```cpp
#include <iostream>
using namespace std;

//menyimpan item daftar kedekatan
struct adjNode {
    int val, cost;
    adjNode* next;
};
//struktur untuk menyimpan edges
struct graphEdge {
    int start_ver, end_ver, weight;
};
class DiaGraph{
    //masukkan node baru ke dalam daftar kedekatan dari grafik yang diberikan
    adjNode* getAdjListNode(int value, int weight, adjNode* head){
        adjNode* newNode = new adjNode;
        newNode->val = value;
        newNode->cost = weight;
        newNode->next = head;
        return newNode;
    }
    //jumlah node dalam grafik
    int N;
public:
    //daftar kedekatan sebagai array pointer
    adjNode **head;
    // Constructor
    DiaGraph(graphEdge edges[], int n, int N)  {
        // mengalokasikan simpul baru
        head = new adjNode*[N]();
        this->N = N;
        // initialize head pointer for all vertices
        for (int i = 0; i < N; ++i)
            head[i] = nullptr;
        //inisialisasi penunjuk kepala untuk semua simpul
        for (unsigned i = 0; i < n; i++)  {
            int start_ver = edges[i].start_ver;
            int end_ver = edges[i].end_ver;
            int weight = edges[i].weight;
            // masukkan di awal
            adjNode* newNode = getAdjListNode(end_ver, weight, head[start_ver]);
```

```
            head[start_ver] = newNode;
             }
    }
      // Destructor
     ~DiaGraph() {
    for (int i = 0; i < N; i++)
        delete[] head[i];
        delete[] head;
     }
};
// cetak semua simpul yang berdekatan dari simpul yang diberikan
void display_AdjList(adjNode* ptr, int i)
{
    while (ptr != nullptr) {
        cout <<  i << " -> " << "[" << ptr->val
            << ", " << ptr->cost << "] ";
        ptr = ptr->next;
    }
    cout << endl;
}
// implementasi grafik
int main()
{
    // array tepi grafik
    graphEdge edges[] = {
        // (x, y, w) -> tepi dari x ke y dengan bobot w
        {1,2,5},{2,3,1},{4,1,3},{2,4,1},{3,1,1}
    };
    // Jumlah simpul dalam grafik
    int N = 4;
    // hitung jumlah rusuknya
    int n = sizeof(edges)/sizeof(edges[0]);
    // construct graph
    DiaGraph diagraph(edges, n, N);
    // cetak representasi daftar kedekatan grafik
    for (int i = 0; i < N; i++)
    {
        // menampilkan simpul yang berdekatan dari simpul i
        display_AdjList(diagraph.head[i], i);
    }
    return 0;
```

```
1 -> [2, 5]
2 -> [4, 1] 2 -> [3, 1]
3 -> [1, 1]

Process returned 0 (0x0)   execution time : 0.575 s
Press any key to continue.
```