Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

ISTI Technical Reports

RSIEVE: considerazioni sulla generazione dei numeri primi

Marco Righi, CNR-ISTI, Pisa, Italy

ISTI-TR-2021/021

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

RSIEVE: considerazioni sulla generazione dei numeri primi
Righi M.
ISTI-TR-2021/021

Abstract
Appunti sulla generazione dei numeri primi

Numeri primi, Matematica, Algebra

Citation
Righi M. *RSIEVE: considerazioni sulla generazione dei numeri primi* ISTI Technical Reports 2021/021. DOI: 10.32079/ISTI-TR-2021/021

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Area della Ricerca CNR di Pisa
Via G. Moruzzi 1
56124 Pisa Italy
http://www.isti.cnr.it

RSIEVE: considerazioni sulla generazione dei numeri primi

Marco Righi (marco.righi@isti.cnr.it)

08/11/2021

1 Problema

Si crivella fino a $\boldsymbol{O}$ (test fatti solo per $\boldsymbol{O}$ pari)

2 Idea di base con ottimizzazioni minime per la riduzione della complessità

Le ottimizzazioni realizzate per la riduzione della complessità si dividono in più semplici qui esposte e più complicate spiegate più avanti. Questa sezione mostra quelle definite più semplici mentre le complicate sono esposte più avanti.

1. Round 1
   (a) si scrivono i numeri primi fino a $\sqrt{\boldsymbol{O}}$(questi primi devono essere disponibili). Questo insieme prende il nome di `base`.
   (b) si moltiplicano tra loro escludendo quelli maggiori di $\boldsymbol{O}$ calcolando quello che chiamiamo insieme dei `numbers_generated_by_base` (osservazione. essendo presente il numero 2 sono tutti numeri pari)
   (c) dall'insieme dei `numbers_generated_by_base` maggiori di $\sqrt{\boldsymbol{O}}$ (fino a $\sqrt{\boldsymbol{O}}$ abbiamo già l'elenco completo dei primi) si toglie 1 e si crea l'insieme dei `possible_primes`
   (d) l'insieme dei `possible_primes` si crivella con `criteri` di divisibilità e divisione classica per numeri fino alla radice del numero in oggetto utilizzando i numeri primi a disposizione
   (e) otteniamo un sottoinsieme detto dei `calculated_primes` che è sottoinsieme di tutti i numeri primi fino a $\boldsymbol{O}$
   (f) l'insieme dei `calculated_primes` viene copiato sull'insieme dei `primes_to_explore`
   (g) si ordina in ordine crescente l'insieme dei `numbers_generated_by_base`
2. Round 2.....T: questo round si ripete fino a quando l'elenco dei primi da esplorare non è vuoto
   (a) ogni elemento nell'insieme dei `primes_to_explore` inferiore a $\frac{\boldsymbol{O}}{2}$ (il più piccolo coefficiente dei `numbers_generated_by_base` per la moltiplicazione è 2) si moltiplica per i `numbers_generated_by_base` interrompendo il calcolo non appena il prodotto supera $\boldsymbol{O}$. Man a

mano che si calcola un numero se ne verifica la primalità, se è un nuovo primo si inserisce nell'insieme dei `calculated_primes` e dei `primes_to_explore`

(b) Si ripete il punto 2 fino a quando la lista dei `primes_to_explore` non è vuota.

3 Suddivisione del codice

Il codice è suddiviso in due parti: la prima parte per il calcolo dei dati del Round 1 e le restanti per il calcolo dei successivi Round.

3.1 Calcolo del Round 1

Per effettuare questo calcolo si suddivide il codice in 3 parti: il corpo principale che chiama la funzione `MyNextGoodNumber` che a sua volta si avvale della funzione `MyNextNumber:`

- la funzione `MyNextGoodNumber` ha due scopi principali: generare numeri di valore inferiore a $\boldsymbol{O}$e scartare quanti più numeri possibile per effettuare il minor numero possibile di moltiplicazioni per calcolare i `numbers_generated_by_base` lavorando con numeri che siano i più piccoli possibili. Tutti i `numbers_generated_by_base` sono numeri pari quindi la funzione scarta anche i numeri dispari.
- la funzione `MyNextNumber` calcola il valore successivo secondo una data algebra.

3.1.1 Algebra utilizzata

Nell'algebra utilizzata in questo contesto le cifre meno significative sono a sinistra e le più significative sono a destra. Ogni cifra ha il suo valore di riporto che è indicato nelle variabili `baseMaxPower` e `localbaseMaxPower`. I valori di `localbaseMaxPower` rispecchiano la regola `localbaseMaxPower` $\preccurlyeq$ `baseMaxPower` e seguono l'ordinamento descritto di seguito.

Definizione 1. `localbaseMaxPower` $\preccurlyeq$ `baseMaxPower`.
Siano `baseMaxPower` e `localbaseMaxPower` due vettori di K posizioni con $p_i \in$ `baseMaxPower` e $q_i \in$ `localbaseMaxPower` allora $\forall i \in [1 \dots K]$ abbiamo che $p_i \geqslant q_i$.
La variabile `baseMaxPower` viene calcolata all'inizio del codice e è il limite di salvaguardia per verificare se lo spazio di ricerca è esaurito o meno mentre il limite che viene utilizzato e calcolato dinamicamente più volte nel codice è `localbaseMaxPower.`

Definiamo adesso il successore in questa algebra.

Definizione 2. Successore. Il successore ha riporto solo al superamento del valore di `baseMaxPower`.

Esempio 1. Sia data baseMaxPower=$\{2,5,1,1\}$. Il successore di $\{0,0,0,0\}$ è $\{1,0,0,0\}$. Il successore di $\{2,0,0,0\}$ è $\{0,1,0,0\}$.

Definizione 3. Date le variabili `base`=$\{b_1, \ldots, b_n\}$ e un generico numero `num`=$\{d_1, \ldots, d_n\}$ definiamo il prodotto come segue:

$$prodotto = \prod_{i=1}^{n} b_i^{d_i} \tag{1}$$

3.1.2 MyCalculateLocalbaseMaxPower

Come su indicato, `numbers_generated_by_base` si ottiene moltiplicando i numeri primi tra di loro assegnando a ogni numero un esponente tale da non superare il range di interesse, ovvero $\boldsymbol{O}$.
Per fare questo abbiamo una prima approssimazione come limite superiore da dare ai vari esponenti che hanno i numeri primi della `base`.
I numeri primi costituenti la `base`, che devono essere precalcolati, sono tutti i numeri primi da 2 a $\sqrt{\boldsymbol{O}}$.
Consideriamo la base costituita da $\{b_1, \ldots, b_n\}$. `baseMaxPower` ha valori $\{m_1, \ldots, m_n\}$cosí calcolati:

$$\begin{cases} m_1 = \log_{b_1} \boldsymbol{O} & (m_1 = 2) \\ m_i = \log_{b_i} \dfrac{\boldsymbol{O}}{2} & i > 1 \end{cases} \tag{2}$$

Poiché sono generati solo numeri pari per un valore della base $b_i$dispari ($i > 1$) dovremo rispettare la condizione che $2 \cdot b_i^{m_i} \leq \boldsymbol{O}$.

Esempio 2. Crivellando fino a $\boldsymbol{O} = 1000$ abbiamo:

- `base`=$\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}$
- `baseMaxPower`=$\{9, 5, 3, 3, 2, 2, 2, 2, 1, 1, 1\}$

Al crescere degli esponenti, ovvero della variabile numero, la valore massimo che puè assumere un esponente diminuisce. Poniamo di avere la variabile `num`=$\{d_1, \ldots, d_k, \ldots, d_n\}$ dove $1 < k \leq n$ e $d_k \neq 0$ e $\forall \quad k < i \leq n \quad d_i = 0$.
Possiamo calcolare un nuovo valore di potenza massima locale (`localbaseMaxPower`) limitato da $d_k$come segue:

$$\begin{cases} l_1 = \log_{b_1} \dfrac{\boldsymbol{O}}{b_k^{d_k}} & (m_1 = 2) \\ l_i = \log_{b_i} \dfrac{\boldsymbol{O}}{2 \cdot b_k^{d_k}} & i > 1 \end{cases} \tag{3}$$

La funzione `MyCalculateLocalbaseMaxPower` ha i seguenti parametri:

- Input:
  - `num`
  - `base`
  - $\boldsymbol{O}$
- Output:
  - `localbaseMaxPower`

3.1.3 `MyNextNumber`

La funzione `MyNextNumber` implementa quanto delineato in `3.1.1.`

- Input:
  - un numero della base `num`
  - la `base` di riferimento
  - `localbaseMaxPower`
- Output
  - il numero successivo nella base di riferimento
  - se i numeri da esplorare sono stati finiti

`Considerando num=`$\{d_1, \ldots, d_n\}$ `e baseMaxPower=`$\{m_1, \ldots, m_n\}$, se `num==baseMaxPower` (ovvero $\forall \quad 1 \leq i \leq n \quad d_i = m_i$) allora è terminato lo spazio da esplorare

3.1.4 `MyNextGoodNumber`

`MyNextGoodNumber` ha i seguenti paramenti:

- Input
  - un numero **num** di cui calcolare il successore
  - baseMaxPower (definita in seguito, è una base dell'algebra, viene calcolata all'inizio del programma)
  - ***O***
  - `base`
- Output:
  - n, il successivo numero idoneo
  - il prodotto calcolato da n
  - endNumbers, booleano che indica che i numeri calcolare sono finiti

I paramatri di accettabilità (o idoneinità) di un numero per `MyNextGoodNumber` sono:

- il numero deve essere minore o uguale a ***O***
- il numero deve essere pari primi genererebbe solo un multiplo di 5)

`MyNextGoodNumber` esegue le seguenti istruzioni:

1. calcola `localbaseMaxPower` chiamando la funzione 3.1.2
2. Fino a quando non ha trovato un numero pari che rispetti i criteri di essere pari e inferiore a N oppure sia terminato lo spazio da esplorare esegue quanto segue:
   (a) forza la chiamata al numero successivo (`mustCallMyNextNumber = True`)

(b) Se `mustCallMyNextNumber == True allora chiama il successore di num` che prende il nome di `num+1`

(c) Se il prodotto calcolato come da definizione 3 rispetta i parametri di accettabilità, la funzione `MyNextGoodNumber` restituisce `num`. Osserviamo che il prodotto è effettuato con un ciclo un termine alla volta, questo ciclo termina se sono stati moltiplicati tutti i valori oppure appena il valore calcolato supera $\boldsymbol{O}$ (**isMaxExceed** = **True**).

(d) Se lo spazio di ricerca non è esaurito e **isMaxExceed** == **True** viene generato il prossimo numero pari più piccolo di $\boldsymbol{O}$.Questo calcolo viene descritto nella sezione 3.1.5. Trovato il valore viene posto `mustCallMyNextNumber = True` cosí da evitare la chiamata al calcolo del successivo al prossimo ciclo. Il calcolo effettuato a questo punto puó far terminare il ciclo del punto 2 in quanto lo spazio di ricerca è terminato.

3.1.5 Generazione del successivo valore al Round 1 dopo il superamento della soglia $\boldsymbol{O}$.

L'obiettivo di questo paragrafo è introdurre un metodo per minimizzare i numeri da controllare quando il prodotto $\prod_{i=1}^{k} b_i^{d_i}$ per $i = k \leq n$ supera $\boldsymbol{O}$ quando l'indice da incrementare $i > 1$.

Lo stato in cui si trova l'algoritmo quando è utile questo approccio lo riassumiamo come segue:

- `num`=$\{d_1, \ldots, d_n\}$ con $\sum_{i=1}^{k} d_i > 0$ e $\sum_{i=k+1}^{n} d_i = 0$ e $\sum_{i=2}^{n} d_i > 0$ con $k \leq n$.
- `localbaseMaxPower`=$\{l_1, \ldots, l_n\}$
- `base`=$\{b_1, \ldots, b_n\}$
- massimo numero da crivellare $\boldsymbol{O}$

Appena il calcolo $\prod_{i=1}^{k} b_i^{d_i} > \boldsymbol{O}$ per $i = k \leq n$ abbiamo superato la soglia $\boldsymbol{O}$.

Espresso altrimenti abbiamo che $b_1^{d_1} \cdot \ldots \cdot b_k^{d_k} > \boldsymbol{O}$.
Per calcolare il valore successivo vi sono due alternative:

1. provare a variare i d_i con $i \leq k$
2. incrementare d_{k+1} di 1 unità.

Vediamo in dettaglio il punto 1 appena elencato.
Si calcola il primo valore h (con $1 < h \leq n$) t.che:

$$\begin{cases} 2 \cdot b_h^{d_h+1} \leq \sum_{i=1}^{h} b_i^{d_i} \\ 2 \cdot b_h^{d_h+1} \leq \boldsymbol{O} \\ d_h + 1 \leq l_h \end{cases} \tag{4}$$

Notiamo che la condizione $d_h + 1 \leq l_h$ Ãš ridondante ma utile al calcolo. È ridondante in quanto se questa non è verificata non lo è neppure $2 \cdot b_h^{d_h+1} \leq \boldsymbol{O}$.

Il fattore 2 è presente in quanto si generano solo numeri pari. Se l'eq. 4 non ha soluzione per $h \leq k$ e $d_n < l_n$allora viene fornita come soluzione `num+1`$=\{d_1 = 1, 0, \ldots, 0, d_{k+1} = 1, 0, \ldots, 0, d_n = 0\}$.

Esempio 3. Siano dati:

- $\boldsymbol{O} = 25000$
- `base`=

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	71	73	79	83	89	97	101	103
107	109	113	127	131	137	139	149	151	157			

- `baseMaxPower`=

14	8	5	4	3	3	3	3	3	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	1	1	1	1	1	1	1			

- `localBaseMaxPower`=

11	6	4	3	3	3	3	3	3	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	1	1	1	1	1	1	1			

- $\texttt{num}_a$=

1	3	2	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0			

Il valore ottenuto con $\texttt{num}_a$ è 14850, ovvero viene accettato. Il valore successivo, che chiamiamo $\texttt{num}_b$ è

2	3	2	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0			

ovvero 29700, superiore alla soglia di $\boldsymbol{O} = 25000$. Adesso per il calcolo del successibvo utilizziamo l'algoritmo indicato.
Focalizziamo l'attenzione sul calcolo del successivo utilizzando 3.1.5. Vediamo tutti i passi che vengono eseguiti.

01 iniziando dall'indice $i = 2$ andiamo a calcolare $2^2 \cdot 3^3 \geq 2 \cdot 3^4$ ovvero $108 \geq 81$? Sì, quindi il prossimo numero da prendere è:

1	4	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0			

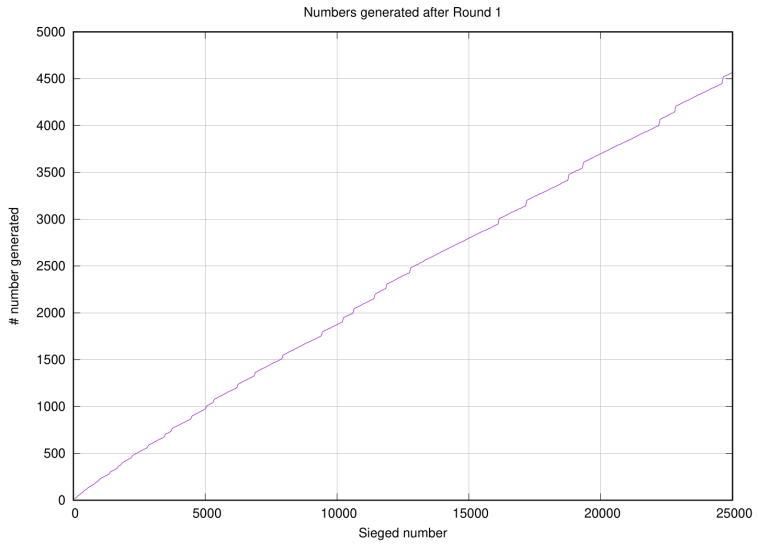

Figura 1: Cardinalità dei `numbers_generated_by_base` dopo il round 1 in funzione del numero crivellato

Appena andiamo a verificare questo numero ci accorgiamo che $2^1 \cdot 3^4 \cdot 5^1 \cdot 11^1 = 44550$ quindi è stata nuovamene superata la soglia $\boldsymbol{O} = 25000$. Ripetiamo il procedimeto già visto:

01 $2^1 \cdot 3^4 \geq 2 \cdot 3^5$ ovvero $162 \geq 486$? No, quindi facciamo scorrere l'indice i.

02 $2^1 \cdot 3^4 \cdot 5^2 \geq 2 \cdot 5^3$ ovvero $4050 \geq 250$? Sì, quindi il prossimo numero da prendere è:

1	0	3	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0			

3.1.6 Round 1: calcolo dei primi

Il calcolo dei primi avviene prendendo i `numbers_generated_by_base` e sottraendovi 1. Questo insieme prende il nome di `possible_primes`. L'insieme dei possibili primi viene crivellato con un l'insieme dei numeri primi disponibili. La quantità dei `numbers_generated_by_base` al round 1 è rappresentato in figura. 1.

3.1.7 Considerazioni sul numero di chiamate alle funzioni `MyNextGoodNumber` e `MyNextNumber`

In figura 2 sono rappresentati i cicli eseguiti dalle funzioni in oggetto.

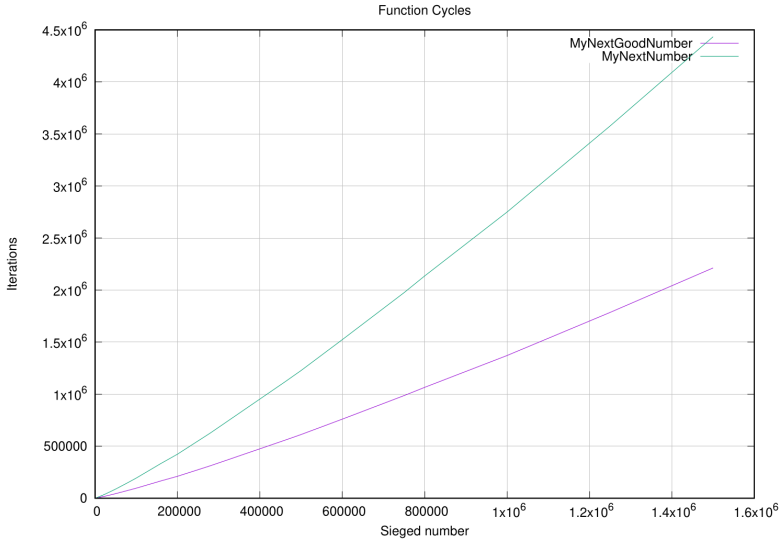

Figura 2: Cicli eseguiti dalle funzioni `MyNextGoodNumber` e `MyGoodNumber`

3.2 Calcolo dei Round successivi al primo

Al termine del Round 1 abbiamo a disposizione l'insieme dei numeri generati dalal base.

Definizione 4. `numbers_generated_by_base`$=\{bg_1, \ldots, bg_g\}$

Questo insieme viene ordinato in ordine crescente e prende il nome di numeri generati dalla base ordinati.

Definizione 5.
`srt_numbers_generated_by_base`=`Sort(numbers_generated_by_base)`.

I numeri primi che costituiscono la `base` e i nuovi numeri primi generati dall'algoritmo al Round 1, sono indicati con `generated_primes(1)`. L'insieme dei numeri primi generati al passo k è dato dall'unione della **base** e dell'insieme `generated_primes(k)` ovvero i primi generati al passo k (eq. 5).

$$generated\_primes\,(k) = \left[\bigcup_{i=1}^{k} new\_generate\_primes\,(i)\right] \bigcup base \qquad (5)$$

L'insieme dei numeri primi generati non è completo ma iterando come spiegato di seguito vengono calcolati tutti i numeri primi fino a $\boldsymbol{O}$. A ogni passo si tiene traccia dei nuovi primi generati e con la notazione `new_generated_primes(k)` si indicano i primi generati al passo k.
Al fine di evitare di controllare più volte gli stessi numeri viene tenuta traccia dei numeri generati detti `generated_numbers` che all'inizio del Round 2 corrisponde ai `srt_numbers_generated_by_base`. Con `generated_numbers(k)` si indicano

i numeri generati alla fine del Round k. Con `new_generated_numbers(k)` si indicano i numeri generati dal passo k.

$$generated\_numbers\,(k) =$$
$$= \left[\bigcup_{i=2}^{k} new\_generated\_numbers\,(i)\right] \bigcup number\_generated\_by\_base =$$
$$= \bigcup_{i=1}^{k} generated\_numbers\,(i) \quad (6)$$

1. L'insieme dei `generated_primes` viene copiato nell'insieme dei `primes_to_explore`=$\{s_1, . , s_q\}$.
   Viene eliminato dall'insieme dei `primes_to_explore` il primo elemento s_1e l'insieme dei primi da esplorare diventa `primes_to_explore`=$\{s_2, . , s_q\}$.
   Viene calcolato l'insieme dei `new_generated_numbers(k)` come da eq. 7:

   $$new\_generated\_numbers\,(k) = \{s_1 \cdot srt_1, \ldots, s_1 \cdot srt_g\} \quad (7)$$

   Per minimizzare lo spazio di memoria occupato per ogni prodotto $s_1 \cdot srt_p$ (con $p \leq g$) si verifica immediatamente se il prodotto appartiene all'elenco dei **generated_numbers**, se vi appartiene si procede con la moltiplicazione successiva, se non vi appartiene viene effettuato un test di primalità sul **possible_prime** calcolato come in eq. 8:

   $$possible\_prime = s_1 \cdot srt_p - 1 \quad (8)$$

   Osservazione 1. Il confronto tra `new_generated_numbers(k)` e `new_generated_numbers(k-1)` per il teorema di unicità della scomposizione in fattori primi dovrebbe sempre indicare che il numero confrontato non è presente nell'insieme.

   Il crivello viene effettuato con i numeri primi a disposizione (questo per evidenziare che l'algoritmo è completo) e:

   (a) se il numero `possible_prime`=$s_1 \cdot srt_p$ è primo:
      i. $s_1 \cdot srt_p$ viene aggiunto all'insieme dei `new_generated_numbers`
      ii. all'insieme dei `new_generated_primes` viene aggiunto il numero $s_1 \cdot srt_p - 1$.
      iii. se $s_1 \cdot srt_p \leq \dfrac{\boldsymbol{O}}{2}$, il numero $s_1 \cdot srt_p - 1$ viene aggiunto anche all'insieme dei `primes_to_explore`. Si noti che questo viene fatto in quanto anche tutit i primi afferenti all'insieme **primes_to_explore** sono moltiplicati per un fattore pari 2 o maggiore di 2.

   (b) se $s_1 \cdot srt_p$ non è primo viene aggiunto solamente all'insieme dei `new_generated_numbers` (ciò avverrà fino a quando non sarà verificata l'oss. 1 in quanto se dovesse essere nuovamente generato non vi si perderanno risorse per stabilire se è primo o no).

   Il calcolo dei numeri generati rispetto al primo s_1 si interrompe nei seguenti due casi:

(a) dopo il calcolo del prodotto $s_1 \cdot srt_n$ per ogni n (ovvero il numero in testa all'inseme dei **primes_to_explore** è moltiplicato con tutti i numeri della lista `srt_numbers_generated_by_base`)

(b) se $\exists t : s_1 \cdot srt_t > \boldsymbol{O}$. In questo caso viene processato $s_1 \cdot srt_t$ se è uguale a $\boldsymbol{O}$, altrimenti viene scartato. In altre parole, essendo la lista `srt_numbers_generated_by_base` ordinata, avremo

$$\forall i \quad 1 \leq i \leq t-1 \quad s_1 \cdot srt_i \leq \boldsymbol{O} \tag{9}$$

e

$$\forall i \quad p \leq i \leq g \quad s_1 \cdot srt_i > \boldsymbol{O} \tag{10}$$

quindi sfruttando l'ordinamento in ordine crescente degli srt_i il calcolo si interrompe non appena s_1 viene moltiplicato per srt_p.

2. Si ripete il punto 1 fino a quando l'insieme dei **primes_to_explore** non è vuoto.

Poiché il codice di testing utilizzato permette di conoscere a ogni round quali numeri primi non sono stati individuati (nelle prove sperimentali vengono sempre individuati tutti) possiamo determinare a quale Round sono stati individuati tutti. Generalmente tutti i numeri primi sono individuati quando la coda dei `primes_to_explore` contiene pochi elementi. Nelle figure 345, per intervalli diversi, sono messi in evidenza i Round necessari a esaurire la coda dei `primes_to_explore` (nella legenda *Tot Round*) e dopo quanti Round sono stati trovati tutti i numeri primi (nella legenda *Round4Prime*).

3.3 Analisi qualitativa dei risultati

Le analisi preliminari dei risultati sono mostrate nelle figure seguenti.
La figura 6 illustra come la densità dei numeri generati dall'algoritmo (nel caso sia generato un numero più volte è considerato una sola volta) tenda a avvicinarsi ala densità dei numeri primi. Ricordiamo che i numeri sono generati moltiplicando tra di loro i numeri primi e sottraendovi uno. Questa informazione puó essere messa in relazione al fatto che i **numbers_generated_by_base** tendono a crescere più lentamente del numero da crivellare (figura 8), tale fatto sembra coerente con quanto riscontrato in figura 6, ovvero che il rapporto tra i numeri calcolato con il prodotto e i numeri primi tende a diminuire.
La figura 7 mostra i tempi di calcolo per crivellare un numero. Essi sono solo indicativi in quanto per massimizzare la raccolta dei dati sono stati eseguiti in parallelo più istanze dell'algoritmo crivellando diversi numeri e inoltre i calcoli sono stati eseguiti su macchine molto diverse (processori i7 dalla generazione 4 alla generazione 10).
L'utilizzo della memoria di questo algoritmo risulta essere estremamente basso mentre le criticità sono da legarsi maggiormente alla quantità di calcoli da fare.
Strumenti di calcolo
Per il calcolo è stato utilizzato Matlab 12 e i codici sono stati eseguiti sulle macchine chiamate *i7*, *skejja* e *e2m* le cui caratteristiche sono di seguito riportate:

- i7

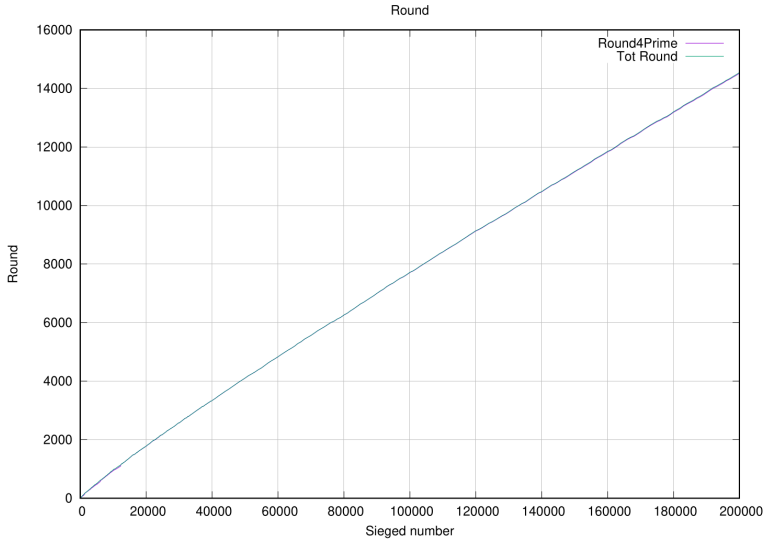

Figura 3: Vista complessiva dei Round necessari a completare l'analisi e a trovare tutti i numeri primi

  - i7 Kernel: 5.10.59-1-MANJARO x86_64 bits: 64 compiler: gcc v: 11.1.0
  - Desktop Mobo: ASUSTeK model: P9X79-WS-SYS v: Rev 1.xx
  - RAM: total: 31.31 GiB speed 1600 MT/s type DDR3
  - CPU: Quad Core model Intel Core i7-4820K

- skejja
  - skejja Kernel: 5.10.59-1-MANJARO x86_64 bits: 64 compiler: gcc v: 11.1.0
  - Desktop System: Apple product: MacPro5,1 v: 0.0
  - RAM: total: 31.34 GiB speed 1066 MT/s type DDR3
  - CPU: Quad Core model Intel Xeon W3565

- e2m
  - e2m Kernel: 5.10.59-1-MANJARO x86_64 bits: 64 compiler: gcc v: 11.1.0
  - Laptop System Dell product Precision 5550
  - RAM: total: 62.56 GiB speed 3200 MT/s type DDR4
  - CPU: 8-Core model Intel Core i7-10875H

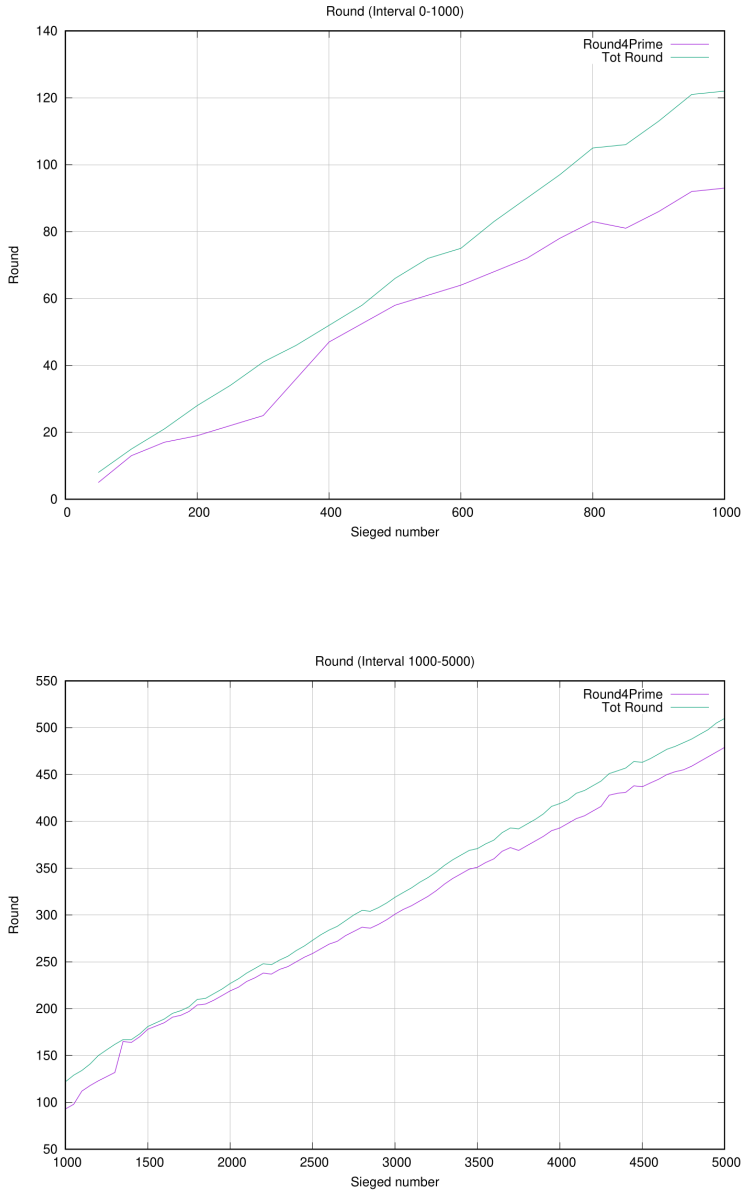

Figura 4: Viste particolari dei Round necessari a completare l'analisi e a trovare tutti i numeri primi

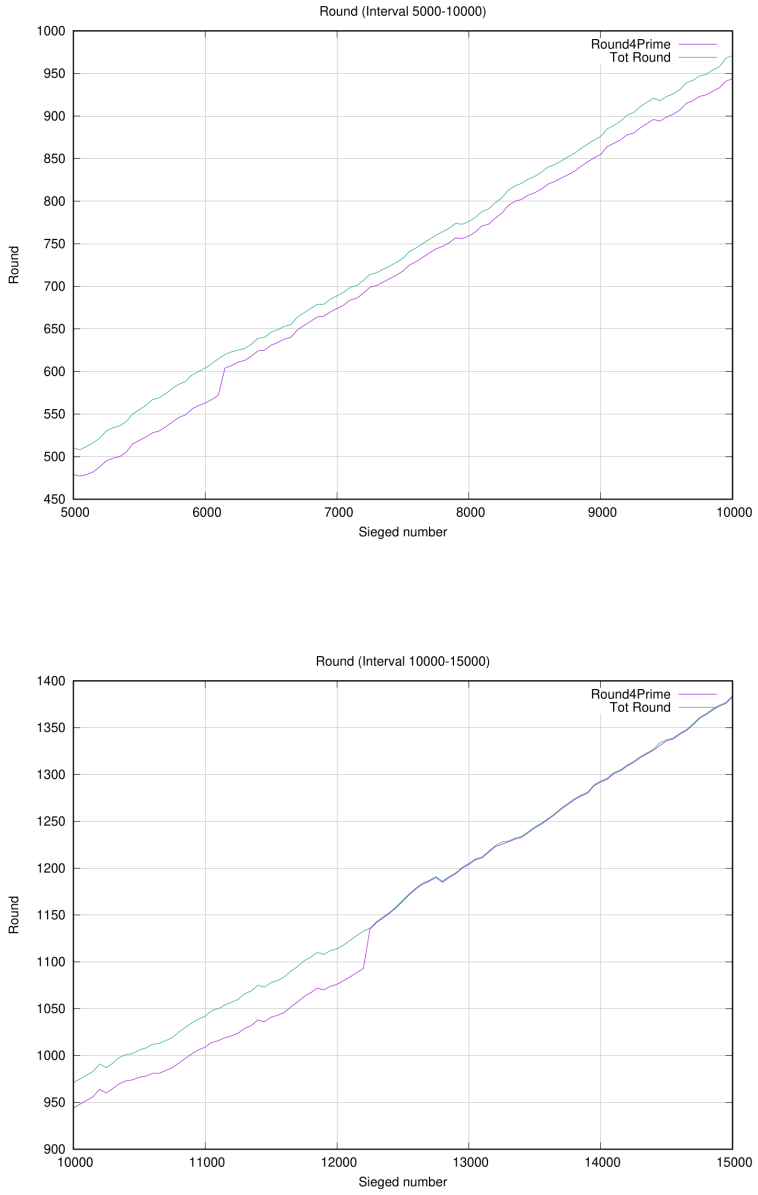

Figura 5: Vista particolare dei Round necessari a completare l'analisi e a trovare tutti i numeri primi

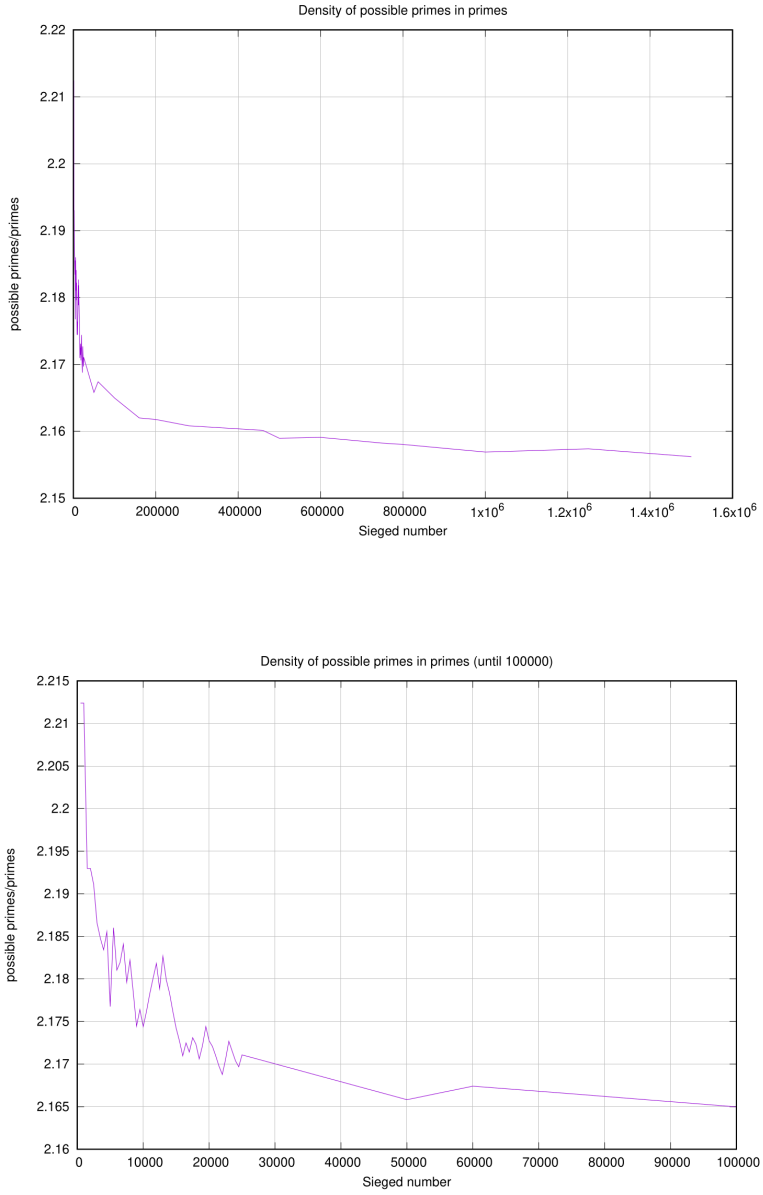

Figura 6: Densità dei `numeri_generated_primes` e i numeri primi

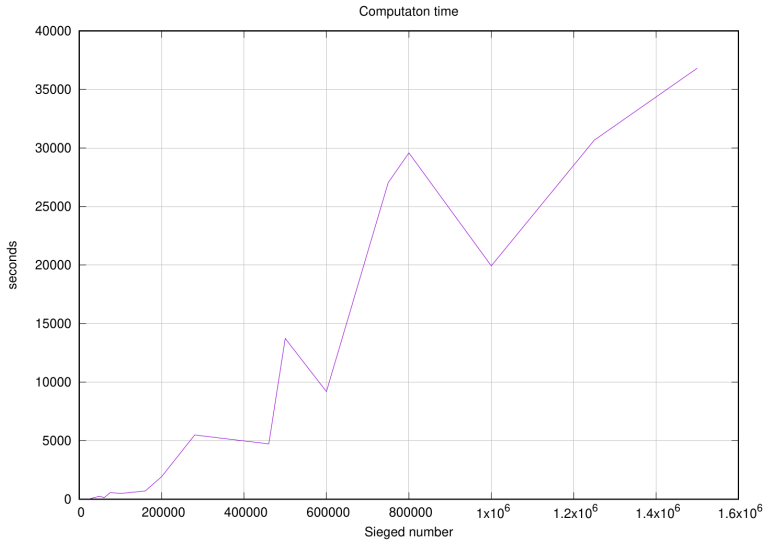

Figura 7: Tempo in secondi per crivellare fino a O

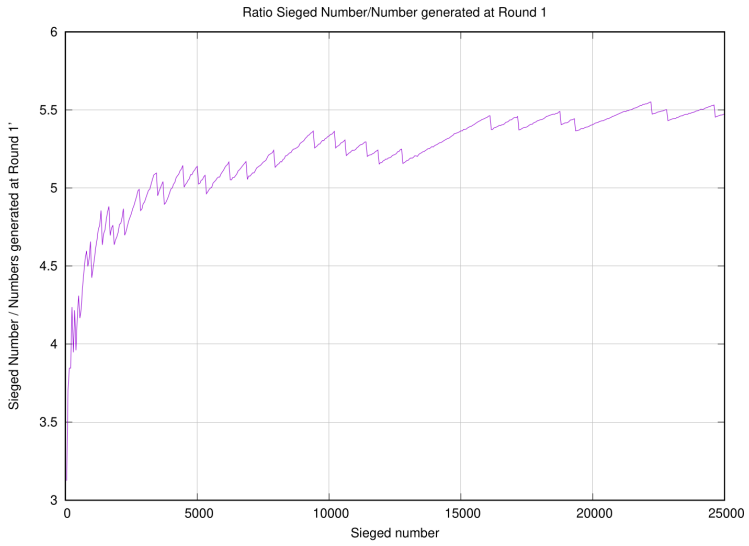

Figura 8: Crescita dei **numbers_generated_by_base** in funzione della crescita del numero O crivellato