DSA Assignment-03

Name:	Syed Muhammad Raza Ali
Enrolment:	02-134231-028
Course:	Data Structures and Algorithms
Faculty:	Miss Lubna Siddiqui

ASSIGNMENT No. 3

Problem Based Learning

Course Title: Data structure & Algorithm
Course Code: CSC-221
Class: BS (CS)-3A,3B — Submission deadline: 4-Jun-24
Course Instructor: Lubna Siddiqui — Marks: 10

Instructions

1. Students will perform this assignment individually.
2. Deadline will not be extended for any reason.
3. Copied assignments will be marked zero
4. Submit your assignment in soft and hard copies both. Make a single pdf file and upload on LMS.

Scenario

Logistics services refers to the management of goods movement from one point to another in an efficient manner. Logistics service provides you with door-to-door service, this includes the collection, customs clearance, and last-mile delivery. ***SwiftEx*** is a new logistics company that receives order and plans the most efficient routes of transporting goods. It processes two types of order, *urgent orders* and *normal orders*. Urgent orders are processed on immediate basis.

Factors such as mode of transportation, distance and deadlines are to be considered while processing orders. When an order is received, from the customers. SwiftEx verifies the order details and also notifies the customer through text messages. To maintain the customer satisfaction and to find economic solution, the manager of SwiftEx Karachi branch wants to manage the transportation of orders/parcels in an efficient way. He planned different schemes to distribute the orders that should be maintained economically.

a. Process urgent orders separately on immediate basis.
b. Normal orders are delivered to the customers in a same way as they ordered.
c. Normal orders are delivered to the customers by calculating the distance of customer's destination. The order having the smallest distance will distribute first.

Assumptions

The riders are given N parcels (orders) each day that they have to deliver to the customers. You are asked to simulate all schemes with at least 15 parcels (orders). The cost of one liter petrol is 289.33 rupees. Assume that on average 35 km distance is covered in 1 liter petrol.

Deliverables.

Create the code in C++ by applying appropriate data structure for scheme a, b & c by considering the assumptions mentioned above. (CLO4, PLO4, C6)

Analyze the time complexity of all schemes. (CLO 3, PLO2, C4)

Explain the use of data structure that you used for all schemes and also provide comparisonwith calculation for consumption of petrol for scheme b and c. (CLO1, PLO1, C2)

Note: Single file is required with title page, index page, codes with output, calculations andcomparisons.

Evaluation Criteria and Targeted CLOs

Deliverables	Evaluation Criteria	Targeted CLOs
Algorithm and Code	50 %	CLO-4,PLO-4,C6
Time complexity Analysis	25 %	CLO-3,PLO-2,C4
Comparative study and use of data structures	25%	CLO-1,PLO-1,C2

Table of Contents

Case a -- Urgent Orders

In this case we have a scenario where we have to process urgent orders and deliver them first then the rest of the orders will be delivered. This implies that there will be some sort of priority related to orders which will indicate that which order will be completed first.

Approach:

This scenario can be implemented through a priority queue. A priority queue is an abstract data-type similar to a regular queue or stack data structure. Each element in a priority queue has an associated priority. In a priority queue, elements with high priority are served before elements with low priority.

There are several ways of implementing the priority queue in our code, however we will implement it through heap data structure and we will be maintaining a Max-heap containing "order number" as data field on its nodes.

Implementation in code:

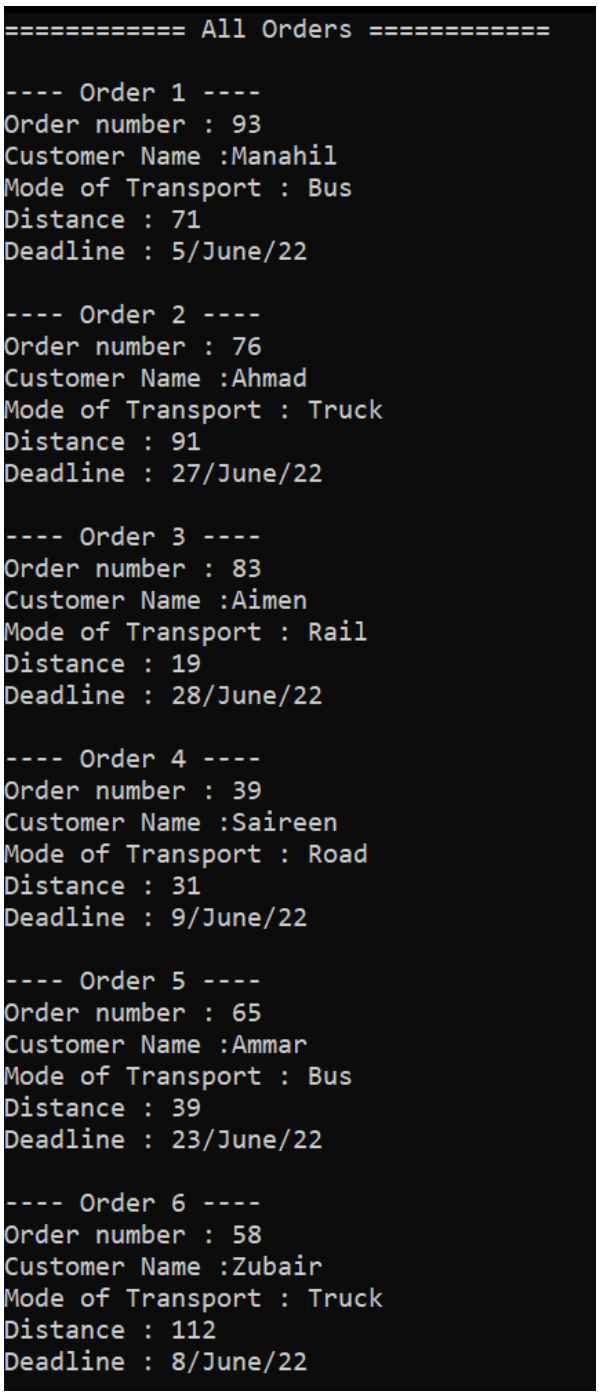
```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;


void swap(int* a, int* b)
{
	int temp = *b;
	*b = *a;
	*a = temp;
}

struct Order {
	string customerName;
	int orderNumber;
	string modeOfTransport;
	int distance;
	string deadline;

	Order(string customerName, int orderNumber, string modeOfTransport, int
distance, string deadline) {
		this->customerName = customerName;
		this->orderNumber = orderNumber;
		this->modeOfTransport = modeOfTransport;
		this->distance = distance;
		this->deadline = deadline;
	}
};

void heapify(vector<Order> &heap,int size,int i) {
	int largest = i;
	int l = (2 * i) + 1;
	int r = (2 * i) + 2;
	if (l < size && heap[l].orderNumber > heap[largest].orderNumber)
		largest = l;
	if (r < size && heap[r].orderNumber > heap[largest].orderNumber)
		largest = r;
```

```
	if (largest != i)
	{
		swap(heap[i].customerName, heap[largest].customerName);
		swap(&heap[i].orderNumber, &heap[largest].orderNumber);
		swap(heap[i].modeOfTransport, heap[largest].modeOfTransport);
		swap(heap[i].distance, heap[largest].distance);
		swap(heap[i].deadline, heap[largest].deadline);

		heapify(heap, size, largest);
	}

}

void insert(vector<Order>& heap, Order order)
{

	heap.push_back(order);
	int size = heap.size();
	for (int i = size / 2 - 1; i >= 0; i--)
	{
		heapify(heap, size, i);
	}
}

void heapSort(vector<Order>& heap) {
	int size = heap.size();

	for (int i = size / 2 - 1; i >= 0; i--)  //heapifying heap if necessary
		heapify(heap, size, i);

	for (int i = size - 1; i >= 0; i--) {    //sorting heap
		swap(heap[0].customerName, heap[i].customerName);
		swap(&heap[0].orderNumber, &heap[i].orderNumber);
		swap(heap[0].modeOfTransport, heap[i].modeOfTransport);
		swap(heap[0].distance, heap[i].distance);
		swap(heap[0].deadline, heap[i].deadline);
		heapify(heap, i, 0);
	}

	int counter = 1;
	for (int i = size - 1; i >= 0; i--) {    //defining priorites
		cout << "---- Priority"<<counter<<" ----\n";

		cout << "Order number : " << heap[i].orderNumber << "\n"
			<< "Customer Name : " << heap[i].customerName << "\n"
			<< "Mode of Transport : " << heap[i].modeOfTransport << "\n"
			<< "Distance : " << heap[i].distance << "\n"
			<< "Deadline : " << heap[i].deadline << "\n";

		cout << "\n";
		counter += 1;
	}
}


void printList(vector<Order>& heap)
{
	for (int i = 0; i < heap.size(); ++i) {
		cout << "---- Order "<<i+1<<" ----\n";

		cout <<"Order number : "<< heap[i].orderNumber << "\n"
			<< "Customer Name :"<<heap[i].customerName << "\n"
```

```
                    << "Mode of Transport : "<<heap[i].modeOfTransport << "\n"
                    << "Distance : "<<heap[i].distance << "\n"
                    << "Deadline : "<<heap[i].deadline << "\n";

            cout << "\n";
      }
}

void expenseCalculation(vector<Order> &heap) {
      float totalDistance = 0;
      for (int i = 0; i < heap.size(); i++) {
            totalDistance += heap[i].distance;
      }
      float petrolUsed = totalDistance / 35;
      float costOfPetrol = petrolUsed * 289.33;
      cout << "Total distance covered (in kilometeres) : " << totalDistance <<
"\n"
            << "Amount of petrol used (in liters) : " << petrolUsed << "\n"
            << "Cost of petrol per liter : 289.33 rupees\n"
            << "Cost of petrol for " << totalDistance << "kms : " <<
costOfPetrol<<"rupees\n";

}

int main() {

      vector<Order> heap;

      Order order1 = Order("Raza",20,"Road",12,"4/June/22");
      Order order2 = Order("Ali", 17, "Truck", 34, "1/June/22");
      Order order3 = Order("Shayan", 34, "Rail", 389, "7/June/22");
      Order order4 = Order("Rehman", 29, "Bus", 31, "4/June/22");
      Order order5 = Order("Aimen", 83, "Rail", 19, "28/June/22");
      Order order6 = Order("Muskan", 16, "Rail", 39, "10/June/22");
      Order order7 = Order("Zubair", 58, "Truck", 112, "8/June/22");
      Order order8 = Order("Alishba", 42, "Air", 921, "6/June/22");
      Order order9 = Order("Saireen", 39, "Road", 31, "9/June/22");
      Order order10 = Order("Ammar", 65, "Bus", 39, "23/June/22");
      Order order11 = Order("Ahmad", 76, "Truck", 91, "27/June/22");
      Order order12 = Order("Zikria", 51, "Road", 23, "30/June/22");
      Order order13 = Order("Manahil", 93, "Bus", 71, "5/June/22");
      Order order14 = Order("Mishkat", 27, "Rail", 480, "4/June/22");
      Order order15 = Order("Saad", 43, "Air", 347, "18/June/22");


      insert(heap, order1);
      insert(heap, order2);
      insert(heap, order3);
      insert(heap, order4);
      insert(heap, order5);
      insert(heap, order6);
      insert(heap, order7);
      insert(heap, order8);
      insert(heap, order9);
      insert(heap, order10);
      insert(heap, order11);
      insert(heap, order12);
      insert(heap, order13);
      insert(heap, order14);
      insert(heap, order15);

      cout << "============ All Orders ============\n\n";
```

```
        printList(heap);

        cout << "============ Orders In Priority ============\n\n";
        heapSort(heap);

        cout << "============ Expense required ============\n\n";

        expenseCalculation(heap);

        return 0;
}
```

Output:



```
---- Order 7 ----
Order number : 43
Customer Name :Saad
Mode of Transport : Air
Distance : 347
Deadline : 18/June/22

---- Order 8 ----
Order number : 17
Customer Name :Ali
Mode of Transport : Truck
Distance : 34
Deadline : 1/June/22

---- Order 9 ----
Order number : 34
Customer Name :Shayan
Mode of Transport : Rail
Distance : 389
Deadline : 7/June/22

---- Order 10 ----
Order number : 29
Customer Name :Rehman
Mode of Transport : Bus
Distance : 31
Deadline : 4/June/22

---- Order 11 ----
Order number : 42
Customer Name :Alishba
Mode of Transport : Air
Distance : 921
Deadline : 6/June/22

---- Order 12 ----
Order number : 16
Customer Name :Muskan
Mode of Transport : Rail
Distance : 39
Deadline : 10/June/22
```

```
---- Order 12 ----
Order number : 16
Customer Name :Muskan
Mode of Transport : Rail
Distance : 39
Deadline : 10/June/22

---- Order 13 ----
Order number : 51
Customer Name :Zikria
Mode of Transport : Road
Distance : 23
Deadline : 30/June/22

---- Order 14 ----
Order number : 20
Customer Name :Raza
Mode of Transport : Road
Distance : 12
Deadline : 4/June/22

---- Order 15 ----
Order number : 27
Customer Name :Mishkat
Mode of Transport : Rail
Distance : 480
Deadline : 4/June/22

============ Orders In Priority ============

---- Priority1 ----
Order number : 93
Customer Name : Manahil
Mode of Transport : Bus
Distance : 71
Deadline : 5/June/22

---- Priority2 ----
Order number : 83
Customer Name : Aimen
Mode of Transport : Rail
Distance : 19
```

```
---- Priority3 ----
Order number : 76
Customer Name : Ahmad
Mode of Transport : Truck
Distance : 91
Deadline : 27/June/22

---- Priority4 ----
Order number : 65
Customer Name : Ammar
Mode of Transport : Bus
Distance : 39
Deadline : 23/June/22

---- Priority5 ----
Order number : 58
Customer Name : Zubair
Mode of Transport : Truck
Distance : 112
Deadline : 8/June/22

---- Priority6 ----
Order number : 51
Customer Name : Zikria
Mode of Transport : Road
Distance : 23
Deadline : 30/June/22

---- Priority7 ----
Order number : 43
Customer Name : Saad
Mode of Transport : Air
Distance : 347
Deadline : 18/June/22

---- Priority8 ----
Order number : 42
Customer Name : Alishba
Mode of Transport : Air
Distance : 921
Deadline : 6/June/22
```

```
---- Priority9 ----
Order number : 39
Customer Name : Saireen
Mode of Transport : Road
Distance : 31
Deadline : 9/June/22

---- Priority10 ----
Order number : 34
Customer Name : Shayan
Mode of Transport : Rail
Distance : 389
Deadline : 7/June/22

---- Priority11 ----
Order number : 29
Customer Name : Rehman
Mode of Transport : Bus
Distance : 31
Deadline : 4/June/22

---- Priority12 ----
Order number : 27
Customer Name : Mishkat
Mode of Transport : Rail
Distance : 480
Deadline : 4/June/22

---- Priority13 ----
Order number : 20
Customer Name : Raza
Mode of Transport : Road
Distance : 12
Deadline : 4/June/22

---- Priority14 ----
Order number : 17
Customer Name : Ali
Mode of Transport : Truck
Distance : 34
Deadline : 1/June/22
```

```
---- Priority15 ----
Order number : 16
Customer Name : Muskan
Mode of Transport : Rail
Distance : 39
Deadline : 10/June/22

============ Expense required ============

Total distance covered (in kilometeres) : 2639
Amount of petrol used (in liters) : 75.4
Cost of petrol per liter : 289.33 rupees
Cost of petrol for 2639kms : 21815.5rupees
```

Time Complexity:

Time complexities for different operations of heap are given as:

- Insertion
  - Best case – O(1)
  - Average case – O(log n)
  - Worst case – O(log n)

- Deletion
  - Best case – O(log n)
  - Average case – O(log n)
  - Worst case – O(log n)

- Heap sort
  - Best case – O(n logn)
  - Average case – O(n logn)
  - Worst case – O(n logn)

Case b -- Normal Orders in FIFO Manner

In this scenario, we have normal orders which will be delivered to the customer in the same manner in which they were placed. In other words we can say that the parcel distribution will obey the property of First In First Out.

Approach:

This can be achieved by using queue data structure. A queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end(rear) of the sequence and the removal of entities from the other end(front) of the sequence.

Since we don't need to take actions on customer's distances and just need to process the orders in the same way in which they were placed, This will be the best approach for this scenario.

Implementation in code:

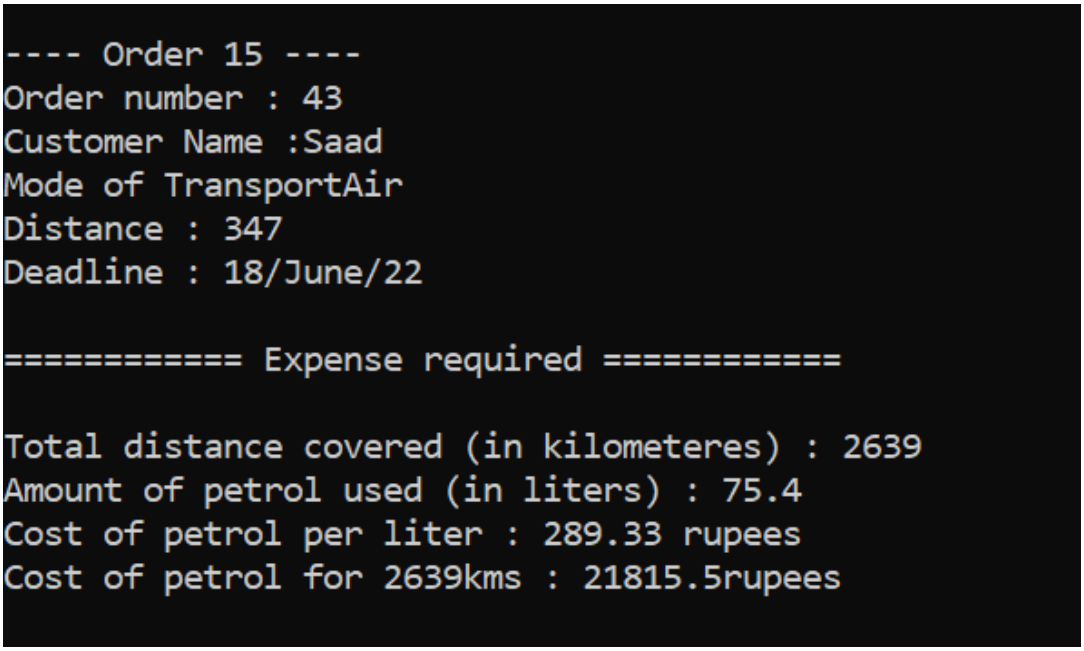
```
#include <iostream>
#include <queue>
using namespace std;

struct Order {
	string customerName;
	int orderNumber;
	string modeOfTransport;
	int distance;
	string deadline;

	Order(string customerName, int orderNumber, string modeOfTransport, int
distance, string deadline) {
		this->customerName = customerName;
		this->orderNumber = orderNumber;
		this->modeOfTransport = modeOfTransport;
		this->distance = distance;
		this->deadline = deadline;
	}
};
float totalDistance = 0;

void printOrderList(queue<Order> orderList) {

	int counter = 1;
	while(orderList.empty() == false){
		Order currentOrder = orderList.front();
		orderList.pop();
		cout << "---- Order " << counter << " ----\n";
		cout << "Order number : " << currentOrder.orderNumber << "\n"
			<< "Customer Name :" << currentOrder.customerName << "\n"
			<< "Mode of Transport" << currentOrder.modeOfTransport <<
"\n"
			<< "Distance : " << currentOrder.distance << "\n"
			<< "Deadline : " << currentOrder.deadline << "\n";
		totalDistance += currentOrder.distance;
		counter++;
```

```
            cout << "\n";
        }
}
void expenseCalculation(queue<Order> orderList) {

        float petrolUsed = totalDistance / 35;
        float costOfPetrol = petrolUsed * 289.33;
        cout << "Total distance covered (in kilometeres) : " << totalDistance <<
"\n"
             << "Amount of petrol used (in liters) : " << petrolUsed << "\n"
             << "Cost of petrol per liter : 289.33 rupees\n"
             << "Cost of petrol for " << totalDistance << "kms : " <<
costOfPetrol << "rupees\n";

}
int main() {
        queue<Order> orderList;

        Order order1 = Order("Raza", 20, "Road", 12, "4/June/22");
        Order order2 = Order("Ali", 17, "Truck", 34, "1/June/22");
        Order order3 = Order("Shayan", 34, "Rail", 389, "7/June/22");
        Order order4 = Order("Rehman", 29, "Bus", 31, "4/June/22");
        Order order5 = Order("Aimen", 83, "Rail", 19, "28/June/22");
        Order order6 = Order("Muskan", 16, "Rail", 39, "10/June/22");
        Order order7 = Order("Zubair", 58, "Truck", 112, "8/June/22");
        Order order8 = Order("Alishba", 42, "Air", 921, "6/June/22");
        Order order9 = Order("Saireen", 39, "Road", 31, "9/June/22");
        Order order10 = Order("Ammar", 65, "Bus", 39, "23/June/22");
        Order order11 = Order("Ahmad", 76, "Truck", 91, "27/June/22");
        Order order12 = Order("Zikria", 51, "Road", 23, "30/June/22");
        Order order13 = Order("Manahil", 93, "Bus", 71, "5/June/22");
        Order order14 = Order("Mishkat", 27, "Rail", 480, "4/June/22");
        Order order15 = Order("Saad", 43, "Air", 347, "18/June/22");

        orderList.push(order1);
        orderList.push(order2);
        orderList.push(order3);
        orderList.push(order4);
        orderList.push(order5);
        orderList.push(order6);
        orderList.push(order7);
        orderList.push(order8);
        orderList.push(order9);
        orderList.push(order10);
        orderList.push(order11);
        orderList.push(order12);
        orderList.push(order13);
        orderList.push(order14);
        orderList.push(order15);


        cout << "============ All Orders ============\n\n";
        cout << "parcels will be delivered in this order...\n";
        printOrderList(orderList);

        cout << "============ Expense required ============\n\n";
        expenseCalculation(orderList);
        return 0;
}
```

Output:

```
============ All Orders ============

parcels will be delivered in this order...
---- Order 1 ----
Order number : 20
Customer Name :Raza
Mode of TransportRoad
Distance : 12
Deadline : 4/June/22

---- Order 2 ----
Order number : 17
Customer Name :Ali
Mode of TransportTruck
Distance : 34
Deadline : 1/June/22

---- Order 3 ----
Order number : 34
Customer Name :Shayan
Mode of TransportRail
Distance : 389
Deadline : 7/June/22

---- Order 4 ----
Order number : 29
Customer Name :Rehman
Mode of TransportBus
Distance : 31
Deadline : 4/June/22

---- Order 5 ----
Order number : 83
Customer Name :Aimen
Mode of TransportRail
Distance : 19
Deadline : 28/June/22

---- Order 6 ----
Order number : 16
Customer Name :Muskan
Mode of TransportRail
Distance : 39
Deadline : 10/June/22

---- Order 7 ----
Order number : 58
Customer Name :Zubair
Mode of TransportTruck
Distance : 112
```

```
---- Order 8 ----
Order number : 42
Customer Name :Alishba
Mode of TransportAir
Distance : 921
Deadline : 6/June/22

---- Order 9 ----
Order number : 39
Customer Name :Saireen
Mode of TransportRoad
Distance : 31
Deadline : 9/June/22

---- Order 10 ----
Order number : 65
Customer Name :Ammar
Mode of TransportBus
Distance : 39
Deadline : 23/June/22

---- Order 11 ----
Order number : 76
Customer Name :Ahmad
Mode of TransportTruck
Distance : 91
Deadline : 27/June/22

---- Order 12 ----
Order number : 51
Customer Name :Zikria
Mode of TransportRoad
Distance : 23
Deadline : 30/June/22

---- Order 13 ----
Order number : 93
Customer Name :Manahil
Mode of TransportBus
Distance : 71
Deadline : 5/June/22

---- Order 14 ----
Order number : 27
Customer Name :Mishkat
Mode of TransportRail
Distance : 480
Deadline : 4/June/22
```



Time complexity:

Time complexities for different operations of queue are given as:

- Insertion
  - Best case – O(1)
  - Average case – O(1)
  - Worst case – O(1)

- Deletion
  - Best case – O(1)
  - Average case – O(1)
  - Worst case – O(1)

- Searching
  - Best case – O(1)
  - Average case – O(n)
  - Worst case – O(n)

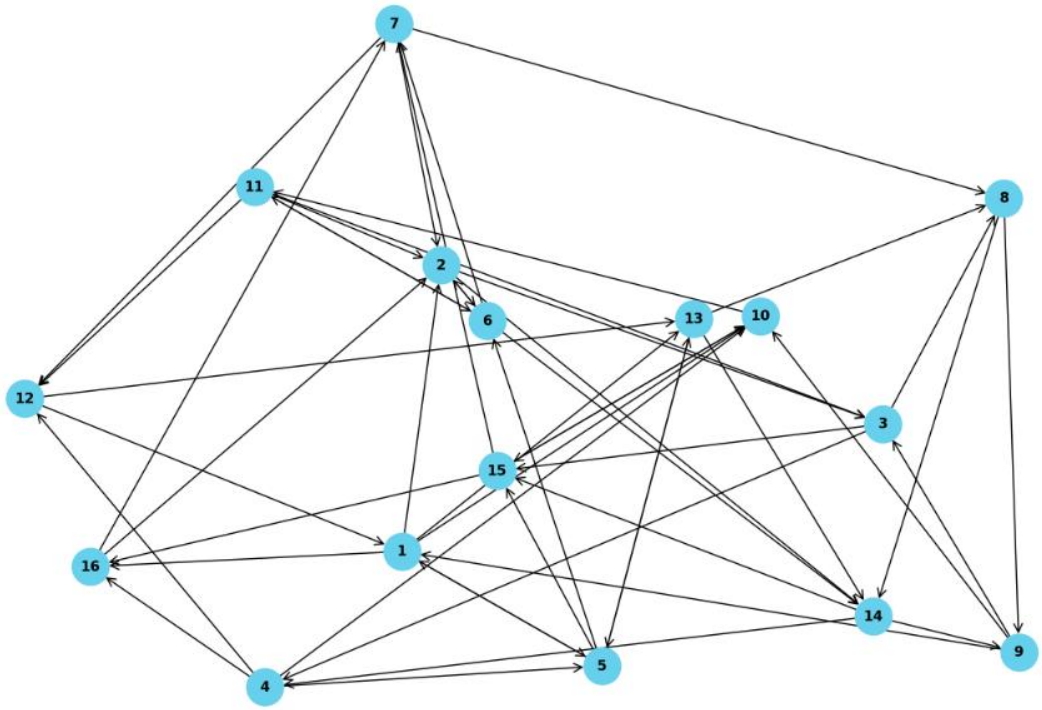
Case c -- Normal Orders with distances

In scheme c, we are asked to implement an efficient and optimized data structure to complete the normal orders of the customers in such a way that we calculate the distances from our warehouse to the customer's destinations and then deliver the parcel having the shortest distance first.

Approach:

This can be achieved by using graphs. All the destinations will represent different vertices on our graph having multiple paths (called edges) connecting them. Then we will take the vertex representing our warehouse as the source and will calculate the distance of each vertex from this source. After that we will start the process of parcel distribution.

We will be using Dijkstra's algorithm for calculating the distances from warehouse to the customer's destination. Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a weighted graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.



In the above graph there are total 16 vertices (1-16) each representing the customer's location. The vertex 1 is the source i.e. the warehouse. In code however, vertices are represented from V_0 to V_{15} where 0 is the source.

Implementation in code:

```
#include <iostream>
#include <queue>
#include <vector>
#include <limits>
#include <algorithm>
using namespace std;

struct Edge {
	int src;
	int dest;
	int wt;

	Edge(int src, int dest, int wt) {
		this->src = src;
		this->dest = dest;
		this->wt = wt;
	}
};

struct Pair {
	int node;
	int dist;

	Pair(int node, int dist) {
		this->node = node;
		this->dist = dist;
	}
	bool operator<(const Pair& other) const {
		return this->dist > other.dist;
	}
};
int totalDistance = 0;
void createGraph(vector<Edge> graph[]) {
	graph[0].push_back(Edge(0, 1, 10));
	graph[0].push_back(Edge(0, 9, 7));
	graph[0].push_back(Edge(0, 4, 2));
	graph[0].push_back(Edge(0, 15, 1));

	graph[1].push_back(Edge(1, 2, 7));
	graph[1].push_back(Edge(1, 5, 8));

	graph[2].push_back(Edge(2, 3, 3));
	graph[2].push_back(Edge(2, 7, 3));

	graph[3].push_back(Edge(3, 4, 7));
	graph[3].push_back(Edge(3, 9, 6));
	graph[3].push_back(Edge(3, 11, 9));

	graph[4].push_back(Edge(4, 5, 3));
	graph[4].push_back(Edge(4, 12, 6));
	graph[4].push_back(Edge(4, 14, 5));

	graph[5].push_back(Edge(5, 6, 4));
	graph[5].push_back(Edge(5,10, 1));
	graph[5].push_back(Edge(5, 13, 2));

	graph[6].push_back(Edge(6, 7, 7));
	graph[6].push_back(Edge(6, 11, 1));
	graph[6].push_back(Edge(6, 1, 7));
```

```
	graph[7].push_back(Edge(7, 8, 8));
	graph[7].push_back(Edge(7, 13, 8));

	graph[8].push_back(Edge(8, 9, 4));
	graph[8].push_back(Edge(8, 0, 7));
	graph[8].push_back(Edge(8, 2, 1));

	graph[9].push_back(Edge(9, 10, 4));
	graph[9].push_back(Edge(9, 14, 2));

	graph[10].push_back(Edge(10, 11, 9));
	graph[10].push_back(Edge(10, 1, 1));
	graph[10].push_back(Edge(10, 2, 5));
	graph[10].push_back(Edge(10, 5, 7));

	graph[11].push_back(Edge(11, 12, 7));
	graph[11].push_back(Edge(11, 0, 7));

	graph[12].push_back(Edge(12, 13, 5));
	graph[12].push_back(Edge(12, 4, 4));
	graph[12].push_back(Edge(12, 7, 7));

	graph[13].push_back(Edge(13, 14, 8));
	graph[13].push_back(Edge(13, 8, 1));
	graph[13].push_back(Edge(13, 3, 1));

	graph[14].push_back(Edge(14, 15, 9));
	graph[14].push_back(Edge(14, 6, 4));
	graph[14].push_back(Edge(14, 9, 1));

	graph[15].push_back(Edge(15, 6, 10));
	graph[15].push_back(Edge(15, 1, 9));


}


void dijkstra(vector<Edge> graph[],int src) {
	priority_queue<Pair> pq;

	int distance[16];
	for (int i = 0; i < 16; i++) {
		if (i != src)
			distance[i] = numeric_limits<int>::max();
	}
	distance[src] = 0;

	bool visited[16];
	for (int i = 0; i < 16; i++)
		visited[i] = false;

	pq.push(Pair(src, 0));

	while (pq.empty() == false) {
		Pair current = pq.top();
		pq.pop();

		if (visited[current.node] == false) {
			visited[current.node] = true;
			for (int i = 0; i < graph[current.node].size(); i++) {
				Edge e = graph[current.node][i];
				int u = e.src;
				int v = e.dest;
```

```
                        if (distance[u] + e.wt < distance[v]) {
                              distance[v] = distance[u] + e.wt;
                              pq.push(Pair(v, distance[v]));
                        }
                  }
            }
      }
      cout << "============================================================\n"
            << "     Distance From Warehouse To Customer's Destination\n"
            << "============================================================\n"
            << "Shortest paths from warehouse to customer's location are given
as\n";


      for (int i = 0; i < 16; i++) {
            if (i == 0)
                  cout << "Warehouse : " << distance[i] << " km\n";
            else
            cout <<"Order "<<i <<" : "<< distance[i] << " km\n";
      }


      for (int i = 0; i < 16; i++) {
            for (int j = 0; j < 15; j++) {
                  if (distance[j] > distance[j + 1])
                        swap(distance[j], distance[j + 1]);
            }
      }
      cout << "============================================================\n"
            << "        Order of Parcel Distribution\n"
            << "============================================================\n"
            << "The order in which the parcels will be distribued is given
as\n";

      for (int i = 0; i < 16; i++) {
            if (i == 0)
                  cout << "Warehouse : " << distance[i] << " km\n";
            else {
                  cout << "Order " << i << " : " << distance[i] << " km\n";
                  totalDistance += distance[i];
            }
      }
}
void expenseCalculation() {
      cout << "============================================================\n"
            << "        Details and Summary of Expenditure\n"
            << "============================================================\n";

      float petrolUsed = totalDistance / 35;
      float costOfPetrol = petrolUsed * 289.33;
      cout << "Total distance covered (in kilometeres) : " << totalDistance <<
"\n"
            << "Amount of petrol used (in liters) : " << petrolUsed << "\n"
            << "Cost of petrol per liter : 289.33 rupees\n"
            << "Cost of petrol for " << totalDistance << "kms : " <<
costOfPetrol << "rupees\n";

}
int main() {

      vector<Edge> graph[16];

      createGraph(graph);
```

```
        dijkstra(graph, 0);

        expenseCalculation();

        return 0;
}
```

Output:

```
============================================================
     Distance From Warehouse To Customer's Destination
============================================================
Shortest paths from warehouse to customer's location are given as
Warehouse : 0 km
Order 1 : 7 km
Order 2 : 9 km
Order 3 : 8 km
Order 4 : 2 km
Order 5 : 5 km
Order 6 : 9 km
Order 7 : 12 km
Order 8 : 8 km
Order 9 : 7 km
Order 10 : 6 km
Order 11 : 10 km
Order 12 : 8 km
Order 13 : 7 km
Order 14 : 7 km
Order 15 : 1 km
```

```
============================================================
              Order of Parcel Distribution
============================================================
The order in which the parcels will be distribued is given as
Warehouse : 0 km
Order 1 : 1 km
Order 2 : 2 km
Order 3 : 5 km
Order 4 : 6 km
Order 5 : 7 km
Order 6 : 7 km
Order 7 : 7 km
Order 8 : 7 km
Order 9 : 8 km
Order 10 : 8 km
Order 11 : 8 km
Order 12 : 9 km
Order 13 : 9 km
Order 14 : 10 km
Order 15 : 12 km
============================================================
              Details and Summary of Expenditure
============================================================
Total distance covered (in kilometeres) : 106
Amount of petrol used (in liters) : 3
Cost of petrol per liter : 289.33 rupees
Cost of petrol for 106kms : 867.99rupees
```

Time complexity:

Time complexities for different operations of adjacency list in a graph are given as:

- Storage
  - Best case – O(V+E)
  - Average case – O(V+E)
  - Worst case – O(V+E)

- Adding an edge
  - Best case – O(1)
  - Average case – O(1)
  - Worst case – O(1)

- Removing an edge
  - Best case – O(1)
  - Average case – O(avg degree)
  - Worst case – O(V)

- Traversing all edges
  - Best case – O(V+E)
  - Average case – O(V+E)
  - Worst case – O(V+E)

- Finding all neighbors
  - Best case – O(degree)
  - Average case – O(degree)
  - Worst case – O(degree)

Time complexity of Dijkstra's algorithm for calculating the shortest paths to vertices depends on its implementation, as we are implementing it through a priority queue, then:

- Overall time complexity
  - Best case – O((V+E) logV)
  - Average case – O((V+E) logV)
  - Worst case – O((V+E) logV)

Comparison of scheme b and c:

In this simulation, we compared the performance of two logistics schemes for delivering parcels: Scheme B, which processes orders in the order they were received (FIFO), and Scheme C, which processes orders based on the shortest distance using Dijkstra's algorithm. Scheme B resulted in a cumulative travel distance that is the straightforward sum of the distances to each delivery point, reflecting a potentially inefficient route as it does not consider the geographic distribution of the destinations. This leads to a higher total distance traveled and, consequently, higher petrol expenditure. Conversely, Scheme C optimizes the route by prioritizing the shortest paths from the warehouse to each destination, thereby minimizing the total distance traveled. As a result, Scheme C demonstrates a more efficient

use of resources, achieving lower petrol consumption and cost. The distance-based approach (Scheme C) significantly reduces operational costs and improves delivery efficiency, proving to be a superior strategy for parcel distribution in terms of both distance covered and petrol expenditure. This efficiency gain can lead to better customer satisfaction due to faster deliveries and lower logistical costs, which is crucial for a competitive advantage in the logistics industry.