
Eötvös Loránd Tudományegyetem

Informatikai Kar

Komputeralgebra Tanszék

# Prímszita algoritmusok összehasonlítása

Vatai Emil
Adjunktus

Nagy Péter
Programtervező Informatikus BSc

Budapest, 2018

# Tartalomjegyzék

# 1. fejezet

# Bevezetés

## Motiváció

Sziták összehasonlítása azonos környezetben. Hatékonyság összehasonlítása az elmélet szerint várható értékekkel. Ellenőrzésként a prímek néhány statisztikájának összevetése az elméleti értékekkel, és az ismert eredményekkel. Hatékonyság és implementálhatóság.

## Dolgozat eredményei

# 2. fejezet

# Felhasználói dokumentáció

**A megoldott feladat**

**A program telepítése és futtatása**

**A szitatábla-generátor**

**Mintaadatbázis karbantartása**

**Szitatáblák ellenőrzése**

**Minta megjelenítése**

**Minta közelítése függvényekkel**

# 3. fejezet

# Fejlesztői dokumentáció

## A program komponensei

## A forráskód felosztása

## Adatszerkezetek

## Numerikus algoritmusok

Egyenletrendszerek. Összeadás. Körül kéne írni, hogy igazán tudjuk, hogy hipotézisvizsgálatra nem vállalkozunk.

## Sziták

Eratosztenész szitája, szegmentáltan is. COLS. Prioritásos sorral. Atkin szitája.

Szegmentált szita inicializálása.

Trial division. Pszeudoprím teszt.

Feltételek. Elméleti sebesség.

## Prioritásos sorok

1: $q \leftarrow$ Új-sor
2: **for** $i \leftarrow 2, n$ **do**
3: **while** $\exists (p, k) \in q : k \leq i$ **do**
4: $(p, k) \leftarrow$ Sor-Eltávolít-Min(q)
5: Megjelöl(i)
6: Sor-Beszúr(q, (p, k+p))

```
    end while
    if ¬ MEGJELÖLT?(i) then
        SOR-BESZÚR(q, (i, 2i))
    end if
end for
```

## Bináris kupac

A mérések grafikonják pixelei alapján lassú. A beszúrásonkénti elméleti $\mathcal{O}(log|q|)$ ideje se biztató.

## Bigyó

A bigyó egy természetesszám-párokat tartalmazó monoton prioritásos sor. A számpárok egy prímet, és a prím egy szitálási pozícióját reprezentálják. A sor monoton, minden állapotához tartozik egy érték, a sor aktuális pozíciója, aminél kisebb vagy egyenlő pozíciójú értéket a sor nem tartalmazhat. A bigyó edények egy végtelen sorozatát is tárolja, a sor elemei ezekbe az edényekbe kerülnek. Egy eltárolt elem helyét a sorozatban az elem pozíciójának és a sor aktuális pozíciójának távolsága határozza meg.

A távolságfüggvény legyen

$$d(x, y) := \lfloor log_2(x \oplus y) \rfloor \qquad (x, y \in \mathbb{N}, y > x \geq 0)$$

ahol $\oplus$ a bitenkénti XOR.

$d(x, y)$ a legnagyobb bit-index, ahol $x$ és $y$ eltér.

Ha $q$ egy bigyó, legyen $q.a$ $q$ aktuális pozíciója, és $q.e[i]$ $q$ $i$. edénye. AZ edények, és a számpárok struktúrája...

Egy q bigyó invariánsa

$$\begin{aligned}
&\forall (p, k) \in q : \\
&\qquad q.a < k \\
&\qquad \forall i \in \mathbb{N}_0 : (p, k) \in q.e[i] \iff i = d(q.a, k) \\
&\forall (p, k) \notin q : \forall i \in \mathbb{N}_0 : (p, k) \notin q.e[i]
\end{aligned}$$

Új, üres sor létrehozása tetszőleges kezdőpozíciótól, és meglévő sorba elem beszúrása...

A sor elemeinek feldolgozása $i$-ig

```
while q.a < i do
```

```
    j ← d(q.a, q.a + 1)
    q.a ← q.a + 1
    for all (p, k) ∈ q.e[j] do
        Edény-Kivesz(q.e[j], (p, k))
        if k = i then
            Visszaad((p, k))
        else
            Edény-Beszúr(d(q.a, k), (p, k))
        end if
    end for
end while
```

**Helyesség**

**Idő**

**Hely**

**Számrendszer**

**Cache**

# Memória

Összes prím, és pozíciója $2^32$-ig. Garbage collector.

# Teszt

## Adatszerkezetek

## Numerikus pontosság és sebesség

## Sziták

## Elmélet vs. mért

# Források