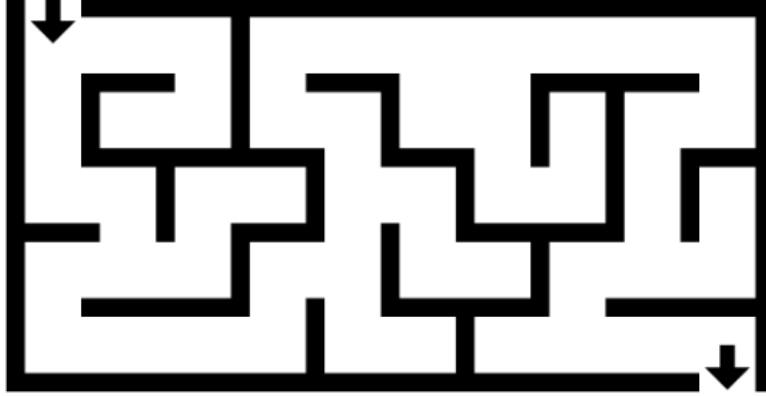


Adı	HIDAYATULLAH	Ders Adı	İLERİ ALGORİTMA ANALİZİ
Soyadı	ARGHANDABI	Lect.	Dr. İLKNUR DÖNMEZ
Ders Kodu	BYL565	Proje Deposu	https://github.com/hidayatarg/Depth-First-Search-DFS



Ödev



Şekildeki gibi bir labirenti dolaşan ve sonunda doğru yolu kendi kendine bulan LabirentCözen fonksiyonu yazınız.

1-Öncelikle labirentinizi graf olarak göstermeniz gerekmektedir.

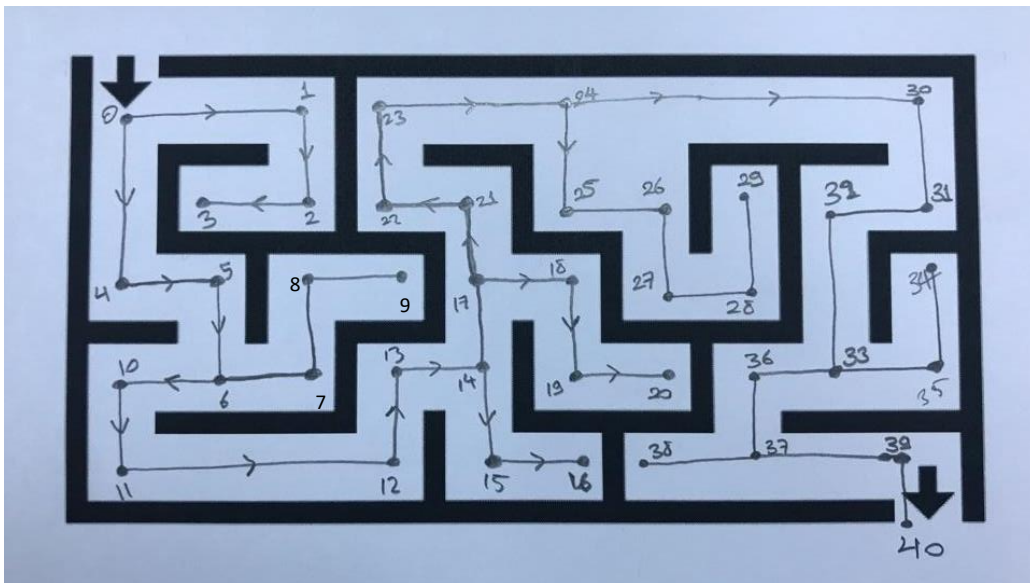
2-Grafı ya komşuluk matrisi kullanarak yada link list kullanarak ifade edebilirsiniz. Yine bunu kendiniz oluşturup kod içinde tanımlamanız gerekmektedir.

3-İstenilen dil kullanılabilir: c, java, c++, python

4-Kod çalıştırılıp uğranılan düğümler gösterilecektir.

Çözüm

Labirentinizi graf



AK = { 1, 4, 2, 3, 5, 6, 7, 10, 8, 9, 11, 12, 13, 14, 15, 17, 16, 18, 21, 19, 20, 22, 23, 24, 25, 30, 26, 27, 28, 29, 31, 32, 33, 34, 36, 35, 37, 38, 39, 40 };

AD = { 0, 2, 3, 4, 4, 5, 6, 8, 9, 10, 10, 11, 12, 13, 14, 16, 17, 17, 19, 20, 21, 21, 22, 23, 24, 26, 27, 28, 29, 30, 30, 31, 32, 33, 35, 36, 36, 37, 39, 39, 40, 40 };

Node Neighbors

0 1, 4

1 0, 2 (We only take the neighbors of forward direction of arrow exp: 2 enough)

2 1, 3 (We take 3)

3 2

4 0, 5 (we take 5)

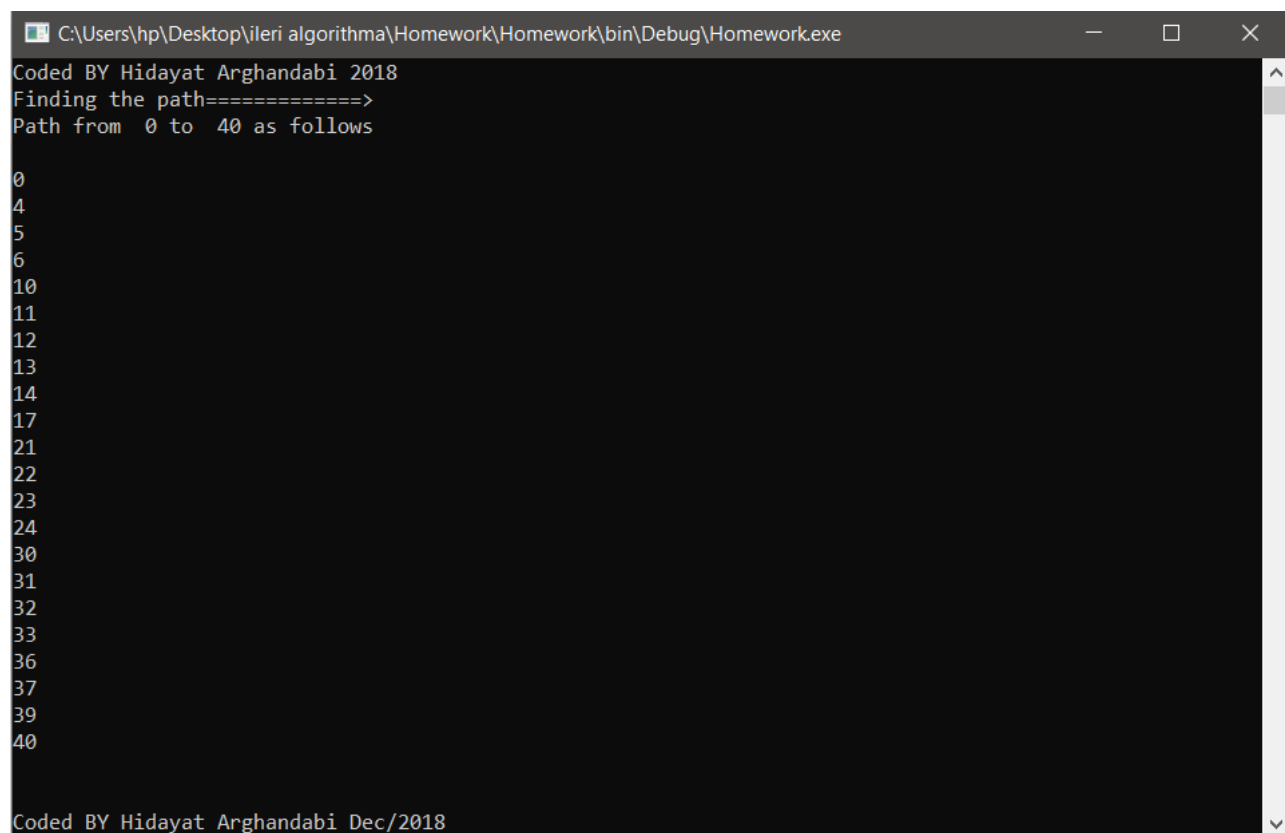
5 4, 6 (we take 6)

6 7, 10 (we take)

...

We complete this and place it AK with according to nodes in AD.

OUTPUT:



```
C:\Users\hp\Desktop\ileri algorithma\Homework\Homework\bin\Debug\Homework.exe
Coded BY Hidayat Arghandabi 2018
Finding the path=====>
Path from 0 to 40 as follows
0
4
5
6
10
11
12
13
14
17
21
22
23
24
30
31
32
33
36
37
39
40
Coded BY Hidayat Arghandabi Dec/2018
```

Code in Csharp

```
using System;
namespace Homework
{
    class Program
    {
        // Static AK and AD arrays
        static int[] AK = { 1, 4, 2, 3, 5, 6, 7, 10, 8, 9, 11, 12, 13, 14, 15, 17, 16, 18,
21, 19, 20, 22, 23, 24, 25, 30, 26, 27, 28, 29, 31, 32, 33, 34, 36, 35, 37, 38, 39, 40 };
        static int[] AD = { 0, 2, 3, 4, 4, 5, 6, 8, 9, 10, 10, 11, 12, 13, 14, 16, 17, 17,
19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 30, 31, 32, 33, 35, 36, 36, 37, 39, 39, 40,
40 };

        // Starting Node
        static int startNode = 0;
        // Ending Node
        static int endNode = 40;
        static void Main(string[] args)
        {
            // Program start
            Console.WriteLine("Finding the path=====>");
            Console.WriteLine("Path from " + startNode + " to " + endNode + " as
follows\n");
            // Print Each node path
            foreach (var y in FindPath(startNode, endNode, DepthFirstSearch(AK, AD)))
                Console.WriteLine(y);
            Console.WriteLine("\n\nCoded BY Hidayat Arghandabi Dec/2018 ");
            // Pause the Program
            Console.ReadLine();
        }

        // Finding the path
        private static int[] FindPath(int startNode, int endNode, int[] parentList)
        {
            int[] path = new int[parentList.Length];
            int currentNode = endNode;

            for (int i = 0; i < parentList.Length; i++)
            {
                path[i] = currentNode;
                if (currentNode == startNode)
                    return FlipArray(TrimArray(path, i + 1));
                currentNode = parentList[currentNode];
            }
            // Flip the array => path
            return FlipArray(path);
        }

        // Find the DFS
        private static int[] DepthFirstSearch(int[] AK, int[] AD)
        {
            int nodeTotal = AD.Length - 1;
            int[] parents = new int[nodeTotal];
            for (int p = 0; p < nodeTotal; p++)
                parents[p] = -1;

            for (int s = 0; s < nodeTotal; s++)
            {
                if (CheckList(parents, s))
                    continue;
                // Finding the Adjacent Node
                if (FindAdjacentNode(AK, AD, s).Length > 0)
                    VisitedDfs(s, parents, FindAdjacentNode(AK, AD, s));
            }
        }
    }
}
```

```

        else
            continue;
    }
    return parents;
}

private static void VisitedDfs(int s, int[] parents, int[] adjacentNode)
{
    // Recursive
    for (int v = 0; v < adjacentNode.Length; v++)
    {
        if (!CheckList(parents, adjacentNode[v]))
        {
            parents[adjacentNode[v]] = s;
            VisitedDfs(adjacentNode[v], parents, FindAdjacentNode(AK, AD,
adjacentNode[v]));
        }
    }
}

// Finding the Adjacent Node
private static int[] FindAdjacentNode(int[] AK, int[] AD, int node)
{
    int[] adjacentNode = new int[AD[node + 1] - AD[node]];
    for (int x = 0; x < adjacentNode.Length; x++)
    {
        adjacentNode[x] = AK[AD[node] + x];
    }
    return adjacentNode;
}

// Flipping the Array
private static int[] FlipArray(int[] array)
{
    int[] flippedArr = new int[array.Length];
    for (int i = 0; i < array.Length; i++)
    {
        flippedArr[array.Length - 1 - i] = array[i];
    }
    return flippedArr;
}

// The value with List
private static bool CheckList(int[] list, int value)
{
    for (int x = 0; x < list.Length; x++)
    {
        if (value == list[x])
            return true;
    }
    return false;
}

// Trimming
private static int[] TrimArray(int[] array, int length)
{
    int[] trimmedArr = new int[length];
    for (int i = 0; i < length; i++)
    {
        trimmedArr[i] = array[i];
    }
    return trimmedArr;
}
}
}

```