OPTIMALIZACE ERATOSTHENOVA SÍTA

Jiří Harčár
Přírodovědecká fakulta, Katedra informatiky a počítačů, 30. Dubna 22, 701 03,
r14278@student.osu.cz

Abstrakt

Tento příspěvek se zabývá optimalizací algoritmu Eratosthenova síta, která je vyčíslena časovým porovnáním generování množin prvočísel v určeném rozsahu.

Eratosthenovo síto je jednoduchý algoritmus pro nalezení všech prvočísel menších než zadaná horní mez. Je pojmenován po řeckém matematikovi Eratosthenovi. Algoritmus funguje „prosíváním" seznamu čísel – na počátku seznam obsahuje všechna čísla v daném rozsahu (2, 3, 4, …, max). Poté se opakovaně první číslo ze seznamu vyjme, toto číslo je prvočíslem; ze seznamu se pak odstraní všechny násobky tohoto čísla (což jsou čísla složená). Tak se pokračuje do doby, než je ze seznamu odstraněno poslední číslo (nebo ve chvíli, kdy je jako prvočíslo označeno číslo vyšší než odmocnina nejvyššího čísla – v takové chvíli už všechna zbývající čísla jsou nutně prvočísly). Časová složitost tohoto algoritmu je O(N*log(log N)), kde N je horní mez rozsahu [1].

Vycházím z Eratosthenova síta, přičemž využívám nápadu, který mi dovoluje vypočítat prvočísla, nebo součiny vyšších prvočísel. Nejjednodušším způsob použití je při rozsahu prvočísla 2. Toto prvočíslo rozdělí řadu přirozených čísel na 2 sloupce: k*2 + 0; k*2 + 1. Sloupec k*2 + 0, můžeme „vyhodit", protože v tomto sloupci nalezneme všechny násobky prvočísla 2. Zatímco 2. sloupec k*2 + 1 můžeme využít, protože víme, že obsahuje všechna ostatní prvočísla. Tento sloupec, ale můžeme vylepšit, když rozšíříme rozsah o následující prvočíslo (nezáleží, kterým prvočíslem rozšiřujeme, ale čím je to prvočíslo menší, tím dosáhneme přesnějších výsledků). Takže teď máme rozsah 2,3 (6). Tím, že rozšiřujeme rozsah prvočíslem 3 tak 3x rozšíříme i sloupec z předchozího rozsahu.

k*2 + 1 -> k*6 +1;~~k*6 + 3~~;k*6 + 5

Teď jsem tento sloupec k*2 + 1 zvýšil na vyšší rozsah, z čehož mi vznikly 3 další sloupce, jejichž součtem bych dostal původní sloupec k*2 + 1. Z těchto nových 3 sloupců odstraníme ten, který je násobkem prvočísla 3, což je prostřední sloupec. Teď nám zůstaly 2 sloupce k*6 + 1;k*6 +5, které můžeme provést dalším zvýšením rozsahu o prvočíslo 5.

k*6 + 1; k*6 +5 -> k*30 + 1; k*30 + 5; k*30 + 7; k*30 + 11; k*30 + 13; k*30 + 17; k*30 + 19; k*30 + 23; k*30 + 25; k*30 + 29

Z původních 2 sloupců rozsahu 2,3(6) jsme rozšířením o prvočíslo 5 dostali rozsah 2,3,5(30) s 8-mi novými sloupci. Tento postup se může aplikovat neomezeně, pouze náročnost na paměť s každým zvýšením rozsahu exponenciálně stoupá.

Při použití této metody při faktorizaci čísel, je potřeba pouze zajistit, aby faktorizované číslo nemělo za dělitele prvočísla, která vytváří rozsah (pro nás teď je to 2,3,5). Dále je potřeba zajistit testovací hodnoty pro faktorizaci, tyto hodnoty stačí vypočíst pouze jednou pro neomezený počet faktorizací. Z výše uvedených sloupců budeme dále ignorovat část s k*30 a bude nás zajímat pouze ta druhá hodnota. Těchto 8 hodnot si dáme do tabulky a budou představovat sloupce i řádky zároveň. Hodnoty v takové tabulce (viz. Tab.1) budou dvojí, první hodnota bude vždy modulo 30 násobku hodnoty řádku se sloupcem. Druhá hodnota bude

(sloupec – řádek)/2 a pokud to bude záporná hodnota, tak se k té hodnotě přidá číslo 15 (polovina rozsahu).

Tabulka 1. Vygenerované hodnoty (testovací hodnota pro faktorizaci/zbytek po dělení)

	1	7	11	13	17	19	23	29
1	1|0	7|3	11|5	13|6	17|8	19|9	23|11	29|14
7	7|12	19|0	17|2	1|3	29|5	13|6	11|8	23|11
11	11|10	17|13	1|0	23|1	7|3	29|4	13|6	19|9
13	13|9	1|12	23|14	19|0	11|2	7|3	29|5	17|8
17	17|7	29|10	7|12	11|13	19|0	23|1	1|3	13|6
19	19|6	13|9	29|11	7|12	23|14	1|0	17|2	11|5
23	23|4	11|7	13|9	29|10	1|12	17|13	19|0	7|3
29	29|1	23|4	19|6	17|7	13|9	11|10	7|12	1|0

Z této tabulky už stačí vypsat, k jakým koncovkám sloupců patří které číslo.

1-0,3,12; 7-3,12; 11-2,5,7,8,10,13; 13-6,9; 17-2,7,8,13;
19-0,6,9; 23-1,4,11,14; 29-1,4,5,10,11,14

Z toho nám vyplývá, že pokud chceme faktorizovat číslo, tak zjistíme výsledek z modula 30 a podle toho výsledku vybereme sadu hodnot, podle kterých budeme testovat výsledek. Testujeme pomocí vzorce $p^2 = c + (a_x + k \times 15)^2$, kde c je faktorizované číslo, a je testovací hodnota a k je přirozené číslo, které se zvyšuje počtem průchodů při testování a p je kvadrát, který pokud je celočíselný, tak jsme provedli úspěšnou faktorizaci.

Př.: Mějme číslo 187, modulo 30 je 7, vybereme testovací hodnoty 3,12 a v cyklu testujeme:
$187 + (3 + 0*15)^2 = 196 = 14^2$ úspěšná faktorizace při 1. průchodu cyklem.

Tyto rozsahy se dají použít nejenom k faktorizaci, ale také k výpočtu prvočísel, přičemž dosahují lepších výsledků než Eratosthenovo síto podle testování (viz. Tab 2).

Tabulka 2. Výsledky porovnání časové náročnosti výpočtu v [ms]

Zvolený rozsah	Eratosthenovo síto	$ERTH_{opt}$
1-510510	13	6
1-9699690	333	125
1-223092870	9025	3000

Princip výpočtu prvočísel pomocí rozsahů je jednoduchý. Vypočte se požadovaný rozsah, podle toho do jakého maxima chceme vypočítat prvočísla (jestli do 1 miliónu, nebo 100 biliard apod.). Po vypočtení požadovaného rozsahu se pokračuje v algoritmu Erastothenova síta, ale až od nejmenšího prvočísla, které není součástí vypočteného rozsahu. Paměťová náročnost tohoto generování je pro příklad prvočíselného rozsahu 19 v řádech GB. Také se tyto rozsahy dají použít k nalezení nejpravděpodobnějšího výskytu prvočíselných párů. Pro nalezení prvočíselných párů stačí počítat samotné rozsahy, ale budou obsahovat jenom prvočíselné páry. Což uvolňuje obrovské množství pamětí a nechává prostor pro nejpravděpodobnější kandidáty pro prvočíselné dvojice. Také tyto rozsahy mě dovedly k nalezení rovnic, které ukazují zajímavé skutečnosti o Goldbachově hypotéze, které by mohly pomoci k nalezení důkazu k této hypotéze.

Klíčová slova: *Eratosthenovo síto; faktorizace, prvočíslo.*

Literatura

[1.] Zdroj Wikipedie [online], [cit. 30. března 2015], Eratosthenovo síto. Dostupné na Internetu: http://cs.wikipedia.org/wiki/Eratosthenovo_s%C3%ADto.