (a == data){  return 1;  }else  return 0;  }  }  void baca(int TQ[N], int R[N]){  int x,y,i;  x=asal;  y=tujuan;  inisialisasiS(&tumpukan);  push(tujuan,&tumpukan);  for(int i=y-1;i>=x-1;i--){  if(R[i]!=x-1){  push(R[i]+1,&tumpukan);  }else{  break;  }  }  push(asal,&tumpukan);  printf("Rute Terpendek : ");  while(!kosongS(&tumpukan)){  printf("%d ",pop(&tumpukan));  }  puts(" ");  printf("Total Beban : %d\n",TQ[tujuan-1]);  printf("Matriks TQ : ");  for(i=0; i<N; i++){  printf("%d ", TQ[i]);  }  printf("\nMatriks R : ");  for(i=0; i<N; i++){  printf("%d ", R[i]);  }  puts("");  }  void tampil(int data[N][N], char \*judul){  printf("%s \n",judul);  for(int i=0; i<N; i++) {  for(int j=0; j<N; j++)  if(data[i][j] >= M)  printf("M ");  else  printf("%d ", data[i][j]);  printf("\n");  }  printf("\n");  }  void inisialisasiQ(queue \*q){  q->count = 0;  q->front = 0;  q->rear = 0;  }  int kosongQ(queue \*q){  if(q->count == 0)  return 1;  else  return 0;  }  int penuhQ(queue \*q){  if(q->count == MAX)  return 1;  else  return 0;  }  void enqueue(int x, queue \*q){  if(penuhQ(q))  printf("Queue penuh, tidak bisa menyimpan data\n");  else{  q->item[q->rear] = x;  q->rear = (q->rear + 1) % MAX;  (q->count)++;  }  }  int dequeue(queue \*q){  int ambil;  if(kosongQ(q))  printf("Queue kosong, tidak bisa mengambil data\n");  else{  ambil = q->item[q->front];  q->front = (q->front + 1) % MAX;  (q->count)--;  }  return(ambil);  }  void inisialisasiS(stack \*s){  s->count = 0;  }  int kosongS (stack \*s){  if(s->count == 0)  return (1);  else  return (0);  }  int penuhS(stack \*s){  if(s->count == MAX)  return (1);  else  return (0);  }  void push(itemType x, stack \*s){  if(penuhS(s))  printf("Stack penuh, tidak bisa menyimpan data\n");  else{  s->data[s->count] = x;  s->count++;  }  }  itemType pop(stack \*s){  itemType x;  if(kosongS(s)){  printf("Stack kosong, tidak bisa mengambil data\n");  }else{  --s->count;  x = s->data[s->count];  return x;  }  } |

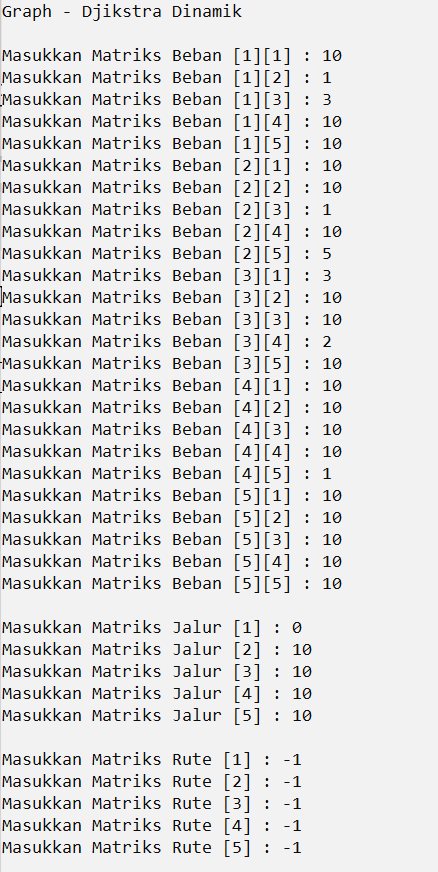
1. Output



# 2. DJIKSTRA DINAMIK

1. Listing Program

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define N 5  #define M 10  #define MAX 30  typedef struct{  int item[MAX];  int front;  int rear;  int count;  }queue;  queue antrian;  typedef int itemType;  typedef struct{  itemType data[MAX];  int count;  }stack;  stack tumpukan;  int asal, tujuan;  int main(){  int Q[N][N];  int TQ[N];  int R[N];  int i, j;  puts("Graph - Djikstra Dinamik\n");  for(i=0; i<N; i++){  for(j=0; j<N; j++){  printf("Masukkan Matriks Beban [%d][%d] : ", i+1,j+1);  scanf("%d", &Q[i][j]);  }  }  puts("");  for(i=0; i<N; i++){  printf("Masukkan Matriks Jalur [%d] : ", i+1);  scanf("%d", &TQ[i]);  }  puts("");  for(i=0; i<N; i++){  printf("Masukkan Matriks Rute [%d] : ", i+1);  scanf("%d", &R[i]);  }  puts("");  tampil(Q, "Matriks Beban Asal(Q)");  printf("Matriks Jalur Asal (TQ)\n");  for(i=0; i<N; i++){  if(TQ[i] >= M)  printf("M ");  else  printf("%d ", TQ[i]);  }  puts("");  puts("");  printf("Matriks Rute Asal(R)\n");  for(i=0; i<N; i++){  printf("%d ",R[i]);  }  puts("");  puts("");  printf("Masukkan asal : ");  scanf("%d", &asal);  printf("Masukkan tujuan : ");  scanf("%d", &tujuan);  puts("");  djikstra(Q, TQ, R);  baca(TQ, R);  }  void djikstra(int Q[N][N], int TQ[N], int R[N]){  int currentNode, i;  inisialisasiQ(&antrian);  enqueue(asal, &antrian);  if(TQ[tujuan]==M){  printf("Tidak ada jalur\n");  }else{  while(!kosongQ(&antrian)){  currentNode = dequeue(&antrian)-1;  i = 0;  while(i<N){  if(Q[currentNode][i] != M){  if((Q[currentNode][i] + TQ[currentNode]) < TQ[i]){  TQ[i] = Q[currentNode][i] + TQ[currentNode];  R[i] = currentNode;  if(i != asal-1 && i != tujuan-1 && (cari(i+1,&antrian)==0))  enqueue(i+1,&antrian);  }  }  i++;  }  }  }  }  void cari(int a, queue \*q){  int x, y, data;  x = q->front;  y = q->count;  while(y!=0){  data = q->item[x];  x = (x+1) % MAX;  y--;  if (a == data){  return 1;  }else  return 0;  }  }  void baca(int TQ[N], int R[N]){  int x,y,i;  x=asal;  y=tujuan;  inisialisasiS(&tumpukan);  push(tujuan,&tumpukan);  for(int i=y-1;i>=x-1;i--){  if(R[i]!=x-1){  push(R[i]+1,&tumpukan);  }else{  break;  }  }  push(asal,&tumpukan);  printf("Rute Terpendek : ");  while(!kosongS(&tumpukan)){  printf("%d ",pop(&tumpukan));  }  puts(" ");  printf("Total Beban : %d\n",TQ[tujuan-1]);  printf("Matriks TQ : ");  for(i=0; i<N; i++){  printf("%d ", TQ[i]);  }  printf("\nMatriks R : ");  for(i=0; i<N; i++){  printf("%d ", R[i]);  }  puts("");  }  void tampil(int data[N][N], char \*judul){  printf("%s \n",judul);  for(int i=0; i<N; i++) {  for(int j=0; j<N; j++)  if(data[i][j] >= M)  printf("M ");  else  printf("%d ", data[i][j]);  printf("\n");  }  printf("\n");  }  void inisialisasiQ(queue \*q){  q->count = 0;  q->front = 0;  q->rear = 0;  }  int kosongQ(queue \*q){  if(q->count == 0)  return 1;  else  return 0;  }  int penuhQ(queue \*q){  if(q->count == MAX)  return 1;  else  return 0;  }  void enqueue(int x, queue \*q){  if(penuhQ(q))  printf("Queue penuh, tidak bisa menyimpan data\n");  else{  q->item[q->rear] = x;  q->rear = (q->rear + 1) % MAX;  (q->count)++;  }  }  int dequeue(queue \*q){  int ambil;  if(kosongQ(q))  printf("Queue kosong, tidak bisa mengambil data\n");  else{  ambil = q->item[q->front];  q->front = (q->front + 1) % MAX;  (q->count)--;  }  return(ambil);  }  void inisialisasiS(stack \*s){  s->count = 0;  }  int kosongS (stack \*s){  if(s->count == 0)  return (1);  else  return (0);  }  int penuhS(stack \*s){  if(s->count == MAX)  return (1);  else  return (0);  }  void push(itemType x, stack \*s){  if(penuhS(s))  printf("Stack penuh, tidak bisa menyimpan data\n");  else{  s->data[s->count] = x;  s->count++;  }  }  itemType pop(stack \*s){  itemType x;  if(kosongS(s)){  printf("Stack kosong, tidak bisa mengambil data\n");  }else{  --s->count;  x = s->data[s->count];  return x;  }  } |

1. Output

