

ICE3020 알고리즘설계

<과제 1>

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2024년 05 월 05 일

학부 정보통신공학

학년 3

성명 이상혁

학번 12201928



1. 개요

* 이번 과제에서는 패턴 매칭의 주요 알고리즘인 kmp, 보이어 무어, 라빈 카프 알고리즘을 특수기호도 검색할 수 있도록 만들고, 검색되는 과정을 보이는 것이다.  
  이후 각각의 시간복잡도를 계산하고 테스트하는 것까지의 구현을 완료하였다.  
  대부분의 코드는 강의노트를 참고하여 작성하였으며 c++로 구현하였다.

1. 상세 설계 내용

* KMP

텍스트, 스크린샷, 디스플레이, 번호이(가) 표시된 사진

자동 생성된 설명

**InitNext** - 이 함수는 패턴 문자열을 입력받아 테이블을 초기화한다. 이 함수는 각 패턴 위치에서의 일치하지 않는 경우 다음으로 어디로 이동해야 하는지를 미리 계산하여 kmpnext 배열에 저장한다.

**KMP:** 주어진 텍스트 문자열에서 패턴을 검색합니다. 이전에 계산한 테이블을 사용하여 텍스트와 패턴을 한 번에 한 문자씩 비교하면서, 일치하지 않는 경우에는 kmpnext 배열을 참조하여 텍스트 문자열의 검색 위치를 조정한다.

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

**Skip** – 패턴 내에 있는 각 문자마다 텍스트에서 건너뛸 위치를 미리 계산한 스킵 테이블을 작성하고 사용한다. 일치하지 않는 문자가 발견되었을 때 패턴을 효율적으로 이동할 수 있다.

**BoyerMoore** - 이 함수는 패턴 문자열을 뒤에서부터 비교한다. 일치하지 않는 문자가 발견되었을 때 skip 배열을 사용하여서 패턴을 더욱 많이 이동할 수 있게 만들어준다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

우선 첫번째 전처리 단계에서 해시 값을 계산한다. 패턴문자열과 텍스트 첫 부분 문자열에 대한 해시 값을 계산한다.

이후 문자열 검색을 진행하는데 해시 값이 일치하는 경우에만 실제 문자열의 비교를 진행하도록 코드를 작성하였다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

이러한 알고리즘을 테스트하기 위하여 정보를 저장하는 구조체를 작성하였다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

* 텍스트 파일 읽기 "RFC2616\_modified.txt" 파일을 열고, 한 줄씩 읽어온다.  
  패턴 검색: 파일에서 한 줄씩 읽어오면서, 각 줄에서 주어진 패턴을 검색합니다. 패턴 검색은 "KMP", "BoyerMoore", "RabinKarp" 세 가지 알고리즘을 사용하여 수행한다.

패턴 발견 위치 기록: 만약 패턴이 특정 줄에서 발견된다면, 해당 위치를 기록하고, 이 정보를 나중에 요약하여 출력한다. 이후 각 패턴이 어느 알고리즘을 사용하여 몇 번 발견되었는지를 기록한후 알고리즘에 대해 발견된 패턴을 요약하여 출력하도록 하였다.

즉 주어진 텍스트 파일에서 패턴을 검색하고 발견된 패턴의 위치 및 횟수를 요약하여 출력하는 작업을 수행하도록 하였다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* 우선 각각의 worst best를 비교하기 위하여 어떤 경우에 알고리즘의 시간이 증가하는지 판단해보았다.
* **KMP**

BEST: 텍스트 문자열에서 패턴이 전혀 발견되지 않거나, 텍스트 문자열과 패턴 문자열이 전혀 일치하지 않을 때 KMP 알고리즘은 효율적이다. 이 경우, 알고리즘은 한 번만 텍스트를 순회하며, 각 문자를 단 한 번씩만 확인하여 시간 복잡도는 O(n)이다. 텍스트가 ABCDEF이고 패턴이 XYZ 인경우

Worst: 텍스트와 패턴이 반복적인 문자로 구성되어 있을 때 KMP 알고리즘은 패턴의 접두사와 접미사가 일치하는 계산을 여러 번 수행해야 하여. 하지만 그래도 시간 복잡도는 동일하다. (예: 텍스트 "AAAAAA...A", 패턴 "AAA...A")

* **Boyer-Moore**BEST: 패턴의 마지막 문자가 존재하지 않는 경우이다. 이때 존재하지않으면 패턴의 길이만큼 건너뛰게 되어 검색이 종료된다. O(n/m) 텍스트가 ABCDEF이고 패턴이 XYZ인 경우

Worst: 텍스트, 패턴이 모두 반복적일 때 패턴이 끝에만 존재하는 경우이다. O(nm) (예: 텍스트 "AAAAAA...A", 패턴 "AAA")

* **Rabin-Karp**BEST: O(n+m)의 시간 복잡도를 가진다. 여기서 n은 텍스트의 길이, m은 패턴의 길이이며. 이는 해시 함수를 통해 패턴의 해시값과 텍스트의 각 부분의 해시값을 빠르게 비교할 수 있다. 텍스트가 ABCDEF이고 패턴이 XYZ 인경우

Worst: O(nm)의 시간 복잡도를 가집니다. 이는 해시 충돌이 빈번히 발생할 때, 즉, 많은 텍스트 부분이 패턴과 동일한 해시값을 가질 때마다 실제 문자열 비교가 필요할 때 발생한다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

따라서 위와 같은 경우를 실제로 생성을 한 이후 시간이 얼마나 걸리는지 측정하는 코드를 작성하였다. 각 케이스별로 완벽하게는 구현할만한 텍스트 파일은 생성하지 못하였으나, 비슷하게 만큼 구현하고자 노력하였다.

이후에는 테스트를 통해 시간의 차이를 확인할 수 있었다.

1. 실행 화면

문제 1,2

텍스트, 스크린샷, 컴퓨터, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

각 알고리즘 별로 어디 라인에서 시작하는 idx를 가지고있는지 우선적으로 측정하였다.

이후에 각 패턴에 따른 개수를 알고리즘별로 카운팅하여 결과를 출력하도록 작성하였다.

텍스트, 스크린샷, 흑백, 폰트이(가) 표시된 사진

자동 생성된 설명

문제 3

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

"patternpattern"이 나타나지않는 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"와 같은 텍스트에 대해서는 Boyer-Moore와 KMP 알고리즘이 Rabin-Karp 알고리즘보다 빠를 것으로 예상하였다.

예상과 동일한 결과가 출력됨을 확인하였다.

"patternpattern"이 자주 나타나는 "patternpatternpatternpatternpatternpatternpattern"과 같은 텍스트에 대해서는 모든 알고리즘이 비슷한 성능을 보일 것으로 예상하였다. 예상외로 보이어무어가 조금 더 좋은 결과를 보였는데 약간의 오차로 생각되었다

"XYZ"와 같이 짧은 패턴이 있는 경우, Boyer-Moore와 KMP 알고리즘이 더 빠를 것으로 예상하였다. 이부분도 예상한 결과와 동일함을 확인하였다.

1. 결론

* 문제 3에서 예상한 결과와 조금 다르게 출력되는 경우가 있었는데, 아무래도 직접 텍스트 파일을 생성하여 테스트한 결과라 오차가 좀 발생한 것으로 예측된다.
* 주어진 텍스트 파일에서의 패턴 검색에 대한 세 가지 알고리즘(KMP, Boyer-Moore, Rabin-Karp)의 성능을 비교하였다. 이를 통해 각 알고리즘의 장단점과 적합성을 평가할 수 있었다.
* KMP 알고리즘은 패턴이 자주 나타나는 경우와 같이 반복되는 패턴이 많은 텍스트에서 뛰어난 성능을 보였다. 이는 KMP 알고리즘이 일치하지 않는 문자열을 사용하여 검색 위치를 스킵하기 때문이다. 따라서, 대규모 텍스트에서 반복되는 패턴을 효율적으로 찾을 때 유용하다.
* Boyer-Moore 알고리즘은 오른쪽 끝부터 패턴을 비교하며, 일치하지 않는 문자열을 사용하여 검색 위치를 스킵하는 특성 때문에 특정한 상황에서 유용하다. 하지만, 특히 패턴이 텍스트 내에서 자주 나타나거나 반복되는 경우에는 성능이 감소할 수 있다.
* Rabin-Karp 알고리즘은 해시 값을 사용하여 패턴을 찾는 특징을 가지고 있다. 따라서, 패턴이 텍스트 내에서 드물게 나타나는 경우에도 효율적으로 동작한다. 그러나, 해시 충돌이 발생할 가능성이 있으며, 이는 정확성과 성능에 영향을 미칠 수 있다.
* 결론적으로, 각 알고리즘은 다양한 상황에서 강점을 보인다. KMP 알고리즘은 반복되는 패턴을 효율적으로 처리할 때 가장 효과적이다. Boyer-Moore 알고리즘은 특정한 패턴이 텍스트 내에서 자주 나타날 때 유용하며, Rabin-Karp 알고리즘은 패턴이 드물게 나타날 때 효과적이다. 선택하는 알고리즘은 주어진 상황과 데이터의 특성에 따라 달라질 것이다.