TUGAS MATA KULIAH STRUKTUR DATA PERTEMUAN KE-8

Nama : Afridho Ikhsan
Kelas : 2A - Informatika
NPM : 2210631170002

Kodingan :

```cpp
#include <iostream>
#define maks 4 //Menentukan maksimum muatan queue

using namespace std;

struct Queue //Deklarasi Queue untuk menampung simpul pada graph agar nantinya bisa ditelusuri sesuai dengan yang diinginkan
{
    int rear, front, data[maks];
} verticesQueue;

int n;

bool isFull() // Untuk mengecek apakah Queue yang menampung representasi simpul pada graph sudah penuh atau belum
{
    return verticesQueue.rear == maks;
}

bool isEmpty() // Untuk mengecek apakah Queue yang menampung representasi simpul pada graph masih kosong
{
    return verticesQueue.rear == 0;
}

void enqueue(int insertedVertice) // Untuk menambah elemen baru ke Queue yang menampung representasi simpul pada graph
{
    if (isFull())
    {
        cout << "Antrian Penuh!";
    }
    else
    {
        verticesQueue.data[verticesQueue.rear] = insertedVertice;
        verticesQueue.rear++;
    }
}
```

```
36  int dequeue() // Untuk mengambil sekaligus menghapus simpul pada
    graph di Queue
37  {
38      int temporaryStored;
39
40      if (isEmpty())
41      {
42          cout << "Antrian masih kosong!";
43      }
44      else
45      {
46          temporaryStored = verticesQueue.data[verticesQueue.front];
47          for (int i = verticesQueue.front; i < verticesQueue.rear; i++)
48          {
49              verticesQueue.data[i] = verticesQueue.data[i + 1];
50          }
51          verticesQueue.rear--;
52      }
53      return temporaryStored;
54  }
```

```
56  void cetakgraph(int G[][5], int n) //Untuk mencetak graph (array 2
    dimensi) yang dimasukkan sebagai parameter ke function cetakGraph
    tersebut
57  {
58      cout << "Cost List : " << endl;
59      int i, j;
60      int simpul1 = 1, simpul2 = 1;
61
62      cout << endl;
63      for (i = 0; i < n; i++)
64      {
65          cout << "\t";
66          for (j = 0; j < n; j++)
67          {
68              cout << G[i + 1][j + 1] << "\t";
69          }
70          cout << endl;
71      }
72      cout << endl;
73  }
```

```
75  void BFS(int G[][5], int startVertice, int amoutOfVertices) //Untuk
    melakukan penelusuran graph berdasarkan aturan Breadth First Search
76  {
77      int i = startVertice, j = 1;
78      int temp, tempdata, finalTempData = 0;
79      bool cekSelesai;
80      int visitedArray[5] = {0};
81
82      cout << i << "--->";
83      visitedArray[i] = 1;
84      enqueue(i);
85
86      while (!isEmpty())
87      {
88          i = dequeue();
89          tempdata = 999;
90          for (j = 1; j <= amoutOfVertices; j++)
91          {
92              if (G[i][j] != 0 && visitedArray[j] == 0)
93              {
94                  if (G[i][j] < tempdata)
95                  {
96                      tempdata = G[i][j];
97                      temp = j;
98                  }
99              }
100         }
101
102         for (int k = 1; k <= amoutOfVertices; k++)
103         {
104             if (visitedArray[k] == 1)
105             {
106                 cekSelesai = true;
107             }
108             else
109             {
110                 cekSelesai = false;
111                 break;
112             }
113         }
114
115         if (!cekSelesai)
116         {
117             finalTempData += tempdata;
118             visitedArray[temp] = 1;
119             cout << temp << "--->";
120             enqueue(temp);
121         }
122     }
123     cout << startVertice << endl;
124     cout << "Minimum Cost : " << finalTempData + G[temp]
    [startVertice] << endl;
125 }
```

```
127    int main()
128    {
129        int G[5][5] = {{0, 0, 0, 0, 0},
130                       {0, 0, 4, 1, 3},
131                       {0, 4, 0, 2, 1},
132                       {0, 1, 2, 0, 5},
133                       {0, 3, 1, 5, 0}};
134        cetakgraph(G, 4);
135        cout << "Jalur Terpendek : " << endl;
136        BFS(G, 1, 4);
137
138        return 0;
139    }
```

Output :

1. **Output untuk penulusuran yang dimulai dari simpul 1 :**

```
Cost List :

        0       4       1       3
        4       0       2       1
        1       2       0       5
        3       1       5       0

Jalur Terpendek :
1--->3--->2--->4--->1
Minimum Cost : 7
```

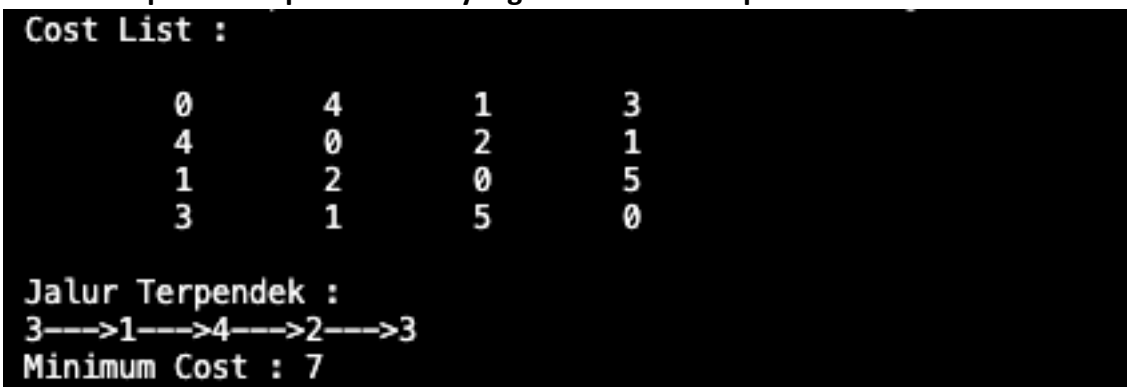
2. **Output untuk penulusuran yang dimulai dari simpul 2 :**

```
Cost List :

        0       4       1       3
        4       0       2       1
        1       2       0       5
        3       1       5       0

Jalur Terpendek :
2--->4--->1--->3--->2
Minimum Cost : 7
```

3. **Output untuk penulusuran yang dimulai dari simpul 3 :**



4. **Output untuk penulusuran yang dimulai dari simpul 4 :**