Лекція 6.

Алгоритми пошуку підрядків в рядках

§1. Основні поняття

Рядок (слово) – це послідовність знаків (букв), з деякої скінченої множини, яка називається алфавітом.

Слова позначають буквами латинського алфавіту.

Наприклад, $X=x[1]x[2]\ldots x[n]$ – слово довжини n, де $x[i]$ (i-та буква), яка належить алфавіту.

Довжина рядка – це кількість знаків у рядку, позначається $\text{Length}(X)=|X|=n$.

Слово, яке не містить жодної букви називається ***порожнім***. Порожнє слово зазвичай позначають буквою *L*, Length(*L*)=0.

Слово *X* називається ***префіксом*** слова *Y*, якщо є таке слово *Z*, що *Y*=*XZ*. Причому саме слово є префіксом для самого себе, так як знайдеться нульове слово *L*, таке що *X*=*XL*.

Приклад: слово ab є префіксом слова abcfa.

Слово *X* називається ***суфіксом*** слова *Y*, якщо є таке слово *Z*, що *Y*=*ZX*. Аналогічно, слово є суфіксом самого себе.

Приклад: слово bfg є суфіксом слова vsenfbfg.

Слово X називається ***підрядком*** рядка Y, якщо знайдуться такі рядки Z_1 и Z_2, що $Y=Z_1XZ_2$.

При цьому Z_1 називають лівим, а Z_2 - правим крилом підрядка.

Підрядком може бути і само слово. При цьому слово X називають входженням в слово Y. Серед всіх входжень слова X в слово Y, входження з найменшою довжиною свого лівого крила називають першим або головним входженням. Входження позначають $X \subseteq Y$.

Приклад: слова hrf та fhr є підрядками слова abhrfhr.

§2. Постановка задачі пошуку

Задача пошуку підрядків полягає наступному:

– існує деякий текст $T[1..n]$ і шуканий текст $P[1..m]$, $m \le n$.

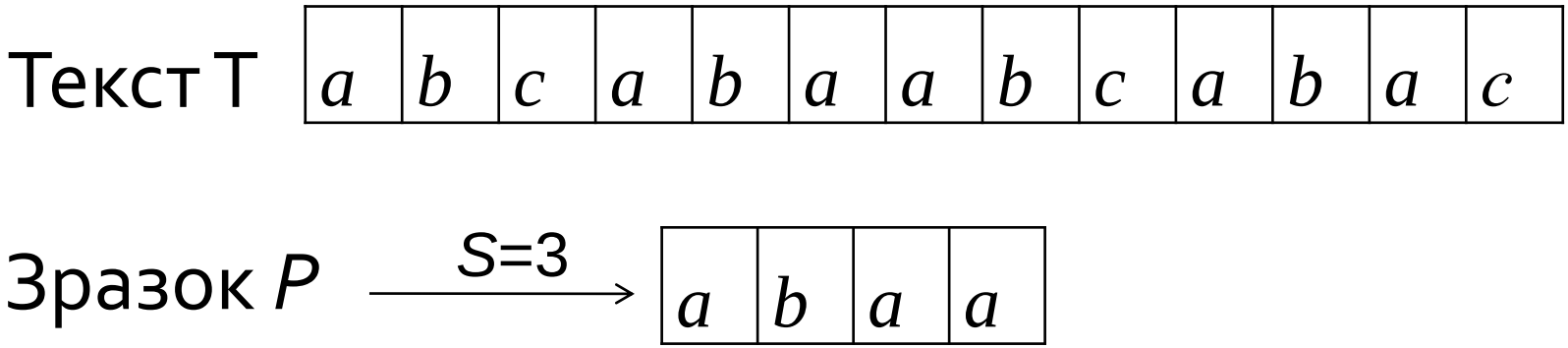
– потрібно знайти всі входження тексту P в текст T.

Текст, який шукають у рядку, називається ***зразком***. Кажуть, що зразок P входить в текст T із зсувом S, якщо

$$P[1\ldots m] = T[S+1\ldots S+m] \text{ і } 0 \le S \le n.$$

Приклад. Необхідно знайти всі входження зразка *P*=*abaa* в текст T=*abcabaabcabac*.

Розв'язок. Зразок *P* зустрічається в тексті лише 1 раз зі зсувом *S* =3



§3. Алгоритм послідовного (прямого) пошуку

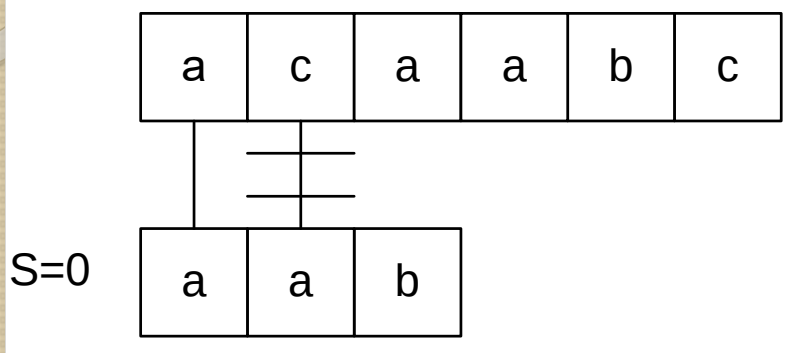
Послідовний алгоритм пошуку – це найпростіший алгоритм, але не найефективніший.

Пошук всіх допустимих зсувів виконується послідовним порівняння зразка P зі всіма підрядками тексту T.

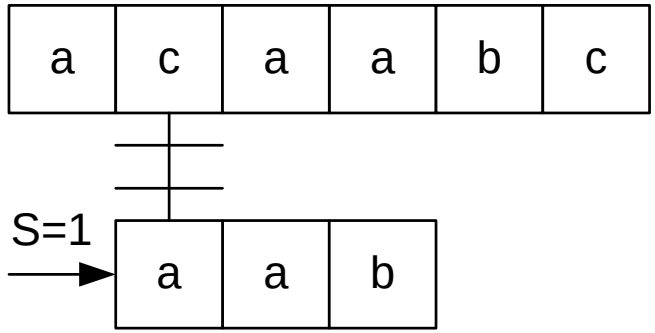
Це легко виконати за допомогою циклу, в якому перевіряється умова $P[1..m]=T[S+1\ldots S+m]$ для кожного з $n-m+1$ можливих значень S.

```
Procedure Search;
n=length(T);
m=length(P);
        for S=0 to n-m do
  begin
    j:=1;
  while (j ≤ m) and (P[j] = T[j+S]) do inc(j);
  if  j > m then writeln (S);  //рядок P
   входить в T зі зсувом S
end;
```

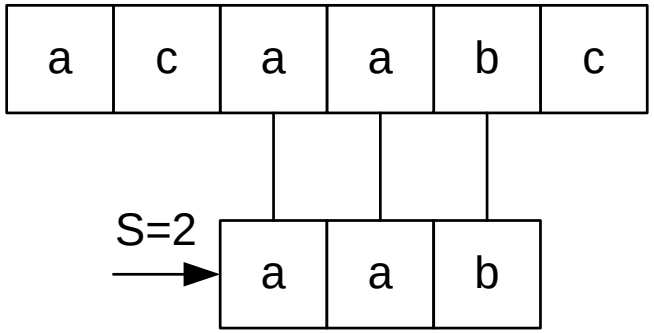
Приклад: Знайти всі входження зразка *P=aab* в тексті *T=acaabc*, використовуючи найпростіший алгоритм пошуку.



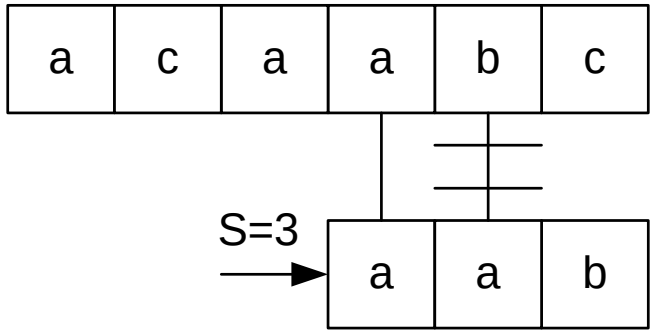
а)



б)



в)



г)

Час роботи алгоритму дорівнює $O(m(n-+1))$.

Найпростіший алгоритм пошуку підрядків є неефективним, оскільки інформація про текст, отримана для одного значення S, повністю ігнорується при розгляді інших значень S.

Наприклад, якщо зразок має вид $P=aaab$ ($T=acaabc$), а значення одного з допустимих зсувів дорівнює нулю, то ні один зі зсувів, рівних 1, 2 або 3 не можуть бути допустимими, оскільки $T[4]=b$. Найпростіший алгоритм це не враховує.

§4. Алгоритм Рабіна-Карпа

Ідея, запропонована Рабіном і Карпом, полягає в тому, щоб поставити у відповідність кожному рядку деяке унікальне число, і замість того, щоб порівнювати самі рядки, порівнювати числа, що набагато швидше.

Розглянемо P і T як великі числа, записані в системі зчислення d=256. Кожен символ $P(i)$ і $T(i)$ будемо вважати цифрою від 0 до 255.

Тоді $P = P[1]d^{m-1} + P[2]d^{m-2} + ...P[m]$

Позначимо частину числа T довжиною m d-ічних цифр, починаючи з зміщення S як:

$$T_S = T[S+1]d^{m-1} + T[S+2]d^{m-2} + ... + T[S+m]$$

Ми шукаємо ті зсуви S, де T_S=P.

Число P можна порахувати за $O(m)$ ітерацій, використовуючи схему Горнера:

$$P = P[m] + d(P[m-1] + d(P[m-2] + ... + d(P[2] + dP[1]))$$

T_S аналогічно обчислюється за $O(m)$ кроків. Знаючи T_S можна обчислити T_{S+1} за $O(1)$ ітерацій:

$$T_{S+1} = (T_S - T[S+1]d^{m-1})d + T[S+1+m]$$

При цьому видаляємо цифру зі старшого розряду і додаємо нову цифру в молодший розряд.

Числа T_S і P можуть бути настільки великими, що не вміщуватимуться в стандартні типи змінних і прийдеться реалізовувати “арифметику з великими числами”.

Вирішенням цієї проблеми є обчислення значень з P і T_S за модулем деякого фіксованого числа q.

Недолік: рівність $T_S = P$ за модулем q не означає, що рядки $P[1+m]$ і $T[S+i...S+m]$ рівні, але вона відкидає завідомо невірні варіанти.

Тому на кожному кроці потрібно перевіряти посимвольно $P[1...m]=T[S+1...S+m]$ при умові виконання $T_S=P$ по модулю q. Враховуючи модуль q:

$$T_{S+1} = d \cdot T_S \bmod q - h \cdot T[S+1] \bmod q + T[S+1+m] \bmod q$$

$$T_{S+1} = (d(T_S - T[S+1] \cdot h) + T[S+m+1]) \bmod q$$

де $h = d^m \bmod q$ - це значення, яке набуває цифра 1 розміщена в старшому розряді m-цифрового текс

Число P, враховуючи модуль q, обчислюється за формулою:

$$P = (P[m] + d(P[m-1] + d(P[m-2] + \ldots + d(P[2] + dP[1]))) \bmod q$$

В якості модуля q можна обрати таке **просте число**, для якого довжина $10q$ не перевищує довжини комп'ютерного слова. Зазвичай в якості q беруть найбільше просте число з урахуванням обмежень: $qd<2^{31}-1$ та $qh<2^{31}-1$. Наприклад, для d=256 найбільше значення q=8388593.

Приклад: *T=abcdeabfgfcakmbddaf* (n=19), *P=bfgfc* (*m*=5), {0,1…9} – алфавіт, *d*=10, $h=d^{m-1}=10^4=10000$; *q*=13.

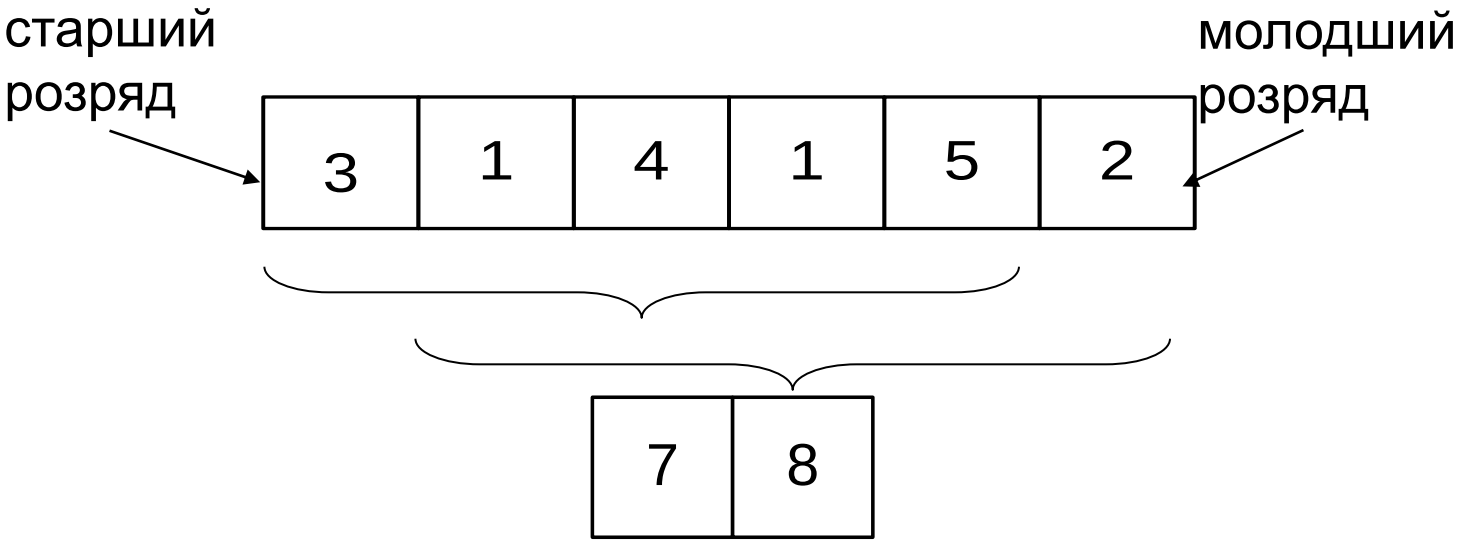
b	f	g	f	c
3	1	4	1	5

P=31415.

P_q =(5+10(1+10(4+10(1+10·3)))) mod 13=

=31415 mod 13=7

Необхідно шукати підрядки, які дорівнюють 7 по модулю 13.



$T_6=31415$, $T_6 \bmod 13=7$.

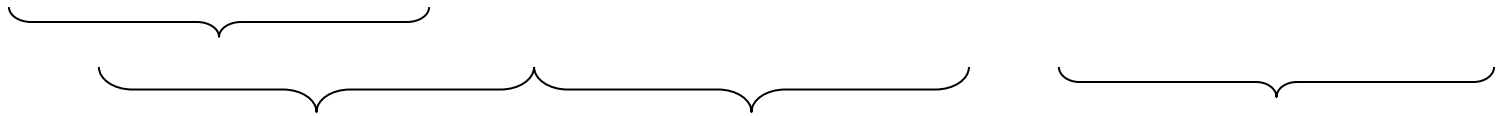
Обчислюємо T_7:

$T_7=10(31415-3\cdot 10000)+2=14152$

$T_7 \bmod 13=8$

Обчислимо значення T_S для всього тексту:

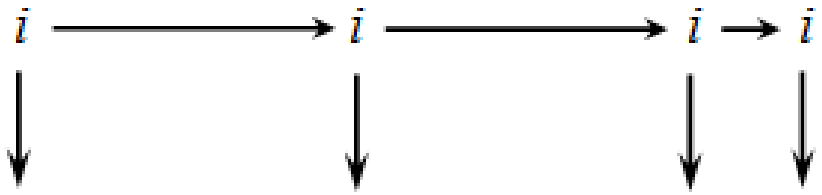
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
a	b	c	d	e	a	b	f	g	f	c	a	k	m	b	d	d	a	f
2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1



8	9	3	11	0	1	7	8	4	5	10	11	7	9	11

§5. Алгоритм Кнута-Морриса-Пратта (КМП)

Ідея алгоритму: при кожному неспівпадінні двох символів тексту та зразка, зразок зсувається на деяку відстань, так як менші зсуви не зможуть привести до повного співпадіння.



Рядок	A	B	C	A	B	C	A	A	B	C	A	B	D
Зразок	**A**	**B**	**C**	**A**	**B**	**D**							
				A	**B**	**C**	**A**	**B**	D				
							A	**B**	C	A	B	D	
								A	**B**	**C**	**A**	**B**	**D**

Алгоритм КМП складається з двох етапів: підготовчого (побудова префікс-функції) і основного (пошуку).

Підготовчий етап.

Побудова префікс-функції.

Нехай задано рядок $P[1\ldots n]$. Необхідно обчислити для нього **префікс-функцію** - масив чисел $f[1\ldots n]$, де $f[i]$ визначається наступним чином: це така найбільша довжина найбільшого власного суфікса підрядка $P[1\ldots i]$, який співпадає з його префіксом. При цьому $f[1]=0$.

$$f[i] = \max_{k=1..i} \{k : P[1...k-1] = P[i-k+2...i]\}$$

Приклад 1: Знайти префікс-функцію рядка *P*=abcabcd.

Префікс	Суфікс	Префікс-функція f [i]
a	a	0
ab	ab	0
abc	abc	0
abca	abc**a**	1
abcab	abc**ab**	2
abcabc	abc**abc**	3
abcabcd	abcabcd	0

Приклад 2: знайти префікс-функцію рядка *P=abababababca*.

Префікс	Суфікс	Префікс-функція f [i]
a	a	0
ab	ab	0
aba	aba	1
abab	abab	2
ababa	ababa	3
abababab	abababab	4
ababababa	ababababa	5
abababababab	abababababab	6
abababababc	abababababc	0
ababababababca	ababababababca	1

Схема побудови префікс функції:

1) Обчислюємо значення префікс-функції $f[i]$ при $i=1..n$. При цьому $f[0]=0$.

2) Для визначення поточного значення $f[i]$ введемо змінну j – поточна довжина рядка, що розглядається: $j= f[i-1]$.

3) Тестуємо зразок довжини j. Порівнюємо $P[i]$ та $P[j]$. Якщо $P[i]=P[j]$, то $f[i]= j+1$, $i= i+1$. Якщо $P[i]\neq P[j]$, то $j=f[j-1]$ і починаємо спочатку.

4) Якщо $j=0$, а співпадіння не знайдено, то $f[i]=0$, переходимо до індексу $i=i+1$.

Основний етап. Пошук.

Відомості про кількість символів, що співпали q дозволяють зробити висновок про те, які символи містяться у відповідних місцях тексту. Це дозволяє одразу визначити, які зсуви будуть недопустимими.

Якщо перші q символів співпали при зсуві S_i, то наступний зсув, який може бути допустимим, дорівнює $S_{i+1}=S_i+(q - f(q))$.

Якщо $q=0$, співпадінь при зсуві S_i не виявлено, то наступний зсув, який може бути допустимим, дорівнює $S_{i+1}=S_i+1$.

Приклад. $f[i]=[0,0,0,1,2,0]$.

1) Зсув $S_0=0$, $q=5$, $f[q]=f[5]=2$,

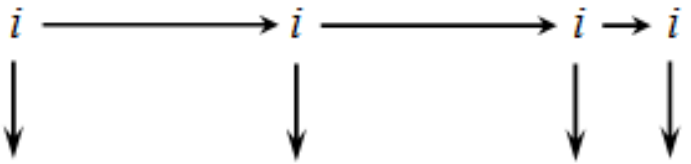
$S_1=S_0+(q-f(q))=0+(5-2)=3$, $i=4$.

2) Зсув $S_1=3$, $q=4$, $f[q]=f[4]=1$,

$S_2=S_1+(q-f(q))=3+(4-1)=6$, $i=7$.

3) Зсув $S_2=6$, $q=1$, $f[q]=f[1]=0$,

$S_3=S_2+(q-f(q))=6+(1-0)=7$, $i=8$.



Строка	A	B	C	A	B	C	A	A	B	C	A	B	D
Подстрока	**A**	**B**	**C**	**A** **A**	**B** **B**	**D** **C**	**A** **A**	**B** **B** **A**	D C **B**	A **C**	B **A**	D **B**	**D**

§6.Порівняння алгоритмів пошуку

Алгоритм	Попередня обробка	Середній час пошуку	Найгірший час
Алгоритм послідовного пошуку	Відсутня	2*n	O(n*m)
Алгоритм Рабіна-Карпа	Відсутня	O(n+m)	O(n*m)
Алгоритм КМП	O(m)	O(n+m)	O(n+m)