-Popis myšlienky:

Algoritmus bude postupne skúšať podmienky ktoré určia či je čokoláda rozdeliteľná a keď u jednej zistí že nie tak napíše nie a skončí.

-Popis dátových štruktúr:

4 inty z vstupom, string s vstupom, 2d pole symbolizujúce začiatky a konce jednotlivých kúskov a int y ktorý obsahuje aktuálnu hodnotu indexu v poli počas zapisovania do pola.

-Popis algoritmu:

Algoritmus zoberie vstup, vytvorí funkciu kt. vypíše nie a skončí program keď zistí že podmienka rozdeliteľnosti neplatí(č. sa nedá rozdeliť). Overovanie rozdeliteľnosti:

1. Algoritmus zistí či je počet kúskov horkej č. rovný počtu trojstenákov.

   Vytvorí pole s rozsahmi častí č. a zapíše doň pozíciu daného kúsku horkej č.(to je min rozsah kúsku)

   Začiatok a koniec jednej časti bude nastavený na pozíciu horkého kúsku.

2. Otestuje základné podmienky rozdeliteľnosti: n*r<k , n*l>k, či nie je prvý kúsok horkej moc ďaleko od začiatku, či nie je druhý kúsok horkej moc blízko začiatku, či nie je posledný kúsok horkej moc ďaleko od konca a či nie je predposledný kúsok horkej moc blízko pri konci.
3. Nastaví min začiatky a konce prvej a poslednej časti čokolády.
4. Otestuje či nie je medzera medzi 2 kúskami horkej viac ako maximálna medzera.
5. Otestuje či sa dá zostrojiť každá časť tak aby bola veľkosť každej časti väčšia ako min veľkosť. To robí tak že skúša zväčšovať časť smerom vľavo a keď je málo miesta naľavo tak pokračuje aj smerom vpravo od kúska horkej v tejto časti, takto neuberá miesto iným častiam.

   Ak ani jedna z týchto podmienok nie je splnená čokoláda sa dá rozdeliť a vypíše sa áno.

-Zdôvodnenie správnosti:

Algoritmus vždy nájde správnu odpoveď preto že skúša všetky podmienky rozdeliteľnosti a nepodarila sa mi nájsť výnimka.

-Odhad časovej zložitosti:

Časová zložitosť je lineárna a závislá od súčtu k a n.

-Odhad pamäťovej zložitosti:

Pamäťová zložitosť je lineárna a závislá od súčtu k a n.

Kód:

```
n,k,l,r =(map(int, input().split(" ")))#vstup
vstup=str(input())#vstup

def nie():   #funkcia vypisie nie a skonci program
    print("nie")
    exit()
if vstup.count("1")!=n :#ked nieje rovnaky pocet kuskov horkej ako ludi nemoze dostat
kazdy 1ks
    nie()
y=0 #vytvorenie int y ktory bude udrziavat aktualnu hodnotu v poli
```

```
pole=[[-1,-1] for i in range(n)] #vytvorenie 2d pola kt. obsahuje zaciatok a koniec kusku cokolady pre i-teho trojstenaka
for i in range(k): #zapisovanie rozsahu kusku
    if vstup[i]=="1":
        pole[y][0]=i #najskor sa zapise rozsah tak aby obsahoval kusok iba 1 ks horkej
        pole[y][1]=i
        y+=1
if n*r<k or n*l>k or pole[0][0]>=r or n-pole[-1][1]>r or pole[1][0]<l:#testovanie zakladnych podmienok rozdelitelnosti
    nie()
if n>1:
    if n-pole[-2][1]-1>l:#testovanie zakladnych podmienok rozdelitelnosti
        nie()

pole[0][0]=0#nastavenie zaciatku 1. kusku
pole[0][1]=max(pole[0][1],l-1) #nastavenie min konca 1. kusku
pole[-1][1]=k-1#nastavenie konca posledneho kusku
pole[-1][0]=min(pole[-1][0],k-l)#nastavenie min zaciatku 1.kusku
for i in range(1,n):
    if pole[i][1]-pole[i-1][0]+1>r*2:#testovanie ci nie je medzera medzi 2 kuskami horkej viac ako maximalna medzera
        nie()

for i in range(1,n-1):#testovanie ci sa da spravit min velkost kusku
    if pole[i][0]-pole[i-1][1]>=l:#najskor program skusi aby kusok zacinal co najviac vlavo aby neuberal miesto dalsiemu
        pole[i][0]-=(l-1)
    else:
        pole[i][0]=pole[i-1][1]+1
        if pole[i][0]+l-1>=pole[i+1][0]:#testovanie ci sa da spravit kusok od konca toho predtym po zaciatok toho potom
            nie()
        else:
            pole[i][1]=pole[i][0]+l-1

print("ano")#ked ani jedna podmienka nie je splnena tak sa da cokolada rozdelit a vypise sa ano a program skonci
```