Technische Universität München
Institut für Informatik
Prof. Dr. Ernst W. Mayr

WS 2000/2001
Aufgabenblatt 11

Praktikum Algorithmen-Entwurf

(Abgabetermin: Montag, den 5.01.2001, 14.^{00} Uhr)

Aufgabe 1 Randomisierter Primzahltest

Implementieren Sie den im Skript beschriebenen Algorithmus zur randomisierten Erkennung von Primzahlen. Ihr Programm sollte folgende Benutzerschnittstelle zur Verfügung stellen:

- Aufruf mit einem Kommandozeilenparameter, der die Anzahl der gewünschten Wiederholungen des Tests angibt.
- Die zu testende Primzahl wird von der Standardeingabe eingelesen (als zusammenhängender String von Ziffern). Führende Leerzeichen sollten hierbei ignoriert werden. Ferner wird angenommen, daß die Eingabe weniger als 1000 Stellen besitzt.
- Auf der Standardausgabe wird `yes` oder `no` ausgegeben, je nachdem, ob die eingelesene Zahl eine Primzahl ist oder nicht.

Implementieren Sie den Primzahltest als eigenständige Funktion und nicht in `main()`, da sie diese Funktion in der nächsten Aufgabe wiederverwenden werden. Als Eingabe können Sie die Dateien `p10.dat` bis `p500.dat`, sowie `np10.dat` bis `np500.dat` verwenden. Hierbei steht `p` für Primzahl (Die Bedeutung von `np` stellen wir als Übungsaufgabe). Die Zahl danach gibt an, wieviele Stellen die Binärdarstellung der Zahl ungefähr besitzt. Die Dateien `cm1.dat` und `cm2.dat` enthalten Carmichael-Zahlen. Ihr Programm rufen Sie dann beispielsweise so auf:

```
primetest 10 < p300
```

Aufgabe 2 Pattern Matching mit Fingerprints

Implementieren Sie den im Skript beschriebenen Pattern Matching-Algorithmus. Ihr Programm sollte folgende Benutzerschnittstelle zur Verfügung stellen:

- Aufruf mit drei Kommandozeilenparametern:
  1. Name des Files, in dem der Text enthalten ist,

2. zu testendes Pattern,
3. τ (Obergrenze für zufällig gewählte Primzahl, sh. Skript).

- Die Ausgabe Ihres Programms erfolgt auf der Standardausgabe und umfaßt Matches und false matches (d.h. deren Positionen)

Ein Aufruf könnte also folgendermaßen aussehen:

```
fingerprint foo.txt Wrzlbrmpft 1234567
```

und die Ausgabe z. B. so:

```
false match: Position 120
** Match **: Position 312
false match: Position 340
```

Suchen Sie das Pattern `virtual` in der Datei `tcshman.txt` und experimentieren Sie mit verschiedenen Einstellungen für τ. Wie groß muß τ in etwa sein, damit keine false matches mehr auftreten? Geben Sie zum Vergleich den aus der theoretischen Analyse hergeleiteten Wert $\tau = n^2 m \log(n^2 m)$ an.
Zur Realisierung der Arithmetik können Sie wieder auf die LEDA-Klasse `integer` zurückgreifen, auch wenn dann die Laufzeit eigentlich nicht mehr $\mathcal{O}(n+m)$ sondern $\mathcal{O}((n+m)\log\tau)$ beträgt. Davon wollen wir aber aus Gründen der Bequemlichkeit absehen.

Wir wünschen frohe Weihnachten und ein gutes neues Jahrtausend!
Erholsame Ferien!