

小 v 是公司的运维工程师，现有一个有关应用程序部署的任务如下：

- 1、一台服务器的**磁盘空间**、**内存**是固定的，现在有 N 个应用程序要部署；
- 2、每个应用程序所需要的**磁盘**、**内存**不同，每个应用程序**允许访问的用户数**也不同，且同一个应用程序不能在一台服务器上部署多个。

对于一台服务器而言，如何组合部署应用程序能够使得单台服务器允许访问的用户数**最多**？

```
import java.io.*;
import java.util.*;

/**
 * Welcome to vivo !
 */

public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String inputStr = br.readLine();
        String[] input = inputStr.split(" ");
        int totalDisk = Integer.parseInt(input[0]);
        int totalMemory = Integer.parseInt(input[1]);
        List<Service> services = parseServices(input[2].split("#"));
        int output = solution(totalDisk, totalMemory, services);
        System.out.println(output);
    }

    private static int solution(int totalDisk, int totalMemory, List<Service> services) {

        // TODO Write your code here
        int len = services.size();
        int[][][] dp = new int[len + 1][totalDisk + 1][totalMemory + 1];
        for(int i = 1; i <= len; i++)
            for(int j = totalDisk; j > 0; j--)
                for(int k = totalMemory; k > 0; k--){
                    if(j >= services.get(i - 1).getDisk() && k >= services.get(i - 1).getMemory()){
                        dp[i][j][k] = Math.max(dp[i - 1][j][k],
                        dp[i - 1][j - services.get(i - 1).getDisk()][k - services.get(i -
1).getMemory()]
                        + services.get(i - 1).getusers());
                    }else{
                        dp[i][j][k] = dp[i - 1][j][k];
                    }
                }
    }
}
```

```

        return dp[len][totalDisk][totalMemory];
    }

    private static List<Service> parseServices(String[] strArr) {
        if (strArr == null || strArr.length == 0) {
            return new ArrayList<Service>(0);
        }
        List<Service> services = new ArrayList<>(strArr.length);
        for (int i = 0; i < strArr.length; i++) {
            String[] serviceArr = strArr[i].split(",");
            int disk = Integer.parseInt(serviceArr[0]);
            int memory = Integer.parseInt(serviceArr[1]);
            int users = Integer.parseInt(serviceArr[2]);
            services.add(new Service(disk, memory, users));
        }
        return services;
    }

    static class Service {
        private int disk;

        private int memory;

        private int users;

        public Service(int disk, int memory, int users) {
            this.disk = disk;
            this.memory = memory;
            this.users = users;
        }

        public int getDisk() {
            return disk;
        }

        public void setDisk(int disk) {
            this.disk = disk;
        }

        public int getMemory() {
            return memory;
        }

        public void setMemory(int memory) {

```

```

        this.memory = memory;
    }

    public int getUsers() {
        return users;
    }

    public void setUsers(int users) {
        this.users = users;
    }
}
}

```

小 v 在 vivo 手机的应用商店中下载了一款名为“一维消消乐”的游戏，介绍如下：

- 1、给出一些不同颜色的豆子，豆子的颜色用数字（0-9）表示，即不同的数字表示不同的颜色；
- 2、通过不断地按行消除相同颜色且连续的豆子来积分，直到所有的豆子都消掉为止；
- 3、假如每一轮可以消除相同颜色的连续 k 个豆子（ $k \geq 1$ ），这样一轮之后小 v 将得到 $k*k$ 个积分；
- 4、由于仅可按行消除，不可跨行或按列消除，因此谓之“一维消消乐”。

请你帮助小 v 计算出最终能获得的**最大**积分。



```

class Solution {
    int n;
    int[][] F;
    int[] prev;

    public int removeBoxes(int[] boxes) {
        n = boxes.length;
        prev = new int[n];
        F = new int[n][n];
        for (int i = 0; i < n; i++) {
            prev[i] = -1;
            for (int j = 0; j < n; j++) {
                F[i][j] = -1;
            }
        }
    }
}

```

```

        }
        for (int j = i - 1; j >= 0; --j) {
            if (boxes[i] == boxes[j]) {
                prev[i] = j;
                break;
            }
        }
    }
    return f(0, n - 1);
}

public int f(int i, int j) {
    if (i > j) {
        return 0;
    }
    if (F[i][j] != -1) {
        return F[i][j];
    }
    if (i == j) {
        return F[i][j] = 1;
    }
    return F[i][j] = g(i, j, prev[j], 1);
}

public int g(int i, int j, int r, int m) {
    if (i > j) {
        return 0;
    }
    if (r < i) {
        return f(i, j - 1) + m * m;
    }
    // 如果相邻，可以消除许多中间状态
    if (r + 1 == j) {
        return g(i, r, prev[r], m + 1);
    }
    int rv = prev[r];
    return Math.max(
        g(i, r, rv, m + 1) + f(r + 1, j - 1),
        g(i, j, rv, m)
    );
}
}

```

小 v 所在的公司即将举行年会，年会方案设计过程中必不可少的一项就是抽奖活动。小 v

在本次活动中被委以重任,负责抽奖活动的策划;为了让中奖的礼物更加精美且富有神秘感,打算采用礼品盒来包装奖品,此时小 v 发挥了自己的创意想捉弄一下获奖的同事,便采取了多重包装来包装奖品。

现给出一个字符串,并假定用一对圆括号()表示一个**礼品盒**,0 表示**奖品**,你能据此帮获奖者算出**最少**要拆多少个礼品盒才能拿到奖品吗?

```
/*
本来是栈, 但是这里很简单的用计数器就可以了
*/
int solution(string str)
{
    int L = 0;
    int sum = 0;
    for (int i = 0;i<str.size();i++){
        if(str[i] == '(')    L++;
        else if(str[i] == ')')    L--;
        else if(str[i] == '0') {
            sum = L;
            break;
        }
    }

    // TODO Write your code here

    return sum;
}
```