Proceedings of International Conference on Scientific Research in Natural and Social Sciences
Hosted online from Toronto, Canada.
Date: 5th December, 2024
ISSN: 2835-5326 **Website:** econferenceseries.com

KATTA SONLARNI FAKTORIZATSIYA QILISH MUAMMOSINI SAMARALI HAL QILISHDA SHOR ALGORITIMDA FOYDALANISHNING BOSHQA ALGORITMLARDA SAMARADORLIGI

1 Toirov Sh. A.
2 Saidakhmedov E. I.
3 Qo'ldoshov Sh. A
techmespeaker@gmail.com
1"Toshkent menejment va iqtisodiyot instetuti"
muxandis va axborot texnologiyalari kafedrasining dotsent
2"Denov tadbirkorlik va pedogogika instituti"
axborot texnologiyalari kafedrasi o'qituvchisi
2"Denov tadbirkorlik va pedogogika instituti"
Kompyuter injiniring yo'nalishi 3-kurs

Annotatsiya:
Ushbu maqoladi katta sonlarni faktoralizatsiya qilish uchun ishlatiladigan Shor algoritimini boshqa RSA, Brute Force, Trial Division, Fermatning foktaralizatsiya usuli, Pollard Rho algoritmidan farqlari pythonning numpy va pyplot va kutubxonalarni qollsh ososida olingan natijalar orqali tahlil qilinadi.

Kalit so'zlar: Faktorizatsiya**,** Shor algoritmi, RSA algoritmi, Brute Force algoritmi, Trial Division algoritmi, Fermatning faktorizatsiya usulli, Pollard's Rho algoritm.

Faktorizatsiya — bu biror butun sonni uning Tub bo'luvchilariga ajratish jarayonidir. Tub bo'luvchilar — bu faqat 1 va o'ziga bo'linadigan sonlardir, boshqacha qilib aytganda, faktorizatsiya biror sonni faqat tub sonlar ko'paytmasi sifatida ifodalashni anglatadi.

Misol uchun, agar siz 12 sonini faktorizatsiya qilmoqchi bo'lsangiz, uni asosiy bo'luvchilarga ajratishingiz kerak:

$$12 = 2 \times 2 \times 3$$

Bu yerda 2 va 3 — asosiy sonlardir.

Shunday qilib, faktorizatsiya orqali 12 ni 2 va 3 kabi tub sonlarning ko'paytmasi sifatida ifodalanadi.

Faktorizatsiya jarayonida ma'lum bir sonni topishga harakat qilamiz:

1. Sonni 2 ga bo‘linadi.
2. Agar bo‘linmasa, keyingi kichik asosiy son — 3, 5, 7, 11 va boshqalar bilan bo‘linadi.
3. Bu jarayonni har safar, bo‘lish mumkin bo‘lgan tub sonni tanlab, davom ettiriladi.

Faktorizatsiya matematika va kriptografiyada muhim rol o‘ynaydi, chunki bu jarayon sonlarning tuzilishini o‘rganish va xavfsiz ma’lumot uzatish tizimlarini yaratishda asosiy rol o‘ynaydi.

Kriptografiya turli algoritmlardan foydalanib ma’lumotlarni xavfsiz tarzda uzatish va saqlash mumkin va bu jarayondan faktorizatsiyadan foydalanadi.

RSA algoritmi (Rivest–Shamir–Adleman) — bu asosan katta sonlarning faktorizatsiyasiga asoslangan shifrlash algoritmi bo’lib sonlarni asosiy bo‘luvchilarga ajratishning qiyinligiga tayanadi.

RSA tizimi ikkita katta asosiy sonni tanlab, ularning ko‘paytmasi sifatida umumiy kalitni hosil qiladi. Agar siz bu kalitni faktorizatsiya qilsangiz, shifrlangan ma’lumotni ochish mumkin bo‘ladi. Shuning uchun, faktorizatsiya qilish juda qiyin bo‘lgani sababli, RSA tizimi xavfsiz bo‘lib, ma‘lumotlarni himoya qiladi va u quydagicha amalga oshiriladi.

1. Kalit juftligini yaratish: RSA algoritmida ikkita kalit mavjud:
2. Umumiy kaliti (**public key**): Bu kalit ma’lumotlarni shifrlashda ishlatiladi va keng tarqatiladi.
3. Maxfiy kalit (**private key**): Bu kalit shifrlangan ma’lumotlarni deshifrlashda ishlatiladi va faqat kalit egasi tomonidan saqlanadi.

Kalitlarni yaratish uchun quyidagi bosqichlar bajariladi:

Ikkita asosiy sonni tanlash: p va q kabi ikkita tasodifiy asosiy sonni tanlang. Bu sonlar bir-biriga mustaqil va katta bo‘lishi kerak.

N ni quyidagicha hisoblang: $N = p \times q$

N bu RSA algoritmining tub soni bo‘lib, u umumiy_kaliti va maxfiy_kalitni yaratishda ishlatiladi. N soni juda katta bo‘lishi kerak, bu uning xavfsizligini ta’minlaydi.

Eulerning totient funktsiyasini (phi) quyidagi tarzda hisoblang:

$$\phi(N) = (p - 1) \times (q - 1)$$

Bu yerda p va q tanlangan tub sonlar.

Umumiy_kaliti uchun e ni tanlash: e soni $1 < e < \phi(N)$ oraliqda bo‘lib, e va $\phi(N)$ ning eng katta umumiy bo‘luvchisi (GCD) 1 bo‘lishi kerak. Bu e ni tanlash uchun shart bo‘ladi. e odatda kichik, masalan, 65537 soni tanlanadi, chunki u xavfsiz va tez hisoblanadi.

Maxfiy kalit uchun d ni hisoblash:

$$d \times e \equiv 1 \pmod{\phi(N)}$$

Bu yerda d ni modul teskari (modular inverse) sifatida hisoblash kerak.

Ma’lumotlarni shifrlash uchun umumiy kaliti (public key) ishlatiladi. Agar M — bu shifrlanadigan xabar bo‘lsa, uni quyidagicha shifrlash mumkin:

$$C = M^e \pmod{N}$$

Bu yerda:

C — shifrlangan xabar (kryptogramma),

e — jamoat kalitining birinchi qismini tashkil etuvchi son,

N — jamoat kalitining ikkinchi qismini tashkil etuvchi son,

M — asl ma’lumot.

Shifrlangan xabarni deshifrlash uchun maxfiy kalit (private key) ishlatiladi.

Deshifrlash jarayoni quyidagicha amalga oshiriladi:

$$M = C^d \pmod{N}$$

Bu yerda:

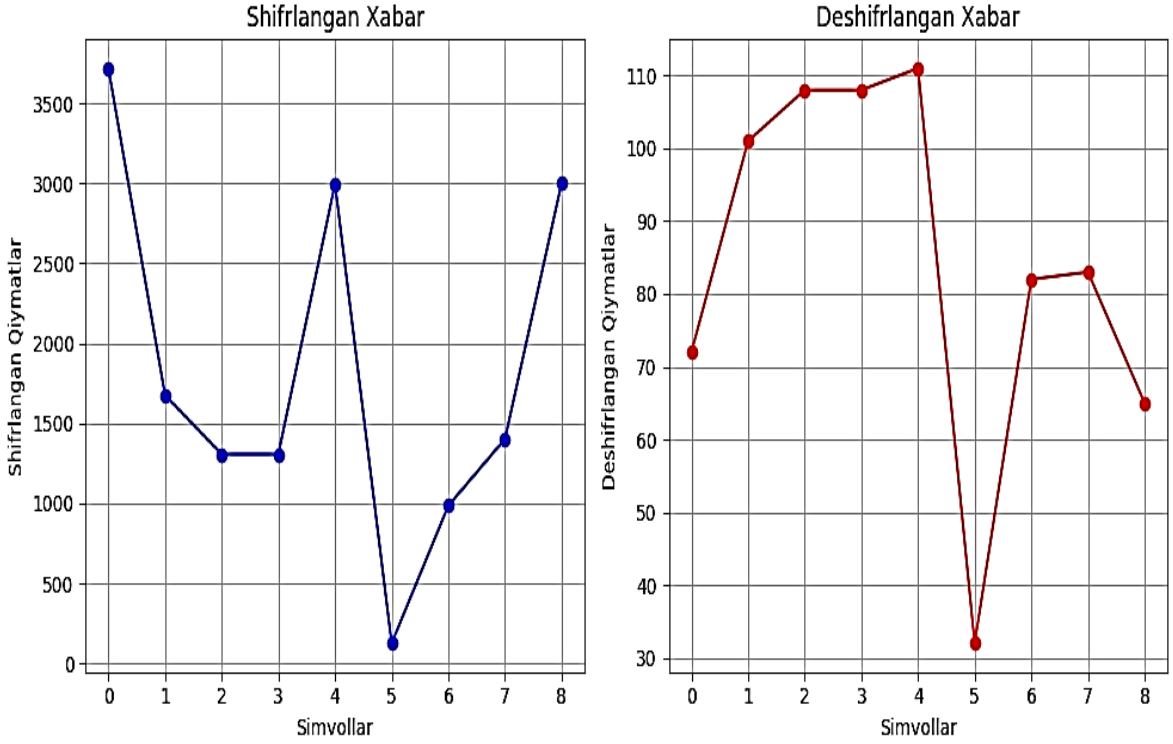
C — shifrlangan xabar

d — maxfiy kalitning birinchi qismini tashkil etuvchi son

N — jamoat kalitining ikkinchi qismini tashkil etuvchi son.

M — asl ma’lumot.

RSA algoritmining python Jupyter notebookda Pythoning PyPlot kutubxonasidan foydalangan holda olingan natija.



Brute Force algoritmi matematik masalalarni hal qilishda "xom kuch" yondashuvidan foydalanadi. Faktorizatsiya uchun bu yondashuv berilgan sonni barcha mumkin bo‘lgan bo‘luvchilar orqali bo‘lib, asosiy omillarni topishga qaratiladi.

Boshlanishi: Faktorizatsiya qilish kerak bo‘lgan sonni, N-ni aniqlash.

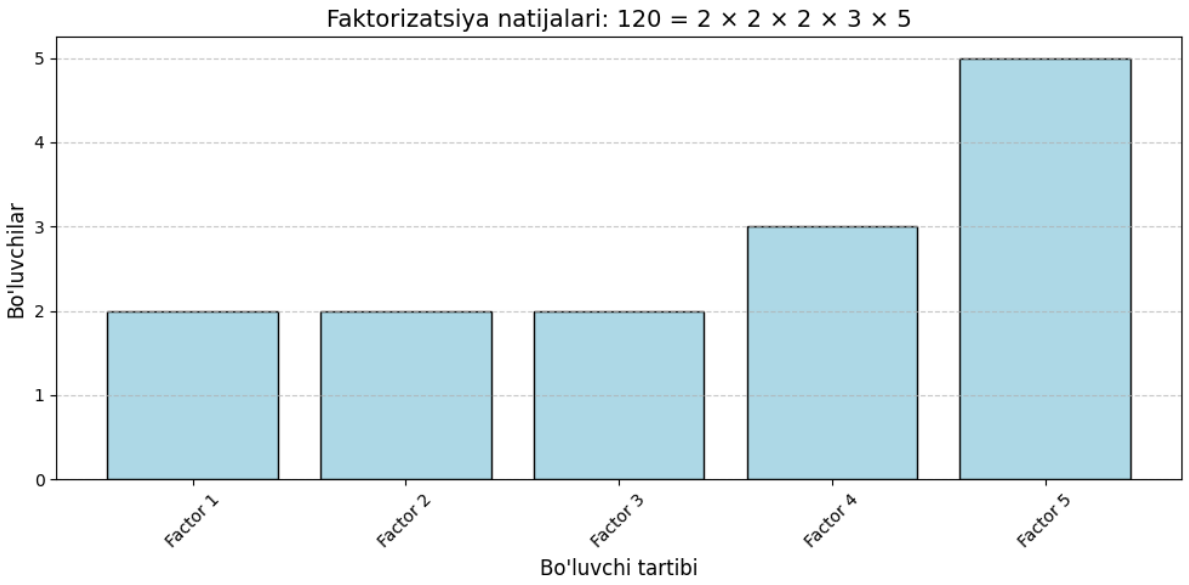
1. Bo‘luvchilarni tekshirish:
- N-ni ketma-ket sonlar bilan bo‘ling, odatda 2 dan boshlanadi.
- Har safar i-ga bo‘linganda qoldiq nol bo‘lsa (N%i==0), demak, i - bu N-ning omili.

2. Sonni kichraytirish:
- Agar bo‘linish muvaffaqiyatli bo‘lsa, N-ni i-ga bo‘linadi. (N=N//i).
- Agar bo‘linish muvaffaqiyatsiz bo‘lsa, i-ni oshiradi. (i+=1).

3. Jarayonni takrorlash:
- Yuqoridagi jarayon N=1 bo‘lgunga qadar davom etadi. Bu vaqtda barcha omillar topilgan bo‘ladi.

4. Natija:
- Topilgan barcha bo‘luvchilar ro‘yxati N-ning asosiy omillari hisoblanadi.[1]

Brute Force algoritmining python Jupyter notebookda Pythoning PyPlot kutubxonasidan foydalangan holda olingan natija.



Trial Division algoritmi oddiy va tushunarli faktorizatsiya usulidir. Uning ishlash prinsipi sonni ketma-ket kichik asosiy (prime) sonlarga bo‘lib chiqish orqali faktorizatsiya qilishga [2] asoslangan.

Quyida Trial Division algoritmini ishlashni tushuntirib beraman:

1. Berilgan son N dan kichik bo‘lgan barcha asosiy sonlarni (prime numbers) topish.
2. Ushbu asosiy sonlar yordamida N ni ketma-ket bo‘lish orqali faktorizatsiya qilish.
3. Har bir bo‘luvchi topilganda, N ni topilgan bo‘luvchiga bo‘lib, jarayonni davom ettirish.

Algoritm qadami.

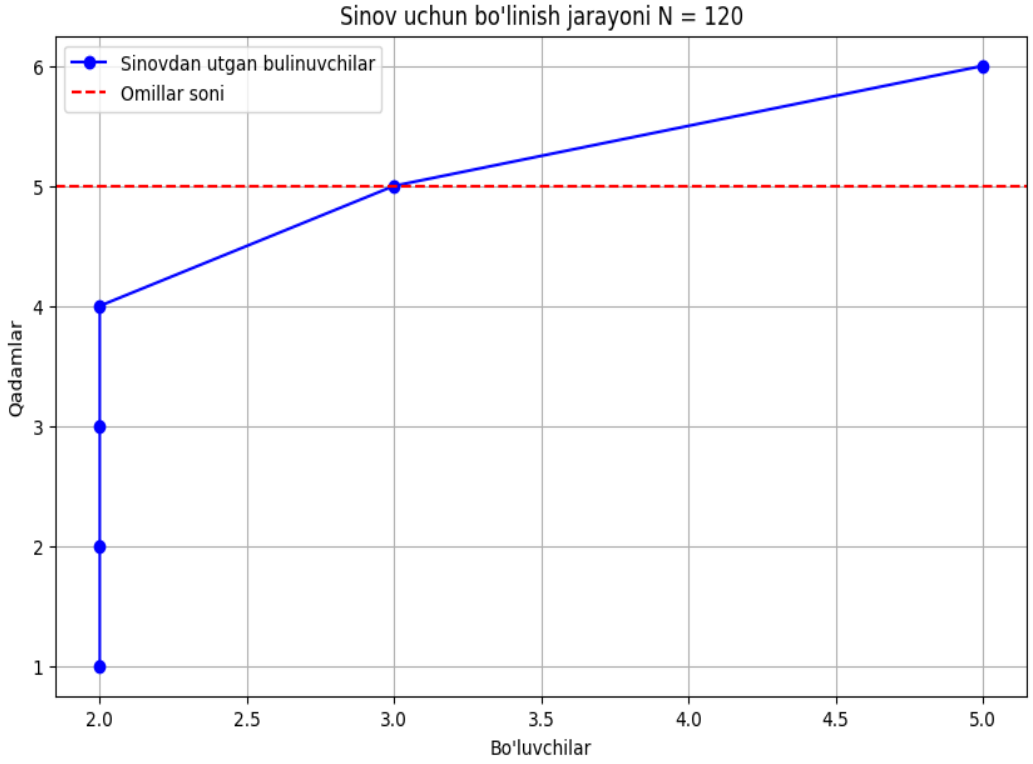
1. Faktorizatsiya qilish kerak bo‘lgan son N ni kiritiladi.
2. Bo‘luvchi d-ni 2-dan boshlanadi.
3. Bo‘luvni tekshirish:
   - Agar N mod d = 0 bo‘lsa, d asosiy bo‘luvchi hisoblanadi.
   - N-ni d-ga bo‘lib (ya’ni N=N/d), jarayonni davom ettiring.
4. Bo‘luvchini oshirish:
   - Agar N mod d≠0 bo‘lsa, d-ni keyingi sondan davom ettiriladi.

5. To‘xtash: Jarayon $d > \sqrt{N}$ bo‘lganda tugaydi yoki N-ning o‘zi asosiy son bo‘lib qoladi.

Misol: N=120

1. Boshlash: N=120.d =2
2. Bo'luvni tekshirish: 120 mod 2= 0 demak 2 asosiy bo'luvchi.
   - 120/2=60. Qolgan N=60
3. Jarayonni davom ettirish:
   - 60mod 2=0, 60/2=30, N=30
   - 30mod 2=0, 30/2=15 , N=15
4. Bo'luvchini oshirish: Endi d=3
   - 15mod 3=0 15/3=5, N=5
5. Yakun: N=5, bu asosiy son, shuning uchun natija: 2,2,2,3,5

Ushbu misolni pythonning pyplot kutubxonasidan foydalangan holdagi dasturiy natijasi.



Fermatning faktorizatsiya usulli sonni ikki kvadratlar farqi sifatida tasvirlashga asoslangan:

$$N = a^2 - b^2$$

Bu ifoda matematik tarzda qayta yozilishi mumkin:

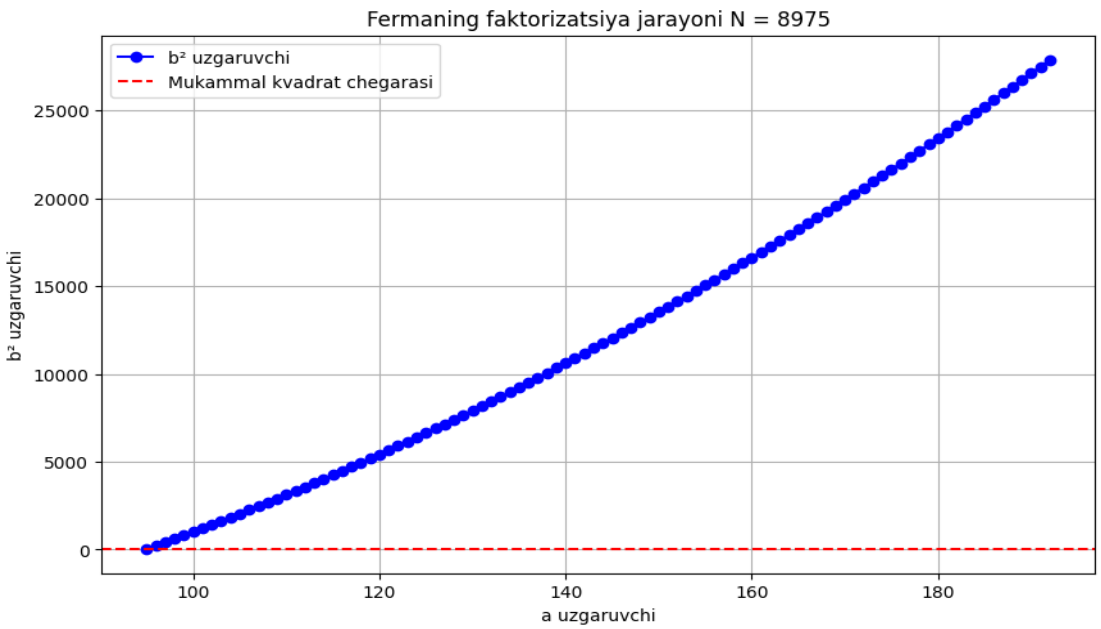
$$N = (a - b)(a + b)$$

Bu yerda N - faktorizatsiya qilinadigan son, a va b- izlanayotgan butun sonlar [3,4].

Algoritmning ishlash qadam-baqadam tavsifi

1. Boshlash: Berilgan N son uchun a ni boshlang'ich qiymat sifatida $\lceil\sqrt{N}\rceil$ qilib olinadi.
   - Bu N-dan katta yoki teng eng kichik butun kvadrat ildizdir.
2. Kvadrat farqni tekshirish:
   - Hisoblang: $b^2 = a^2 - N$.
   - Agar b^2 butun sonning kvadrati bo‘lsa, faktorizatsiya topilgan.
3. To'g'rilash:
   - Agar b^2 kvadrat bo‘lmasa, a-ni 1 ga oshiring (a=a+1) va qayta hisoblang.
4. Faktorizatsiyani topish:
   - Faktorizatsiya topilgach: $p = a - b, q = a + b$ Bu yerda p va q N-ning bo'luvchilari.
5. To'xtash: Algoritm b^2 kvadratga aylanguncha davom etadi.

Fermatning faktorizatsiya usulli ning python Jupyter notebookda Pythoning PyPlot kutubxonasidan foydalangan holda olingan natija.



Pollard's Rho algoritm - bu faktorizatsiya uchun probabilistik algoritm bo'lib, asosan o'rta kattalikdagi sonlarni tezkor faktorizatsiya qilish uchun ishlatiladi. U matematik funksiya yordamida sikl (cycle) topish va GCD (Greatest Common Divisor) orqali bo'luvchi aniqlashga asoslangan.Algoritmning nomi "Rho" (yunoncha ρ) ga o'xshash siklni tasvirlovchi shakl bilan [5,6] bog'liq.

Ishlash prinsipi quydagicha:

1. **Funksiya tanlash**: Oddiy kvadrat funksiya ishlatiladi, masalan:

$$f(x) = (x^2 + c) \mathrm{mod} N$$

Bu yerda:
N- faktorizatsiya qilinadigan son.
c- biror o'zgarmas butun son.

2. iteratsion nuqta yaratish:

x va y boshlang'ich qiymati sifatida tanlanadi (odatda x=2 va y=2).
x bir marta iteratsiya qilinadi: $x \leftarrow f(x)$
y ikki marta iteratsiya qilinadi: $y \leftarrow f(f(y))$

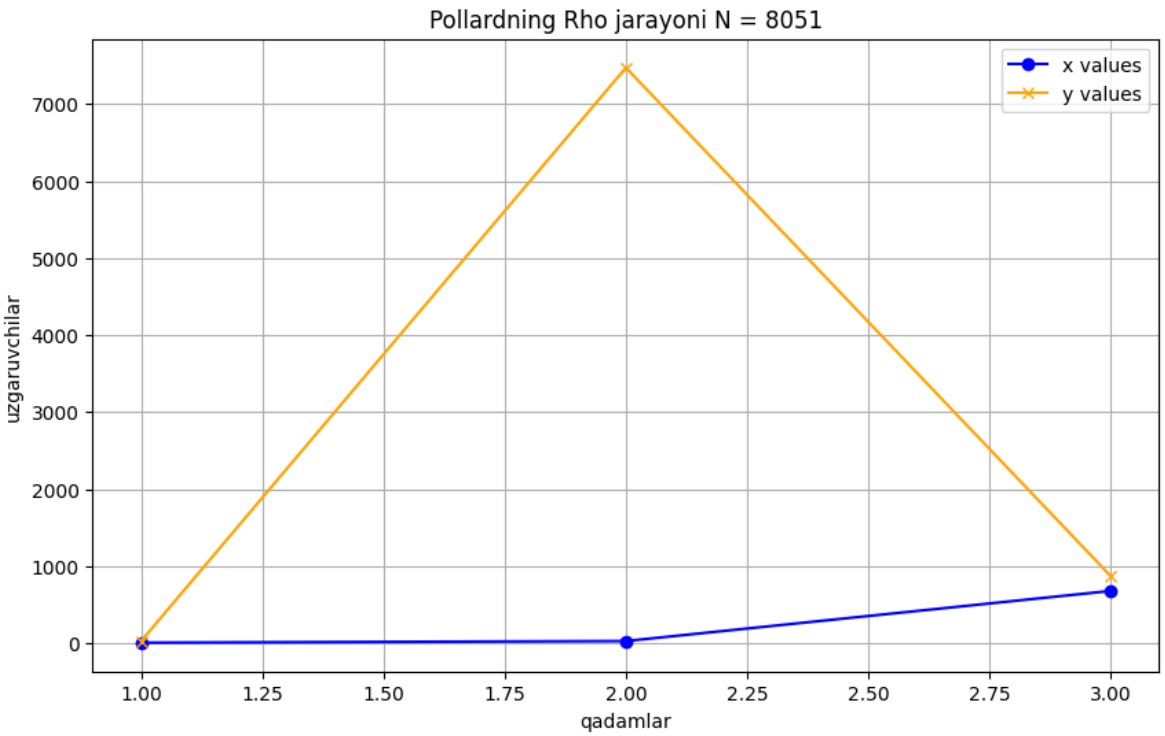
3.GCD hisoblash: Har bir iteratsiyada:

$$d = \mathrm{GCD}(|x - y|, N)$$

Agar $d > 1$ bo'lsa, d , N-ning bo'luvchisi hisoblanadi.

4. To'xtash:

Agar d=N, sikldan chiqib boshqa funksiya yoki boshlang'ich qiymatlarni tanlash kerak.
Agar d asosiy bo'luvchi bo'lsa, u qaytariladi.
Fermatning faktorizatsiya usulli ning python Jupyter notebookda Pythoning PyPlot kutubxonasidan foydalangan holda olingan natija.



Xulosa. Ushbu materialda katta sonlarni faktorizatsiya qilish muammosini yechishda turli algoritmlarning samaradorligi tahlil qilingan. Shor algoritmi, ayniqsa, kvant hisoblash imkoniyatlaridan foydalangan holda, boshqa klassik algoritmlarga nisbatan ustunligi bilan ajralib turadi. Bunda RSA, Brute Force, Trial Division, Fermatning faktorizatsiya usuli va Pollard's Rho kabi klassik algoritmlar bilan taqqoslash keltirilgan.

Asosiy xulosalar:

1. **Shor algoritmi** - kvant kompyuterlarida katta sonlarni faktorizatsiya qilishda eng samarali hisoblanadi. Bu RSA shifrlash tizimining xavfsizligiga jiddiy xavf solishi mumkin.
2. **Klassik algoritmlar**:
   - **Brute Force** - eng oddiy, ammo sekin usul bo'lib, katta sonlar uchun amaliy emas.
   - **Trial Division** - oddiy va intuitiv usul, lekin katta sonlar uchun samaradorligi past.
   - **Fermatning usuli** - sonni ikki kvadrat farqi sifatida ifodalash asosida ishlaydi, lekin samaradorligi sonning strukturasiga bog'liq.

- **Pollard's Rho algoritmi** - o‘rta kattalikdagi sonlarni faktorizatsiya qilish uchun samarali probabilistik usul hisoblanadi.

Bu algoritmlar Python dasturlash tilida amalga oshirilib, **NumPy** va **PyPlot** kutubxonalari yordamida olingan natijalar orqali samaradorlik tahlil qilingan. Shor algoritmi klassik usullardan tezligi va samaradorligi bilan ajralib turishi yana bir bor tasdiqlangan.

Materialga asoslangan holda, kvant hisoblashning rivoji faktorizatsiya asosida qurilgan shifrlash tizimlariga jiddiy o‘zgarishlar kiritadi degan xulosaga kelish mumkin.

Foydalanga adabiyotlar

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms. MIT Press, 2009.
2. Crandall, R., & Pomerance, C. Prime Numbers: A Computational Perspective. Springer, 2005.
3. Riesel, H. Prime Numbers and Computer Methods for Factorization. Birkhäuser Basel, 2012.
4. Bressoud, D. M. Factorization and Primality Testing. Springer, 1989.
5. Pollard, J. M. "A Monte Carlo Method for Factorization." BIT Numerical Mathematics, 15(3), 1975, pp. 331-334.
6. Knuth, D. E. The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley, 1997.
7. Shor, P. W. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring." Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124-134.
8. Nielsen, M. A., & Chuang, I. L. Quantum Computation and Quantum Information. Cambridge University Press, 2010.