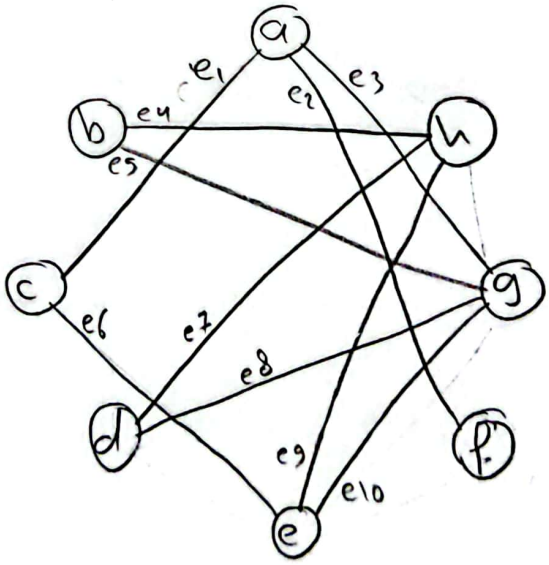
Malek Alismail /20253833

a) Örtü sayısı:

Bir alışveriş merkezinde birden fazla koridoru izleyen 8 kamera vardır. Yönetici, kameraların aynı koridorları izleyebilmesi için kamera sayısı azaltmak istiyor. Tüm koridorları kapsayacak şekilde seçilebilecek <u>en az</u> kamera sayısı nedir?

b)



$$f = (a+c)(a+f)(a+g)(b+g)(b+h)(c+e)(d+g)(d+h)(e+g)(e+h)$$

$$= (a+cf)(a+g) \cdots$$

$$= (a+cfg)(b+g) \cdots$$

$= (ab + \cancel{bcfg} + ag + cfg)(b+h)$ - ——— ———

$= (ab + agh + bcfg + cfgh)(c+e)$ - ——— ———

$= (abc + abe + acgh + aegh + bcfg + cfgh)(d+g)$ - —

$= (abcd + abcg + abde + abeg + \cancel{acdgh} + acgh + \cancel{adeg}$
$+ aegh + bcfg + cfgh)(d+h)$ - —

$= (abcd + \cancel{abcgh} + abde + \cancel{abegh} + acgh + aegh$
$+ bcdfg + \cancel{befgh} + cfgh)(e+g)(e+h)$

$= (\cancel{abcde} + abcdg + abde + acgh + aegh + bcdfg$
$+ cfgh)(e+h)$

$= (\cancel{abcdeg} + \cancel{abcdgh} + abde + acgh + aegh + bcdefg$
$+ \cancel{bcdfgh} + cfgh)$

$= (abde + acgh + aegh + bcdefg + cfgh)$

$\Rightarrow$ **Örtü sayısı = 4**

- $\{abde, acgh, aegh, cfgh\}$ Bu gruplardan birini seçip tüm koridorları kapsayan en az kamera sayısı buluruz (4)

Java kullanarak örtü sayısı buldum.

Derste kullandığımız algoritmayı kodlayıp gördüğümüz tüm örnekleri uygulayarak yaptım.

4 tane örnek graf var deneyebilirsiniz (yaptığım problem + derste gördüğümüz 3 örenği)

Örtü kümeleri ve örtü sayısı bulan bir koddur.

Ekran çıktısı en son sayfada var.

```java
import java.util.*;
import java.util.ArrayList;
import java.util.Arrays;
public class MinimumVertexCover {

    public static int leastCoveringNumber(int[][] graph , int edgesCount) {
        int n = graph.length; // vertices count

        String[] vertices = new String[n];

        String[] alphabet =
{"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s",
"t","u","v","w","x","y","z"};

        System.arraycopy(alphabet, 0, vertices, 0, n);

        int c=0; //counter for edges to add items to it
        String[][] edges = new String[edgesCount][2];

        //giving the edges names as letters
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) {
                if (graph[i][j] == 1){
                    edges[c][0] = vertices[j];
                    edges[c][1] = vertices[i];
                    c++;
                }
            }
        }
        System.out.println(Arrays.deepToString(edges));



        String first, second; // for the identical strings when 2 elements
are concatenated
        List<String> f = new ArrayList<>();
        List<String> newf = new ArrayList<>();

        //spread first two paranthesis of edges, first 2 edges
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                first = edges[0][i];
                second = edges[1][j];
                if (first==second){
                    f.add(first);
                }
                else{
                    f.add(first+second);
                }
            }
        }
```

```
        //remove elements which includes another element as a substring
        for (String element : f){
            for (String otherElement : f){
                if (element != otherElement &&
otherElement.contains(element)){
                    newf.add(otherElement);
                }
            }
        }
        f.removeAll(newf);
        newf.clear();



        int s; //size of f
        boolean containsAll; //check if an element is a substring of another
element

        //starting from 3rd edge to the last one
        for (int i = 2; i < edgesCount; i++) {
            s=f.size(); //size of f is changing in every loop so it should be
a const once per loop
            for (int j = 0; j < 2; j++) {
                for (int k = 0; k< s; k++) {
                    if (f.get(k) == edges[i][j] ||
f.get(k).contains(edges[i][j])){
                        f.add(f.get(k));
                    }
                    else {
                        f.add(f.get(k) + edges[i][j]);
                    }
                }
            }

            f.subList(0,s).clear(); //remove the elements from the last loop




            //remove duplicate elements
            for (int j = 0; j < f.size(); j++) {
                for (int k = j+1; k < f.size(); k++) {
                    if (f.get(j) == f.get(k)){
                        f.remove(f.get(j));
                        break;
                    }
                }
            }
```

```
            //remove the elements that have another element as a substring
            for (String element : f){
                for (String otherElement : f){
                    if (element != otherElement){
                        containsAll = true;
                        for (int h = 0; h < element.length(); h++) {
                            char character = element.charAt(h);
                            if (otherElement.indexOf(character) == -1){
                                containsAll = false;
                                break;
                            }
                        }
                        if (containsAll){
                            newf.add(otherElement); //add the elements to
another list to subtract the elements we want to remove
                        }
                    }
                }
            }
            f.removeAll(newf); //subtract the lists
            newf.clear();
        }
        System.out.println(f);



        //get covering number
        int minSize = Integer.MAX_VALUE;
        for (String element : f) {
            int elementSize = element.length();
            if (elementSize < minSize) {
                minSize = elementSize;
            }
        }


        return minSize;
    }

    public static void main(String[] args) {
        int m = 0; //edges number
        int[][] graph = {
                {0, 0, 1, 0, 0, 1, 1, 0},
                {0, 0, 0, 0, 0, 0, 1, 1},
                {1, 0, 0, 0, 1, 0, 0, 0},
                {0, 0, 0, 0, 0, 0, 1, 1},
                {0, 0, 1, 0, 0, 0, 1, 1},
                {1, 0, 0, 0, 0, 0, 0, 0},
                {1, 1, 0, 1, 1, 0, 0, 0},
                {0, 1, 0, 1, 1, 0, 0, 0}
        };
```

```
        int [][] graph2 = {
                {0, 1, 0, 0, 0, 1, 0, 0},
                {1, 0, 1, 1, 0, 1, 0, 0},
                {0, 1, 0, 0, 1, 0, 0, 0},
                {0, 1, 0, 0, 1, 0, 1, 0},
                {0, 0, 1, 1, 0, 0, 0, 1},
                {1, 1, 0, 0, 0, 0, 1, 0},
                {0, 0, 0, 1, 0, 1, 0, 1},
                {0, 0, 0, 0, 1, 0, 1, 0}
        };

        int [][] graph3 = {
                {0, 1, 1, 0, 0, 0, 0},
                {1, 0, 0, 1, 1, 1, 0},
                {1, 0, 0, 0, 1, 0, 0},
                {0, 1, 0, 0, 1, 0, 0},
                {0, 1, 1, 1, 0, 0, 1},
                {0, 1, 0, 0, 0, 0, 1},
                {0, 0, 0, 0, 1, 1, 0}
        };

        int [][] graph4 = {
                {0, 1, 0, 0, 0, 1},
                {1, 0, 1, 1, 1, 0},
                {0, 1, 0, 0, 0, 0},
                {0, 1, 0, 0, 0, 0},
                {0, 1, 0, 0, 0, 1},
                {1, 0, 0, 0, 1, 0}
        };

        //calculating edges number
        for (int i = 0; i < graph.length; i++) {
            for (int j = 0; j < graph.length; j++) {
                if (graph[i][j] == 1){
                    m++;
                }
            }
        }
        m=m/2;


        int leastCoveringNumber = leastCoveringNumber(graph,m);
        System.out.println("Least covering number: " + leastCoveringNumber);
    }

}
```

Graph: Kendi yaptığım probleminden

Graph 2: Hafta 2 sayfa 52

Graph 3: Hafta 2 sayfa 56

Graph 4: Hafta 2 sayfa 58

Not: Başka bir graf denemek için kodun son sayfadaki for loop içinde 3 tane “graph” yerine + en sondan 2. satır “graph” yerine, “graph2, graph3 …” yazabilirsiniz.

```
no usages
public static void main(String[] args) {
    int m = 0; //edges number
    int[][] graph = {
            {0, 0, 1, 0, 0, 1, 1, 0},
            {0, 0, 0, 0, 0, 0, 1, 1},
            {1, 0, 0, 0, 1, 0, 0, 0},
            {0, 0, 0, 0, 0, 0, 1, 1},
            {0, 0, 1, 0, 0, 0, 1, 1},
            {1, 0, 0, 0, 0, 0, 0, 0},
            {1, 1, 0, 1, 1, 0, 0, 0},
            {0, 1, 0, 1, 1, 0, 0, 0}
    };
```

MinimumVertexCover

```
C:\Users\abobr\.jdks\openjdk-20\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ ID
[[a, c], [c, e], [a, f], [a, g], [b, g], [d, g], [e, g], [b, h], [d, h], [e, h]]
[aebd, cfgbde, aegh, cagh, cfgh]
Least covering number: 4

Process finished with exit code 0
```

```
            {1, 1, 0, 1, 1, 0, 0, 0},
            {0, 1, 0, 1, 1, 0, 0, 0}
    };

    int [][] graph2 = {
            {0, 1, 0, 0, 0, 1, 0, 0},
            {1, 0, 1, 1, 0, 1, 0, 0},
            {0, 1, 0, 0, 1, 0, 0, 0},
            {0, 1, 0, 0, 1, 0, 1, 0},
            {0, 0, 1, 1, 0, 0, 0, 1},
            {1, 1, 0, 0, 0, 0, 1, 0},
            {0, 0, 0, 1, 0, 1, 0, 1},
            {0, 0, 0, 0, 1, 0, 1, 0}
    };

    int [][] graph3 = {
            {0, 1, 1, 0, 0, 0, 0},
```

MinimumVertexCover

```
C:\Users\abobr\.jdks\openjdk-20\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
[[a, b], [b, c], [b, d], [c, e], [d, e], [a, f], [b, f], [d, g], [f, g], [e, h], [g, h]]
[acdfeg, bcdfeg, beag, beadfh, acdfh, bcdfh, acdbgh]
Least covering number: 4

Process finished with exit code 0
```

```
                {0, 0, 0, 1, 0, 1, 0, 1},
                {0, 0, 0, 0, 1, 0, 1, 0}
        };

        int [][] graph3 = {
                {0, 1, 1, 0, 0, 0, 0},
                {1, 0, 0, 1, 1, 1, 0},
                {1, 0, 0, 0, 1, 0, 0},
                {0, 1, 0, 0, 1, 0, 0},
                {0, 1, 1, 1, 0, 0, 1},
                {0, 1, 0, 0, 0, 0, 1},
                {0, 0, 0, 0, 1, 1, 0}
        };

        int [][] graph4 = {
                {0, 1, 0, 0, 0, 1},
```

MinimumVertexCover

```
C:\Users\abobr\.jdks\openjdk-20\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
[[a, b], [a, c], [b, d], [b, e], [c, e], [d, e], [b, f], [e, g], [f, g]]
[bcef, abef, adef, bceg, abeg, bcdg]
Least covering number: 4

Process finished with exit code 0
```

```
                {0, 1, 0, 0, 1, 0, 0},
                {0, 1, 1, 1, 0, 0, 1},
                {0, 1, 0, 0, 0, 0, 1},
                {0, 0, 0, 0, 1, 1, 0}
        };

        int [][] graph4 = {
                {0, 1, 0, 0, 0, 1},
                {1, 0, 1, 1, 1, 0},
                {0, 1, 0, 0, 0, 0},
                {0, 1, 0, 0, 0, 0},
                {0, 1, 0, 0, 0, 1},
                {1, 0, 0, 0, 1, 0}
        };

        //calculating edges number
        for (int i = 0; i < graph4.length; i++) {
```

MinimumVertexCover

```
C:\Users\abobr\.jdks\openjdk-20\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
[[a, b], [b, c], [b, d], [b, e], [a, f], [e, f]]
[bae, acde, bf]
Least covering number: 2

Process finished with exit code 0
```