|  |  |
| --- | --- |
| Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej | |
| Wstęp do Informatyki | |
| Zajęcia 5 – Częste problemy | |
| Imię i Nazwisko | Daniel Rubak |
| Data wykonania ćwiczenia | 06.11.2017 |

1. **Kod programu (wersja podstawowa)**

Podstawowa wersja algorytmu polegała na wykonaniu jednego pełnego cyklu znajdowania liczb pierwszy z zakresu od 2 do wartości podanej przez użytkownika oraz zmierzeniu czasu wykonania się owego cyklu. Kod algorytmu zamieszczono poniżej.

*# Napisz program, który możliwie najefektywniej znajdzie wszystkie liczby pierwsze w zakresie od 1 do max, przyjmij max = 1000. Zmierz czas obliczeń*

**import** math  
**import** time  
  
n = int(input(**"Podaj liczbę całkowitą większą od 1: "**))  
  
start\_time = time.time()  
sqrt\_n = math.ceil(math.sqrt(n))  
A = []  
A.append(0)  
A.append(0)  
**for** i **in** range (2,n+1):  
 A.append(1)  
  
**for** i **in** range (2, sqrt\_n):  
 **if** A[i] == 1:  
 **for** j **in** range (i\*i, n+1, i):  
 A[j]=0  
  
print(**"--- %s seconds ---"** % (time.time() - start\_time))  
  
print(**"Wszystkie liczby pierwsze w przedziale (2, "**,n,**"):"**,sep=**""**)  
**for** i **in** range (2, n+1):  
 **if** A[i]==1:  
 print(i)

1. **Kod programu (wersja rozszerzona)**

Wersja rozszerzona algorytmu działała w pętli for, gdzie dla każdej iteracji zwiększano zakres obliczeń, począwszy od liczby 1000000, a skończywszy na liczbie 10000000. Co więcej, w każdej iteracji zapisywano uzyskane pomiary do pliku xml wykorzystując do tego bibliotekę xltw. Kod algorytmu zamieszczono poniżej.

*# Napisz program, który możliwie najefektywniej znajdzie wszystkie liczby pierwsze w zakresie od 1 do max, przyjmij max = 1000. Zmierz czas obliczeń***import** math  
**import** time  
**import** xlwt  
  
**from** tempfile **import** TemporaryFile  
wb = xlwt.Workbook()  
ws = wb.add\_sheet(**'Duration from value'**)  
  
k = 0  
**for** n **in** range (1000000, 10000001, 1000000):  
 print(k)  
 start\_time = time.time()  
 sqrt\_n = math.ceil(math.sqrt(n))  
 A = []  
 A.append(0)  
 A.append(0)  
 **for** i **in** range (2,n+1):  
 A.append(1)  
  
 **for** i **in** range (2, sqrt\_n):  
 **if** A[i] == 1:  
 **for** j **in** range (i\*i, n+1, i):  
 A[j]=0  
  
 duration = (time.time() - start\_time)  
 ws.write(k, 0, n)  
 ws.write(k, 1, duration)  
 k += 1  
  
wb.save(**'data.xls'**)  
wb.save(TemporaryFile())

1. **Tabela zależności czasu działania algorytmu od wartości zadanej.**

Wartości uzyskane w eksperymencie umieszczono w tabeli poniżej. Kolumna lewa zawiera wartość liczbową, do której algorytm wyszukiwał liczb pierwszych. Z kolei kolumna prawa zawiera czas wykonania algorytmu podany w sekundach.

|  |  |
| --- | --- |
| Wartość liczbowa | Czas |
| 1000000 | 0,328150988 |
| 2000000 | 0,765707254 |
| 3000000 | 1,513885498 |
| 4000000 | 1,843935966 |
| 5000000 | 2,173677444 |
| 6000000 | 2,692678452 |
| 7000000 | 2,862760305 |
| 8000000 | 3,087696791 |
| 9000000 | 3,777535677 |
| 10000000 | 4,677649736 |

1. **Wykres zależności czasu działania algorytmu od wartości zadanej.**
2. **Wnioski**

Wyniki są dość zaskakujące. Dla dużo mniejszych zakresów, tj. od 1000000 do 10000000, czas wykonania poszczególnych cykli był losowy. Zdarzały się sytuacje, w których dla większych wartości algorytm kończył działanie dużo szybciej niż dla mniejszych. W związku z tym zdecydowałem się zwiększyć zakres obliczeń. W przypadku dużo większych wartości, czas wykonania algorytmu ewidentnie rośnie, jednak nie jestem w stanie wskazać żadnej konkretnej zależności ponieważ nie jest on ani liniowy ani wykładniczy.