1 Wstęp

Rozważamy problem znajdowania wszystkich wystąpień wzorca x o długości m w tekście y o długości n. Zakładamy, że m jest istotnie mniejsze niż n, nie jest istotny dla nas czas preprocessingu wzorca x, lecz liczba porównań liter z tekstem y. Zakładamy również, że czytelnik jest obeznany z algorytmem Morrissa-Pratta na wyszukiwanie wszystkich wystąpień wzorca w tekście.

Algorytm Morrissa-Pratta wykonuje pesymistycznie około $2n$ porównań liter z tekstu x z literami z tekstu y. Jako przykład weźmy $x = a^{m-1}b$, $y = a^n$. Wówczas algorytm Morrisa-Pratta, po porównaniu pierwszych $m-1$ liter a i stwierdzeniu, że są one zgodne, dla każdej kolejnej litery tekstu y:

1. będzie miał zapamiętane, że znalazł już $m-1$ liter wzorca x;
2. sprawdzi, że aktualna litera tekstu nie pasuje do wzorca x, tj $y[i] = a \neq b = x[m]$;
3. dokona przesunięcia zgodnie z tablicą P wzorca x, najdłuższym presikso-sufiksem słowa $a^{m-1} = x[1 \ldots m-1]$ jest a^{m-2};
4. sprawdzi, że aktualna litera tekstu pasuje do wzorca x na pozycji $m-1$, tj $y[i] = a = x[m-1]$;
5. uaktualni informację, że znalazł już $m-1$ liter wzorca x i przesunie się do następnej litery tekstu.

W sumie, algorytm wykona $2n-m+1$ porównań, co przy założeniu $m \ll n$ jest bliskie $2n$. Poniżej przedstawimy algorytm, który w pesymistycznym wypadku dokonuje $\frac{3}{2}n$ porównań tekstu z wzorcem przy preprocessingu wzorca x w czasie $O(n)$ i pamięci $O(n)$. Algorytm ten jest nieznaczną modyfikacją algorytmu Morrissa-Pratta. O tekście x zakładamy, że zawiera przynajmniej dwie różne litery; znajdowanie wystąpień wzorca a^m w dokładnie n porównaniach mamy nadzieję, że nie przedstawia Czytelnikowy żadnych problemów.

2 Algorytm

2.1 Preprocessing

O tekście x chcemy wiedzieć następujące dwie rzeczy:

1. Chcemy mieć skonstruowaną tablicę P wzorca x, tą samą co używana jest w algorytmie Morrissa-Pratta; tj $P[i]$ jest długością najdłuższego prefikso-sufiksu słowa $x[1 \ldots i]$.

2. Załóżmy, że pierwszą literą słowa x jest a, zaś pierwszą różną od a literą sło"wa x jest b i pierwszy raz występuje ona na pozycji K. Innymi słowy, $x[1 \ldots K] = a^{K-1}b$. Zauważmy, że $2 \leqslant K \leqslant m$. W trakcie preprocessingu tekstu x zapamiętujemy wartość K.

Zauważmy, że dla każdego $i \geqslant K$ mamy $i - P[i] \geqslant K$, gdyż wiemy, że $x[1 \ldots K] = a^{K-1}b$: prefikso-sufiks słowa zaczynającego się do $a^{K-1}b$ musi zaczynać się za tą literą b w wystąpieniu jako sufiks.

2.2 Treść algorytmu

Poniżej zamieszczamy pseudokod algorytmu:

```
var L, I, J;

procedure SPRAWDZ_LITERY_A;
var s:integer;
{
  for s := (I + L) to (I + K - 1) do
    if y[s] != 'a' then
      return false;
  return true;
}

procedure ZNAJDZ_WSZYSTKIE_WYSZUKANIA
{
  L := 0; I := 1; J := 0;
  while (I <= n - m + 1) do {
    if (J = m - K + 1) or (x[J + K] != y[I + J + K - 1]) then {
      if (J = m - K + 1) and SPRAWDZ_LITERY_A then
        ZNALEZIONO_WZORZEC(I);
      if (J = 0) then {
        L := max(L-1, 0);
        I := I + 1;
      } else if (P[J + K - 1] <= K - 1) then {
        L := P[J + K - 1];
        I := I + (J + K - 1 - P[J + K - 1]);
        J := 0;
      } else {
        L := K - 1;
        I := I + (J + K - 1 - P[J + K - 1]);
```

```
        J := P[J + K - 1] - (K - 1);
      }
    } else
      J := J + 1;
  }
}
```

i już śpieszymy go wyjaśnić. Zmienne kolejno oznaczają:

- Aktualnie staramy się sprawdzić, czy wzorzec występuje w tekście poczynając od pozycji I tekstu.
- Wiemy, że litery $x[K \ldots K+J-1]$ się zgadzają z tekstem na tej pozycji, tj pierwszych J liter poczynając od pozycji K wzorca się zgadza z tekstem.
- Wiemy, że litery $x[1 \ldots L]$ się zgadzają z tekstem na tej pozycji, tj pierwszych L liter a wzorca się zgadza z tekstem.

Algorytm stara się znaleźć wzorzec w tekście podobnie jak algorytm Morrissa-Pratta, ale wpierw sprawdzając litery od pozycji K wzorca (czyli pozycje K, $K+1$ itd.). Dopiero, jeśli wszystkie spasują, sprawdza początkowe $K-1$ liter a, czy pasują.

Funkcja `SPRAWDZ_LITERY_A` sprawdza, czy litery a na pozycjach wzorca od $L+1$ do $K-1$ pasują do tekstu.

Tłumacząc dokładniej, algorytm w pętli wykonuje:

- Jeśli już znaleźliśmy wszystkie litery wzorca poza początkowymi literami a ($J = m - K + 1$) lub sprawdzamy kolejną literę i nie pasuje ($x[J+K] \neq y[I+J+K-1]$) to:
  - Jeśli to jest przypadek, że znaleźliśmy wszystkie litery wzorca od pozycji K, to sprawdzamy brakujące litery a i jesłi się zgadzają, wypisujemy, że znaleziono wystąpienie.
  - Niezależnie od tego, czy jest wystąpienie, czy właśnie litera wzorca nie spasowała, musimy wykonać odpowiednik przejścia przez tablicę P w algorytmie Morrissa-Pratta. Czyli:
    - Jeśli nie było wcześniej żadnego spasowania ($J = 0$), to przesuwamy się do następnej pozycji ($I := I + 1$). J wciąż zostaje 0, zaś jeśli wcześniej wiedzieliśmy, że pasuje nam L liter a na początku słowa, to teraz wiemy tylko o $L-1$ literach — przesunęliśmy się o jeden w prawo.

    * Jeśli było spasowanie, dokonujemy przejścia przez tabliće P jak w algorytmie Morrissa-Pratta, tj. następną możliwą pozycją do szukania wzorca jest $J + K - 1 - P[J + K - 1]$. O tyle przesuwamy I. Zauważyliśmy już, że to przesunięcie będzie wynosiło co najmniej K. Wobec tego wszystkie litery z początkowych $P[J + K - 1]$ nasz algorytm już sprawdził że pasują. Jeśli $P[J + K - 1] \leqslant K - 1$, czyli nasza wiedza obejmuje tylko litery a, zapamiętujemy tę wiedzę w zmiennej L i ustawiamy J na 0. Jeśli wiemy o większej ilości liter, zapamiętujemy to w L ($L := K - 1$) i w J.

- W przeciwnym przypadku, czyli jeśli nie znaleźliśmy jeszcze całego wzorca po pozycji K, a kolejna litera pasuje, po prostu zwiększamy J o jeden.

Poprawność algorytmu, czyli to, że algorytm znajduje wszystkie wystąpienia wzorca w tekście, jest oczywista — algorytm sprawdza wszystkie możliwe wystąpienia wzorca tak jak algorytm Morrissa-Pratta, korzystając z tablicy P. W tym tekście skupimy się na dowodzie, że ten algorytm wykonuje pesymistycznie $\frac{3}{2}n$ porównań wzorca z tekstem.

2.3 Dowód

W dowodzie poprawności posłużmy się następującą nomenklaturą: na początku mamy $\frac{3}{2}n$ królików, za każde porównanie wzorca z tekstem płacę jednego królika.

Udowodnię, że w każdym stanie mam przynajmniej

$$\frac{3}{2}(n - I - J + 1) + \frac{1}{2}(K + (K - L) + J) - K = \frac{3}{2}(n - I + 1) - J - \frac{1}{2}L$$

królików. To na początku wynosi $\frac{3}{2}n$. W stanie końcowym $n - I - J + 1 \geqslant K - 1$, a $K + (K - L) + J \geqslant K$, czyli liczba królików wynosi co najmniej $\frac{3}{2}(K-1)+\frac{1}{2}K-K = K-\frac{3}{2} > 0$, co by zakończyło dowód. Wystarczy wykazać teraz przez indukcję, że jeśli przed krokiem algorytmu mieliśmy przynajmniej tyle królików, to po też.

Rozpatrzmy więc kolejne przypadki naszego algorytmu:

1. W przypadku, gdy litera wzorca spasowała nam się z literą tekstu, nastąpiło jedynie zwiększenie J o 1. Zapłaciliśmy jednego królika za porównanie, ale liczba wymaganych królików spadła o jeden. Czyli wszystko się zgadza.

2. W przypadku, gdy litera wzorca nie spasowała się z literą tekstu, ale $J = 0$, wówczas zmieniliśmy: I się zwiększyło o 1, zaś L zmniejszyło o jeden. Zapłaciliśmy jednego królika za porównanie, ale liczba wymaganych królików też spadła o jeden ($-\frac{3}{2}$ za zwiększenie I, $+\frac{1}{2}$ za zmniejszenie L).

3. W przypadku, gdy litera wzorca nie spasowała się z literą tekstu, ale $J > 1$, wówczas są dwa podprzypadki:

   - Gdy $P[J + K - 1] \leqslant K - 1$, wówczas zmieniliśmy: $J := 0$, $L := P[J + K - 1]$, $I := I + J + K - 1 - P[J + K - 1]$. Policzmy zmianę liczby wymaganych królików:
     - w związku ze zmianą J: $+J$
     - w związku ze zmianą L: $+\frac{1}{2}(L - P[J + K - 1])$
     - w związku ze zmianą I: $-\frac{3}{2}(J + K - 1 - P[J + K - 1])$

     różnica wynosi

     $$J + \frac{1}{2}(L - P[J + K - 1]) - \frac{3}{2}(J + K - 1 - P[J + K - 1]) =$$

     $$= -\frac{1}{2}(J+K-1-P[J+K-1]) - \frac{1}{2}(K-1-L) - \frac{1}{2}(K-1-P[J+K-1]) \leqslant$$

     $$\leqslant -\frac{1}{2}(J + K - 1 - P[J + K - 1]),$$

     ale jak już zauważyliśmy $J + K - 1 - P[J + K - 1] \geqslant K \geqslant 2$, to rórńica wynośi co najmniej 1, czyli mamy królika do zapłacenia za porównanie.
   - W przeciwnym przypadku mamy $L := K - 1$, $J := P[J + K - 1] - K + 1$ i $I := I + J + K - 1 - P[J + K - 1]$. Policzmy zmianę liczby wymaganych królików:
     - w związku ze zmianą J: $+J + K - 1 - P[J + K - 1]$
     - w związku ze zmianą L: $-\frac{1}{2}((K - 1) - L)$
     - w związku ze zmianą I: $-\frac{3}{2}(J + K - 1 - P[J + K - 1])$

     różnica wynosi $-\frac{1}{2}(K-1-L) - \frac{1}{2}(J+K-1-P[J+K-1])$, więc tak jak poprzednio mamy przynajmniej 1 królika na opłacenie tego porównania.

4. W przypadku, gdy znaleźliśmy wystąpienie wzorca, wykonujemy $K - 1 - L$ porównań sprawdzając, czy litery a zgadzają się w tekście z

wzorcem, po czym wykonujemy pracę taką samą, jak w poprzednim punkcie. Zauważmy, że w pierwszym przypadku:

$$-\frac{1}{2}(J+K-1-P[J+K-1])-\frac{1}{2}(K-1-L)-\frac{1}{2}(K-1-P[J+K-1]) \leqslant$$

$$\leqslant -\frac{1}{2}K-\frac{1}{2}(K-1-L) \leqslant -(K-1-L)$$

Zaś w drugim przypadku:

$$-\frac{1}{2}(K-1-L)-\frac{1}{2}(J+K-1-P[J+K-1]) \leqslant \frac{1}{2}(K-1-L)-\frac{1}{2}K \leqslant -(K-1-L)$$

Czyli w obu przypadkach wymagana liczba królików spada na tyle, że jesteśmy w stanie zapłacić za porównania.

Czyli zawsze mamy wymaganą liczbę królików, na początku wynosi ona $\frac{3}{2}n$, na końcu jest nieujemna, czyli wykonujemy maksymalnie $\frac{3}{2}n$ porównań.