

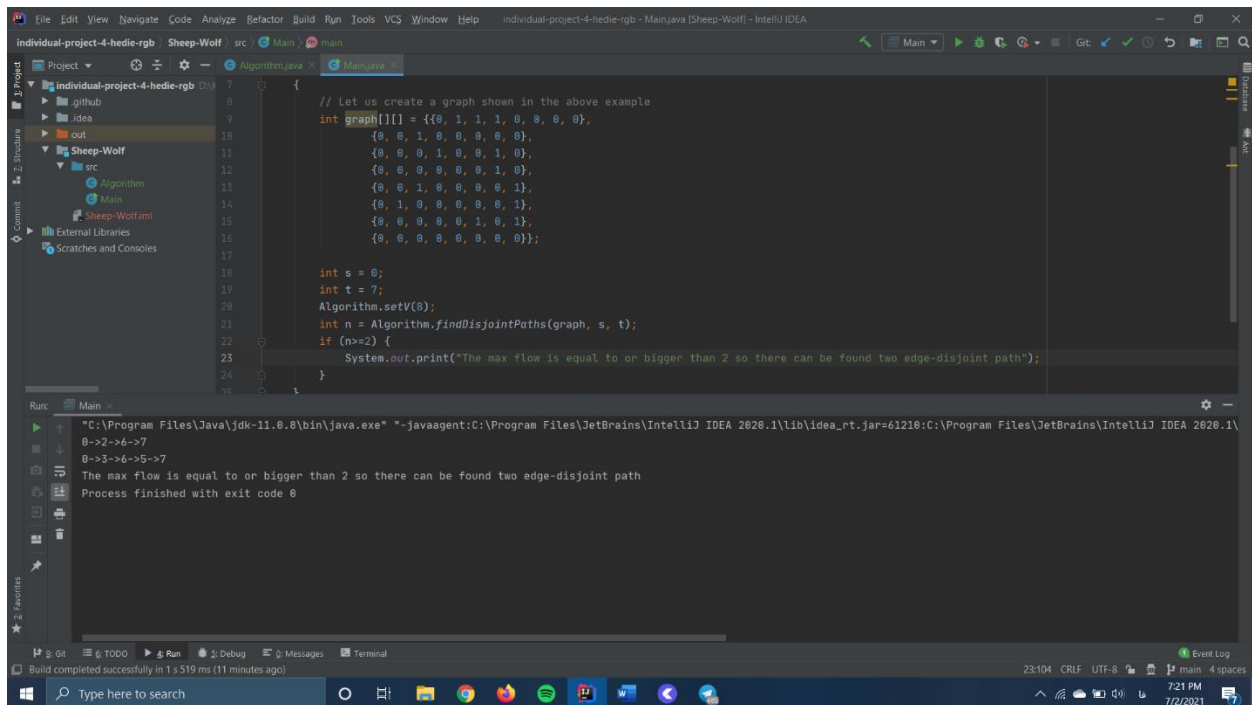
گرگ و گوسفند

این مسئله با کاهش به مسئله ی بیشینه شار حل می شود.

Source and destination -> Source and sink

Unit capacity for edges

با اعمال الگوریتم Ford-Fulkerson بیشینه شار پیدا می شود. چون ظرفیت به یک کاهش پیدا کرده است پس یال دوبار استفاده نمی شود پس بیشینه شار با بیشینه مسیر بدون یال مشترک برابر است.



```
1 // Let us create a graph shown in the above example
2 int graph[][] = {{0, 1, 1, 1, 0, 0, 0, 0},
3 {0, 0, 1, 0, 0, 0, 0, 0},
4 {0, 0, 0, 1, 0, 0, 1, 0},
5 {0, 0, 0, 0, 0, 0, 1, 0},
6 {0, 0, 1, 0, 0, 0, 0, 1},
7 {0, 1, 0, 0, 0, 0, 0, 1},
8 {0, 0, 0, 0, 0, 1, 0, 1},
9 {0, 0, 0, 0, 0, 0, 0, 0}};
10
11 int s = 0;
12 int t = 7;
13 Algorithm.setV(8);
14 int n = Algorithm.findDisjointPaths(graph, s, t);
15 if (n>2) {
16     System.out.print("The max flow is equal to or bigger than 2 so there can be found two edge-disjoint path");
17 }
18
19 Run: Main
20 "C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.1\lib\idea_rt.jar=61218:C:\Program Files\JetBrains\IntelliJ IDEA 2020.1\
21 0->2->6->7
22 0->3->6->5->7
23 The max flow is equal to or bigger than 2 so there can be found two edge-disjoint path
24 Process finished with exit code 0
```

در این الگوریتم از BFS استفاده شده است که در بدترین حالت قابل کاهش به $O(VE^2)$ هست اما چون بهینه سازی نشده است برابر $O(VE^3)$ هست.

هدیه آقاجانی

۹۶۹۷۹۴۳