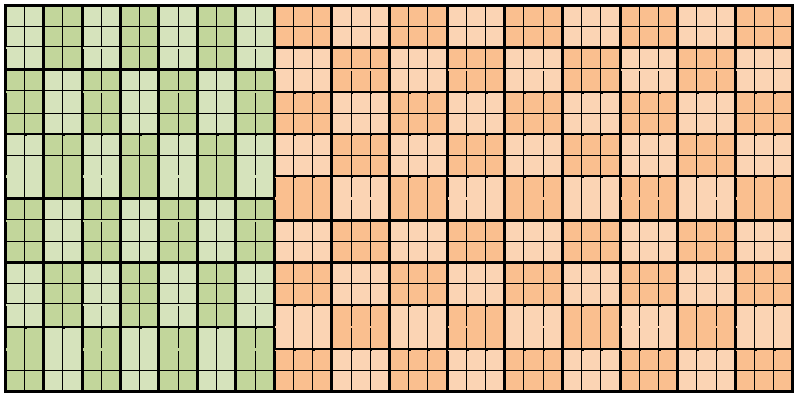
Pokrivanje pravokotnika

Peter Nose

19. marec 2009

Poskušajmo rešiti naslednji problem:

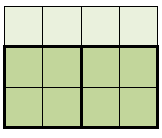
Ali je možno pravokotnik velikosti $v \times s$ pokriti z dominami velikosti $a \times b$?



Slika 1: Pokritje pravokotnika 18×41 z dominami 2×3

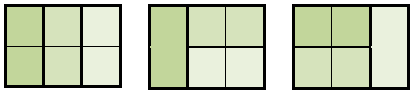
Predno se lotimo splošnega problema si najprej oglejmo dva konkretna primera:

- Vzemimo pravokotnik velikosti 3×4 in ga poskušajmo pokriti z dominami velikosti 2×2. Seveda vsi poskusi propadejo, saj če bi bilo pravokotnik možno pokriti z dominami, potem bi morali biti tako višina $v = 3$ kot širina $s = 4$ deljivi z 2.



Slika 2: Neuspešen poskus pokritja pravokotnika 3×4 z dominami 2×2

- Za drugi primer vzemimo pravokotnik velikosti 2×3 in ga poskušajmo pokriti z dominami velikosti 1×2. V tem primeru nimamo enoličnega pokritja, temveč je možnih več pokritij.



Slika 3: Vsa možna pokritja pravokotnika 2×3 z dominami 1×2

Na osnovi zgornjih primerov lahko opazimo, da pokritje ne obstaja vedno. Prav tako ni nujno, da je enolično. Pokritja se med seboj razlikujejo v postavitvi posameznih domin, vsem pa je skupno to, da je vsaka domina postavljen ali navpično ali vodoravno.

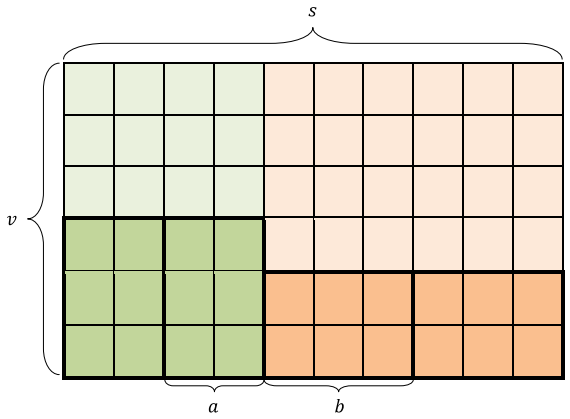
Za začetek bomo pravokotnik $v \times s$ s pomočjo navpične črte razdelili na 2 dela. Levi del bomo poskušali pokriti z navpično obrnjenimi dominami $b \times a$, medtem ko bomo desni del poskušali pokriti z vodoravno obrnjenimi dominami $a \times b$. Število domin, ki se nahajajo v posamezni vrsti levega oz. desnega dela bomo označili z x oz. z y (pokritje na sliki 1 ima v vsaki vrsti levega dela $x = 7$ domin obrnjenih navpično in v vsaki vrsti desnega dela $y = 9$ domin obrnjenih vodoravno). Ker obe števili označujeta število domin, sta obe nenegativni. V primeru, da tako pokritje obstaja, morajo veljati naslednje lastnosti.

(a) Če je $x > 0$, potem mora širina domine deliti višino celotnega pravokotnika. Torej $b \mid v$.

(b) Če je $y > 0$, potem mora višina domine deliti višino celotnega pravokotnika. Torej $a \mid v$.

(c) Ker je celotni pravokotnik pokrit, je pokrit tudi spodnji rob pravokotnika. En del je pokrit s prvo vrstico levega dela, drugi del pa s prvo vrstico desnega dela. Ker je na levem delu x domin obrnjenih navpično in ker je na desnem delu y domin obrnjeno vodoravno, mora veljati enačba

$$a \cdot x + b \cdot y = s. \tag{1}$$



Slika 4: Oznake

Enačba (1) ima neskončno mnogo rešitev. Kadar nas zanimajo samo celoštevilske rešitve, pravimo enačbi *linearna diofantska enačba*. Seveda pa vsaka enačba take oblike ni vedno rešljiva. Za primer lahko vzamemo enačbo $2x+4y = 1$. Tu je leva stran enačbe deljiva z 2, medtem ko desna stran ni. Ker iščemo samo celoštevilske rešitve, ta enačba ni rešljiva. Rešljivosti diofantske enačbe lahko preverimo s pomočjo *Evklidovega algoritma* za izračun največjega skupnega delitelja. Izkaže se, da je enačba rešljiva natanko tedaj, ko največji skupni delitelj števil a in b deli s.

Diofantske enačbe sodijo v področje matematike, ki mu pravimo *teorija števil*. Slednje je svoj hiter razvoj doživelo predvsem po *Fermatu* (1601-1665), ki ga uvrščamo med največje francoske matematike.

Diofantsko enačbo (1) rešimo tako, da najprej poiščemo eno rešitev (*partikularna rešitev*), ter nato s pomočjo le te poiščemo splošno rešitev. Eno samo rešitev lahko poiščemo s pomočjo *verižnih ulomkov*. Ulomek a/b zapišemo z *verižnimi koeficienti*. Nato zadnji koeficient zbrišemo in koeficiente pretvorimo nazaj v ulomek. Imenovalec in števec sta tako (do predznaka natančno) rešitvi diofantske enačbe. Označili ju bomo z x_0 in y_0. Splošna rešitev diofantske enačbe je

$$x = x_0 - b \cdot n \text{ in } y = y_0 + a \cdot n,$$

kjer je n poljubno celo število.

V našem konkretnem primeru vsaka rešitev diofantske enačbe ne ustreza pokritju pravokotnika. Ustrezne so samo nenegativne rešitve, saj števili x in y predstavljata število domin. Ker mora naša rešitev zadoščati tudi točkama (a) in (b), je dobro preveriti dve rešitvi enačbe (1). Pri prvi rešitvi število y_1 postavimo na najmanjšo možno nenegativno vrednost, pri drugi pa x_2. Če sta ustrezni števili x_1 oz. y_2 tudi nenegativni, preverimo še točki (a) in (b). Torej ali je višina pravokotnika *kongruentna* 0 po *modulu* a oz. b, v primeru da je rešitev x oz. y različna od 0.

Do sedaj smo predpostavljali, da je pravokotnik z navpično črto razdeljen na levi in desni del. Podobno bi lahko pravokotnik s pomočjo vodoravne črte razdelili na zgornji in spodnji del, ter nato vsak del pokrili posebej. Tako pokritje pa lahko poiščemo tudi z zgornjim postopkom, če zamenjamo višino in širino pravokotnika.

Naloga pa s tem še ni popolnoma rešena. Namreč naš postopek poskuša najti samo tiste rešitve, pri katerih je možno levi oz. zgornji del v celoti pokrit z navpično obrnjenimi dominami in desni oz. spodnji del pokriti z vodoravno obrnjenimi dominami. Izkaže se, da vsaj ena taka rešitev obstaja, če za pravokotnik obstaja pokritje. Premislek te trditve prepustimo bralcu.

Šifriranje s pomočjo pokritja pravokotnika

Denimo da imamo osebi Anito in Bojana, ki bi med seboj rada izmenjevala *skrivnostna sporočila*. Za lažje razumevanje bomo predpostavili, da bo pisma pošiljal Bojan, medtem ko jih bo Anita prejemala.

V nadaljevanju bomo s pomočjo pokritja pravokotnika skonstruirali preprosto metodo šifriranja. Skonstruirana šifra bo temeljila na skupnem skrivnem ključu, ki bo znan samo Aniti in Bojanu. S pomočjo tega ključa bomo lahko besedila šifrirali in odšifrirali. Takim šifram pravimo tudi simetrične šifre, saj uporabljajo isti ključ za šifriranje in odšifriranje. Če šifrirni in odšifrirni ključ nista enaka, potem govorimo o

asimetričnih šifrah. Primer asimetrične šifre je *metoda RSA*, ki je ena izmed najbolj znanih šifrirnih metod v uporabi.

Problem simetričnih šifer je v tem, da se morata Anita in Bojan dogovoriti za skupni ključ. To lahko storita tako, da se v tajnosti srečata. Taka izmenjava ključev ni ravno primerna, če Anita in Bojan ne živita blizu. V ta namen obstaja kar nekaj načinov, kako se na daljavo dogovoriti za skupni ključ. En izmed njih je tudi *Diffie-Hellmanov dogovor o ključu*.

Predpostavili bomo, da si bosta Anita in Bojan izmenjavala samo besedila napisana z velikimi črkami brez šumnikov, kjer bodo besede ločene s presledkom. Črke bomo *zakodirali* s števili od 1 do 22 ('A' se zakodira v 1, 'B' v 2, ..., 'Z' v 22), presledek pa bomo zakodirali z 0.

Anita in Bojan si izbereta dve praštevili večji od 22. Označili ju bomo z a in b in bosta predstavljali velikost domin, s katerimi bosta pokrivala pravokotnik. Skupni ključ je torej par (a, b). Šifriranje poteka na naslednji način:

1. Bojan razbije besedilo na pare znakov (če je besedilo lihe dolžine doda na konec besedila presledek).

2. Za vsak par znakov (z_{2i}, z_{2i+1}):

   (a) znaka z_{2i} in z_{2i+1} zakodira v števili x in y,

   (b) Aniti pošlje število $s_i = a \cdot x + b \cdot y$ (s_i predstavlja širino pravokotnika).

Odšifriranje poteka na naslednji način:

1. Anita za vsako prejeto število s_i naredi naslednje:

   (a) pokrije pravokotnik velikosti $ab \times s_i$ z dominami velikosti $a \times b$ in dobi rešitev (x, y),

   (b) iz števil x in y odkodira znaka z_{2i} in z_{2i+1}.

Če bi za skupni skrivni ključ vzeli $(31, 37)$, potem bi se sporočilo

"BI SLA Z MENOJ V KINO"

zašifriralo v seznam števil

$$[463, 734, 477, 882, 549, 741, 903, 845, 475, 865, 533]$$

in odšifriralo v

"BI SLA Z MENOJ V KINO ".

Skonstruirali smo preprosti algoritem za šifriranje sporočil. Vprašanje, ki se sedaj pojavi je, ali lahko iz šifriranega sporočila razberemo osnovno sporočilo, ne da bi poznali tajni ključ (a, b). Dandanes je odgovor na to vprašanje da, včasih pa so take *šifre* veljale za nezlomljive. Da je taka šifra šibka nam pove že podatek, da se par črk (z_1, z_2) vedno zašifrira v isto število s.

1 Dodatek

1.1 Pokritje pravokotnika

```
# Preveri ce je mozno pravokotnik velikosti v x s pokriti z
   dominami velikosti a x b
def pokritje(v, s, a, b):
    if pokritje1(v, s, a, b) == () or pokritje1(s, v, a, b)
       == ():
        print "Ne"
    else:
        print "Da"

# Vrne resitev pokritja pravokotnika z visino v in sirino s
   z dominami velikosti a x b
def pokritje1(v, s, a, b):
    # preveri ce je mozno domine postaviti tako, da
       pokrijejo celotno sirino
    resitev = resiDiofantskoEnacbo(a, b, s)

    if len(resitev) == 0:   # diofantska enacba nima resitev
        return ()

    if resitev[0] < 0 or resitev[1] < 0:
        print "Diofantska enacba nima nenegativne resitve"

    x = resitev[0]  # stevilo domin, ki smo jih postavili
       navpicno (torej z visino b in sirino a)
    y = resitev[1]  # stevilo domin, ki smo jih postavili
       vodoravno (torej z visino a in sirino b)

    # polozili smo vsaj eno domino navpicno, zato mora njena
        dolzina deliti celotno visino povrsine
    if x != 0 and v % b != 0:
        return ()

    # polozili smo vsaj eno domino vodoravno, zato mora
       njena visina deliti celotno visino povrsine
    if y != 0 and v % a != 0:
        return ()

    return (x, y)
```

1.2 Evklidov algoritem za iskanje največjega skupnega delitelja

```
# Evklidov algoritem za izracun najvecjega skupnega
   delitelja naravnih stevil a in b
def gcd(a, b):
    if b == 0:
        return a
    return gcd(b, a % b)
```

1.3 Iskanje rešitve diofantske enačbe

```
# Resi diofantsko enacbo ax + by = c. Ce enacba nima resitev
    vrne prazen seznam.
# Sicer vrne par (x, y) ki ustreza diofantski enacbi.
def resiDiofantskoEnacbo(a, b, c):
    print "Resujem diofantsko enacbo %dx + %dy = %d" % (a, b
       , c)

    # okrajsaj enacbo in preveri ce je resljiva (gcd(a,b)
       mora deliti c)
    g = gcd(abs(a), abs(b))
    if c % g != 0:
        print "Diofantska enacba nima resitve, saj gcd(%d, %
           d) = %d ne deli %d" % (a, b, g, c)
        return ()

    a, b, c = a/g, b/g, c/g

    # poisci partikularno resitev s pomocjo veriznih ulomkov
    koeficienti = verizni_koeficienti(abs(a), abs(b))
    koeficienti.pop()

    if len(koeficienti) == 0:
        v, u = 1, 1
    else:
        v, u = ulomek_iz_koeficientov(koeficienti)

    # po potrebi spremenimo steviloma u in v predznak, da
       zadoscata enacbi au + bv = 1
    if a * -u + b * v == 1:
        u = -u
    elif a * u + b * -v == 1:
        v = -v
    elif a * -u + b * -v == 1:
        u, v = -u, -v
```

```
    # u in v zadoscata enacbi au + bv = 1, zato obe strani
       enacbe mnozim s c
    x0 = u * c
    y0 = v * c

    print "Partikularna resitev je x = %d, y = %d" % (x0, y0
       )

    # splosna resitev je x = x0 - bn, y = y0 + an
    print "Splosna resitev diofantske enacbe je x = %d - %dn
       , y = %d + %dn" % (x0, b, y0, a)

    # najdi pozitivni resitvi
    n = (-y0 - 1) / a + 1
    x = x0 - b * n
    y = y0 + a * n

    print "Potencialni najmanjsi pozitivni resitvi sta x = %
       d in y = %d" % (x, y)

    return (x, y)
```

1.4 Verižni ulomki

```
# Zapis ulomka a/b z veriznimi koeficienti (iterativno)
def verizni_koeficienti(imenovalec, stevec):
    koeficienti = []
    while stevec != 1:
        koeficienti.append(imenovalec / stevec)
        imenovalec, stevec = stevec, imenovalec % stevec
    return koeficienti + [imenovalec]

# Zapis ulomka a/b z veriznimi koeficienti (rekurzivno)
def verizni_koeficienti_r(imenovalec, stevec):
    if stevec == 1:
        return [imenovalec]
    else:
        return [imenovalec / stevec] + verizni_koeficienti_r
           (stevec, imenovalec % stevec)

# Izracun ulomka predstavljenega z veriznimi koeficienti
def ulomek_iz_koeficientov(koeficienti):
    stevec = 1
    imenovalec = koeficienti.pop()
```

```
    while len(koeficienti) > 0:
        imenovalec, stevec = koeficienti.pop() * imenovalec
           + stevec, imenovalec
    imenovalec, stevec = stevec, imenovalec

    return stevec, imenovalec
```

1.5 Šifriranje in odšifriranje

```
znaki = [' ', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', '
   J', 'K', 'L', 'M', 'N', 'O', 'P', 'R', 'S', 'T', 'U', 'V'
   , 'Z']

def sifriraj(besedilo, a, b):
    # ce ima besedilo liho stevilo crk, dodaj na koncu
       presledek
    if len(besedilo) % 2 == 1:
        besedilo += ' '

    sifrirano_sporocilo = []

    pos = 0
    while pos < len(besedilo):
        x = zakodiraj(besedilo[pos])
        y = zakodiraj(besedilo[pos + 1])
        sifrirano_sporocilo += [a * x + b * y]
        pos += 2

    return sifrirano_sporocilo

def desifriraj(sifrirano_sporocilo, a, b):
    sporocilo = []

    pos = 0
    while pos < len(sifrirano_sporocilo):
        s = sifrirano_sporocilo[pos]
        (x, y) = pokritje1(a * b, s, a, b)
        sporocilo += [odkodiraj(x)] + [odkodiraj(y)]
        pos += 1

    return "".join(sporocilo)

def zakodiraj(crka):
    return znaki.index(crka)
```

```
def odkodiraj(koda):
    return znaki[koda]
```

Literatura

[1] France Križanič, *Diofantske enačbe*.

[2] Martin Juvan, *O Evklidovem algoritmu*.

[3] Janez Stare, *Nekaj o teoriji števil*.

[4] Matjaž Omladič, *Zagonetni Fermat*.

[5] Peter Legiša, *Verižni ulomki*.

[6] Marino Pavletič, *Verižni približki*.

[7] France Forstenič, *O kongruencah*.

[8] Dušan Pagon, *Kongruence in Eulerjev izrek*.

[9] Tomaž Pisanski, *Skrivnostno sporočilo*.

[10] Marija Vencelj, *Šifriranje z javnim ključem*.

[11] Aleksandar Jurišić, *Diffie Hellmanov dogovor o ključu 2. del*.

[12] Aleksandar Jurišić, *Napake niso za vedno*.

[13] Marija Vencelj, *To in ono o tajnopisih*.