# Teoria SIQS

Notazione:

N: numero da fattorizzare

FB: factor base insieme contenente tutti i primi della factor base senza molteplicità.

: un generico prodotto di alcuni primi della factor base con molteplicità.

Si sceglie un numero “a” come prodotto di “s” primi della factor base FB che si avvicina alla soglia di a ideale pari a

Da un valore singolo di a si ottengono 2^(s-1) “b” diversi che danno luogo a 2^(s-1) polinomi diversi, quindi non ha senso generare valori uguali di a, perché vengono generati gli stessi valori di “b”. Quando si calcola il valore di “a” utilizzare più candidati, ossia stilare una classifica di “a” candidati dal migliore al peggiore e scegliere il primo “a” migliore non utilizzato.

Si calcola il polinomio dove

.

Se l(j) viene moltiplicato per a si ottiene

che modulo N da:

Quindi se l(j) è un numero divisibile rispetto alla factor base si è ottenuta una relazione quadratica del tipo

. Ricorda che “a” contiene fattori della factor base quindi l(j)\*”a” continua a contenere fattori primi della factor base).

Combinando/Moltiplicando tra di loro queste relazioni è possibile ottenere relazioni del tipo

Che è equivalente a

E a quel punto è possibile calcolare

## Quando si trovano sicuramente dei fattori non banali di N?

Se valgono contemporaneamente

1)

2)

3)

Allora i gcd restituiscono uno il fattore p e l’altro il fattore q.

Questo succede perché la prima relazione ci dice che :

Combinando insieme le 3 affermazioni si ottiene che è necessario che un gcd ritorni p e l’altro ritorni q.

N.B.: il gcd(N,x) con N=p\*q può restituire 1,p,q oppure p\*q=N

## Relazioni semi B\_smooth

Se l(j) non è esprimibile come prodotto di fattori primi della factor base la parte che avanza dalla divisione è chiamata residuo P e se compreso tra B e B^2 la relazione semi B\_smooth trovata può essere memorizzata perché è utile per calcolare delle nuove relazioni B\_smooth.

La relazione che si ottiene è:

alcuni primi della factor base con molteplicità (ossia è esprimibile come prodotto di primi della factor base) e P

non è esprimibile come prodotto della factor base.

Se si ottiene un’altra relazione semi\_B\_smooth con lo stesso residuo è possibile ottenere una relazione B\_smooth:

Moltiplicate tra loro danno:

Che portando i termini al lato corretto danno:

, infatti equivale ad:

## Relazioni utili/indipendenti e inutili/ridondanti

Accumulare più relazioni utili possibili.

E’ possibile fattorizzare avendo a disposizione un piccolo sottoinsieme delle relazioni inutili? Se si, limitare le relazioni inutili nella maniera più veloce possibile, eliminare completamente tutte le relazioni inutili potrebbe essere computazionalmente complesso. Avere relazioni inutili si ripercuote sulla risoluzione del sistema lineare che a quel punto troverà come dimensione della base un numero più basso rispetto al caso in cui le relazioni sono completamente indipendenti.

Assunzione: N non divisibile per ogni numero primo della factor base, se è divisibile lo dividiamo e finisce l’algoritmo, (per fare questo verificare durante la creazione della factor base che N non è divisibile per ogni primo p della factor base). Questa assunzione ci serve per dire che:

Gli elementi di una relazione quadratica sono 3:

# Relazioni quadratiche senza residuo

Nel caso in cui le relazioni quadratiche sono senza residuo (oppure semplicemente considerando P=1), 2 relazioni possono essere in relazione in 4 modi diversi:

Le 2 relazioni hanno:

1. Stessa , stesso relazioni utili in alcuni casi, vedi paragrafo successivo
2. Stessa , diverso , matematicamente impossibile
3. Diversa , stesso , matematicamente impossibile
4. Diversa , diverso le 2 relazioni sono utili in ogni caso

# Relazioni quadratiche senza residuo utili del punto 1

Caso Stessa , stesso .

Stesso

1. in questo caso le relazioni sono ridondanti perché sono completamente identiche
2. in questo caso le relazioni sono ridondanti, vedi calcoli successivi

Calcoli dei punti 1. e 2. : Le 2 relazioni:

Sono entrambe valide ma sono ridondanti infatti eseguendo i calcoli e moltiplicando tra di loro le relazioni si ha che:

Invece le relazioni:

Formano una coppia di relazioni valide, quindi stessa factor base ma diversi quadrati possono essere utili.

Una relazione di questo tipo può esistere in quanto il valore

, m

Probabilisticamente è poco probabile ottenere una coppia di relazioni del tipo

Ma sapere che sono inutile suggerisce il fatto che non ha senso generare facilmente a “mano” questo tipo di relazioni, in quanto sono inutili.

# Relazioni quadratiche con residuo

Nel caso in cui è presente anche il residuo, invece, abbiamo 8 possibilità:

1. Stessa , stesso , stesso residuo P, relazioni utili in alcuni casi vedi paragrafi successivi, e possono essere accoppiate per ottenere nuove relazioni quadratiche
2. Stessa , stesso , diverso residuo P, matematicamente impossibile
3. Stessa , diverso , stesso residuo P, matematicamente impossibile
4. Stessa , diverso , diverso residuo P, sempre utile, ma queste 2 relazioni non possono essere accoppiate per ottenere una nuova relazione quadratica
5. Diversa , stesso , stesso residuo P, matematicamente impossibile
6. Diversa , stesso , diverso residuo P, matematicamente impossibile
7. Diversa , diverso , stesso residuo P, sempre utile e possono essere accoppiate per ottenere una nuova relazione quadratica.
8. Diversa , diverso , diverso residuo P, sempre utile, ma non possono essere accoppiate per formare un’altra relazione quadratica.

# Relazioni quadratiche con residuo utili del punto 1

Caso Stessa , stesso , stesso residuo P.

Stesso

1. in questo caso le relazioni sono ridondanti perché sono completamente identiche
2. in questo caso le relazioni sono ridondanti, vedi calcoli successivi

Calcoli dei punti 1. e 2. : Le 2 relazioni:

Sono entrambe valide ma sono ridondanti infatti eseguendo i calcoli e moltiplicando tra di loro le relazioni si ha che:

Invece le relazioni:

Formano una coppia di relazioni valide, quindi stessa factor base ma diversi quadrati possono essere utili.

Una relazione di questo tipo può esistere in quanto il valore

, m

Probabilisticamente è poco probabile ottenere una coppia di relazioni del tipo

Ma sapere che sono inutile suggerisce il fatto che non ha senso generare facilmente a “mano” questo tipo di relazioni, in quanto sono inutili.

# Come raggruppare 3 o più relazioni con lo stesso residuo?

Se abbiamo 3 o più relazioni con lo stesso residuo come possiamo raggrupparle per ottenere il massimo numero di relazioni quadratiche utili/indipendenti?

Consideriamo le 3 relazioni semi B smooth:

I modi in cui è possibile raggrupparle per ottenere nuove relazioni quadratiche (potenzialmente non inutili) sono:

1. Relazione 1+relazione 2
2. Relazione 1+relazione 3
3. Relazione 1+ relazione 3
4. Relazione 2 + relazione 3

Di queste 4 possibilità doppiamo capire quali sono utili e quali sono ridondanti non indipendenti.

Intanto con la 1) E la 2) Posso generare 2 relazioni quadratiche indipendenti.

La nuova relazione che si formerebbe con le relazioni semi B smooth 1. e 3. una volta formate le 2 nuove relazioni B smooth (1+2 e 2+3) rappresenta una relazione dipendente e quindi inutile [infatti moltiplicando relazione B smooth data da 1+2 con relazione B smooth data da 2+3 e sfruttando la relazione semi Bsmooth 2. Si ottiene la relazione B smooth 1+3]. Quindi il caso 3) non aggiunge informazione rispetto alle relazioni B smooth già ottenute, infatti si forma un ciclo di relazioni (1,2), (1,3) ,(2,3). Lo stesso discorso vale per il caso 4). Quindi è possibile date queste 3 relazioni semi B smooth ottenere solamente 2 nuove relazioni B smooth.

Naturalmente esistono vari modi di raggruppare queste 3 relazioni per ottenere le 2 nuove relazioni, ad esempio possiamo ottenerle con:

1+3,2+3 oppure 1+2,2+3 oppure 1+2,1+3, sta nella nostra comodità sceglierle, forse generarle con 1,2 e 1,3 è più semplice viene scelta sempre la 1 e si scansiona sulle altre 2.

E se avessi 4 o più relazioni?

Consideriamo di avere queste 4 relazioni:

Sicuramente possiamo accoppiare (1+2),(1+3),(1+4).

Non possiamo dopo aver generato queste ottenere nuove relazioni con 2+3,2+4,3+4, infatti è come avere 3 relazioni del caso precedente e formare un ciclo di relazioni.

Con una tripla di relazioni non otteniamo nessuna relazione.

Con tutte e 4 le relazioni messe insieme si forma una relazione B smooth inutile infatti moltiplicando tra di loro

le relazioni 1+2,1+3 e 1+4 e sfruttando la relazione semi B smooth 1. si ottiene la relazione 1+2+3+4

Conclusione:

E’ possibile e conveniente accoppiare le relazioni semi B smooth in modo tale che venga utilizzata sempre la prima con ognuna delle altre relazioni semi B smooth. Con m relazioni semi B smooth con lo stesso residuo si riescono ad ottenere m-1 relazioni B smooth valide. Un approccio algoritmico potrebbe prevedere il calcolo di queste relazioni in modo incrementale e parallelo da parte dei threads (man mano che i threads eseguono calcolano le nuove relazioni). Uno dei thread pubblica la prima relazione semi B smooth con un certo residuo P e le successive relazioni non vengono aggiunte ma vengono direttamente calcolate a partire da questa prima relazione. Per aggiungere la prima relazione può essere fatto in modo esclusivo tramite istruzione atomica CAS, se fallisce CAS allora un altro ha già pubblicato la relazione con lo stesso residuo o insiste su una struttura dati condivisa necessaria ad entrambi per pubblicare la propria relazione.

Le relazioni potrebbero essere pubblicate in una struttura ad hash table con liste di collisione, la chiave di ogni elemento di una lista di collisione è il residuo P.

# Fase di sieving: somma dei log P in modo veloce sfruttando cache locality

Dato un array di sieving dei numeri da fattorizzare, Memorizzare un altro array che contiene nella cella all’indice x la somma dei log di P del numero presente all’indice x nell’array di sieving. Questi 2 array hanno 2M+1 elementi ossia vanno dall’indice 0 all’indice 2M.

Per ogni primo p della factor base si calcolano 2 indici iniziali tali per cui il numero dell’array di sieving è divisibile per p e da quelli tutti gli indici ottenuti sommando ogni volta p determinano i nuovi numeri divisibili per p nell’array di sieving.

Considerando card(FB) arrays ognuno con il relativo numero p a salti di p in base alla divisibilità dei numeri dell’array sieving quello che si vuole fare è sommare tra di loro questi card(FB) arrays per ottenere un array risultato R che contiene quindi la somma di tutti i log di P.

Per esempio supponendo di avere M=4 e 2M+1=9 con FB={2,3,5} la somma dei log di P è pari a:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| indice | risultato | P=2 | P=3 | P=5 |
| 0 | 3 | 0 | 3 | 0 |
| 1 | 2 | 2 | 0 | 0 |
| 2 | 5 | 0 | 0 | 5 |
| 3 | 5 | 2 | 3 | 0 |
| 4 | 0 | 0 | 0 | 0 |
| 5 | 2 | 2 | 0 | 0 |
| 6 | 3 | 0 | 3 | 0 |
| 7 | 7 | 2 | 0 | 5 |
| 8 | 0 | 0 | 0 | 0 |

Questa rappresentazione mostra che l’array di sieving è memorizzato in maniera contigua in memoria per verticale.

Ora per ottenere la somma dei log di P abbiamo 2 possbilità:

1. Somma dei log di P per colonna: si inizia con l’array risultato con tutti 0, per ogni primo della factor base si sommano tutti gli elementi per verticale fino a quando non si arriva alla fine dell’array. Una volta fatto ciò si passa al primo della factor base successivo.

In pratica vengono scansionati nel loop esterno i primi della factor base e nel loop interno gli indici dell’array

1. Somma dei log di P per riga: si inizia con l’array risultato con tutti 0. Si analizzano un blocco di indici per volta. Per ogni primo della factor base si vede qual è l’indice che appartiene al blocco degli indici corrente e che quindi deve essere sommato all’array risultato. Una volta fatto ciò si passa al primo della factor base successivo.

In pratica vengono scansionati nel loop esterno i blocchi degli indici e nel loop interno i primi della factor base. La dimensione del blocco degli indici può essere anche 1 ma conviene utilizzare come blocco la dimensione della cache, ossia il numero di indici che rientrano in un blocco di cache.

Quale approccio è migliore?

Assolutamente il primo, infatti nonostante non minimizzi il numero di caricamenti delle linee di cache rispetto al secondo approccio, esegue molti meno calcoli. Infatti supponendo che ci siano solamente 2 primi grandi P1 e P2, il primo approccio esegue salti di P1 e P2 eseguendo cosi pochissime somme nell’array di dimensione 2M+1, invece il secondo approccio siccome guarda blocchi di indici per volta si ritrova a scansionare blocchi di indici a cui non deve quasi mai essere sommato il logaritmo di P1 e logaritmo di P2.