Autor: Paweł Salwa

Chciałem poprawić ten program, dlatego przesyłam go jeszcze raz. Tym razem zaimplementowałem funkcję z przesunięciami bitowymi która ma złożoność obliczeniową logarytmiczną O(log(n)). Wiem , że ona znowu wykorzystuje pętlę, która jest niewydajna w pythonie, ale na podstawie wyników programu zauważyłem, że dla dużych wykładników jest ona szybsza (zgadza się to z logarytmiczną złożonością).

Ponieważ to w żaden sposób nie udowadnia, że mnożąc szybciej osiągniemy wynik, porównałem potęgowanie ** i mnożenie * zapisując je wprost. Co się okazało- dla prawie wszystkich danych wejściowych czas trwania obliczeń był niemalże identyczny (może zatem operator ** w pythonie wykorzystuje po prostu zwykłe mnożenie?).

Przesyłam także program w c++ który liczy to samo i wreszcie wyniki są zgodne z tym o co proszono w zadaniu. Okazuje się , że i metoda z wykorzystaniem przesunięcia binarnego, i metoda iteracyjna są szybsze od wbudowanej funkcji dla wykładników mniejszych od ok. 60. Dla większych pow() wbudowana jest szybsza.

Output programu pythonowego:

a= 2
b= 10
n= 100000
time pow_lib = 0.0111503248772
time pow_moja = 0.0101532555273

a= 2
b= 100
n= 100000
time pow_lib = 0.110493306507
time pow_moja = 0.0101593994332

a= 2
b= 1000
n= 100000
time pow_lib = 0.143562480822
time pow_moja = 0.0101617399923

a= 2
b= 10000
n= 100000
time pow_lib = 1.38894540315
time pow_moja = 0.0102217162785
wszystko sie zgadza

0.00139583858254 <-czas 3**2
0.00136277847923 <-czas 3*3

0.00141017437272 <-czas 3**4
0.00156201681239 <-czas 3*3*3*3

0.00135838997085 <-czas 3**8
0.00134668727696 <-czas 3*3*3*3*3*3*3*3

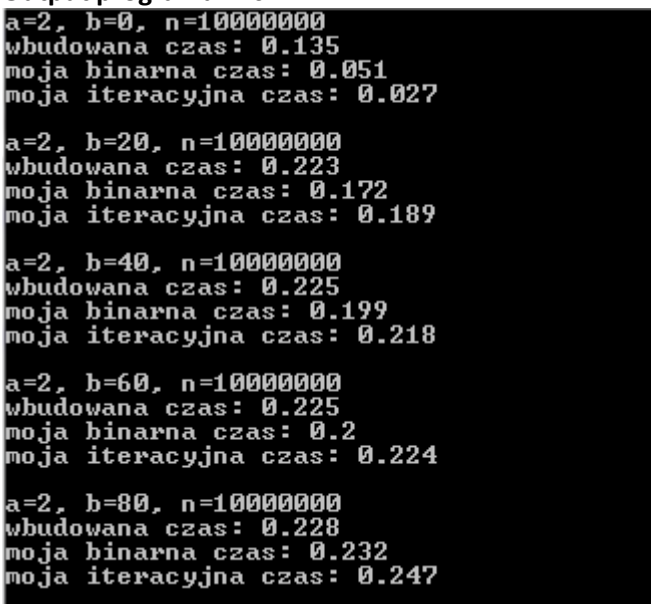
0.00134815010824 <-czas 3**16
0.0013563419925 <-czas 3*3*3*3*3*3*3*3*3*3*3*3*3*3*3*3

Kod w pythonie :

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 27 17:00:42 2016

@author: salwus     VERSION v2
"""
import timeit

a = 2
b = 10
n = 10**5 #ilosc wykonan

def pow_lib():
   global a,b
   return a**b

def pow_moja():
        global a,b



        wynik = 1
        while(b):
                if (b & 1):
                        wynik *= a
                b>>=1
                a *=a
        return wynik

a = 2
b = 10
n = 10**5 #ilosc wykonan

print "a=" ,a
print "b=",b
print "n=" ,n
#============ponizsze funkcje zwracaja czas trwania moich funkcji=======
print "time pow_lib = " , timeit.timeit( pow_lib, number=n )
print "time pow_moja = ",  timeit.timeit( pow_moja, number=n )

a = 2
b = 100
n = 10**5 #ilosc wykonan

print "\na=" ,a
```

```
print "b=",b
print "n=" ,n

print "time pow_lib = " , timeit.timeit( pow_lib, number=n )
print "time pow_moja = ",  timeit.timeit( pow_moja, number=n )

a = 2
b = 1000
n = 10**5 #ilosc wykonan

print "\na=" ,a
print "b=",b
print "n=" ,n

print "time pow_lib = " , timeit.timeit( pow_lib, number=n )
print "time pow_moja = ",  timeit.timeit( pow_moja, number=n )

a = 2
b = 10000
n = 10**5 #ilosc wykonan

print "\na=" ,a
print "b=",b
print "n=" ,n

print "time pow_lib = " , timeit.timeit( pow_lib, number=n )
print "time pow_moja = ",  timeit.timeit( pow_moja, number=n )
if(pow_lib() == pow_moja()):
        print 'wszystko sie zgadza\n\n'

print timeit.timeit('3**2', number=n) , '<-czas 3**2'
print timeit.timeit('3*3', number=n) , '<-czas 3*3\n'

print timeit.timeit('3**4', number=n) , '<-czas 3**4'
print timeit.timeit('3*3*3*3', number=n) , '<-czas 3*3*3*3\n'

print timeit.timeit('3**8', number=n) , '<-czas 3**8'
print timeit.timeit('3*3*3*3*3*3*3*3', number=n) , '<-czas 3*3*3*3*3*3*3*3\n'

print timeit.timeit('3**16', number=n) , '<-czas 3**16'
print timeit.timeit('3*3*3*3*3*3*3*3*3*3*3*3*3*3*3*3', number=n) , '<-czas
3*3*3*3*3*3*3*3*3*3*3*3*3*3*3*3\n'
================================================================================
================================================================================
```

Output programu w c++:



Kod w c++:

```
#include <iostream>
#include <cmath>
#include <ctime>

using namespace std;



double pow_bin(double a, int b){
   double wynik = 1;
   while (b)
   {
      if (b & 1)
         wynik *= a;
      b >>= 1;
      a *= a;
   }

   return wynik;
}

double pow_i( double liczba, unsigned long long int potega){
   double wynik=liczba;
   while(potega>1) {
      wynik *= wynik;
      potega /= 2;
   }
   return wynik;

}

int main()
```

```
{
        double elapsed_secs ;
   double elapsed_secs2;
//================== a=2, b= 20*j============================
        int a=2;
   int b;
   for (int j=0 ; j<5 ; j++){
        b=20*j;
        //=================
          clock_t begin = clock();
          for(int i=0;i<10000000; i++){
                pow(a,b);
          }
          clock_t end = clock();
          //=================
          clock_t begin2 = clock();
          for(int i=0;i<10000000; i++){
                pow_i(a,b);
          }
          clock_t end2 = clock();
          //=================
          clock_t begin3 = clock();
          for(int i=0;i<10000000; i++){
                pow_bin(a,b);
          }
          clock_t end3 = clock();
          //=================

                double elapsed_secs = double(end - begin)/ CLOCKS_PER_SEC;
          double elapsed_secs2 = double(end2 - begin2)/ CLOCKS_PER_SEC;
          double elapsed_secs3 = double(end3 - begin3)/ CLOCKS_PER_SEC;
          cout<< "a=2, b="<<b<< ", n=10000000"<<endl;
          cout<< "wbudowana czas: " << elapsed_secs << endl;
          cout<< "moja binarna czas: " << elapsed_secs2 << endl;
          cout<< "moja iteracyjna czas: " << elapsed_secs3 << endl <<endl;


        }
}
```