15 - AMALIY MASHG‘ULOT. *STRINGLARDA QISM STRINGLARNI IZLASH ALGORITMLARI.*

Ishdan maqsad: Talabalarda stringlarda qism stringlarni izlash algoritmlari bo‘yicha bilim va ko‘nikmalarini oshirish. C++ dasturlash tilida qism stringlarni izlash algoritmlari bo‘yicha yaratilgan dasturlar bilan tanishish.

Nazariy qism: Stringlarda qism stringlarni izlash algoritmlari, asosan bir matn ichidagi maqbul qism yoki segmentlarni izlash va aniqlash uchun ishlatiladi. Ular bir nechta sohalar uchun muhim bo’lib, masalan, matn tahlili, matn ma’lumotlarini izlash, matn tuzilmasi va boshqa amallarni o‘z ichiga oladi. Quyidagi algoritmlar aynan stringlarda qism stringlarni izlashga mo‘ljallangandir:

Naive Substring Search Algorithm: Bu oddiy usulda, biz qism stringni izlash uchun barcha matn ustida, har bir pozitsiya uchun qism stringni solishtiramiz. Bu usul oson va tushunarli, ammo katta matnlarda yoki katta qism stringlarini izlashda qiyinliklarga duch kelishi mumkin.

Knuth-Morris-Pratt (KMP) Algorithm: KMP algoritmi qism stringni izlashda o’zaro bog’lanishni aniqlaydigan tezkor algoritm hisoblanadi. U qidiruvda foydalaniladigan qism stringning boshi hisoblanib, so’ng qidiruvni davom ettirishga yordam beradi. Bu algoritmning muhim xususiyati, qidiruvda qo’shimcha solishtirmalarni o’tkazishdan foydalanishdir, bu esa asosiy ish vaqtini kamaytiradi.

Boyer-Moore Algorithm: Bu algoritm qidiruv jarayonini tezlashtirish uchun qidiruvda o’zgaruvchilardan foydalanadi. U qism stringni o’rganish uchun o’ng tomondan boshlab qidiruvni amalga oshiradi va tekshirishni eng kam tartibda o’tkazadi. Bu esa undagi muhim solishtirishlarni chiqarishda foydalanishning oson va tez bo’lishini ta’minlaydi.

Rabin-Karp Algorithm: Bu algoritm qism stringni izlashda xeshlashdan foydalanadi. U matndagi har bir qismni xeshlash va uni qidirishda qo’llanadi. Bu algoritmning afzalliklari, u tekshirish uchun tezligini ta’minlashi va bir nechta qism stringlarini bir vaqtning o’zida izlashga imkoniyat berishi.

Bu algoritmlar matnlarda qidiruvni optimallashtirishda va qism stringlarini topishda yordam beradi. Qulayliklar, xususiyatlar va ilovalarning talablari taqdim etilgan qism stringini izlash uchun eng mos algoritmini tanlashda muhimdir.

Amaliy qism.

Naive Approach (Oddiy yondoshma): Eng oddiy usul, qism stringni stringdagi har bir pozitsiyaga mos ravishda solishtirib, mos kelganlikni tekshirib chiqadi. Agar topilmasa, bir pozitsiyaga o’tib keyinroq tekshirib chiqiladi.

```
#include <iostream>
#include <string>
```

```
using namespace std;
void naiveSearch(const string& text, const string& pattern) {
    int n = text.length();
    int m = pattern.length();
    for (int i = 0; i <= n - m; ++i) {
        int j;
        for (j = 0; j < m; ++j) {
            if (text[i + j] != pattern[j])
                break;
        }
        if (j == m){
            cout << "Pattern found at index: " << i << endl;
    }
}
int main() {
    string text = "xabcdabcy";
    string pattern = "abc";
    naiveSearch(text, pattern);
    return 0;
}
```

Knuth-Morris-Pratt (KMP) Algorithm: Bu algoritm qism stringni o'zlashtiradi va undagi shablonlariga mos kelishlar jadvalidini yaratadi. Keyin, asosiy matnda harakatlanib, mos kelganlikni tekshiradi va kerakli o'tishlarni bajaradi.

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
void computeLPSArray(const string& pattern, vector<int>& lps) {
    int m = pattern.length();
    lps.resize(m, 0);
    int len = 0;
    int i = 1;
    while (i < m) {
        if (pattern[i] == pattern[len]) {
            len++;
            lps[i] = len;
            i++;
        } else {
            if (len != 0) {
                len = lps[len - 1];
            } else {
```

```
            lps[i] = 0;
            i++;
        }
    }
  }
}
void KMPSearch(const string& text, const string& pattern) {
  int n = text.length();
  int m = pattern.length();
  vector<int> lps;
  computeLPSArray(pattern, lps);
  int i = 0, j = 0;
  while (i < n) {
    if (pattern[j] == text[i]) {
      j++;
      i++;
    }
    if (j == m) {
      cout << "Pattern found at index " << i - j << endl;
      j = lps[j - 1];
    } else if (i < n && pattern[j] != text[i]) {
      if (j != 0) {
        j = lps[j - 1];
      } else {
        i++;
      }
    }
  }
}
int main() {
  string text = "xabcdabcy";
  string pattern = "abc";
  KMPSearch(text, pattern);
  return 0;
}
```

Boyer-Moore Algorithm: Bu algoritm qism stringni chapdan o'ngga tekshiradi va shablonlarda juda katta qadam o'tkazadi. Agar mos kelmagan bo'lsa, shablonni qanday qadamda o'ngga o'tkazish kerakligini aniqlaydi.

```
#include <iostream>
#include <string>
#include <vector>
```

```
using namespace std;
// #define NO_OF_CHARS 256
void badCharHeuristic(const string& str, int size, int badChar[NO_OF_CHARS]) {
    for (int i = 0; i < NO_OF_CHARS; i++)
        badChar[i] = -1;
    for (int i = 0; i < size; i++)
        badChar[(int)str[i]] = i;
}
void search(const string& txt, const string& pat) {
    int m = pat.size();
    int n = txt.size();
    int badChar[NO_OF_CHARS];
    badCharHeuristic(pat, m, badChar);
    int s = 0;
    while (s <= (n - m)) {
        int j = m - 1;
        while (j >= 0 && pat[j] == txt[s + j])
            j--;
        if (j < 0) {
            cout << "Pattern found at index " << s << endl;
            s += (s + m < n) ? m - badChar[txt[s + m]] : 1;
        } else
            s += max(1, j - badChar[txt[s + j]]);
    }
}
int main() {
    string txt = "xabcxxdabcyxx";
    string pat = "xx";
    search(txt, pat);
    return 0;
}
```

Regular Expressions (Regex):

Bu misolda, regex kutubxonasidan foydalanamiz. Bu kutubxona C++ standartida mavjud bo'lgan bir qismidir.

```
#include <iostream>
#include <regex>
using namespace std;
int main() {
    string txt = "xabxxcdabxxcy";
    regex pattern("xx");
```

```
    smatch matches;
    if (regex_search(txt, matches, pattern)) {
        cout << "Pattern found at index " << matches.position() << endl;
    }
    return 0;
}
```

Mustaqil bajarish uchun topshiriqlar

1. Berilgan matndan kiritilgan qismni izlash algoritmini yozing.
2. Berilgan matndan kiritilgan qismni izlash uchun KMP algoritmini qo'llang.
3. Berilgan matndan kiritilgan qismni izlash uchun Boyer-Moore algoritmini qo'llang.
4. Berilgan matndan kiritilgan qismni izlash uchun regexdan foydalaning.
5. Naive Approach yoki KMP algoritmini kuchaytirib, ko'p qism stringlarni izlash uchun yuzaga kelgan qism stringini toping.
6. Berilgan stringda barcha qism stringlarni toping va ularni konsolga chiqaring.

Masalan: Berilgan string: "abcabcabc"

Natija:

7. Berilgan stringda faqat unikal qism stringlarni toping va ularni konsolga chiqaring.

Masalan: Berilgan string: "abccbaabc"

Natija:

8. Berilgan stringda qaysi qism stringning nechta marta takrorlandigini aniqlang va takrorlanuvchi stringlarni konsolga chiqaring.

Masalan: Berilgan string: "abcabcabcb"

Natija:

```
abc - 2 marta
abcb - 1 marta
b - 3 marta
```

9. Berilgan stringning barcha substringlarini toping va ularni konsolga chiqaring.

Masalan: Berilgan string: "abcd"

Natija:

```
a
ab
abc
abcd
b
bc
bcd
c
cd
d
```

10. Berilgan stringning eng katta uzunligdagi qism stringni aniqlang va uni konsolga chiqaring.

Masalan: Berilgan string: "abcabcaab"

Natija: abcabcaab

11. Berilgan stringda bir qator belgi orqali boshlangan qism stringni toping va uni konsolga chiqaring.

Masalan: Berilgan string: "abc,def,ghi,jkl"

Natija: def,ghi,jkl

12. Berilgan stringda berilgan qism stringni izlash va uni konsolga chiqaring.

Masalan: Berilgan string: "The quick brown fox jumps over the lazy dog" Izlanayotgan string: "quick"

Natija: quick

13. Berilgan stringda berilgan qism stringlarni toping va ularni ichiga o'tkazilgan hamma stringlarni konsolga chiqaring.

Masalan: Berilgan string: "abccbabccbabccbabccbabccba" Izlanayotgan string: "abc"

Natija:

14. Berilgan stringda berilgan qism stringni necha marta takrorlanganligini aniqlang va natijani konsolga chiqaring.

Masalan: Berilgan string: "abccbabccbabccbabccbabccba" Izlanayotgan string: "abc"

Natija: abc - 6 marta

15. Berilgan stringda berilgan qism stringni qanday darajada takrorlanganligini aniqlang va natijani konsolga chiqaring.

Masalan: Berilgan string: "abccbabccbabccbabccbabccba" Izlanayotgan string: "abc"

Natija: abc - ikkita darajada