Przedmowa

W ostatnich latach na naszym rynku wydawniczym pojawiło się wiele poważnych pozycji, omawiających algorytmy teorii liczb i kryptografii. Bibliografia na końcu książki zawiera tylko część z nich. Są to zarówno tłumaczenia bestsellerów rynku światowego, jak i interesujące opracowania polskich autorów. Dzieło *Kryptografia stosowana* [15] to prawdziwa encyklopedia kryptografii. Niemal równie rozbudowana jest książka *Kryptografia dla praktyków. Protokoły, algorytmy i programy źródłowe w języku C* [16]. Czytelnik zainteresowany dowodami podstawowych faktów matematycznych wykorzystywanych w kryptografii ma do dyspozycji takie książki, jak *Wprowadzenie do kryptografii* [3], *Wykład z teorii liczb i kryptografii* [10] czy *Teoria liczb w informatyce* [23].

Proponowana przez nie ścieżka uczenia się jest jednak dość stroma. Aby omówić w sposób wyczerpujący podstawowe zagadnienia kryptografii, teorię liczb przedstawia się w nich bardzo zwięźle. Dla pewnej części czytelników rozpoczynających poznawanie kryptografii szybka powtórka z teorii liczb może się okazać niewystarczająca. Do takich Czytelników skierowane jest niniejsze opracowanie.

W prezentowanym przez nas podejściu znacznie więcej miejsca niż wymienione pozycje poświęcamy elementarnym faktom teorii liczb całkowitych. W związku z tym tempo dochodzenia do treści kryptograficznych jest znacznie wolniejsze. Rezygnujemy natomiast z próby pełnego omówienia kryptografii, ograniczając się do najbardziej popularnych algorytmów. Mamy nadzieję, że zapoznanie się z naszym opracowaniem może znacznie ułatwić studiowanie wymienionych wcześniej obszernych monografii, zawierających pominięte przez nas treści.

Poznając algorytmy teorii liczb i kryptografii, spotykamy wiele przykładów, wymagających rachunków na dużych liczbach całkowitych. Dla zrozumienia algorytmów podstawowe znaczenie ma aktywne podejście do tych przykładów i możliwość samodzielnego eksperymentowania. Wymaga to często użycia szybkich narzędzi komputerowych. Oczywistym wyborem może być język C/C++ wraz z bibliotekami udostępniającymi liczby całkowite dowolnej wielkości. Jeśli podstawowe znaczenie ma szybkość obliczeń, wybór taki jest uzasadniony.

W prezentowanym tekście staramy się przedstawić inną możliwość. Znacznie mniejszym nakładem czasu na programowanie można przeprowadzać eksperymenty ze stosunkowo dużymi liczbami całkowitymi, wykorzystując odpo-

wiednie pakiety matematyczne. Można tu użyć najbardziej znanych pakietów komercyjnych, takich jak Mathematica czy Maple (zob. np. [13]).

Ze względu na dużą cenę tych narzędzi trudno przyjąć założenie, że osoby pragnące aktywnie poznawać kryptografię dysponują legalnymi kopiami wymienionych pakietów na komputerach domowych. Nie chcąc ograniczać grona zainteresowanych czytelników, proponujemy wykorzystanie mniej znanych, lecz wolnodostępnych pakietów GP/Pari i Sage. Naszym zdaniem pakiety te w teorii liczb i kryptografii sprawują się równie dobrze, jak wymienione odpowiedniki komercyjne.

Spora liczba przykładowych obliczeń teorii liczb i kryptografii zawarta w tekście ma na celu zachęcenie czytelnika do własnych eksperymentów i ograniczenie czasu potrzebnego do odnalezienia odpowiednich poleceń w dokumentacji systemów. Przedstawiając przykłady, staraliśmy się, by były możliwie proste. Nie należy ich zatem traktować jako wzorzec, a raczej jako punkt wyjściowy do własnych obliczeń, ulepszania przedstawionych metod i usuwania dostrzeżonych błędów.

Uwagi wstępne o Sage i GP/Pari

Zakładamy, że czytelnik będzie poznawał wykorzystywane przez nas programy jednocześnie ze studiowaniem książki. W dalszym ciągu ograniczymy się tylko do kilku uwag, pozwalających na rozpoczęcie samodzielnych obliczeń.

Pakiet Sage można pobrać ze strony *http://www.sagemath.org* (około 450 MB w wersji binarnej, 250 MB w wersji źródłowej). Do wyboru jest wersja źródłowa, skompilowane pakiety dla wybranych systemów linuksowych, wersja dla Windows, wymagająca VMware lub VirtualBox, a także niewymagający instalacji Sage Live-CD. Do instalacji należy wykorzystać wskazówki z podanej strony. Autor nie napotkał problemów przy instalacji ze źródeł w systemach Debian, Ubuntu, Fedora. Należy się jednak liczyć z długim czasem kompilacji (około 3 godzin) i zajęciem na dysku ponad 1 GB. Przy instalacji Sage, pakiet GP/Pari również zostanie zainstalowany, można go jednak łatwo zainstalować niezależnie ze strony *pari.math.u-bordeaux.fr* (około 2 MB w wersji dla Linuksa, 6 MB dla Windows).

Do komunikacji z systemami można wykorzystać terminal (w przypadku Sage również przeglądarkę *www*). Otwarcie systemów sprowadza się do otwarcia terminalu i wydania polecenia `sage` lub `gp`. Znak zachęty (po którym należy wpisywać polecenia) w przypadku Sage ma postać:

```
sage:
```

natomiast w przypadku GP/Pari jest to znak zapytania:

```
?
```

Uruchomienie GP/Pari wewnątrz Sage jest możliwe po wydaniu polecenia:

```
sage: gp_console()
```

Jeśli GP/Pari nie jest zainstalowany niezależnie od Sage, to aby go móc uruchamiać w Linuksie poleceniem `gp`, należy jako użytkownik `root` wydać polecenie:

```
sage: install_scripts("/usr/local/bin")
```

W obydwu systemach pomoc dotyczącą polecenia można uzyskać, wpisując nazwę polecenia ze znakiem zapytania (w przypadku GP/Pari jest to drugi znak zapytania przed poleceniem, w przypadku Sage znak zapytania może być umieszczony po poleceniu). Gorąco zachęcamy czytelników do używania znaku zapytania z każdym nowo napotkanym poleceniem. Dopisując na końcu polecenia Sage dwa znaki zapytania, możemy przejrzeć kod polecenia.

Bardzo pożyteczna jest funkcja autodopełniania, czyli możliwość wpisania początkowych liter polecenia i użycia przycisku *Tab*, aby się zorientować, czy jakieś polecenia zaczynają się od wpisanych liter. Gdy możliwych dopełnień jest wiele, można uzyskać listę wszystkich takich poleceń. W przypadku Sage można również wykorzystać autodopełnianie do przeglądania operacji możliwych do wykonania na obiekcie danego typu. Wystarczy wpisać po znaku zachęty nazwę obiektu z kropką na końcu, a następnie użyć klawisza tabulacji *Tab*:

```
sage: li=[1,2]                          # definicja obiektu
sage: li.[TAB]                          # po kropce uzyto [TAB]
li.append   li.index    li.remove
li.count    li.insert   li.reverse
li.extend   li.pop      li.sort
sage: li. #system oczekuje na dopisanie wybranego rozszerzenia
```

Znaku `#` użyto tu dla oznaczenia komentarza Sage. Znakiem komentarza w GP/Pari jest \\. Z myślą o możliwości kopiowania kodu z wersji elektronicznej tekstu unikamy polskich znaków w komentarzach.

Jeśli chcemy utrwalić ciągi poleceń, możemy zapisywać je w plikach tekstowych. Dla Sage, pliki takie mają rozszerzenie `sage`, a dla GP/Pari `gp`. Po zapisaniu plików w bieżącym katalogu polecenia w nich zawarte wykonujemy w sposób następujący. W Sage:

```
sage: attach "plik.sage"   # lub  load
```

natomiast w GP/Pari:

```
? \r plik.gp
```

GP/Pari ma własny język programowania, ze zwięzłą składnią. Jest to pakiet napisany głównie dla zainteresowanych algebraiczną teorią liczb. Jego możliwości, w zestawieniu z niezwykle małą objętością, są imponujące.

Sage jest szeroko zakrojonym projektem, będącym w trakcie intensywnego rozwoju. Ambicją jego twórców jest zbudowanie środowiska obliczeń o otwartym kodzie, będącego zamiennikiem dla komercyjnych pakietów: Mathematica, Maple, Matlab, Magma. Sage zawiera w sobie kilkadziesiąt niezależnych pakietów i bibliotek matematycznych o otwartym kodzie, takich jak Maxima, Singular, GAP, GP/Pari, R, NumPy, SciPy, GMP, Linbox, MPFR, NTL, GSL, BLAS, Lapack, ATLAS, stąd duża objętość pakietu i długi czas kompilacji.

Wielką zaletą Sage jest to, że wykorzystuje on składnię języka obiektowego ogólnego przeznaczenia Python. Czytelnik nie musi się jednak obawiać, że wcześniejsze poznanie Pythona będzie niezbędne do zrozumienia poniższego

tekstu. Warto jedynie uprzedzić, że w prezentowanych przykładach kodu Sage, tak jak w Pythonie, wcięcia są ważną częścią kodu i muszą być stosowane bardzo uważnie, gdyż odgrywają rolę nawiasów z innych języków programowania.

Staraliśmy się uczynić przedstawiane opracowanie na tyle samowystarczalnym i niezależnym od wcześniejszego przygotowania czytelnika, na ile to jest możliwe. Ocenę, czy zamierzenie to się powiodło, pozostawiamy samym Czytelnikom.