Algorytmy 2

Sprawozdanie 2 - **Porównanie algorytmów KMP oraz naiwnego wyszukiwania**

Grupa:

Marek Wala 252841

Rok i semestr studiów: **3 rok, 5 semestr**

Kierunek: **Informatyka-przemysłowa**

Rok akademicki**: 2022/2023**

Semestr: **zimowy**

Data oddania: 23**.01.2023**

Prowadzący: **Dr Inż. Beata Laszkiewicz**

1.Specyfikacja sprzętu

1. Procesor: i7 wersja z 2017
   Dysk SSD PCI express
   Środowisko Xcode 11

2.Teoretyczne podstawy

1.Opis algorytmu naiveSearch

Prostym sposobem zlokalizowania określonej pozycji na liście rzeczy jest naiwny algorytm wyszukiwania. Działa na zasadzie rozpoczynania od pierwszej pozycji na liście i porównywania jej z poszukiwaną pozycją. Algorytm powtarza procedurę i przechodzi do następnej pozycji na liście, jeśli pozycja nie może zostać wykryta. Do momentu odnalezienia poszukiwanej pozycji lub zakończenia listy, proces ten jest realizowany. Słaba złożoność czasowa algorytmu naiwnego wyszukiwania jest jedną z jego największych wad. Algorytm zajmuje więcej czasu, aby znaleźć żądany element wraz ze wzrostem rozmiaru listy. Wykorzystując bardziej wyrafinowane metody, takie jak wyszukiwanie binarne, naiwny algorytm wyszukiwania można ulepszyć przykładowo algorytm tablic haszujących. Naiwna technika wyszukiwania ma złożoność czasową O(n), co oznacza, że wraz ze wzrostem liczby elementów na liście wymagany czas rośnie liniowo. Algorytm wyszukiwania naiwnego ma stałe zapotrzebowanie na pamięć ze względu na złożoność przestrzeni O(1). Chociaż naiwna technika wyszukiwania jest prosta w użyciu i zrozumiałe, ogromne zbiory danych nie są dla niej odpowiednie. Algorytm ten, określany jest jako przeszukiwanie liniowe lub przeszukiwanie sekwencyjne. W przypadku bardziej złożonych algorytmów wyszukiwania, takich jak algorytm A*, podstawę stanowi naiwny algorytm wyszukiwania.

2. Opis algorytmu KMP

Aby zlokalizować wygląd wzorca wewnątrz tekstu, algorytm Knutha-Morrisa-Pratta (KMP) dopasowuje ciągi znaków. Jest to wydajny algorytm, który eliminując bezsensowne porównania, w przeciwieństwie do naiwnego algorytmu wyszukiwania. Wzorzec jest wstępnie przetwarzany przez algorytm KMP w celu dostarczenia częściowej tabeli dopasowań i funkcji błędów. Jeśli występuje niezgodność, funkcja niepowodzenia jest wykorzystywana do zidentyfikowania następnego punktu w tekście do dopasowania. Czasowa złożoność algorytmu KMP wynosi O(n + m), gdzie n to długość tekstu, a m to długość wzorca. Algorytm KMP ma złożoność przestrzenną O(m), gdzie m jest długością wzorca. KMP jest szybszym algorytmem niż naiveSearch. Można ten algorytm również wykonać w sposób iteracyjny lub rekurencyjny. Jest często używany do rozwiązywania problemów z dopasowywaniem ciągów i służy jako kluczowy element konstrukcyjny wielu różnych metod dopasowywania. Algorytm KMP jest często używany w kompilatorach, edytorach tekstu i systemach przetwarzania

tekstu. Algorytm KMP jest dobrze ilustruje, jak przetwarzanie wstępne może znacznie poprawić wydajność algorytmu.

3.Wykonanie

Kod implementacji w pliku rar projekt2.rar.

Z wykorzystaniem biblioteki chrono:: zostały porównane ilość znalezionych fraz oraz czas wykonywania poszczególnych funkcji. Wyniki zostały zebrane w tabeli.

Napotkane zostały pewne problemy z implementacją danego algorytmu:
- problem z wczytywaniem pliku przez środowisko Xcode
- funkcja pierwotnie zimpementowana nie liczyła poprawnie ilości wystąpień danej frazy przy naiwnym algorytmie wyszukiwania
- funkcja wyboru wczytywania wzorca zwracała exit code -19421213
- do testów naiveSearch musiałem niestety skorzystać z internetowego rozwiązania ponieważ mój kod nie był skuteczny

Tabela 1. średnie czasy wykonywania się algorytmów

	naiwny	Kmp	naiwny	Kmp	naiwny	Kmp
	lorem		ipsum		dolor	
	33	27	27	23	35	33
	31	34	28	36	23	21
	37	37	27	24	35	35
	24	22	27	24	37	36
	24	31	36	24	35	36
	29,8	30,2	29	26,2	33	32,2

Algorytm:	Naiwny			KMP		
Wzorzec:	lorem	ipsum	dolor	lorem	ipsum	dolor
Ilość znalezione	3	6	7	3	6	7
Średni czas dzia	29,8	29	33	30,2	26,2	32,2

Tabela 2. porównanie czasów wykonywania dla poszczególnych fraz

4.Podsumowanie

Zarówno naiwny algorytm wyszukiwania, jak i algorytm KMP są używane do znajdowania występowania wzorca w tekście. Naiwny algorytm wyszukiwania porównuje każdy znak tekstu ze wzorcem jeden po drugim, podczas gdy algorytm KMP wstępnie przetwarza wzorzec w celu utworzenia funkcji niepowodzenia, znanej również jako tablica częściowych dopasowań. Algorytm wyszukiwania naiwnego ma złożoność czasową O(n), gdzie n to długość tekstu, podczas gdy algorytm KMP ma złożoność czasową O(n + m), gdzie n to długość tekstu, a m to długość wzoru.

Algorytm wyszukiwania naiwnego ma złożoność pamięciową O(1), podczas gdy algorytm KMP ma złożoność pamięciową O(m), gdzie m jest długością wzorca.

Przewagi algorytmu KMP:
- jest bardziej wydajny niż algorytm wyszukiwania naiwnego, zwłaszcza gdy wzorzec jest dłuższy i wielokrotnie występuje w tekście.
- gwarantuje znalezienie wszystkich wystąpień wzorca w tekście, podczas gdy naiwny algorytm wyszukiwania może znaleźć tylko pierwsze wystąpienie.
- gwarantuje znalezienie wszystkich wystąpień wzorca w tekście, podczas gdy naiwny algorytm wyszukiwania może znaleźć tylko pierwsze wystąpienie.
- lepiej nadaje się do dużych zbiorów danych lub zbiorów danych, w których wzorzec można znaleźć wiele razy w tekście, podczas gdy naiwny algorytm wyszukiwania najlepiej nadaje się do małych zbiorów danych.
- jest szybszy od naiwnego algorytmu wyszukiwania w przypadku powtarzających się wzorców w tekście.
- jest mniej podatny na błędy niż naiwny algorytm wyszukiwania.
- jest bardziej odpowiedni dla aplikacji wielowątkowych, ponieważ wykorzystuje funkcję niepowodzenia, która zmniejsza liczbę porównań.

Algorytm KMP lepiej nadaje się do dużych zbiorów danych lub zbiorów danych, w których wzorzec można znaleźć wiele razy w tekście, podczas gdy naiwny algorytm wyszukiwania najlepiej nadaje się do małych zbiorów danych. Algorytm KMP jest klasycznym przykładem tego, jak wstępne przetwarzanie może prowadzić do znacznej poprawy wydajności algorytmów, podczas gdy naiwny algorytm wyszukiwania jest prostą metodą brute force. Algorytm KMP jest bardziej wydajny i odpowiedni dla dużych zbiorów danych i powtarzalnych wzorców, podczas gdy naiwny algorytm wyszukiwania jest prosty i łatwy do wdrożenia, odpowiedni dla małych zbiorów danych. Algorytm KMP jest szeroko stosowany w edytorach tekstu, systemach przetwarzania tekstu i kompilatorach, podczas gdy naiwny algorytm wyszukiwania jest używany jako element budulcowy dla bardziej zaawansowanych algorytmów.