Algorithms project 2022

**Small world phenomenon**

(Using breadth-first search)

Team members:

**Kareem Mohamed abd el-Hamed Mohamed Ali**

20191700456 – section 6 – IS

**Adel Amgad salama Rizk**

20191700325 - section 3 – IS

**Kareem Mohamed Ibrahem ahmed**

20191700454 – section 6 - IS

*Project repository in GitHub:*

[https://github.com/polaamgad88/Small-World-phenomenon](%20https://github.com/polaamgad88/Small-World-phenomenon)

The small-world phenomenon is the principle that we are all linked by short chains of acquaintances. Two main terminologies are of interest to us in this project:

1. Degree of Separation Between any two persons, is defined as the shortest chain of links that leads one person to the other.
2. Relation Strength Between any two persons, is defined as the total number of common "films" between each two persons in the shortest chain of links. If there're many shortest chains of links with the same length, the relation strength is then the one with max common "films".
3. Project depends on graph algorithm and an example on web crawling.

Functions:

1. **strength**(int s1, int s2)

Strength is function for calculating the relation strength between 2 actors by calculating intersection of their movies in complexity O(N^2).

Text

Description automatically generated

1. **degre\_of\_seprations**(List<int> s)

degree of separation is function for calculating the degree of separation of the path which is equal path’s elements – 1 in complexity O(1).

Graphical user interface

Description automatically generated with medium confidence

1. **graph()**

graph is function which responsible for creating adjacency List of actors in graphs representation to loop on it using BFS in another function in complexity O(N^3). **Text

Description automatically generated**

1. **chain\_of\_movies**(List<int> path)

chain of movies is function for finding the chain of movies by dictionary for saving the common movies between 2 actors in it and if there’s more than 1 common movie , it prints all of it in complexity O(N^2).

Text

Description automatically generated

1. **load\_queries()**

load queries is function to load queries of each 2 actors to get degree of sep. and relation strength of them in complexity O(N).

1. **BFS(int actor1\_str, int actor2\_str)**

BFS is function responsible for searching breadth first search on graph level by level and get information about path through this process in complexity O(v+e).

1. **load\_movies()**

load movies is function which read movies of file by store all lines of file in variable then work on it in complexity O(N^2).

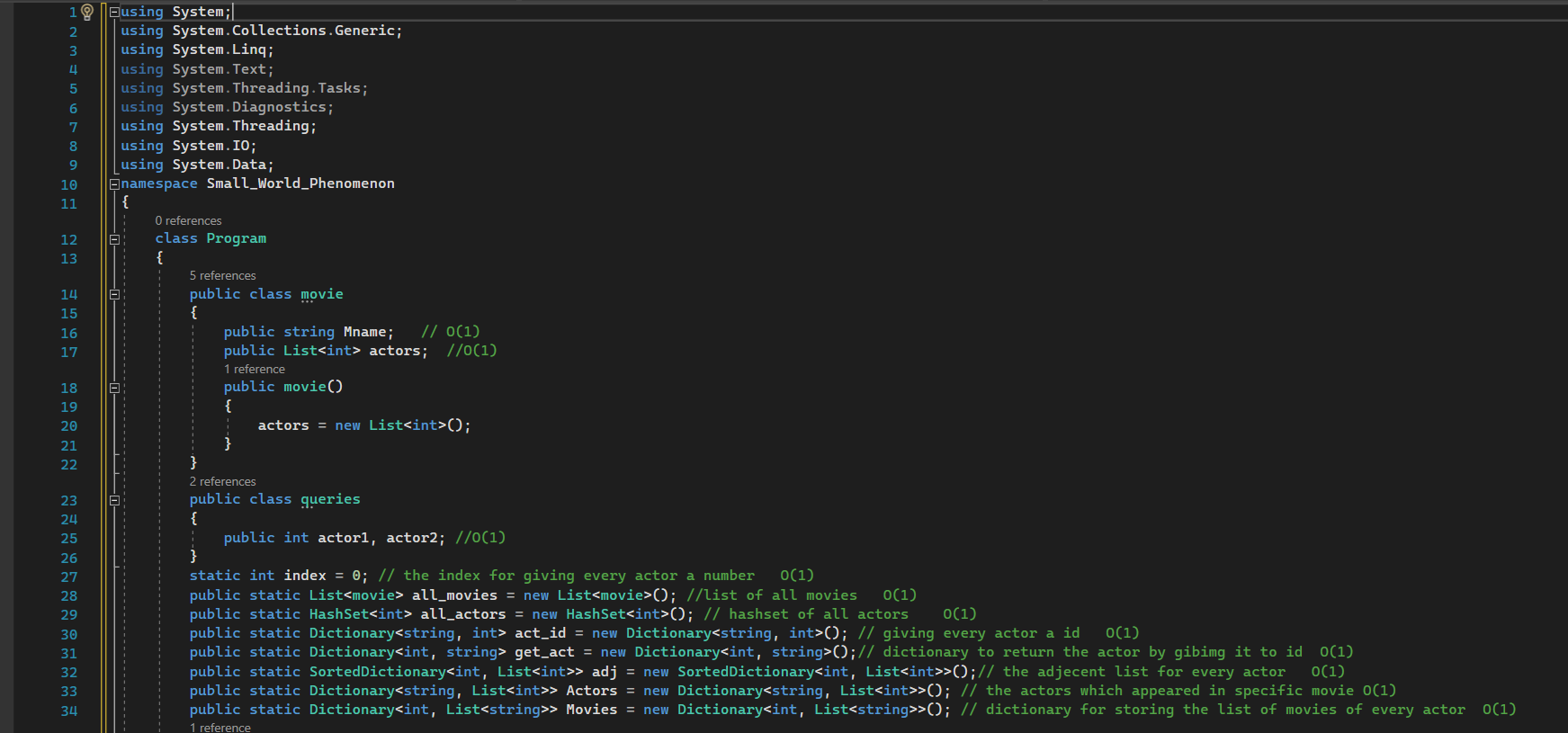
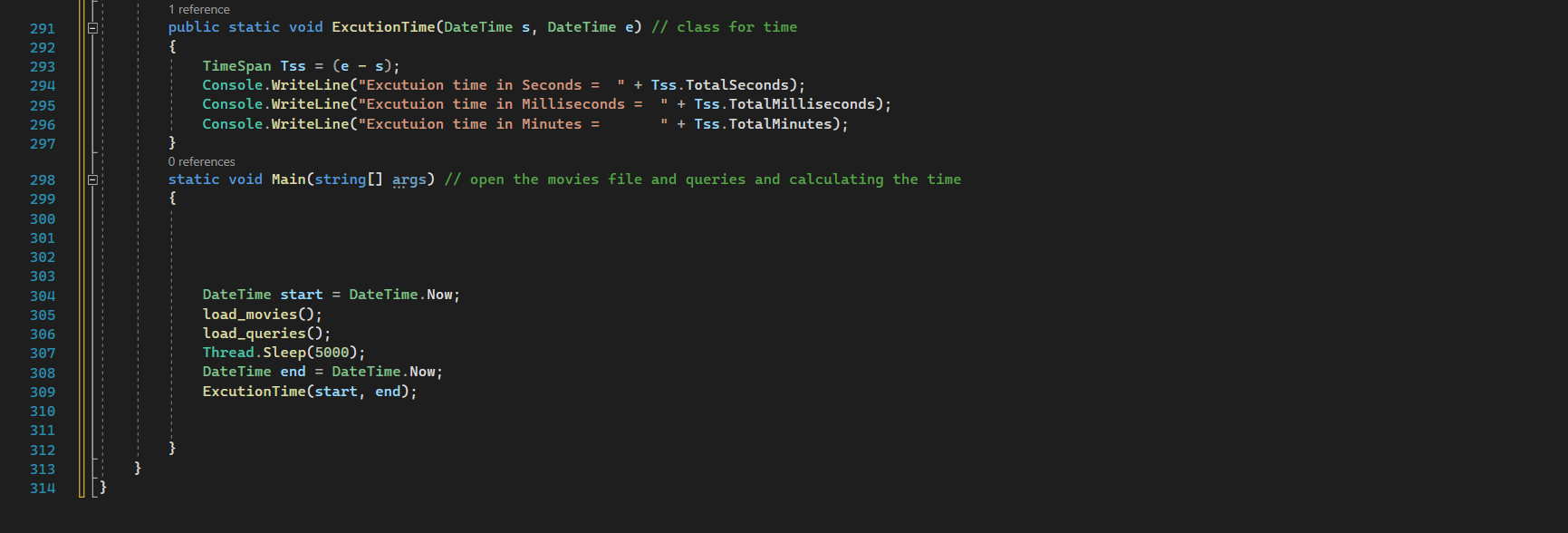
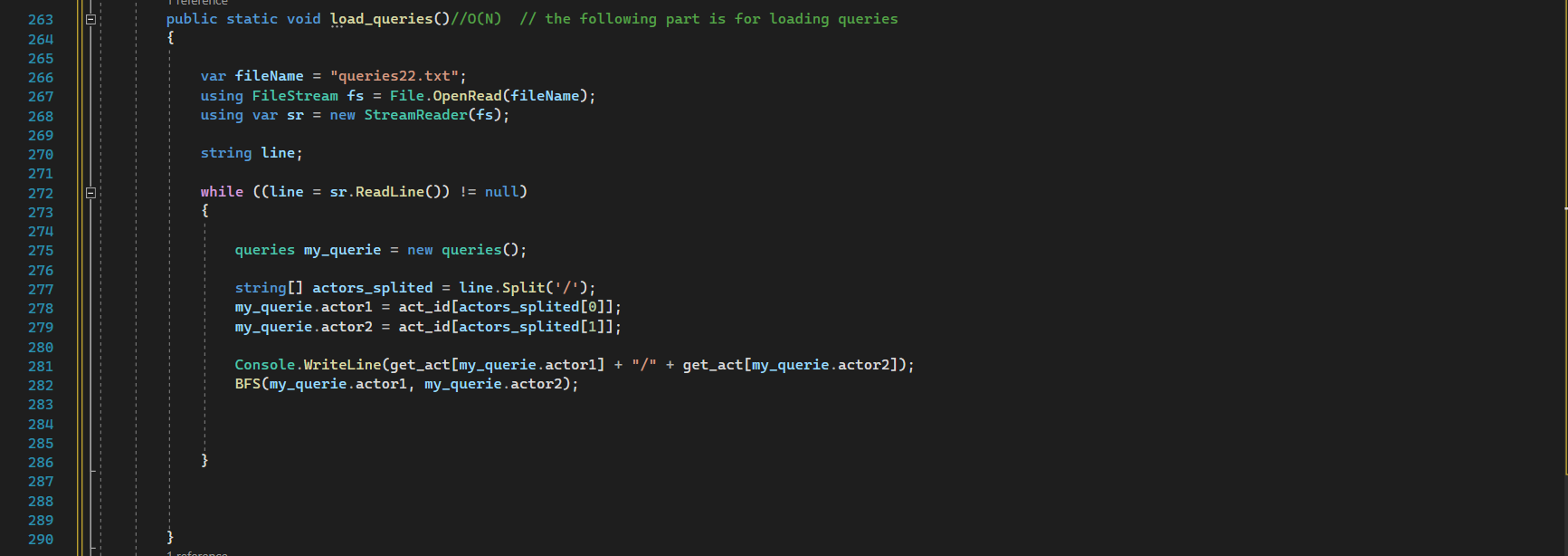
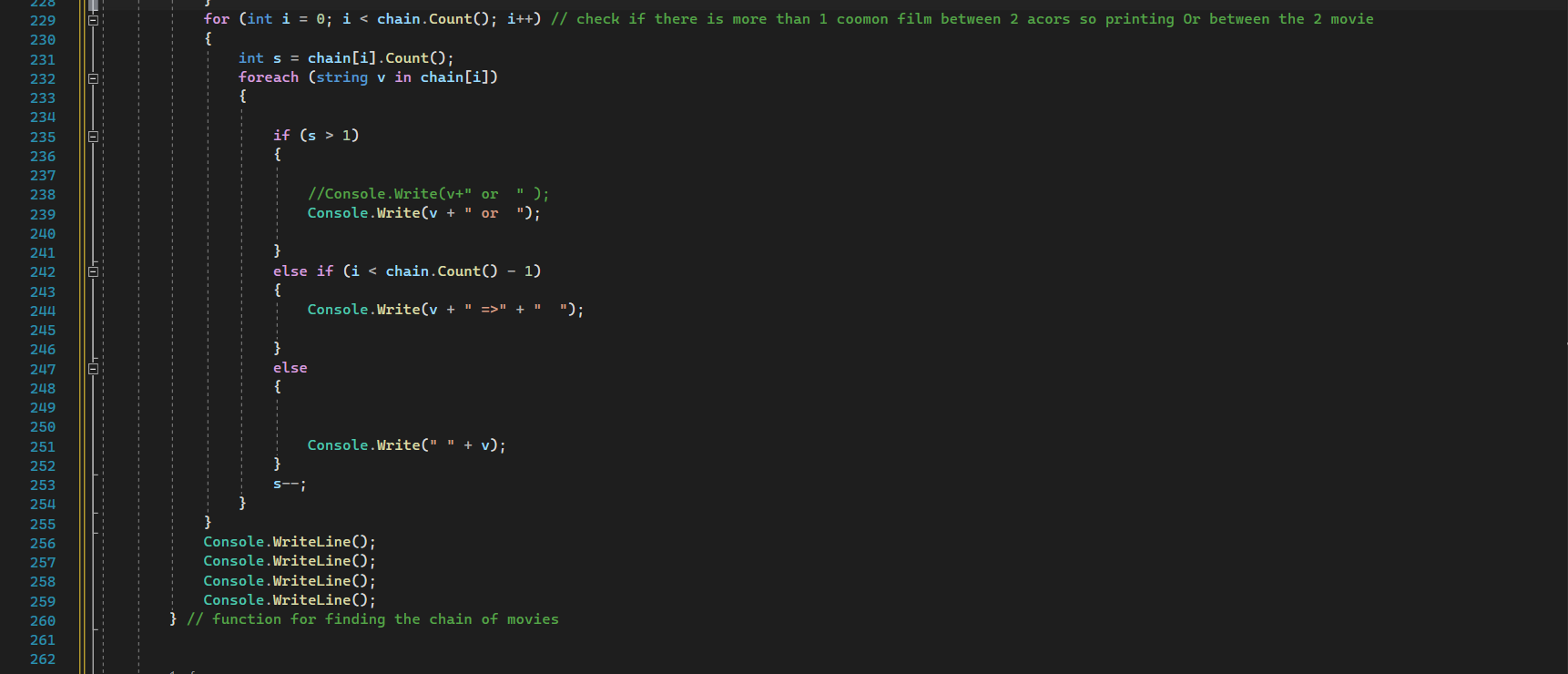
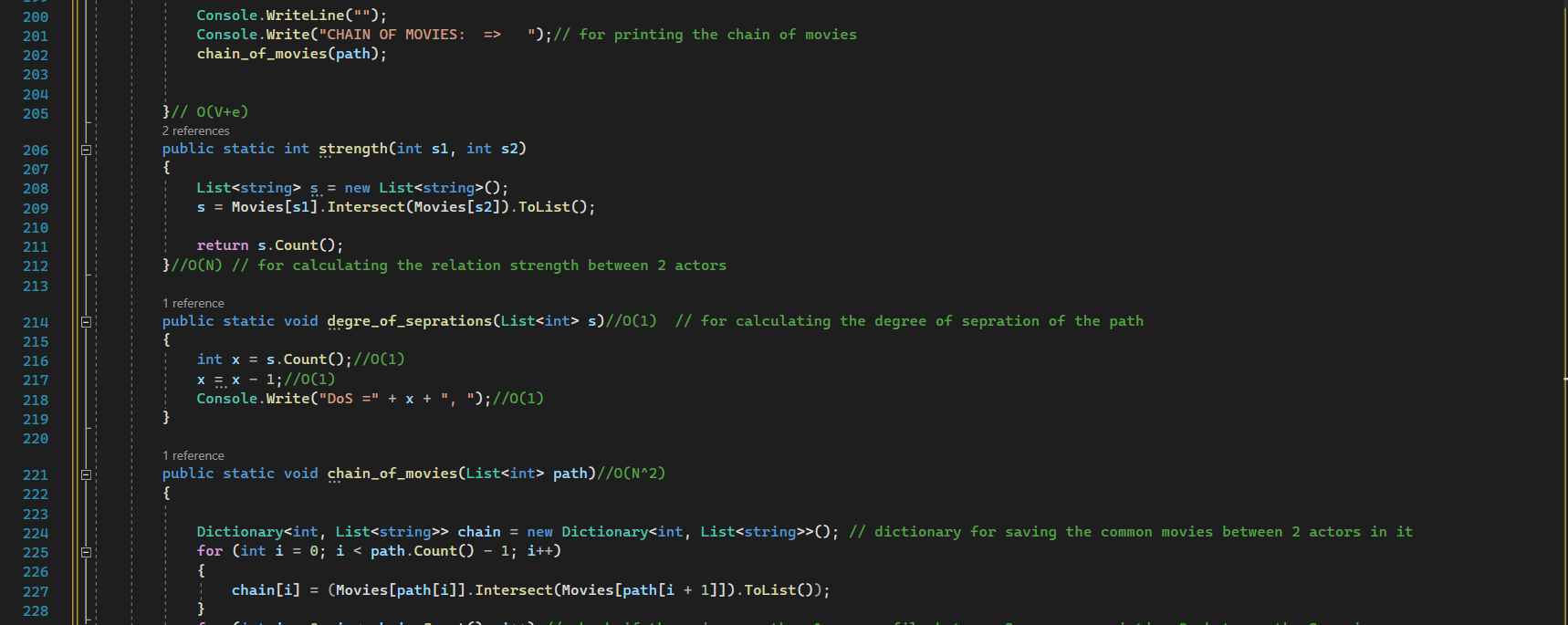
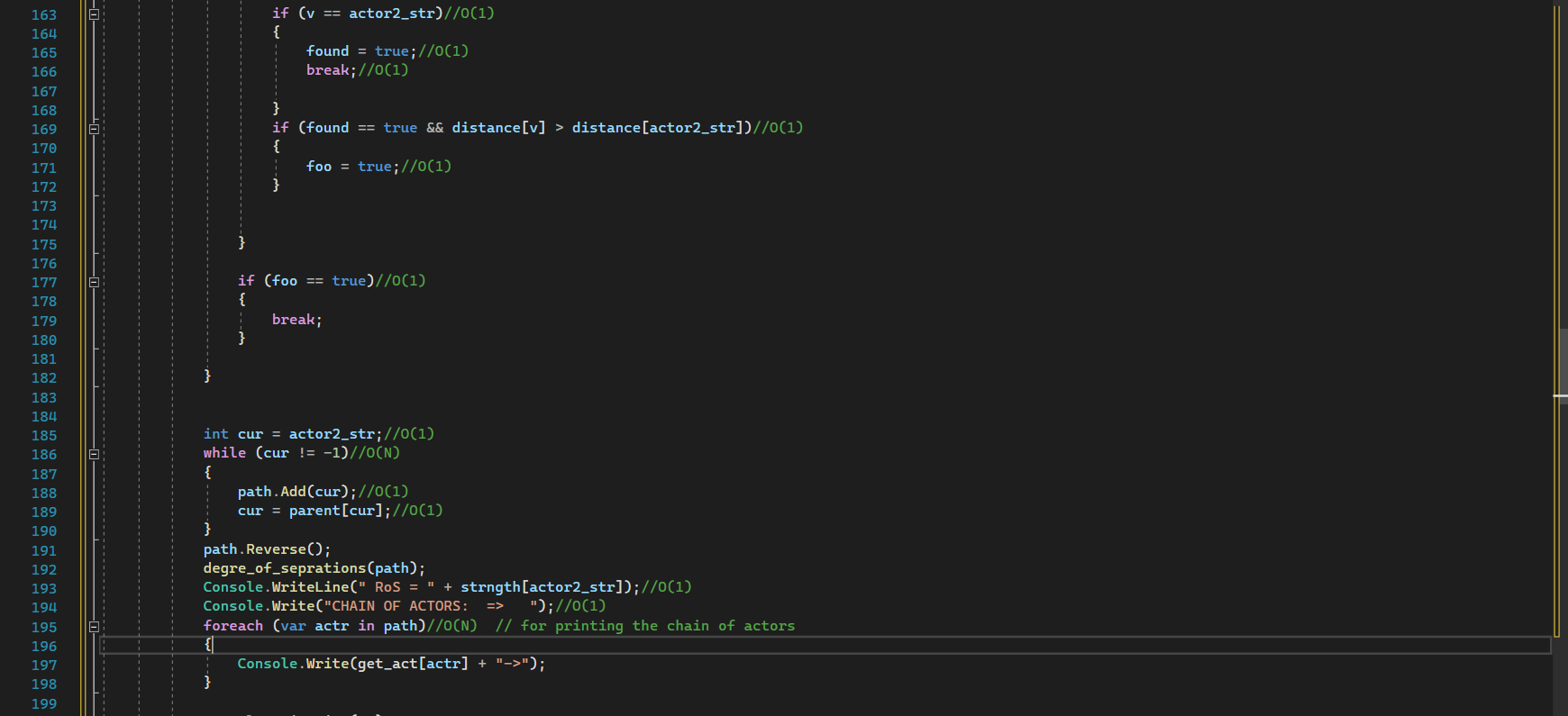
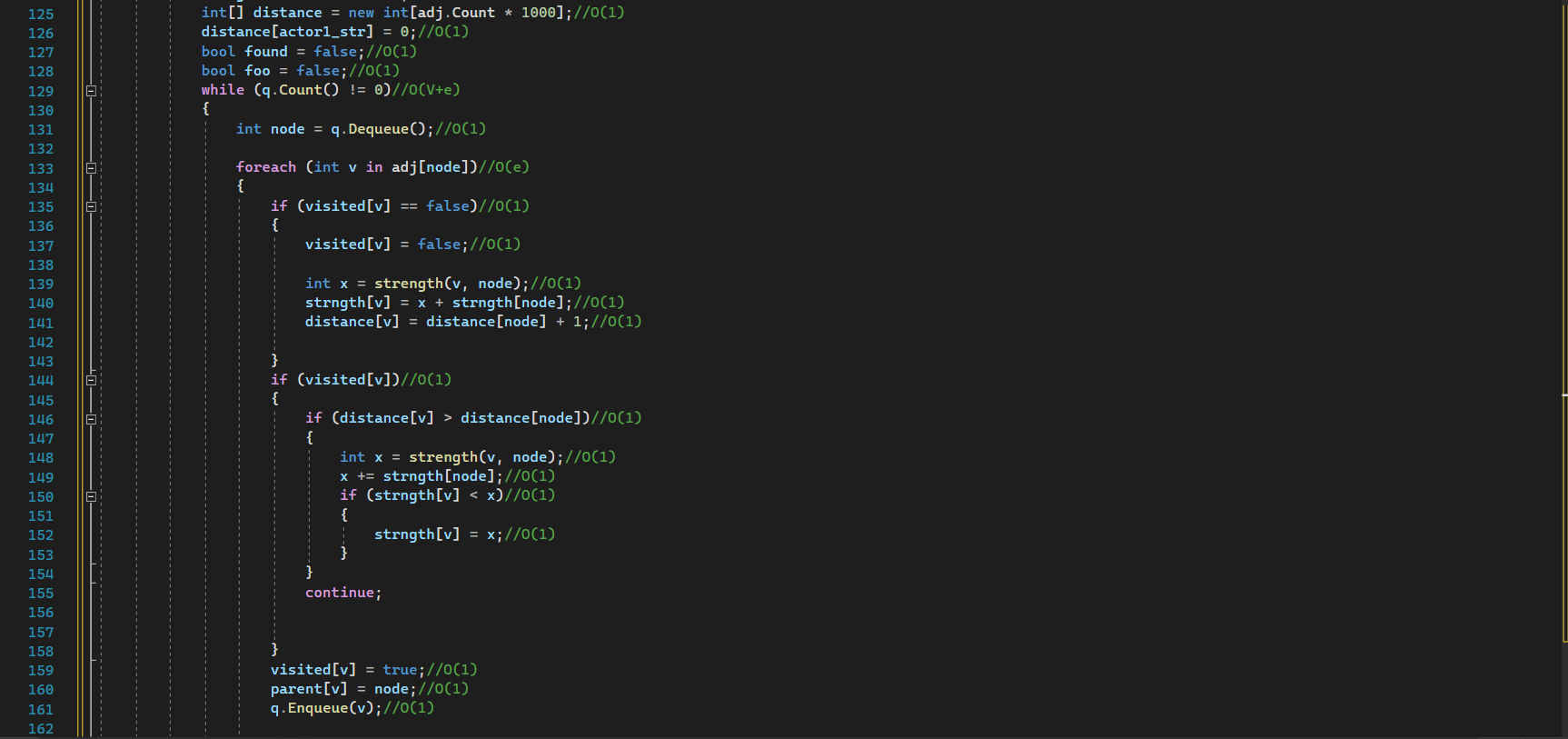
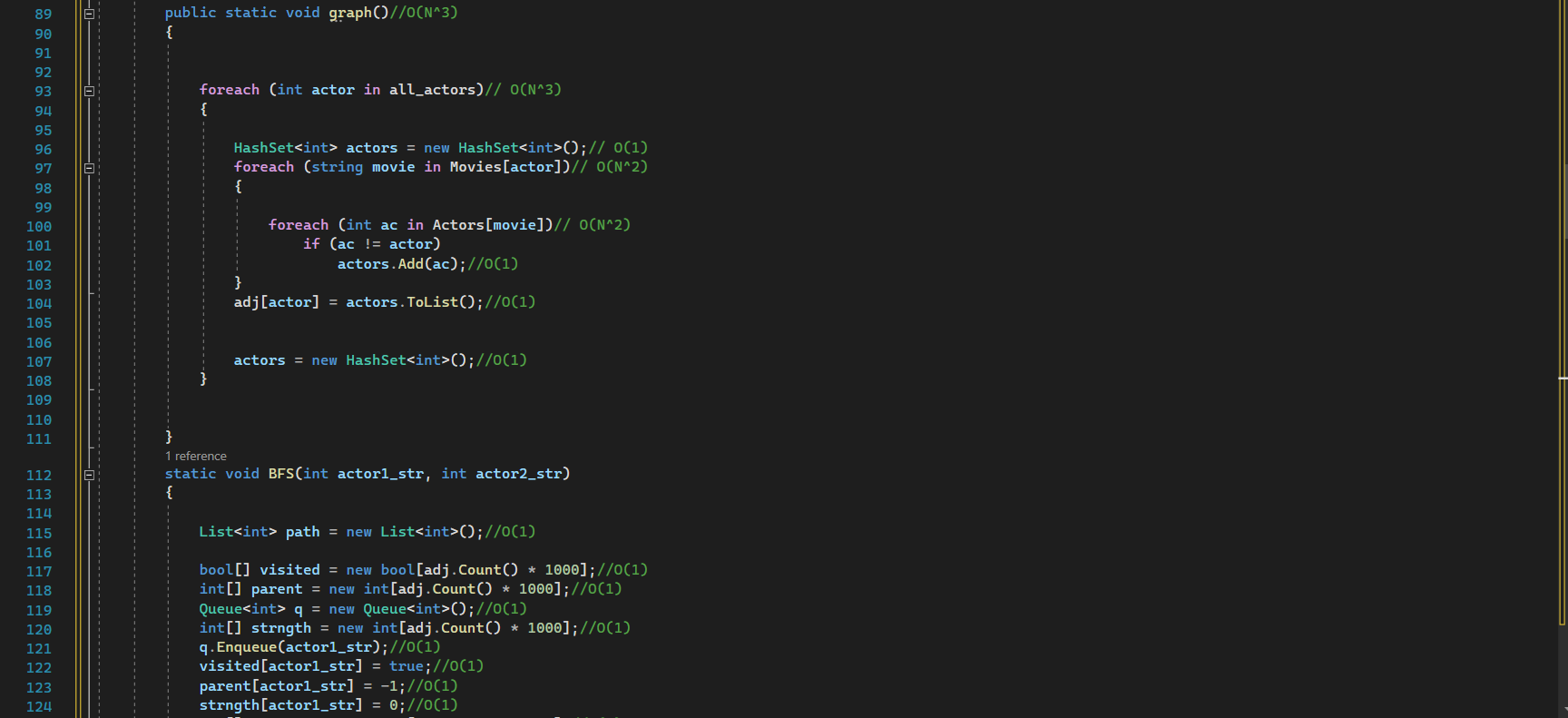
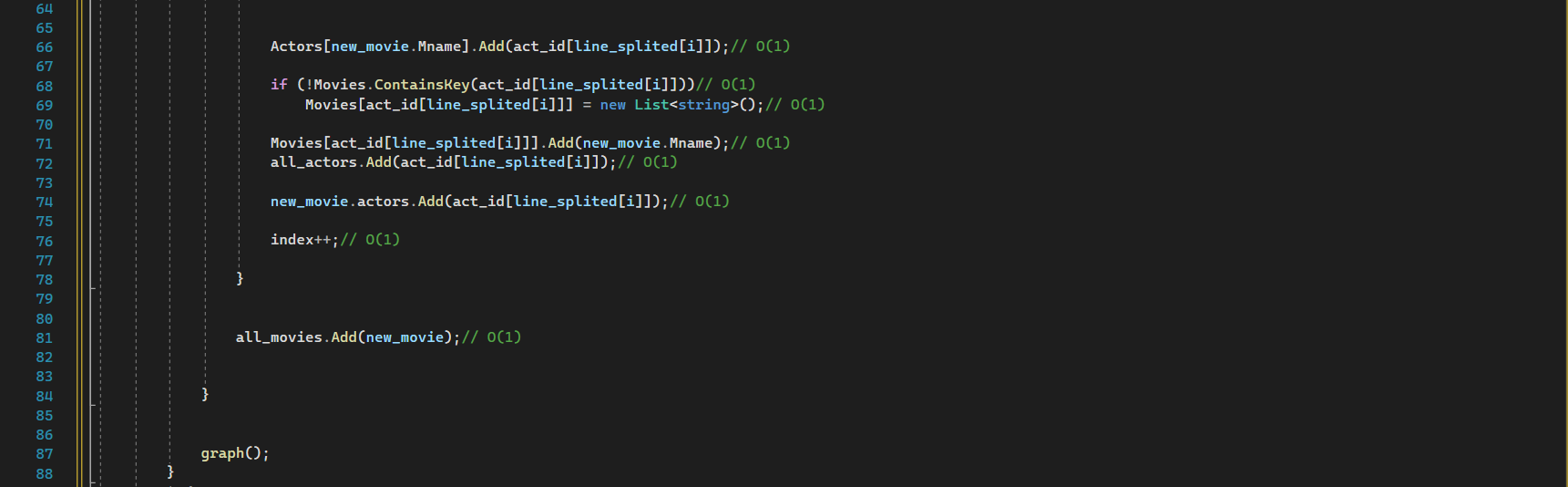
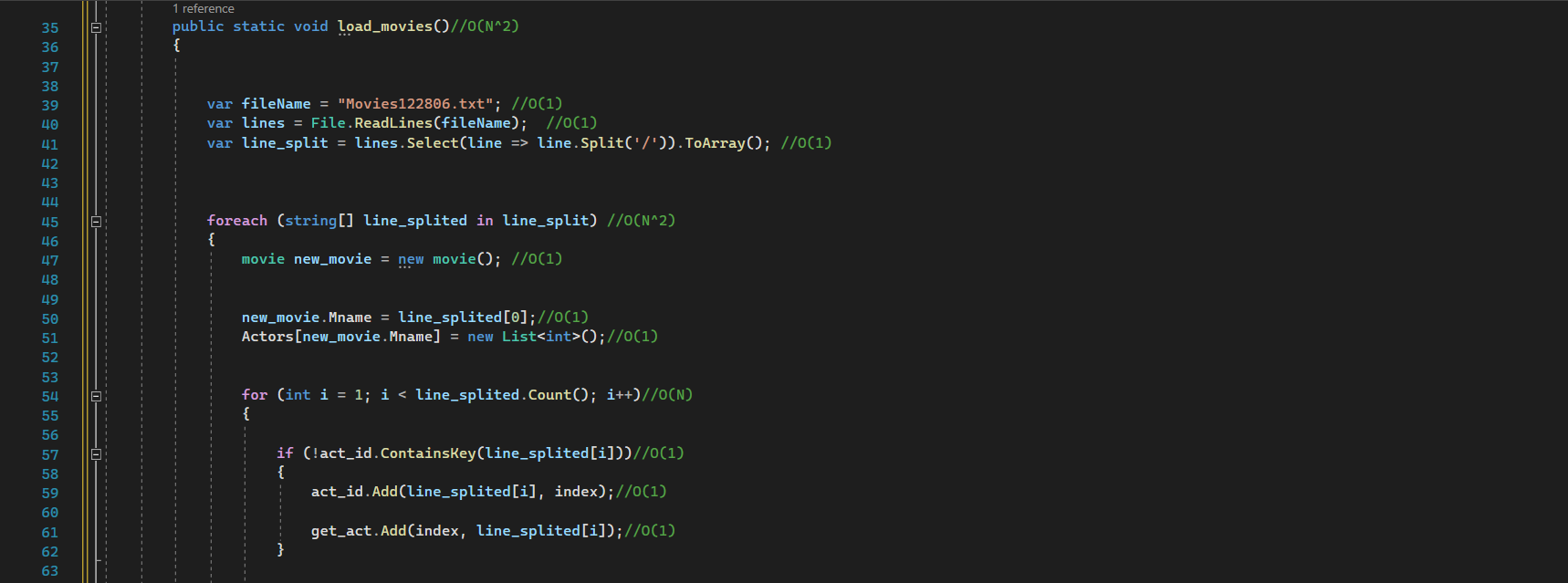
*Code complexity of code is O(V+E).*

**Time of execution of all test cases before and after optimization:**

|  |  |  |
| --- | --- | --- |
|  | before | After |
| Sample test time | **0.2 sec** | **0.03 sec ,32ms** |
| Small test case 1 time | **5.9 sec** | **0.6 sec, 698ms** |
| Small test case 2 time | **7.7 sec** | **1.4 sec, 1400ms** |
| Medium test case 1 query 85 time | **10.5 sec** | **3.2 sec, 3217ms** |
| Medium test case 1 query 4000 time | **241.3 sec** | **95sec, 95022.0ms** |
| Medium test case2 query 110 time | **21.4 sec** | **8 sec, 8197ms** |
| Medium test case2 query 2000 time | **292.73 sec** | **64 sec, 64831ms** |
| large test case query 26 time | **109.7 sec** | **61 sec, 61023ms** |
| large test case query 600 time | **Time exceeds 3 mins** | **35 sec, 35534ms** |
| Extreme test case query 22 time | **Time exceeds 3 mins** | **358 sec, 358504ms** |
| Extreme test case query 200 time | **Time exceeds 3 mins** | **70 sec, 70001ms** |

***In next pages***

***Screenshots of optimized source code in C# :***

****

\*\*\*\*\*\*\*\*

**Bonus point 2 (printing chain of actors related to strongest relation strength)**

**Text

Description automatically generated**