AISDI – laboratorium z wyszukiwania wzorca w tekście

Jakub Stępniak
Przemysław Wyziński
Lab 102, grupa 4

Cel ćwiczenia

Celem ćwiczenia była implementacja i testowanie 3 algorytmów wyszukiwania wzorca w tekście:

- algorytm N (naiwny)
- algorytm KMP (Knutha-Morrisa-Pratta)
- algorytm KR (Krapa-Rabina)

Sprawdzenie poprawności implementacji

Pierwszym zestawem eksperymentów było przetestowanie poprawności działania algorytmów dla alfabetu 2-literowego. Wzorzec był losowym ciągiem znaków z alfabetu o długości pomiędzy 3 a 5, a badany tekst losowym ciągiem znaków z alfabetu o długości 100. Dodatkowo wyświetlany jest wynik pochodzący z wbudowanej funkcji pythona find() służący do sprawdzenia poprawności działania algorytmów.

Przykładowe wyniki eksperymentów:

Eksperyment 1:

```
Text: baaabbbbbbaababbbbbbbbaabbbabbbabaababababaabaababababbabaaaabbbaaaaabbbaaaabababbbbbbbaaabaaaabbbabbaaaaab
Searching pattern: abb
Build-in find method: [3, 13, 23, 27, 48, 55, 63, 72, 87, 91]
Algorithm N found answer: [3, 13, 23, 27, 48, 55, 63, 72, 87, 91]
Algorithm KMP found answer: [3, 13, 23, 27, 48, 55, 63, 72, 87, 91]
Algorithm KR found answer: [3, 13, 23, 27, 48, 55, 63, 72, 87, 91]
```

Eksperyment 2:

```
Text: bbabbaaaabaababbabbababababababbbbbbbbababbaabaabbbaaababbababababaaaaaaabbabaababaabaaaaaaaabaaabbbbbb
Searching pattern: aabaa
Build-in find method: [7, 42, 79, 88]
Algorithm N found answer: [7, 42, 79, 88]
Algorithm KMP found answer: [7, 42, 79, 88]
Algorithm KR found answer: [7, 42, 79, 88]
```

Eksperyment 3:

```
Text: aaaabbbbabbbabbbaababaaaabaababababbbaaaaaaabbbbbbbabbabaaaabaabaaaabbbbbbababababaabbbbbbbaaaaaabababbba
Searching pattern: abaa
Build-in find method: [19, 24, 51, 56, 59, 75, 92]
Algorithm N found answer: [19, 24, 51, 56, 59, 75, 92]
Algorithm KMP found answer: [19, 24, 51, 56, 59, 75, 92]
Algorithm KR found answer: [19, 24, 51, 56, 59, 75, 92]
```

Wnioski:

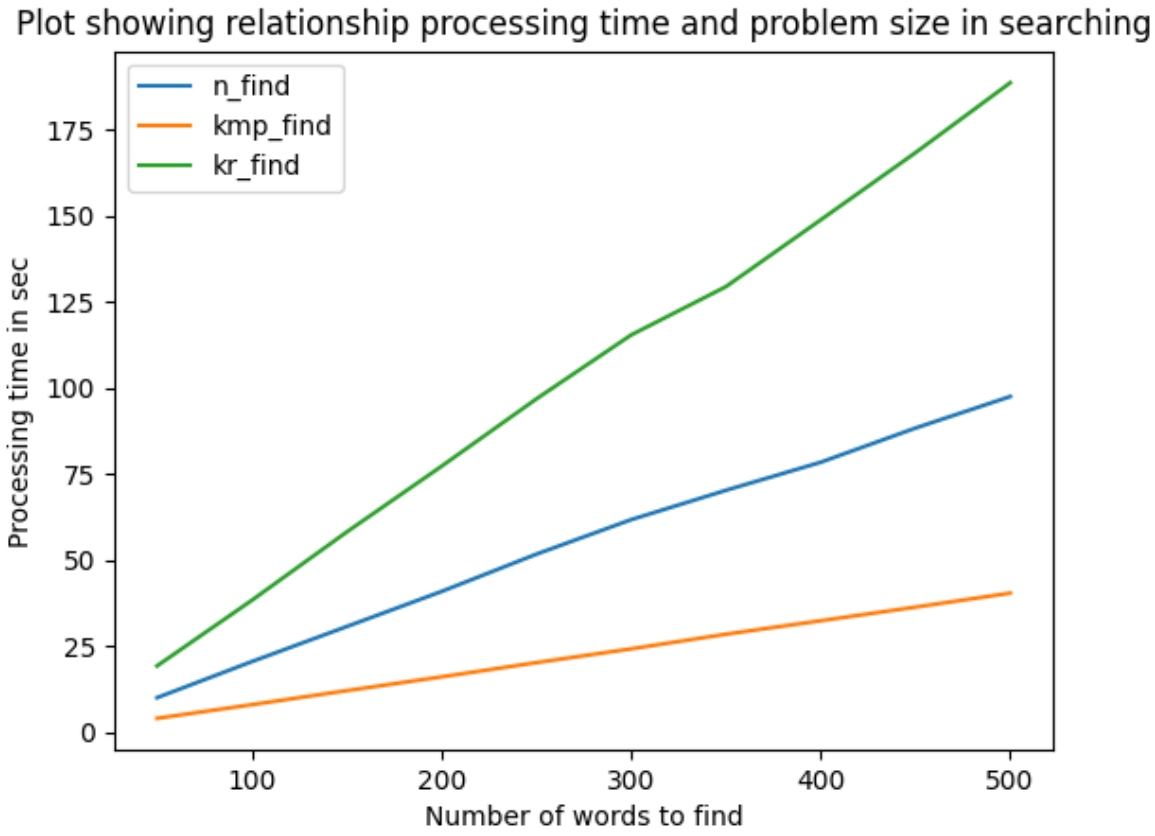
Wyniki wszystkich algorytmów są identyczne – zaimplementowane algorytmy działają poprawnie.

Porównanie szybkości algorytmów

Do porównania szybkości algorytmów użyliśmy tekstowego pliku z treścią Pana Tadeusza znanego polskiego wieszcza Adama Mickiewicza. W treści tego utworu znajdują się różne znaki specjalne spoza alfabetu polskiego, dlatego alfabet wykorzystywany do wyszukiwania został wygenerowany w pliku pan_tadeusz_letters.py.

Do testów jednostkowych wykorzystywany jest alfabet polski, dlatego dla testów Pana Tadeusza trzeba odkomentować odpowiedni alfabet w plikach kmp.py oraz kr.py.

Rezultaty eksperymentu:



Wnioski końcowe:

Najszybszy okazał się algorytm KMP (Knutha-Morrisa-Pratta). Jego teoretyczna złożoność $O(M+N)$ pokrywa się ze złożonością na wykresie (funkcja liniowa). Zastosowanie deterministycznego automatu skończonego powoduje, że osiąga najlepsze wyniki kosztem większego wykorzystania pamięci.

Najwolniejszy okazał się algorytm KR (Krapa-Rabina). Chociaż jego teoretyczna złożoność czasowa to $O(N)$, to okazał się gorszy od algorytmu naiwnego. Powodów może być kilka: Funkcja haszująca może dawać takie same wyniki dla różnych danych, co sprawia, że potrzebna jest funkcja check(), która wydłuża wyszukiwanie. Dodatkowo algorytm wymaga dużej ilości obliczeń (w szczególności dla długich słów) do poprawnego działania. Te dodatkowe wymagania wydłużają wyszukiwanie wzorca i powodują tak słabe rezultaty na tle innych algorytmów.

Algorytm naiwny okazał się nie być najlepszy, ale też nie być najgorszy. Można było się tego spodziewać z uwagi na jego prostotę.