ALGORYTM SUNDAY'A

ALGORYTM SZYBKIEGO WYSZUKIWANIA - QUICK-SEARCH ALGORITHM

OPRACOWANIE: ALICJA SZMYT
352281
pod opieką merytoryczną: dr. inż. PIOTRA BELINGA

UNIWERSYTET ŁÓDZKI
WYDZIAŁ MATEMATYKI I INFORMATYKI

ALGORYTM SUNDAY'A

- Znany również jako QUICK-SEARCH ALGORITHM – algorytm szybkiego wyszukiwania.
- Jedna z bardzo prostych i szybkich odmian algorytmu BOYERA-MOORA opracowana przez Daniela M. Sunday'a w 1990r.
- Używa usprawnionego systemu przesunięcia okna wzorca.
- Łatwy w implementacji.

ALGORYTM SUNDAY'A

- **W algorytmie dane są dwa łańcuchy znaków:**
  - łańcuch znakowy T o długości n - nazywany tekstem,
  - łańcuch znakowy P o długości m - nazywany wzorcem.
  - Łańcuchy nazywane są również słowami.
- **Oba łańcuchy zawierają znaki należące do pewnego skończonego alfabetu:**
  - alfabet oznaczany jest grecką literą Σ (sigma)
  - liczba znaków należących do alfabetu jest nazywana jego rozmiarem i oznaczana przez $|\Sigma|$
- **Działanie na większej ilości znaków**
  - Jeśli algorytm będzie działał na większej ilości znaków niż jeden – rozmiar alfabetu powinien być zwielokrotniony odpowiednią ilość razy.
    - Jeśli działamy na parach znaków – rozmiar alfabetu podnosimy do kwadratu
    - Jeśli działamy na trójkach znaków – rozmiar alfabetu podnosimy do sześcianu itd.

Ogólne działanie algorytmu

- **Łańcuch znaków tekstu zostaje przeszukany i porównany z łańcuchem znaków wzorca.**
- **Nie ma znaczenia od jakiego znaku rozpoczyna się porównywanie(przeszukiwanie) tekstu z wzorcem.**
  - przeszukiwanie od strony lewej do prawej(od pierwszego znaku) wzorca,
  - od prawej do lewej(od ostatniego znaku) wzorca – jak m.in. w algorytmie Boyer-Moore,
  - od dowolnego znaku znajdującego się we wzorcu.
- **Algorytm tworzy pomocniczą tabelę - tabela wstępnego przetwarzania, która informuje nas:**
  - czy znak znajdujący się w porównywanym tekście znajduje się we wzorcu,
  - jaka jest pozycja ostatniego wystąpienia tego znaku we wzorcu.

Ogólne działanie algorytmu

- **Jeśli algorytm nie napotka niedopasowania znaku w przeszukiwanym tekście ze znakiem z wzorca:**
  - Sprawdza kolejne znaki, aż:
    - dopasuje cały wzorzec (tu można zakończyć działanie algorytmu lub kontynuować wyszukiwanie kolejnych dopasowań),
    - napotka niedopasowanie,
    - w przypadku sprawdzenia już wszystkich znaków zakończy działanie algorytmu.
- **Jeśli napotka niedopasowanie znaku w przeszukiwanym tekście ze znakiem z wzorca:**
  - Przerywa porównywanie i sprawdza następny znak po oknie wyszukiwania:
    - jeśli ten znak również się nie zgadza (nie występuje we wzorcu) to okno zostaje przesunięte o długość wzorca +1(ilość znaków okna + kolejny sprawdzony niepasujący znak);
    - jeśli ten znak się zgadza, program mając informację z tabeli pomocniczej przesuwa wzorzec do ostatniego pasującego wystąpienia znaku we wzorcu i kontynuuje przeszukiwanie.
- **Jeśli porówna cały tekst z wzorcem - działanie algorytmu zostaje zakończone.**

Definicja zmiennych

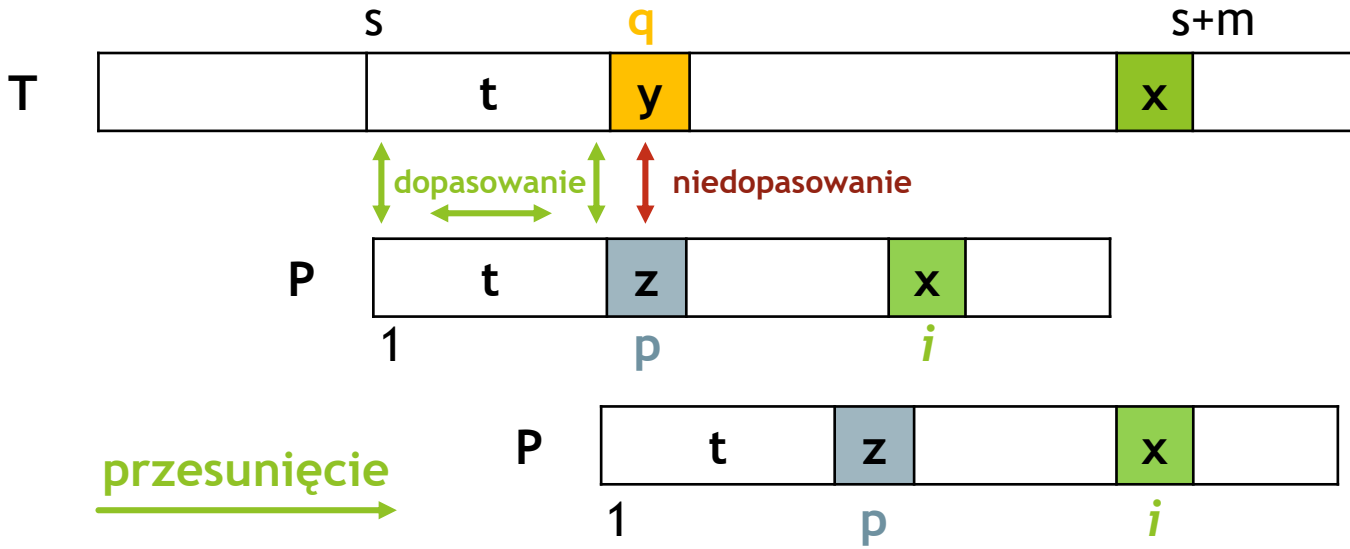
- **T**: łańcuch znakowy – tekst.
- **P**: łańcuch znakowy – wzorzec.
- T_s: pierwszy znak łańcucha T jest zgodny z wzorcem P.
- P_l: pierwszy znak wzorca P wyrównuje się do łańcucha T.
- T_j: znak j-tej pozycji łańcucha T.
- P_i: znak i-tej pozycji wzorca P.
- P_f: ostatni znak wzorca P.
- **n**: długość łańcucha T.
- **m**: długość łańcucha P.

Definicja problemu

- **Wejście:**
  - łańcuch znakowy T(tekst) o długości n,
  - łańcuch znakowy P(wzorzec) o długości m.
- **Wyjście:**
  - wszystkie wystąpienia P(wzorca) w T(tekście).

Reguła szybkiego wyszukiwania

- Załóżmy, że P_1 jest wyrównany do T_s, teraz wykonujemy porównanie między tekstem T i wzorcem P od lewej do prawej.
- Załóżmy, że pierwsze niedopasowanie występuje przy porównywaniu T_q z P_p. Ponieważ $T_q \neq P_p$, przesuwamy wzorzec P w prawo tak, że największa pozycja i po prawej P_i równa jest T_{s+m}
  Możemy przesunąć wzorzec o (m-i) pozycji w prawo.



Szybkie wyszukiwanie - Tabela wstępnego przetwarzania (tabela pomocnicza)

- **Jedyne co chcemy zrobić to zbudować tabelę w następujący sposób:**
  - Niech x będzie znakiem w tekście.
  - Jeśli x istnieje w P, zapisujemy pozycję ostatniego występującego x w P, indeksując pozycje od końca.
  - Jeśli x nie istnieje w P, zapisujemy ją jako m + 1.
- **Przykład:**

Wzorzec i pozycja znaków

	7	6	5	4	3	2	1
P =	B	C	A	A	C	C	A

Tabela pomocnicza

x	A	B	C	W
pos['x']	1	7	2	8

czyli: pos['A'] = 1; pos['B'] =7;

pos['C'] = 2; pos['W'] = m+1=7+1=8;

Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

A	B	A	B	B	C	A	A	C	C	A	W	A	C	A	C	A	W	C	C	A

B	C	A	A	C	C	A

Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

ROZPOCZĘTO PRZESZUKIWANIE(PORÓWNYWANIE)

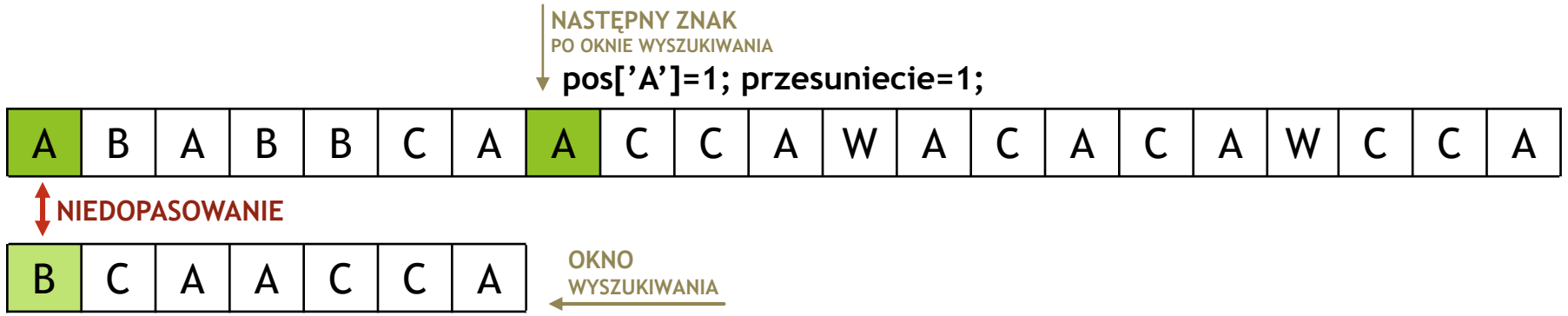
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

A	B	A	B	B	C	A	A	C	C	A	W	A	C	A	C	A	W	C	C	A

B	C	A	A	C	C	A

pos['A'] = 1; przesunięcie = 1;

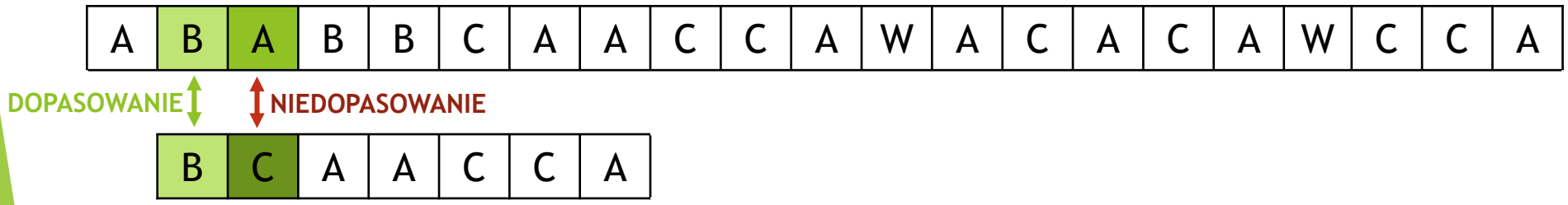
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



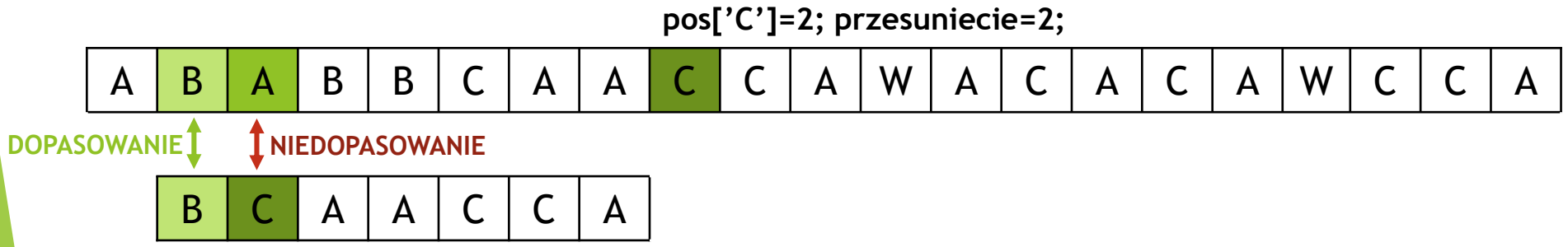
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

A	B	A	B	B	C	A	A	C	C	A	W	A	C	A	C	A	W	C	C	A

B	C	A	A	C	C	A

pos['C'] = 2; przesunięcie = 2;

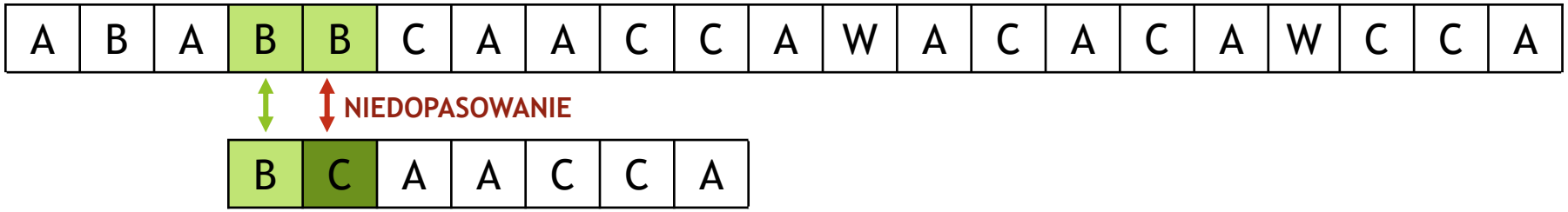
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



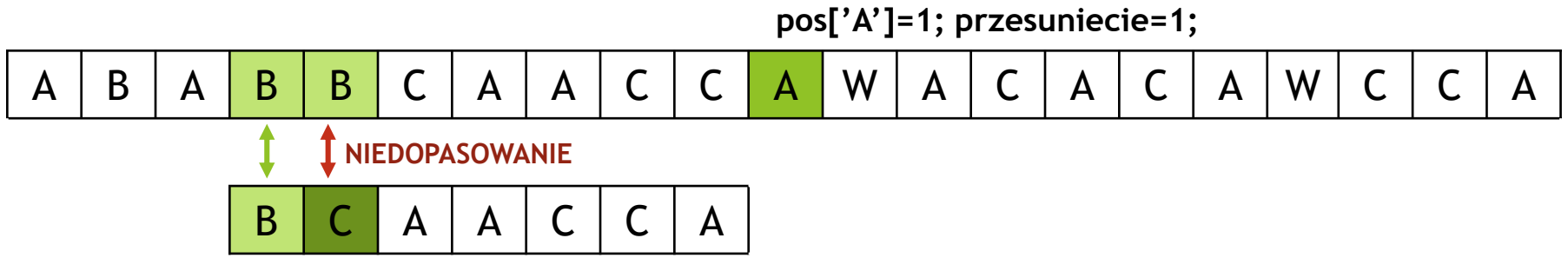
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

pos['A']=1; przesuniecie=1;

A	B	A	B	B	C	A	A	C	C	A	W	A	C	A	C	A	W	C	C	A

B	C	A	A	C	C	A

pos['C'] = 2; przesunięcie = 1;

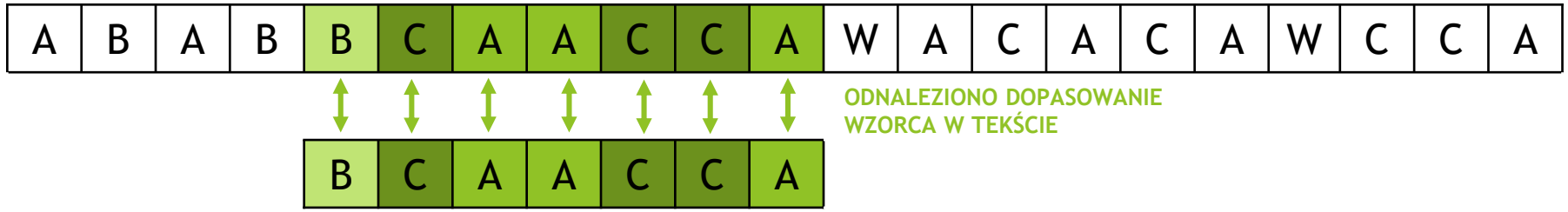
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



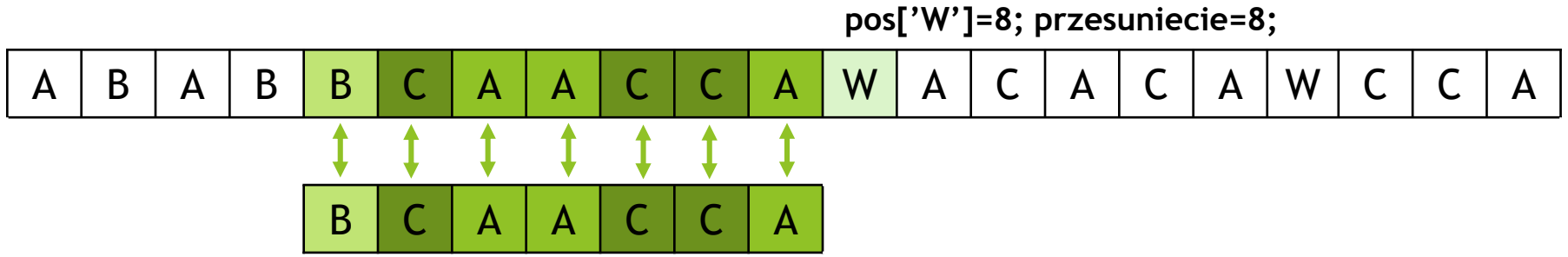
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



ODNALEZIONO DOPASOWANIE CAŁEGO WZORCA W TEKŚCIE, ABY DOWIEDZIEĆ SIĘ CZY MOŻNA ZNALEŹĆ KOLEJNE DOPASOWANIA MUSIMY KONTYNUOWAĆ SPRAWDZANIE

Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

A	B	A	B	B	C	A	A	C	C	A	W	A	C	A	C	A	W	C	C	A

B	C	A	A	C	C	A

pos['W'] = 8; przesunięcie = 8;

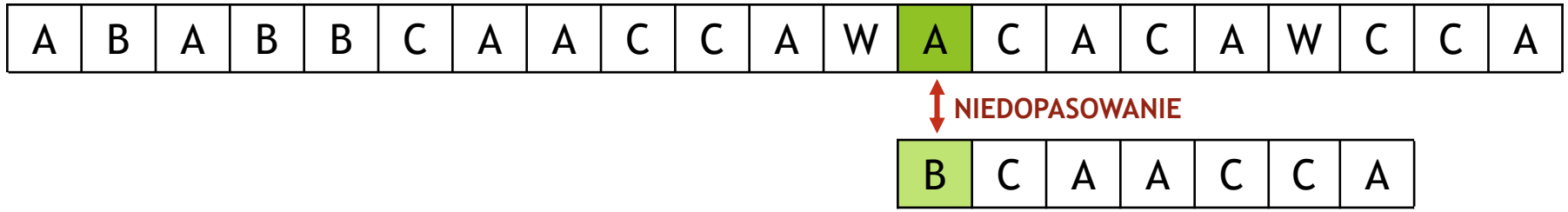
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



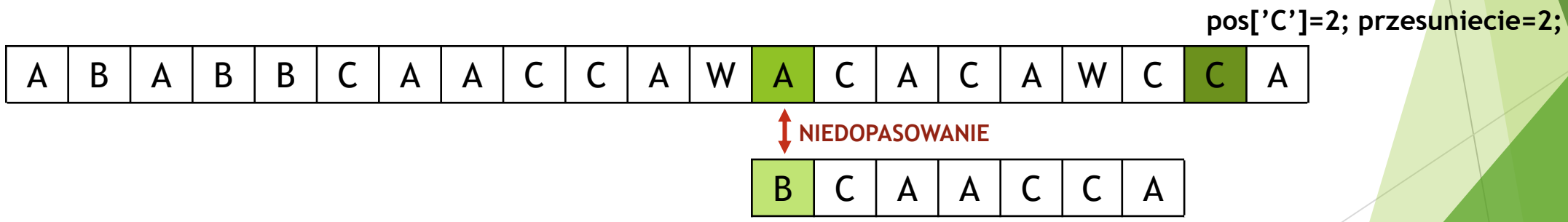
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8

A	B	A	B	B	C	A	A	C	C	A	W	A	C	A	C	A	W	C	C	A

B	C	A	A	C	C	A

pos['C'] = 2; przesunięcie = 2;

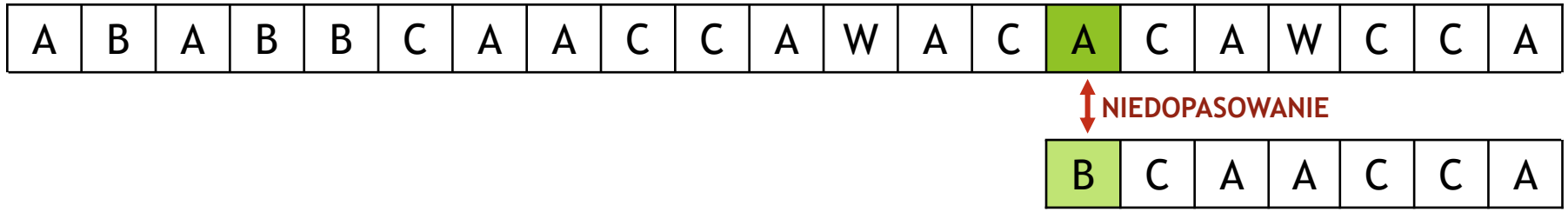
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



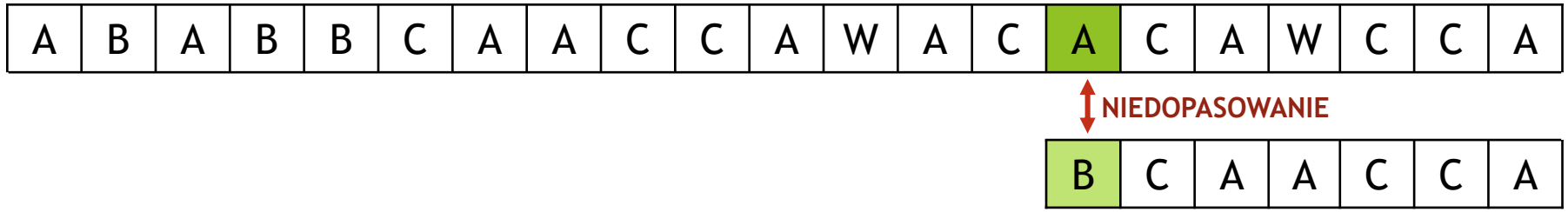
Przykład

znajdźmy wszystkie wystąpienia wzorca w tekście

- **Łańcuch tekstu**
  - T = ABABBCAACCAWACACAWCCA
- **Łańcuch wzorca**
  - P = BCAACCA

- **Tabela pomocnicza**

x	A	B	C	W
pos['x']	1	7	2	8



WYSTĄPIŁO NIEDOPASOWANIE I NIE MA NASTĘPNEGO ZNAKU PO OKNIE WYSZUKIWANIA - NIE MOŻNA PRZESUNĄĆ OKNA.
PRZESZUKANY ZOSTAŁ CAŁY TEKST - ALGORYTM KOŃCZY DZIAŁANIE
ZNALEZIONO WSZYSTKIE WYSTĄPIENIA WZORCA W TEKŚCIE; ILOŚĆ WYSTĄPIEŃ = 1;

Złożoność obliczeniowa

- **Całego algorytmu**
  - **Złożoność obliczeniowa pamięciowa pesymistyczna $O(|\Sigma|)$**
- **Faza wstępna**
  - **Złożoność obliczeniowa czasowa pesymistyczna $O(m + |\Sigma|)$**
- **Faza wyszukiwania**
  - **Złożoność obliczeniowa czasowa pesymistyczna $O(mn)$**
  - **Złożoność obliczeniowa czasowa optymistyczna $O(n/m)$**

Bibliografia

- **SUNDAY D.M., 1990,** *A very fast substring search algorithm, Communications of the ACM . 33(8):132-142.*
- **H.W. Lang,** *Sunday algorithm* http://www.inf.fh-flensburg.de/lang/algorithmen/pattern/sundayen.htm
- **C. Charras, T. Lecroq,** *EXACT STRING MATCHING ALGORITHMS - Animation in Java, Quick Search algorithm* http://www-igm.univ-mlv.fr/~lecroq/string/node19.html#SECTION00190
- **Jacek Widuch,** *Analiza porównawcza algorytmów wyszukiwania wzorca w tekście* https://pdfs.semanticscholar.org/024a/623cb057814068e3716fec4605c8e8e77263.pdf