

Sundaramovo síto

A srovnání se sítí Eratosthenovým

Autor: **Marco Souza de Joode**, G. Nad Štolou

1. května 2021

Eratosthenovo síto

- Předpokládáme, že všechna čísla jsou prvočísla
- Postupně odstraňujeme všechny celočíselné násobky
- Případně pouze celočíselné násobky čísel, které nebyly odstraněny.

Myšlenka za Sundaramovým sítím

Všechna přirozená čísla jsou buď sudá nebo lichá. Všechna prvočísla, až na číslo 2, jsou lichá. **Sundaramovo síto** selektivně vybírá složená lichá čísla a odstraňuje je.

Součiny čísel podle parity

Označíme-li si sudá čísla jako S a lichá čísla jako L , pak platí

$$S \cdot S = S$$

$$S \cdot L = S$$

$$L \cdot S = S$$

$$L \cdot L = L$$

protože

$$2n \cdot 2m = 4nm = 2 \cdot 2nm$$

$$2n \cdot (2m + 1) = 4nm + 2n = 2 \cdot (2nm + n)$$

$$(2n + 1) \cdot (2m + 1) = 4nm + 2n + 2m + 1 = 2 \cdot (n + m + 2nm) + 1$$

Složená lichá čísla

Všechna lichá čísla jsou buď prvočísla, nebo čísla složená. Každé liché složené číslo lze zapsat jako

$$\begin{aligned}(2i + 1)(2j + 1) \\&= 4ij + 2i + 2j + 1 \\&= 2\underbrace{(i + j + 2ij)}_U + 1\end{aligned}$$

Základní princip Sundaramova síta

- Mějme $m = \lfloor N/2 \rfloor$
- Pro všechna $i \in \{1, 2, \dots, m\}$ a pro všechna $j \in \{1, 2, \dots, m\}$ nalezněme množinu čísel L , které **nelze** zapsat jako $U = i + j + 2ij$, když $U < m$.
- Všechna lichá prvočísla lze zapsat jako $2U + 1$ pro každé U v množině L .

Optimalizace Sundaramova síta

- Mějme $m = \lfloor N/2 \rfloor$
- Pro všechna $i \in \{1, 2, \dots, m\}$, aby $U = i + j + 2ij < m$, musí platit omezení na j :

$$\begin{aligned}i + j + 2ij &< m \\j(2i + 1) &< m - i \\j &< \frac{m - i}{2i + 1}\end{aligned}$$

a zároveň nemusíme prověřovat $j \leq i$. Pak

$$j \in \{i, i + 1, \dots, \frac{m-i}{2i+1}\}$$

- Všechna lichá prvočísla lze zapsat jako $2U + 1$ pro každé U v množině L .

Eratosthenovo síto

```
1 def eratosthenes(n):
2     multiples = []
3     primes = []
4     for i in range(2, n+1):
5         if i not in multiples:
6             primes.append(i)
7             for j in range(i*i, n+1, i):
8                 multiples.append(j)
9     return np.array(primes)
```

Budované pomocí dvou seznamů

- Seznamu násobků
- Seznamu složených čísel

Naivní Eratosthenovo síto pomocí Booleovských masek

```
1 def eratosthenes_boolean(n):
2     L = np.ones((n,), dtype="bool")
3     for i in range(2,n):
4         k = 2
5         while k*i < n:
6             L[k*i] = False
7             k += 1
8     L[0], L[1] = False, False
9     return np.array([i for i, l in enumerate(L) if l])
10
```

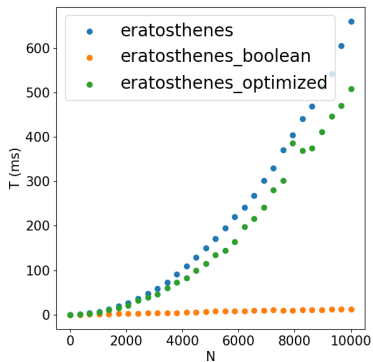
Používá pouze jedno numpy pole zaplněné booleovskými hodnotami True, poté odstraňuje (naivně) všechny násobky všech čísel.

Optimalizované Eratosthenovo síto

```
1 def eratosathenes_optimized(n):
2     primes = [i for i in range(2,n+1)]
3     for i in primes:
4         k = 2
5         while k*i <= n:
6             try:
7                 primes.remove(k*i)
8             except ValueError:
9                 pass
10            k+= 1
11     return primes
```

Na začátku mají všechna čísla status prvočísla. Následně jsou odstraňovány všechny násobky zbývajících čísel s tímto statusem.

Vzájemné srovnání



Vyhrává síto využívající booleovské masky.

Časová složitost Eratosthenova síta

Něco jako nejlepší nebo nejhorší případ zde nehraje roli - vždy procházíme tentýž seznam čísel. Za předpokladu, že odstranění složeného čísla provedeme v čase $O(1)$, tak odstranění všech násobků vyžaduje minimálně

$$O\left(\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \dots + \frac{N}{p}\right) \quad (1)$$

kde p je nejvyšší prvočíslo takové, že $p < N$.

Časová složitost Eratosthenova síta

Euler (1737) dokazuje, že

$$\sum_{p \text{ je prv.} < N} \frac{1}{p} \geq \log \log(N+1) - \log \frac{\pi^2}{6} \quad (2)$$

Což pro velké hodnoty N

$$\log \log(N+1) - \log \frac{\pi^2}{6} \approx \log \log(N), \quad (3)$$

a tak

$$O\left(\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \dots + \frac{N}{p}\right) \approx O(N \log \log N). \quad (4)$$

Naivní Sundaramovo síto

```
1 def sundaram_naive_boolean(n):
2     m = n//2
3     L = [True] * n
4
5     for i in range(1, m):
6         L[2*i] = False
7         for j in range(1, m):
8             U = i + j + 2*i*j
9             if U < m:
10                L[2*U+1] = False
11
12     L[0], L[1] = False, False
13     L[2] = True
14     return np.array([l for i, l in enumerate(L) if l])
```

Pomocí seznamu Booleovských hodnot. Odstraňuje čísla na indexech $U = i + j + 2ij$ menší než $m = \lfloor N/2 \rfloor$.

Časová složitost naivního Sundaramova síta

Prochází všechny body v kartézském součinu

$$i \in \{2, \dots, \lfloor N/2 \rfloor\} \times j \in \{2, \dots, \lfloor N/2 \rfloor\}$$

a operuje tedy minimálně s časem

$$O(\lfloor N/2 \rfloor^2) \approx O\left(\frac{1}{4}N^2\right) \approx O(N^2) \quad (5)$$

což oproti nejlepšímu Eratosthenovu sítu představuje výrazné zhoršení.

Naivní Sundaramovo síto, implementace pomocí množin

```
1 def sundaram_naive_set(n):
2     m = n // 2
3     L = {l for l in range(1,m)}
4     for i in range(1,m):
5         for j in range(i,m):
6             U = i + j + 2*i*j
7             if U < m:
8                 L.discard(U)
9     primes = deque([2*l+1 for l in L])
10    primes.extendleft([2])
11    return np.array(primes)
```

Povšimneme si, že není třeba procházet celý prostor

$$i \in \{2, \dots, \lfloor N/2 \rfloor\} \times j \in \{2, \dots, \lfloor N/2 \rfloor\}$$

ale stačí pouze

$$i \in \{2, \dots, \lfloor N/2 \rfloor\} \times j \in \{i, i+1, \dots, \lfloor N/2 \rfloor\}$$

Optimalizace Sundaramova síta

```
1 def sundaram_optimized_set(n):
2     m = n // 2
3     L = {l for l in range(1,m)}
4     for i in range(1,m):
5         limit = (m-i)//(2*i+1)
6         for j in range(i, limit+1):
7             U = i + j + 2*i*j
8             L.discard(U)
9     primes = deque([2*l+1 for l in L])
10    primes.extendleft([2])
11    return np.array(primes)
```

Optimalizace Sundaramova síta

Protože

$$i + j + 2ij < m$$

$$j(2i + 1) < m - i$$

$$j < \frac{m - i}{2i + 1}$$

a zároveň nemusíme prověřovat $j \leq i$. Pak $j \in \{i, i + 1, \dots, \frac{m-i}{2i+1}\}$

Optimalizace Sundaramova síta pomocí Booleovské masky

```
1 def sundaram_optimized_boolean(n):
2     m = n//2
3     L = [True] * n
4
5     for i in range(1, m):
6         L[2*i] = False
7         limit = (m-i)//(2*i+1)
8         for j in range(i, limit+1):
9             U = i + j + 2*i*j
10            if U < m:
11                L[2*U+1] = False
12
13     L[0], L[1], L[2] = False, False, True
14     return np.array([l for i, l in enumerate(L) if l])
```

Idealizovaná časová složitost

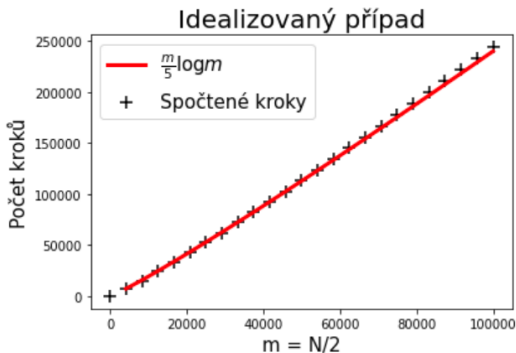
Časová složitost je závislá na počtu kroků, který si můžeme spočítat uměle:

```
1 steps = []
2 M = np.linspace(0,10**5, 25, dtype="int64")
3
4 for m in M:
5     s = 0
6     for i in range(1,m):
7         limit = (m-i)//(2*i+1) + 1
8         for j in range(i, limit):
9             s += 1
10    steps.append(s)
```

Idealizovaná časová složitost

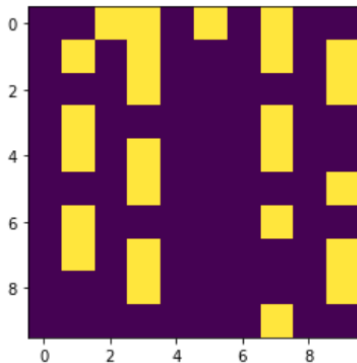
Bylo dokázáno (Sundaram, 1934), že časová složitost Sundaramova síta v této podobě se chová jako

$$O\left(\frac{m}{5} \log m\right) = O\left(\frac{\lfloor N/2 \rfloor}{5} \log \lfloor N/2 \rfloor\right) < N \log N \quad (6)$$

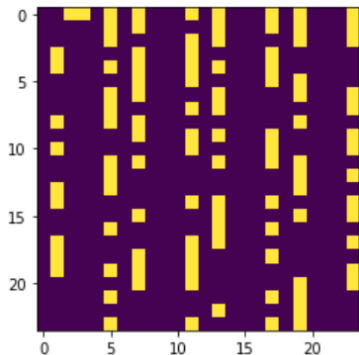


Vizualizace prvočísel: 100

Vizualizace prvočísel

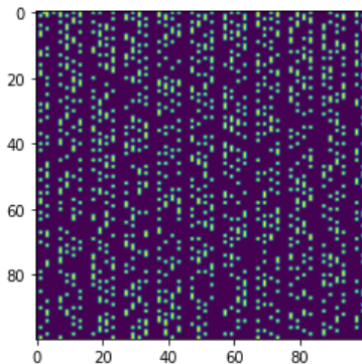


Vizualizace prvočísel: 576

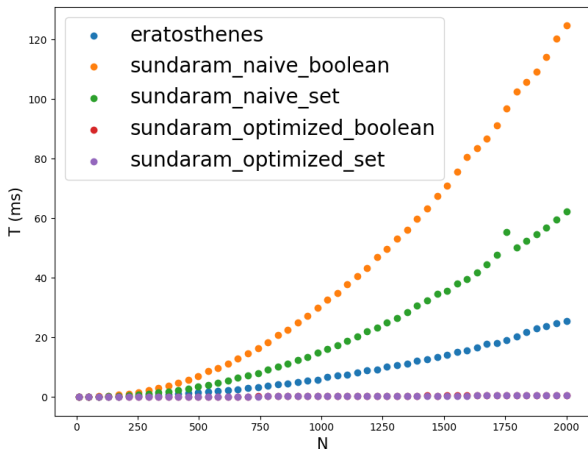


Vizualizace prvočísel: 10000

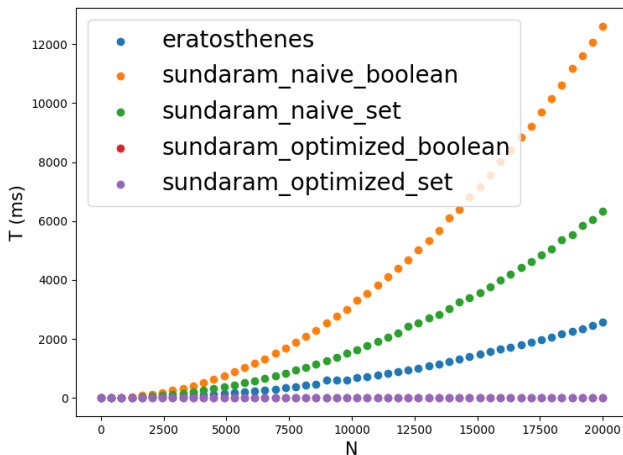
Vizualizace prvočísel



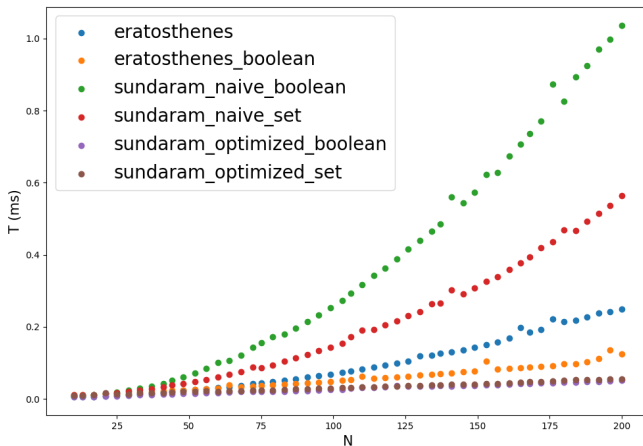
Srovnání sít



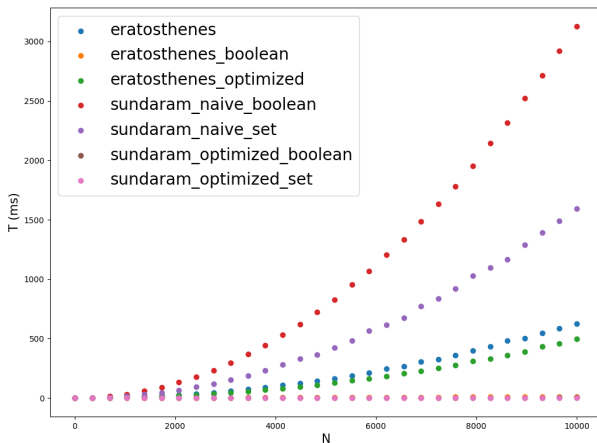
Srovnání sít



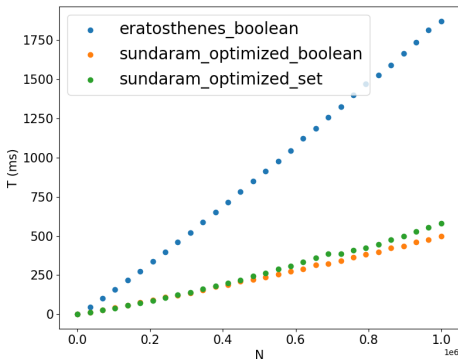
Srovnání sít



Srovnání sít



Nejlepší síta: „výherci“!



Pozor, osa N je v řádu 10^6

Děkuji za pozornost.