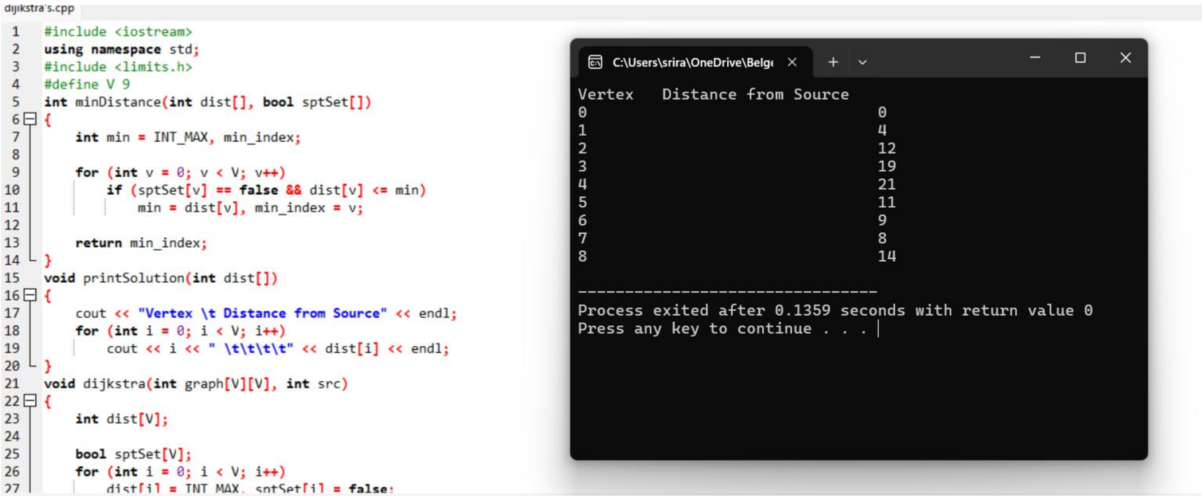
1.Dijikstrs algorithm

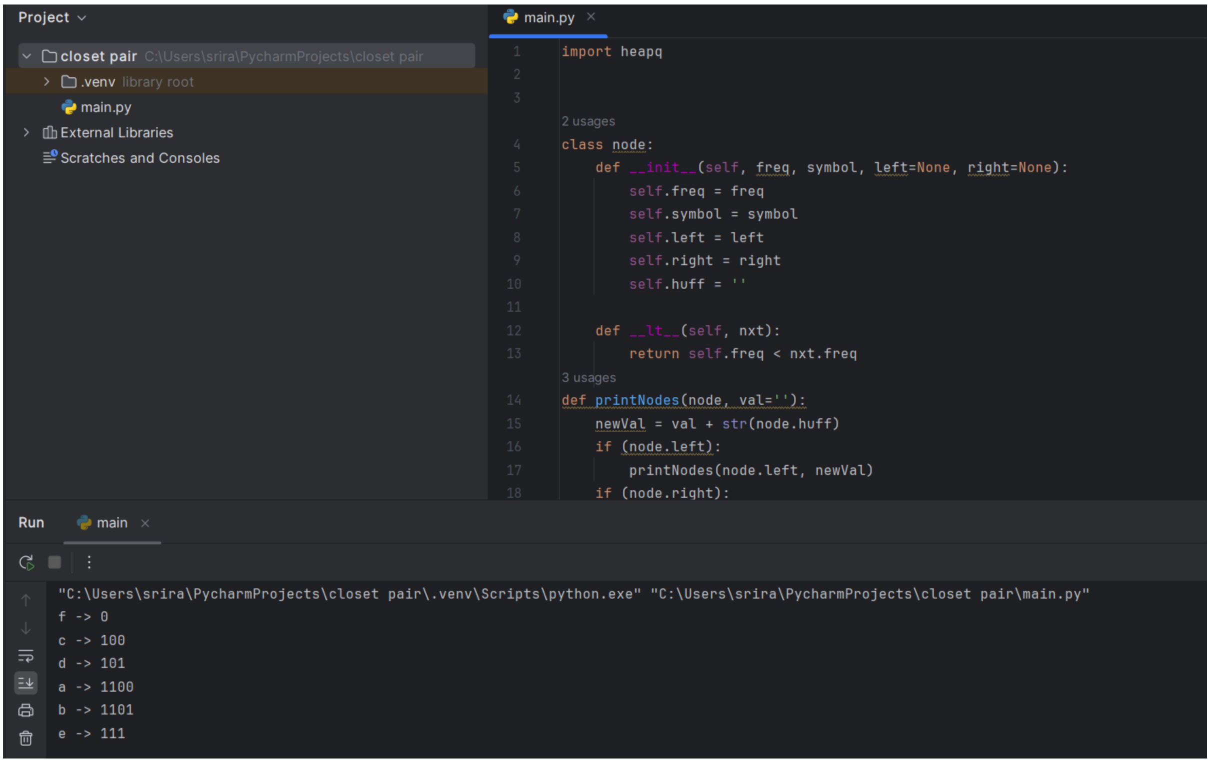


dijikstra's.cpp

```cpp
#include <iostream>
using namespace std;
#include <limits.h>
#define V 9
int minDistance(int dist[], bool sptSet[])
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}
void printSolution(int dist[])
{
    cout << "Vertex \t Distance from Source" << endl;
    for (int i = 0; i < V; i++)
        cout << i << " \t\t\t\t" << dist[i] << endl;
}
void dijkstra(int graph[V][V], int src)
{
    int dist[V];

    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;
```

```
C:\Users\srira\OneDrive\Belg(
Vertex   Distance from Source
0                            0
1                            4
2                            12
3                            19
4                            21
5                            11
6                            9
7                            8
8                            14

--------------------------------
Process exited after 0.1359 seconds with return value 0
Press any key to continue . . .
```

2.Huffman codes



Project
closet pair C:\Users\srira\PycharmProjects\closet pair
.venv library root
main.py
External Libraries
Scratches and Consoles

main.py

```python
import heapq


2 usages
class node:
    def __init__(self, freq, symbol, left=None, right=None):
        self.freq = freq
        self.symbol = symbol
        self.left = left
        self.right = right
        self.huff = ''

    def __lt__(self, nxt):
        return self.freq < nxt.freq
3 usages
def printNodes(node, val=''):
    newVal = val + str(node.huff)
    if (node.left):
        printNodes(node.left, newVal)
    if (node.right):
```

Run main

```
"C:\Users\srira\PycharmProjects\closet pair\.venv\Scripts\python.exe" "C:\Users\srira\PycharmProjects\closet pair\main.py"
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
```

3.Container loading

jikstra's.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
double cont[1000][1000];
void num_of_containers(int n,
                       double x)
{
    int count = 0;
    cont[1][1] = x;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            if (cont[i][j] >= (double)1) {
                count++;
                cont[i + 1][j]
                    += (cont[i][j]
                        - (double)1)
                       / (double)2;

                cont[i + 1][j + 1]
                    += (cont[i][j]
                        - (double)1)
                       / (double)2;
            }
        }
    }
    cout << count;
}
int main()
```

C:\Users\srira\OneDrive\Belg

```
4
--------------------------------
Process exited after 0.1547 seconds with retur
Press any key to continue . . .
```

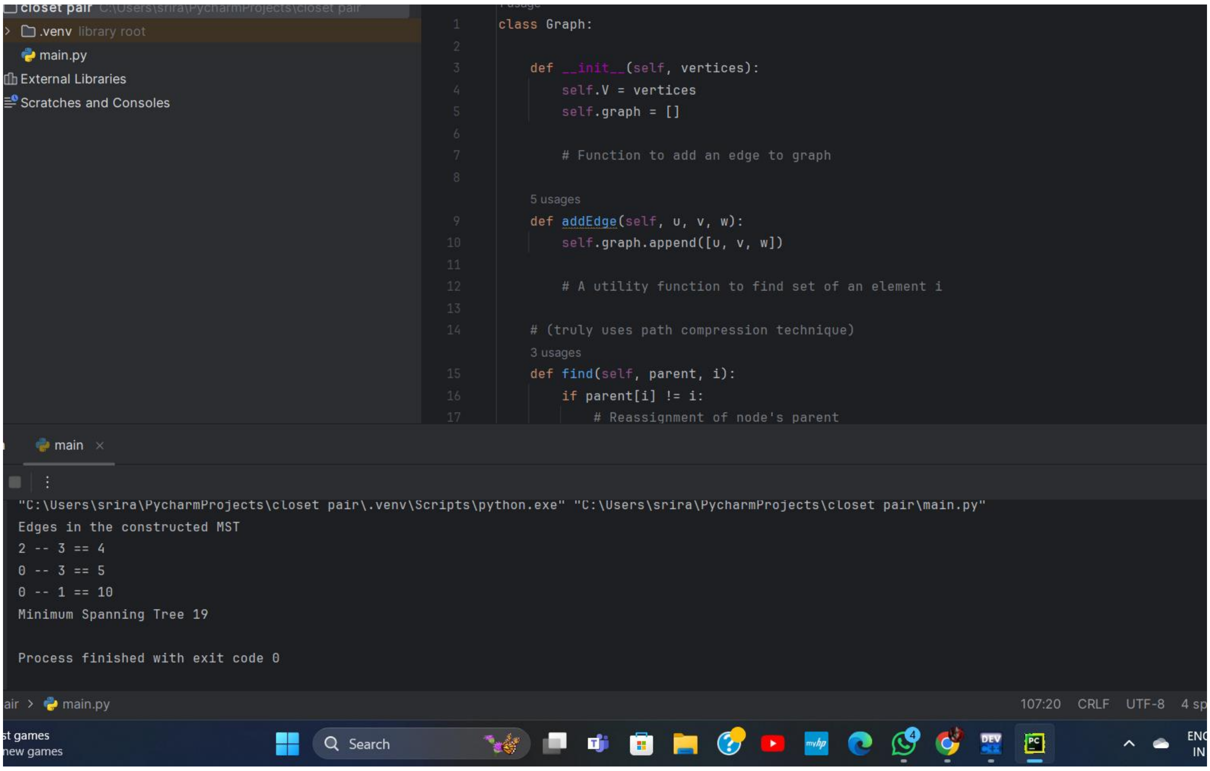
4.Minimum spanning tree

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX 30

typedef struct edge {
    int u, v, w;
} Edge;

typedef struct graph {
    int V, E;
    Edge edges[MAX];
} Graph;

int parent[MAX];

int find(int i) {
    while (parent[i])
        i = parent[i];
    return i;
}

void union_ij(int i, int j) {
    parent[i] = j;
}

void kruskalMST(Graph graph) {
    int minCost = 0;
    for (int i = 0; i < MAX; i++)
        parent[i] = 0;

    int edgeCount = 0;
    for (int i = 0; edgeCount < graph.V - 1; i++) {
        int x = find(graph.edges[i].u);
        int y = find(graph.edges[i].v);
```

Select E:\DAA\LAB\LAB PROGRAMS\Untitled1545.exe

```
0 - 1
0 - 2
0 - 3
Minimum Cost: 21

--------------------------------
Process exited after 0.1878 seconds with return value 0
Press any key to continue . . .
```

5.Kruskals algorithm



6.Boruvkas algorithm