"ГРУБА СИЛА" У ПРОГРАМИРАЊУ

Зоран Будимац, Ђура Паунић,
Природно-математички факултет, Нови Сад

"Грубом силом" (енгл. brute force) се зове техника програмирања (или конструкције алгоритама) по којој се до решења задатог проблема долази тако што се генеришу сви могући кандидати за решење, па се испитује да ли сваки кандидат задовољава услове решења.

"Груба сила" је обично прва "техника" коју науче млади програмери за решавање комбинаторних проблема, а често је користе и искусни програмери, као први корак ка коначном решењу задатог проблема. Јасно је да "грубу силу" треба избегавати кад год је то могуће, пре свега зато што је најчешће скуп кандидата за решење огроман, а решења су малобројна. Ако користимо "грубу силу" једини начин да повећамо ефикасност нашег решења је да купимо бржи рачунар.

"Груба сила" се често може избећи неком од стандардних техника конструкције алгоритама: алгоритмима "похлепе", методом "подели па владај", бектреком, динамичким програмирањем и слично. Са бектреком смо се упознали у првом броју часописа, док ћемо се са осталим техникама упознавати у наредним бројевима. У овом чланку ћемо међутим, на једном познатом проблему показати како се почетно, очигледно решење које се добија применом "грубе силе" може значајно побољшати. При томе нећемо користити никакву стандардну технику конструкције алгоритама, већ само "здрав разум" и мало математике.

Питагорине тројке

Из геометрије је позната Питагорина теорема, чињеница да је у правоуглом троуглу збир квадрата катета једнак квадрату хипотенузе. Постоје случајеви када су и хипотенуза и катете природни бројеви (на пр. 3, 4, 5 или 5, 12, 13). Проблем који треба да се реши је да се нађу све тројке природних бројева, означимо их са (a, b, c), за које важи да је $a^2 + b^2 = c^2$. Овакве тројке природних бројева се називају Питагорине тројке и претпоставићемо да су a, b, c природни бројеви мањи од унапред задате границе n.

Најједноставније решење је да се генеришу све могуће тројке природних бројева a, b и c мањих од n и да се за сваку тројку посебно испита да ли она задовољава услов, $a^2 + b^2 = c^2$, тј. да ли је Питагорина тројка. Када се овај поступак налажења Питагориних тројки испрограмира на Паскалу, добија се следећи програм:

```
program PitagorineTrojke;
var  n, a, b, c: integer;
begin
  repeat
    writeln('Unesite granicu za Pitagorine trojke (<= 100)');
    readln(n)
  until (1 <= n) and (n <= 100);
  writeln;
  writeln('Pitagorine trojke do ',n,' su');
  writeln;
  for a := 1 to n do
    for b := 1 to n do
      for c := 1 to n do
        if sqr(a) + sqr(b) = sqr(c) then
          writeln('(',a,',',b,',',c,')')
end.
```

Ограничење на величину броја n мора да се стави, јер је скуп целих бројева, који се у Паскалу назива `integer`, коначан скуп. Дакле, у Паскалу мора да буде задовољено $a^2 + b^2 <$ `maxint` да би програм коректно радио. Када је `maxint` $= 32767$, тада a и b морају бити мањи од 128.

Овакво решење је очигледно решење "грубом силом". Да бисмо оценили сложеност неког алгоритма треба је измерити на неки начин. Једна могућност је мерити време извођења, али то време зависи од брзине рада рачунара на коме се програм изводи па није објективно мерило. Други начин за мерење сложености алгоритма је број корака који је потребан да се програм изведе. Међутим и ту треба бити пажљив. Очигледно број корака зависи од величине полазних података, па број корака треба да се изрази као функција од величине полазних података. За то се користи тзв. "велико О" нотација. Ако је $f(n)$ позитивна реална функција дефинисана за сваки природан број n, (n је величина улазних података), тада се са $O(f(n))$ означава да је за довољно велико n ($n > n_0$) број корака алгоритма ограничен функцијом $cf(n)$, за неки позитиван реалан број c.

Рачунска сложеност овог алгоритма је $O(n^3)$, што значи да је број корака потребан за израчунавање пропорционалан са n^3 корака. При том константа пропорционалности није јако важна, јер ако се број полазних података удесетостручи, тј. уместо n имамо $10n$ података тада ће израчунавање трајати $(10n)^3 = 1000n^3$ – хиљаду пута дуже од израчунавања за n. Интуитивно, $O(n^3)$, у нашем случају, значи да се израчунавање врши унутар троструке петље, од којих се свака извршава n пута.

Иако коректно, претходно решење је неефикасно. За $n = 1000$, испитивање би се изводило чак 1000000000 пута, што би и на најбржим рачунарима трајало "довољно" дуго. Уместо куповине бржег рачунара, јефтиније је поправити алгоритам.

Прва идеја је искористити чињеницу да је могуће уредити бројеве a, b и c по величини $a < b < c$, па да друга петља иде од $a + 1$ до n (уместо од 1 до n), а трећа петља од $b + 1$ до n (уместо од 1 до n), а када се нађе Питагорина тројка $a < b < c$ да се тада штампа и она и тројка b, a, c. На овај начин се алгоритам убрзава, али му сложеност и даље остаје $O(n^3)$. Израчунајте број корака у овом случају.

Међутим могуће је значајно побољшати алгоритам коришћењем једноставне чињенице да ако су познати a и b тада се c израчунава са $c = \sqrt{a^2 + b^2}$ и нема потребе за унутрашњом петљом (петљом по c). Да би у оваквом решењу добијена тројка била Питагорина, потребно је једино проверити да ли је тако израчунато c природан број. Ако јесте, нашли смо Питагорину тројку – штампати је, ако није – наставити тражење. При том треба пазити на рачунање са реалним бројевима на рачунару при провери да је c цео број. Наиме, при израчунавању квадратног корена је могуће да дође до мале грешке, најчешће на последњој децимали броја. Да се та грешка избегне, уместо упоређивања једнакости два реална броја, треба проверити да ли су они "довољно" блиски са `abs(a-b)<0.01`. Шта значи "довољно блиско" зависи од прецизности рачунања са реалним бројевима на датом рачунару и у датом програмском језику на њему.

Рачунска сложеност овог решења је за ред величине мања. Како имамо само две петље једну у другој то је сложеност овог алгоритма $O(n^2)$, па се испитивање услова за Питагорину тројку за $n = 1000$ врши "само" 1000000 пута (или ако се улазни подаци повећају 10 пута програм ће рачунати само 100 пута дуже).

```
program PitagorineTrojke1;
  var n, a, b: integer;
      c: real;
begin
  repeat
    writeln('Unesite granicu za Pitagorine trojke (<= 100)');
    readln(n)
  until (1 <= n) and (n <= 100);
  writeln;
  writeln('Pitagorine trojke do ',n,' su');
  writeln;
  for a := 1 to n do
    for b := 1 to n do
      begin
        c := sqrt(sqr(a) + sqr(b))
        (* provera da li je z ceo broj *)
        if abs(c - round(c)) < 0.01 then
          writeln('(',a,',',b,',',round(c),')')
      end
end.
```

И овај алгоритам је могуће мало убрзати на исти начин као и претходни тиме што унутрашња петља може да иде од $a+1$ до n, а када се нађе Питагорина тројка a, b, c, тада се штампа и тројка b, a, c.

Опишимо Питагорине тројке на други начин. Очигледно је да је

$$a^2+b^2=c^2 \quad \Longleftrightarrow \quad \left(\frac{a}{c}\right)^2+\left(\frac{b}{c}\right)^2=1,$$

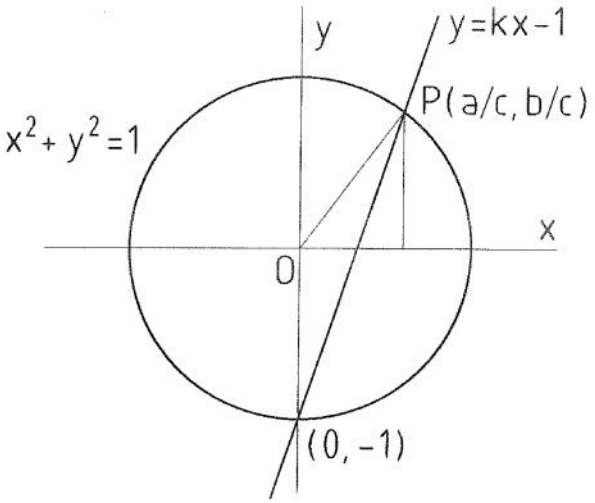
па свакој Питагориној тројки $P(a,b,c)$, одговара тачка P на луку јединичног круга у првом квадранту $P(a/c,b/c)$ или $P(b/c,a/c)$, чије су обе координате рационални бројеви. Поставимо праву p кроз тачке $(0,-1)$ и $P(a/c,b/c)$. Користећи да једначина праве кроз две тачке

$$y-y_1=\frac{y_2-y_1}{x_2-x_1}(x-x_1),$$

добија се да је једначина праве кроз тачке $(0,-1)$ и $P(a/c,b/c)$

$$y=kx-1 \quad \text{у којој је} \quad k=\frac{\frac{b}{c}+1}{\frac{a}{c}}=\frac{b+c}{a},$$

а када је $P(b/c,a/c)$ добија се да је $k=(a+c)/b$. У оба случаја је тачка P на луку круга у првом квадранту па је увек $k>1$ (види слику).



Међутим, важи и обрнуто. Свакој правој $p: \ y=kx-1$ чији је коефицијент правца рационалан број већи од 1, $k>1$, одговара једна Питагорина тројка, која се добија из координата тачке пресека праве p и јединичног круга различите од $(0,-1)$. Наиме права p сече јединични круг $x^2+y^2=1$ у две тачке чије се координате добијају решавањем система једначина

$$y=kx-1, \qquad x^2+y^2=1.$$

Ако се елиминише y добија се да је $x^2 + k^2x^2 - 2kx + 1 = 1$ или $x((k^2 + 1)x - 2k) = 0$. Једно решење је тачка чије су координате $x_1 = 0$, $y_1 = -1$, а друго решење је тачка која има координате

$$x_2 = \frac{2k}{k^2+1}, \quad y_2 = kx_2 - 1 = \frac{2k^2}{k^2+1} - 1 = \frac{k^2-1}{k^2+1}.$$

Како је $k > 1$ ова тачка је у првом квадранту. Ако заменимо координате тачке (x_2, y_2) у једначину круга $x^2 + y^2 = 1$ добија се да је

$$\left(\frac{2k}{k^2+1}\right)^2 + \left(\frac{k^2-1}{k^2+1}\right)^2 = 1 \quad \Longleftrightarrow \quad (2k)^2 + (k^2-1)^2 = (k^2+1)^2,$$

а када ставимо да је $k = m/n$ и средимо

$$\left(2\frac{m}{n}\right)^2 + \left(\left(\frac{m}{n}\right)^2 - 1\right)^2 = \left(\left(\frac{m}{n}\right)^2 + 1\right)^2 \Longleftrightarrow (2mn)^2 + (m^2-n^2)^2 = (m^2+n^2)^2$$

па је $a = 2mn$, $b = m^2 - n^2$, $c = m^2 + n^2$ или $a = m^2 - n^2$, $b = 2mn$, $c = m^2 + n^2$ Питагорина тројка.

Тиме смо доказали теорему:

Теорема 1 *За сваки пар природних бројева m и n, $m > n$, бројеви*

$$a = 2mn, \quad b = m^2 - n^2, \quad c = m^2 + n^2,$$

чине Питагорину тројку.

Свака тројка генерисана на основу теореме 1 јесте Питагорина и нема потребе за генерисањем свих могућих кандидата и додатним испитивањем о задовољености услова. Морамо уочити да се ефикасност решења на основу теореме 1 не може једноставно упоредити са ефикасношћу претходна два решења (погледати задатке на крају чланка).

Посебне Питагорине тројке

Могуће је међу Питагориним тројкама тражити тројке одређеног облика.

1. Досада наведени алгоритми посматрају Питагорине тројке $(3, 4, 5)$, $(6, 8. 10)$, $(9, 12, 15)$ итд. дакле тројке облика $(3k, 4k, 5k)$, $k = 1, 2, 3, \ldots$, као различите Питагорине тројке. Математички гледано ове тројке нису суштински различите, јер су правогли троуглови који им одговарају слични. Због тога се уводи појам примитивне Питагорине тројке. Примитивна Питагорина тројка (a, b, c) је она у којој a, b и c немају заједнички фактор.

Дакле примитивним Питагориним тројкама одговарају правоугли троуглови који нису међусобно слични.

2. Следећа могућност ограничења је решење проблема: генерисати све Питагорине тројке (до одређене границе n) код којих се једна катета за 1 разликује од хипотенузе. Решење "грубом силом" (генерисање свих могућих тројки, па испитивање да ли су Питагорине и да ли задовољавају овај додатни услов) је могуће, али врло неефикасно. Покажите да се коришћењем "грубе силе" добија алгоритам сложености $O(n^3)$. Уместо таквог решења (сложености $O(n^3)$), објаснимо одмах много ефикаснији алгоритам, чија је сложеност $O(n)$.

Због услова задатка мора да је $c = b + 1$ и да је $a = \sqrt{c^2 - b^2}$. Питагорину тројку смо нашли ако је a природан број. Програм за овај поступак има само једну петљу (по b), а све остало се израчунава у њој па је сложеност овог поступка $O(n)$.

```
program PitagorineTrojke2;
var b, c, n : integer;
    a: real;
begin
  repeat
    writeln('Unesite granicu za Pitagorine trojke (<= 100)');
    readln(n)
  until (1 <= n) and (n <= 100);
  writeln;
  writeln('Pitagorine trojke do ',n,' za koje je  c - b = 1');
  writeln;
  for b := 1 to n do
    begin
      c := b + 1;
      a := sqrt(sqr(c) - sqr(b))
      if abs(a - round(a)) < 0.01 then
        (* da li je a ceo broj ? *)
        writeln('(',round(a),',',b,',',c,')')
    end
end.
```

3. Природно се поставља и слично ограничење на катете, тј. могуће је решавати проблем: наћи све Питагорине тројке код којих је разлика између две катете (тј. a и b) 1. Ако се и овај проблем покуша решавати "грубом силом" убрзо ћемо се уверити да је број таквих Питагориних тројки релативно врло мали, а алгоритам је опет реда $O(n^3)$ (Докажите). За a из интервала од 1 до 25000, на начин аналоган претходном решењу, добијају се следеће Питагорине тројке:

```
    3      4      5
   20     21     29
  119    120    169
  696    697    985
 4059   4060   5741
23660  23661  33461
```

Очигледно је да у овом случају ни решење рачунске сложености од $O(n)$ није довољно добро. На велики број испитаних кандидата долази врло мали број оних који задовољавају услове и потребно нам је неко боље решење овог проблема.

У задацима на крају овог чланка је дато упутство за проналажење законитости на основу које је могуће директно израчунати следећу Питагорину тројку код којих је разлика између две катете 1.

4. Разматрање Питагориних тројки закључујемо следећим проблемом: Пронаћи Питагорину тројку у којој је задати природни број b једна од катета. Решење применом "грубе силе" овде готово да и нема смисла, иако је наравно могуће. Уместо тога покушајмо да пронађемо неку законитост, на основу које ћемо директно доћи до решења овог проблема.

Ако је b паран број, тада нам решење нуди теорема 1. Ако за n из теореме одаберемо 1, тада за b можемо узети да је $2m$, па је (на основу теореме) друга катета $a = m^2 - 1$, а хипотенуза је $c = m^2 + 1$.

Покушајмо да нађемо законитост када је задати број b непаран. На основу дефиниције је $a^2 + b^2 = c^2$, следи $b^2 = c^2 - a^2$.

Ако је b непарно, тада се може записати као $2m + 1$ (за неко m), а b^2 је тада: $(2m + 1)^2 = 4m^2 + 4m + 1$, што можемо записати и као $(2m^2 + 2m + 1) + (2m^2 + 2m)$. Због једноставности уводимо да је $k = 2m^2 + 2m$, па на крају имамо да је $b^2 = (k + 1) + k$. Уколико овом изразу и додамо и одузмемо $k^2 + k$, добијамо:

$$b^2 = (k+1)+k = (k^2+k+k+1)-k^2-k+k = (k^2+2k+1)-k^2 = (k+1)^2-k^2$$

што нам на крају даје тражену законитост:

$$b^2 = (2m + 1)^2 = (2m^2 + 2m + 1)^2 - (2m^2 + 2m)^2 = c^2 - a^2$$

Сада је једноставно написати (врло ефикасан) програм за решење датог проблема:

```
program PitagorineTrojke3;
var a, b, c, m, mNa2: integer;
begin
  repeat
    writeln('Unesite prirodan broj za Pitagorinu trojku ');
```

```
  readln(b)
until (2 < b) and (b <= 100);
m := b div 2;
mNa2 := sqr(m);
if odd(b) then begin (* ucitani broj je neparan *)
    a := 2*mNa2 + 2*m;
    c := b + 1
end
else begin (* ucitani broj je paran *)
    a := mNa2 - 1;
    c := mNa2 + 1;
end;
writeln('Pitagorina trojka za broj ',b,' je ',a,',',b,',',c)
end.
```

Више о Питагориним тројкама је могуће наћи у књигама: Р. Тошић, В. Вукославчевић : Елементи теорије бројева, Алеф, Нови Сад 1995. и В. Мићић, З. Каделбург : Увод у теорију бројева, Друштво математичара СР Србије, Београд 1989.

ЗАДАЦИ

1. Написати програм за испис Питагориних тројки према теореми 1.

2. Упоредити ефикасност решења на основу теореме 1 и претходна два решења. Односно, изразити ефикасност решења на основу теореме преко n – задате границе до које испитујемо тројке.

3. Написати програм који ефикасно израчунава и исписује примитивне Питагорине тројке.

4. Написати програм који исписује Питагорине тројке код којих се хипотенуза и једна катета разликују за 2.

5. Написати програм који израчунава и исписује Питагорине тројке код којих је разлика између хипотенузе и катете унапред задати број k који је потпун квадрат $k = t^2$ или двоструки потпун квадрат $k = 2t^2$.

6. Пронаћи неколико следећих Питагориних тројки код којих је разлика између катета 1. За решење овог задатка је потребно пронаћи законитост између таквих Питагориних тројки и бројева m и n из теореме 1. У утврђивању законитости, занемарити разлику између катете a и катете b – треба их третирати равноправно.

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 1995/96 година