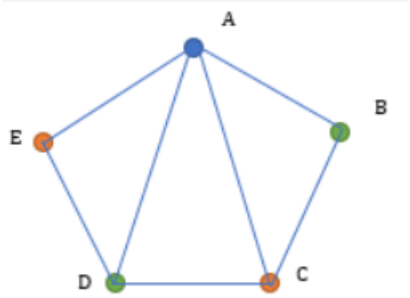
BAB 8.GRAF

8.1 Definisi Graf

Graf adalah himpunan benda-benda yang disebut "simpul" (vertex atau node) yang terhubung oleh "sisi" (edge) atau "busur" (arc). Biasanya graf digambarkan sebagai kumpulan titik-titik (melambangkan "vertex" / "simpul") yang dihubungkan oleh garis-garis (melambangkan "edge" / "Sisi") atau garis berpanah (melambangkan "busur"). Suatu sisi dapat menghubungkan suatu simpul dengan simpul yang sama. Sisi yang demikian dinamakan "gelang" (loop)

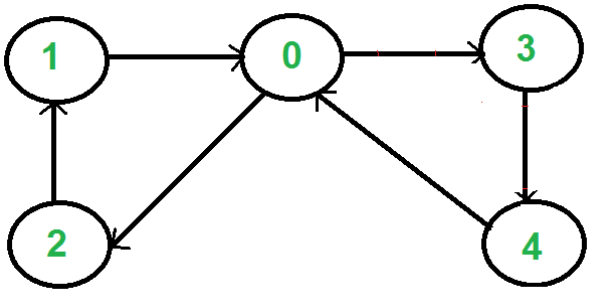
Graf dibagi menjadi berapa macam yaitu :

1. **Undirected Graf**



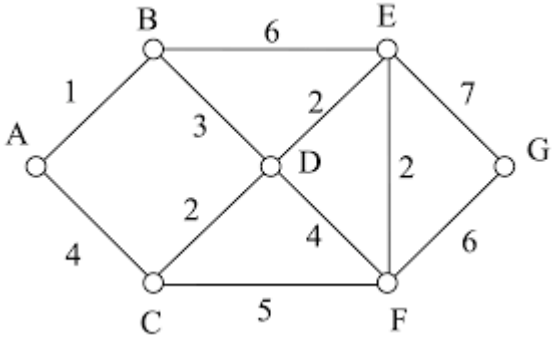
Sumber : https://www.foldertips.com/tik/graph-pohon/

2. **Directed Graf**



Sumber : https://www.geeksforgeeks.org/what-is-directed-graph-directed-graph-meaning/

3. **Weighted Graf**



Sumber : https://webwork.moravian.edu/100.2/sec_shortPaths.html

8.2 Operasi pada Graf

Add Vertex	Menambah Total Simpul
Delete Vertex	Menghapus Simpul
Add Edge	Menambah Masing masing garis Antara simpul
Delete Edge	Menghapus salah satu garis antara simpul
SearchPath	Mencari Jalur antara simpul satu dengan yang lain

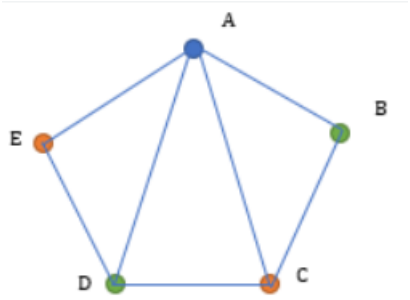
8.3 Konsep Implementasi Graf pada Adjency Matriks

Adjacency matrix atau matriks ketetanggaan adalah salah satu representasi graf yang paling umum. Representasi ini menggunakan matriks untuk merepresentasikan hubungan antar simpul pada sebuah graf.

Dalam adjacency matrix, setiap baris dan kolom merepresentasikan sebuah simpul pada graf, dan nilai pada matriks menunjukkan apakah ada sisi yang menghubungkan dua simpul tersebut atau tidak. Untuk undirected dan directed graf Jika ada sisi yang menghubungkan dua simpul, maka nilai pada matriks adalah 1, dan jika tidak, maka nilai pada matriks adalah 0.namun untuk weighted graph nilai pada matriks untuk kedua simpul diisi dengan nilai beban dari edge atau garis

Contoh 1 :

Buat Program untuk merepresentasikan graph diatas dengan konsep adjacency matriks



Sumber : https://www.foldertips.com/tik/graph-pohon/

Pertama kita buat fungsi untuk menambahkan sisi (edge)

```
#include <iostream>
#include <conio.h>
#include <windows.h>

using namespace std;

#define MAX 999
int graph[MAX][MAX];
int n;
char simpul1 = 'A';
char simpul2 = 'A';

void addedge(int n){
    int i,j;
    cout << "Beri Nilai 1 jika edge di kedua simpul yang terhubung dan 0 untuk tidak"<<endl<<endl;
    for(i=0;i<n;i++){
        cout << "Simpul "<<simpul1++<<" Terhubung dengan"<<endl;
        for(j=0;j<n;j++){
            cout << "Simpul "<<simpul2++<<" = ";
            cin>>graph[i][j];
        }
        simpul2='A';
        cout << endl;
    }
}
```

Kemudian buat fungsi untuk menampilkan graph dalam bentuk matriks

```
27  void cetakgraph(int n){
28      cout << "Cetak Adjacency Matriks"<<endl<<endl;
29      int i,j;
30      cout << "  ";
31      simpul1='A';
32      simpul2='A';
33      for(i=0;i<n;i++){
34          cout << simpul1++<<" ";
35      }
36      cout <<endl;
37      for(i=0;i<n;i++){
38          cout << simpul2++<<" ";
39          for(j=0;j<n;j++){
40              cout << graph[i][j]<< " ";
41          }
42          cout << endl;
43      }
44  }
```

Lalu buat fungsi untuk mencari jalur antar simpul

```
47  void searchpath(char x, char y)
48  {
49      char source=x-65;
50      char destination=y-65;
51      int visited[MAX] = {0};
52      int dist[MAX];
53      int parent[MAX];
54      for (int i = 0; i < n; i++) {
55          dist[i] = MAX;
56          parent[i] = -1;
57      }
58
59      dist[source] = 0;
60      visited[source] = 1;
61      parent[source] = -1;
62
63      int queue[MAX];
64      int front = 0;
65      int rear = 0;
66      int z=0;
67      queue[rear++] = source;
68      while (front != rear) {
69          int u = queue[front++];
70          for (int v = 0; v < n; v++) {
71              if (graph[u][v] && !visited[v]) {
72                  visited[v] = 1;
73
74                  dist[v] = dist[u] + 1;
75                  parent[v] = u;
76                  queue[rear++] = v;
77              }
78          }
79      }
```

```
    if (!visited[destination]) {
        cout << "Tidak ada jalur dari " << x << " ke " << y << endl;
    } else {
        cout << "Jarak terpendek dari " << x << " ke " << y << " adalah " << dist[destination] << endl;

        cout << "Jalur terpendek adalah: ";
        int u = destination;
        while (u != -1) {
            simpul1='A';
            simpul1+=u;
            cout <<simpul1<< " ";
            u = parent[u];
        }
        cout << endl;
    }
}
```

Yang terakhir buat fungsi untuk menghapus sisi dan simpul

```
void deleteEdge(char x,char y){
     int i = x-65;
     int j = y-65;
     graph[i][j]=0;
     graph[j][i]=0;
     cout << "Garis antara Simpul "<<x<<" dan "<<y<<" Berhasil terhapus!\n";
}

void deleteVertex(char z) {
     int v = z-65;
     if (v > n) {
         cout << "Simpul Tidak ada." << endl;
         return;
     }


     for (int i = v; i < n-1; i++) {
         for (int j = 0; j < n; j++) {
             graph[j][i] = graph[j][i+1];
         }
     }


     for (int i = v; i < n-1; i++) {
         for (int j = 0; j < n; j++) {
             graph[i][j] = graph[i+1][j];
         }
     }
     n--;
     cout << "Simpul " << z << " Berhasil Terhapus." << endl;
}
```

Lalu kita akan panggil semua fungsi yang tadi dibuat pada fungsi utama.

```
129  int main()
130  {
131      first:
132      system("cls");
133      char name = 'A',x,y;
134      int source, destination,j,i;
135      int pil;
136
137      cout<<"================================"<<endl;
138      cout<<"         Adjency Matrik"<<endl;
139      cout<<"================================"<<endl;
140      cout<<"1.Tambah simpul dan Sisi"<<endl;
141      cout<<"2.Cetak Graph"<<endl;
142      cout<<"3.Cari Jalur"<<endl;
143      cout<<"4.Hapus Simpul"<<endl;
144      cout<<"5.Hapus Sisi"<<endl;
145      cout<<"\nMasukkan Pilihan : ";
146      cin>>pil;
147      if (pil==1){
148          system("cls");
149          cout << "Masukkan jumlah n: ";
150          cin >> n;
151          addedge(n);
152          cout<<"\nsimpul berhasil dibuat, tekan apa saja untuk lanjut";
153          getch();
154          goto first;
155      }
156      else if (pil==2){
157          system("cls");
158          cetakgraph(n);
159          getch();
160          goto first;
161      }
162      else if (pil==3){
163          system("cls");
164          cout << "Mencari Jalur Terpendek \n";
165          cout << "Masukkan node asal   = ";cin >> x;
166          cout << "Masukkan node tujuan = ";cin >> y;
167          searchpath(x, y);
168          getch();
169          goto first;
170      }
```

```
171     else if (pil==4){
172         system("cls");
173         cetakgraph(n);
174         cout << "\nMenghapus Simpul = ";cin >> x;
175         deleteVertex(x);
176         getch();
177         goto first;
178     }
179     else if (pil==5){
180         system("cls");
181         cetakgraph(n);
182         cout << "\nMenghapus garis antara simpul ";cin >> x;
183         cout << "Dengan simpul ";cin >> y;
184         deleteEdge(x,y);
185         getch();
186         goto first;
187     }
188     else
189         cout<<"input yang anda masukkan salah";
190     return 0;
191 }
```

Berikut adalah output dari program Ketika di run

```
================================
         Adjency Matrik
================================
1.Tambah simpul dan Sisi
2.Cetak Graph
3.Cari Jalur
4.Hapus Simpul
5.Hapus Sisi

Masukkan Pilihan :
```

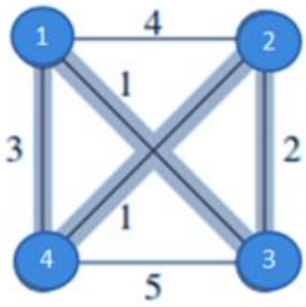
Tambah simpul dan sisi

```
Masukkan jumlah n: 5
Beri Nilai 1 jika edge di kedua simpul yang terhubung dan 0 untuk tidak

Simpul A Terhubung dengan
Simpul A = 1
Simpul B = 2
Simpul C = 3
Simpul D = 4
Simpul E = 5

Simpul B Terhubung dengan
Simpul A = 2
Simpul B = 3
Simpul C = 1
Simpul D = 2
Simpul E = 5

Simpul C Terhubung dengan
Simpul A = 3
Simpul B = 4
Simpul C = 2
Simpul D = 3
Simpul E = 7

Simpul D Terhubung dengan
Simpul A = 6
Simpul B = 5
Simpul C = 8
Simpul D = 9
Simpul E = 7

Simpul E Terhubung dengan
Simpul A = 6
Simpul B = 7
Simpul C = 6
Simpul D = 4
Simpul E = 7
```

Cetak Graph

```
Cetak Adjacency Matriks

  A B C D E
A 1 2 3 4 5
B 2 3 1 2 5
C 3 4 2 3 7
D 6 5 8 9 7
E 6 7 6 4 7
```

LATIHAN:



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

Di atas kita dapat melihat grafik dan matriks biaya yang lengkap yang mencakup jarak antara masing-masing desa. Kita dapat mengamati bahwa matriks biaya adalah simetris yang berarti jarak antara desa 2 hingga 3 sama dengan jarak antara desa 3 hingga 2. Masalah di sini adalah penjual keliling ingin mengetahui turnya dengan biaya minimum. Katakanlah Tukang post berada dititik 1 dia berkeliling semua titik lainnya yang belom dikunjungi dan kembali ke titik awal dengan biaya minimum

Buatlah program dengan c++ dengan mengimplementasikan adjacency matriks dan beberapa algoritma untuk menemukan solusi dari Tukang Pos,untuk titik awal berupa inputnya

Contoh Outputnya

```
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
```