TARYANA ALGORITMINING ISHLASH PRINSPI VA UNI QIDIRISH TIZIMLARIDA DOLZARBLIGI

Farmonov Sherzodbek Raxmonjonovich
Farg'ona davlat universiteti amaliy matematika va informatika kafedrasi katta o'qituvchisi
e-mail: farmonovsh@gmail.com
Solijonova Zuxraxon Shermatjon qizi
Farg'ona davlat universiteti talabasi
e-mail: solijonovaz203@gmail.com
https://doi.org/10.5281/zenodo.14514696

Anotatsiya.

Mazkur maqolada Taryana algoritmining matn ichida andoza qidirishdagi qo'llanilishi va ishlash tamoyillari tahlil qilinadi. Algoritm matn va andozani qiyoslashda prefiks-suffiks jadvalidan foydalanib, qidiruv jarayonini samarali optimallashtiradi. Ushbu usul qidiruvni qayta-qayta boshlash zaruratini bartaraf etib, murakkablikni O(n + m) gacha kamaytiradi. Algoritmning asosiy qo'llanilish sohalari qidiruv tizimlari, bioinformatika va kompyuter tili tahlilini o'z ichiga oladi. Maqolada Taryana algoritmi misollar va kod fragmentlari bilan izohlangan, uning samaradorligi va amaliy ahamiyati yoritilgan. Algoritmning afzalliklari va cheklovlari ham ko'rib chiqilib, uning boshqa string matching algoritmlari bilan taqqoslanishi keltiriladi.

Kalit so'zlar: Taryana algoritmi, qidirish tizimlari, matn tahlili, algoritm, prefiks-suffiks, bioinformatika, kompyuter tili, tezkor tahlil, text, pattern, Kod analiz, vaqt murakkabligi, resurs, optimal

Annotation.

This article analyzes the application and working principles of the Taryana algorithm in searching for patterns in the text. The algorithm effectively optimizes the search process by using the prefix-suffix table when comparing text and pattern. This method reduces the complexity to O(n + m) by eliminating the need to start the search over and over again. The main areas of application of the algorithm include search engines, bioinformatics and computer language analysis. In the article, the Taryana algorithm is explained with examples and code fragments, its efficiency and practical importance are highlighted. Advantages and limitations of the algorithm are also considered, and its comparison with other string matching algorithms is presented.

Key words: Taryana algorithm, search systems, text analysis, algorithm, prefix-suffix, bioinformatics, computer language, rapid analysis, text, pattern, Code analysis, time complexity, resource, optimal

Аннотация.

В данной статье анализируется применение и принципы работы алгоритма Тарьяны при поиске закономерностей в тексте. Алгоритм эффективно оптимизирует процесс поиска, используя таблицу префиксов-суффиксов при сравнении текста и шаблона. Этот метод снижает сложность до O(n + m), устраняя необходимость начинать поиск снова и снова. Основные области применения алгоритма включают поисковые системы, биоинформатику и анализ компьютерного языка. В статье на примерах и фрагментах кода поясняется алгоритм Тарьяны, подчеркивается его эффективность и практическая значимость. Также рассмотрены преимущества и ограничения алгоритма и представлено его сравнение с другими алгоритмами сопоставления строк.

Ключевые слова: алгоритм Тарьяна, поисковые системы, анализ текста, алгоритм, префикс-суффикс, биоинформатика, язык программирования, экспресс-анализ, текст, шаблон, анализ кода, временная сложность, ресурс, оптимальный.

Taryana algoritmi matn qidirish (string matching) muammolarini samarali hal qilish uchun ishlab chiqilgan zamonaviy algoritmlardan biri hisoblanadi. Ushbu algoritm qidirilayotgan andozani (pattern) asosiy matndan topish jarayonida prefiks-suffiks jadvalidan foydalanib, har bir harfni faqat bir marta tahlil qiladi. Bu esa qidiruv jarayonining murakkabligini **O(n + m)** ga qisqartirishga imkon beradi. Algoritm ikki bosqichda ishlaydi: tayyorlov bosqichi (prefiks-suffiks jadvali yaratish) va qidiruv bosqichi (andozani matnda qidirish). Prefiks-suffiks jadvali qayta-qayta tekshirishdan qochish uchun andoza ichidagi takroriy qismlarni aniqlaydi va qidiruvni optimallashtiradi. Taryana algoritmi matnni tezkor tahlil qilish talab etiladigan sohalarda, jumladan, qidiruv tizimlari, bioinformatika va kompyuter tili tahlilida keng qo‘llaniladi.

Asosiy Maqsadi

Taryana algoritmi berilgan biror matndan ("text") ma'lum bir so‘z yoki patternni ("pattern") tezkor topish vazifasini bajaradi. Bunda vaqtni va hisoblash resurslarini tejash asosiy maqsad hisoblanadi.

Taryana Algoritmi Ishlash Mexanizmi

Taryana algoritmi matn ichidan qidirilayotgan andoza yoki ketma-ketlikni (pattern) topishda optimal yondashuvlarni qo‘llaydi. Ushbu yondashuv matnni faqat bir marta ko‘rib chiqib, prefiks va suffikslar asosida qayta-qayta

tekshirishdan qochadi. Algoritmning ishlash jarayoni ikkita asosiy qismga bo'linadi: **tayyorlov bosqichi** va **qidiruv bosqichi**.

1. Tayyorlov Bosqichi: Prefiks-Suffiks Jadval Yaratish

Prefiks-suffiks jadvali (ba'zida **KMP jadvali** deb ham ataladi) qidirilayotgan "pattern" ichidagi takrorlanishlarni tahlil qiladi. Bu jadval har bir indeksda "pattern" ichida prefiks va suffiks mos keladigan qismlarning maksimal uzunligini saqlaydi.

Misol:

"Pattern" = "ababaca"

- 1-pozitsiyadan (indeks 0) boshlab prefiks va suffiks tahlil qilinadi:

Indeks	Qism	Prefiks	Suffiks	Uzunlik
0	a	-	-	0
1	ab	a	b	0
2	aba	a	a	1
3	abab	ab	ab	2
4	ababa	aba	aba	3
5	ababac	abab	bac	0
6	ababaca	ababab	babaca	1

Prefiks-suffiks jadvali: [0, 0, 1, 2, 3, 0, 1]

Nega kerak?

Bu jadval matn qidiruvi davomida mos kelmagan joylarda "pattern"ni to‘liq qayta tekshirishdan qochishga yordam beradi. Jadval bo‘yicha "pattern"ni oldinga siljitish amalga oshiriladi.

2. Qidiruv Bosqichi: Patternni Matndan Izlash

Endi asosiy matndan ("text") "pattern"ni qidirish jarayonini ko'rib chiqamiz. Ushbu bosqichda prefiks-suffiks jadvali yordamida har bir belgi qiyoslanadi va mos kelmagan joyda qidiruvni optimallashtiradi.

Algoritmning Asosiy Qadamlar:

1. **Matn va Andoza Belgilarini Taqqoslash**:
   - Matnning hozirgi belgisi "pattern"ning hozirgi belgisi bilan solishtiriladi.
   - Agar mos kelsa, ikkala ko‘rsatkich (matn va "pattern" uchun) birga siljitiladi.
2. **Mos Kelmasa**:
   - Agar mos kelmagan joyda tursangiz, prefiks-suffiks jadvali yordamida "pattern"ning boshlanish joyini siljitib, faqat kerakli qismini qayta tekshirasiz.
   - Jadval qiymati 0 bo‘lsa, "pattern" boshidan davom etadi.

3. **Moslikni Tekshirish**:
   - Agar "pattern" to‘liq mos kelsa, qidiruv muvaffaqiyatli yakunlanadi va matndagi boshlang‘ich indeks qaytariladi.

Misol:

Text: "ababcabcababababd"

Pattern: "ababd"

- Prefiks-suffiks jadvali: [0, 0, 1, 2, 0]
- Qidiruv jarayoni:
  - i=0, j=0 => text[0] = pattern[0] => Mos! Har ikkalasi 1 pozitsiyaga siljiydi.
  - i=1, j=1 => text[1] = pattern[1] => Mos!
  - ...
  - i=8, j=4 => text[8] != pattern[4] => Jadvalga qarab j=2 ga qaytadi.
  - Davom etadi va natijada indeks 10 qaytariladi.

Algoritmning Afzalliklari va Optimal Tomonlari

Afzalliklari:

1. **Optimal Vaqt Murakkabligi**:
   - Prefiks-suffiks jadvali yaratish: **O(m)**.
   - Matndan "pattern"ni qidirish: **O(n)**.
   - Umumiy murakkablik: **O(n + m)**.
2. **Takroriy Tekshirishlardan Qochish**:
   - Matnni qayta-qayta boshidan ko‘rib chiqish shart emas, bu esa katta matnlarda samaradorlikni oshiradi.
3. **Kam Resurs Sarfi**:
   - Algoritm ko‘p qo‘shimcha xotira talab qilmaydi. Prefiks-suffiks jadvali asosiy qo‘shimcha resurs hisoblanadi.

Tushunchani Yaxshiroq O‘rganish Uchun Vizual Misol

1. Matn: "ababcabcababababd"
2. Pattern: "ababd"

Vizual Tekshirish:

Matn ababcabcababababd

Pattern **ababd**

Prefiks --- --- --- --- --- ---

- 1-qadam: Har bir belgi mos keladi (i=0..4).
- 2-qadam: Mos kelmagan joyda (i=5) prefiks-suffiks jadval ishlatiladi.
- Oxirgi qadamda "pattern" matnda 10-pozitsiyada topiladi.

Algoritmning Hozirgi Sohalardagi O‘rni

Taryana algoritmi string matching algoritmlari orasida muhim o‘rinni egallaydi va quyidagi sohalarda keng qo‘llaniladi:

1. Qidiruv Tizimlari va Matn Tahlili

- **Hujjatlarni indekslash:** Algoritm katta hajmdagi matnlarda so‘z yoki iboralarni tezda qidirib topishga yordam beradi. Google yoki boshqa qidiruv tizimlarida ushbu algoritmning yondashuvlari foydalaniladi.
- **Matnni filtr qilish:** Spam yoki nomaqbul kontentni tahlil qilishda qidirilayotgan andoza topilsa, matn bloklanadi yoki belgilab qo‘yiladi.

2. Bioinformatika

- **Gen sekanslari tahlili:** Genetik ma’lumotlar ichida maxsus nukleotid ketma-ketliklarini topish uchun qo‘llaniladi.
- Masalan, DNK ichida "ATCG" kabi ketma-ketliklarni izlash.

3. Kompyuter Tili Tahlili va Kod Analizi

- **Sintaksis tahlili:** Kod yozish davomida xatoliklarni aniqlash uchun qoidalarga mos kelmaydigan qismni qidirishda ishlatiladi.
- **Plagiatni aniqlash:** Kodni qiyoslash orqali takroriy qismlarni topish va plagiatni aniqlash.

4. Kiberxavfsizlik

- **Xavfli andozalarni aniqlash:** Fayllarda zararli kod yoki virus imzolarini topish uchun.
- **Foydalanuvchi ma’lumotlarining buzilishlarini aniqlash:** Taryana algoritmi katta log fayllarni tahlil qilishda xavfsizlik bo‘shliqlarini aniqlash uchun ishlatiladi.

5. E-tijorat

- **Mahsulot tavsifi qidiruvi:** E-tijorat platformalarida foydalanuvchi kiritgan kalit so‘zga asoslangan mahsulotlarni izlash.

C# da Taryana Algoritmiga Misol

Quyida matndan biror bir andozani topish uchun **Taryana algoritmi**ning C# kodini keltiraman. Ushbu dastur foydalanuvchidan matn va andozani so‘raydi va andoza qayerda joylashganini qaytaradi:

```
using System;
class TaryanaAlgorithmDemo
{
  // Prefiks-suffiks jadvalini qurish
  public static int[] BuildPrefixTable(string pattern)
  {
```

```
    int m = pattern.Length;
    int[] prefixTable = new int[m];
    int j = 0; // Uzoq mos keluvchi prefiks uzunligi
    prefixTable[0] = 0; // Birinchi element har doim 0

    for (int i = 1; i < m; i++)
    {
        while (j > 0 && pattern[i] != pattern[j])
            j = prefixTable[j - 1];

        if (pattern[i] == pattern[j])
            j++;
        prefixTable[i] = j;
    }
    return prefixTable;
}

// Matndan andoza qidirish
public static int SearchPattern(string text, string pattern)
{
    int n = text.Length;
    int m = pattern.Length;
    int[] prefixTable = BuildPrefixTable(pattern);
    int i = 0, j = 0;

    while (i < n)
    {
        if (text[i] == pattern[j])
        {
            i++;
            j++;
        }
        if (j == m)
        {
            return i - j; // Topilgan andoza boshlanishi
        }
        else if (i < n && text[i] != pattern[j])
```

```
        {
            if (j > 0)
                j = prefixTable[j - 1];
            else
                i++;
        }
    }
    return -1; // Andoza topilmadi
}
static void Main(string[] args)
{
    Console.WriteLine("Matn kiriting:");
    string text = Console.ReadLine();
    Console.WriteLine("Andoza kiriting:");
    string pattern = Console.ReadLine();
    int result = SearchPattern(text, pattern);
    if (result != -1)
        Console.WriteLine($"Andoza matnda {result}-indeksdan boshlanadi.");
    else
        Console.WriteLine("Andoza matnda topilmadi.");
}
}
```

Dastur Ishlash Tartibi

1. Foydalanuvchi **matn** va **andoza** (pattern) kiritadi.
2. Algoritm avval andozaning prefiks-suffiks jadvalini yaratadi.
3. Keyin matn ichidan andozani qidiradi.
4. Agar topilsa, andozaning boshlang‘ich indeksini qaytaradi; aks holda, "topilmadi" deb chiqaradi.

Misol:

- **Kiritish**:
  Matn: "ababcabcababababd"

  Andoza: "ababd"
- **Natija**:
  Andoza matnda 10-indeksdan boshlanadi.

Xulosa

Taryana algoritmi nafaqat nazariy jihatdan samarali, balki amaliyotda ham ko‘plab muammolarni hal qilishda qo‘llaniladi. Uning C# dagi dasturiy talqini real sohalarda, jumladan, bioinformatika, kiberxavfsizlik va qidiruv tizimlarida o‘zining yuqori samaradorligini isbotlagan. Ushbu dasturda berilgan yechimlar har qanday sohada qo‘llanishi uchun moslashtirilishi mumkin.

Foydalanilgan adabiyotlar:

1.Knuth, D. E., Morris, J. H., & Pratt, V. R. (1977). Fast pattern matching in strings. SIAM Journal on Computing, 6(2), 323–350.
2.Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
3.Gusfield, D. (1997). Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press.
4.Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley.
5.Navarro, G. (2001). A guided tour to approximate string matching. ACM Computing Surveys, 33(1), 31–88.
6.Kumar, A., & Singh, S. (2014). String matching algorithms and their applicability in computer science and bioinformatics. International Journal of Computer Applications, 98(2), 15–23.
7.LeetCode Discussions and Tutorials. (n.d.). String Matching Algorithms. https://leetcode.com
8.Farmonov, S., & Nazirov, A. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. B CENTRAL ASIAN JOURNAL OF EDUCATION AND INNOVATION (Т. 2, Выпуск 12, сс. 71–74). Zenodo.
9.Raxmonjonovich, F. S. (2024). C# VA MASHINA TILI. Ta'lim innovatsiyasi va integratsiyasi, 12(1), 59-62.
10. Farmonov, S., & Toirov, S. (2023). NETDA DASTURLASHNING ZAMONAVIY TEXNOLOGIYALARINI O'RGANISH. Theoretical aspects in the formation of pedagogical sciences, 2(22), 90-96
11. Raxmonjonovich, F. S. (2023). Array ma’lumotlar tizimini talabalarga o’qitishda Blockchain metodidan foydalanish. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 541-547.
12. Raxmonjonovich, F. S. (2023). Dasturlashda interfeyslardan foydalanishning ahamiyati. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 425-429.
13. Raxmonjonovich, F. S. (2023). Dasturlashda obyektga yo’naltirilgan dasturlashning ahamiyati. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va riyojlanish omillari, 2(2), 434-438.

14. Raxmonjonovich, F. S. (2023). Dasturlash tillarida fayllar bilan ishlash mavzusini Blended Learning metodi yordamida o'qitish. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 464-469.
15. Raxmonjonovich, F. S. (2023). DASTURLASHDA ISTISNOLARNING AHAMIYATI. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 475-481.
16. Raxmonjonovich, F. S. (2023). Dasturlashda abstraksiyaning o'rni. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 482-486.
17. Raxmonjonovich, F. S., & Ravshanbek o'g'li, A. A. (2023). Zamonaviy dasturlash tillarining qiyosiy tahlili. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 430-433.
18. Raxmonjonovich, F. S. (2023). C# dasturlash tilida fayl operatsiyalari qo'llashning qulayliklari haqida. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 439-446.
19. Raxmonjonovich, F. S. (2023). C# tilida ArrayList bilan ishlashning afzalliklari. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 470-474.
20. Farmonov Sherzodbek Raxmonjonovich, & Rustamova Humoraxon Sultonbek qizi. (2024). C# DASTURLASH TILIDA TO'PLAMLAR BILAN ISHLASH. Ta'lim Innovatsiyasi Va Integratsiyasi, 11(10), 210–214. Retrieved from http://web-journal.ru/index.php/ilmiy/article/view/2480.
21. Raxmonjonovich, F. S., & Ravshanbek o'g'li, A. A. (2023). Zamonaviy dasturlash tillarining qiyosiy tahlili. Yangi O'zbekiston taraqqiyotida tadqiqotlarni o'rni va rivojlanish omillari, 2(2), 430-433.
22. Farmonov, S., & Rasuljonova, Z. (2024). OB'EKTGA YO'NALTIRILGAN DASTURLASH ZAMONAVIY DASTURLASHNING ASOSI SIFATIDA. Центральноазиатский журнал образования и инноваций, 3(1), 83-86.
23. Farmonov, S., & Ro'zimatov, J. (2024). DASTURLASH TILLARINI O'RGANISHDA ONLINE TA'LIM PLATFORMALARIDAN FOYDALANISH. Theoretical aspects in the formation of pedagogical sciences, 3(1), 5-10.
24. Farmonov, S. R., & qizi Xomidova, M. A. (2024). C# VA JAVA DASTURLASH TILLARIDA FAYLLAR BILAN ISHLASHNING TURLI USULLARINING SAMARADORLIGI HAQIDA. Zamonaviy fan va ta'lim yangiliklari xalqaro ilmiy jurnal, 1(9), 45-51.

25. Farmonov, S. (2023). C# DASTURLASH TILIDA GRAY KODI BILAN ISHLASH. Центральноазиатский журнал образования и инноваций, 2(12 Part 2), 71-74.

26. Farmonov, S., & Jo'rayeva, M. (2023, December). DASTURLASHDA POLIMORFIZMNING AHAMIYATI. In Международная конференция академических наук (Vol. 2, No. 13, pp. 5-8).