REPORT

lab 09

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 과목명 : | | | 컴퓨터알고리즘과실습 | | | | | | | | |
| 담당교수 : | | | 주종화 | | | | | | | 교수님 | |
| 제출일 : | 2021 | | | 년 | | 05 | 월 | | 13 | | 일 |
| 공과 | | 대학 | | | 컴퓨터공학 | | | 과 | | | |
| 학번 : | | 2016112154 | | | 이름: | | | 정동구 | | | |

문제1

-결과

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

각각 15 30 45 길이의 문자열을 10000의 길이를 가진 문자열에서 검색한 결과이다. 일반적으로 랜덤한 문자열을 검색하였을 경우 검색이 되는 경우가 거의 없어, 생성한 문자열 안에서 15 30 45 길이만큼을 골라 내어 검색하였다.

-분석

문자열 검색 완료까지의 시간을 측정하여 그래프화 하였을 경우

위 그래프와 같은 양상을 보인다. x축은 원본 문자열(plain text)의 길이이고 y축은 걸린 시간, 꺾은 선 그래프 선의 색은 각각 파랑 : 15길이 패턴 주황 :30 길이 패턴 회색: 45길이 패턴을 나타낸다. 패턴의 길이에 관계없이 원본 문자열이 길어질수록 시간이 오래걸린다. 패턴의 길이가 길수록 무조건 탐색시간이 길어야 할 것 같았지만 의외로 크게 차이가 없거나 30길이 문자열을 탐색하는 것이 더 오래걸리는 등의 결과도 나왔다.

|  |  |  |  |
| --- | --- | --- | --- |
| brute | 15 | 30 | 45 |
| 10000 | 336 | 336 | 385 |
| 100000 | 833 | 784 | 794 |
| 1000000 | 6416 | 5891 | 5426 |
| 10000000 | 42450 | 48669 | 47434 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| RK | 15 | 30 | 45 |
| 10000 | 259 | 265 | 7 |
| 100000 | 930 | 941 | 307 |
| 1000000 | 8013 | 8546 | 7303 |
| 10000000 | 76291 | 68801 | 71041 |
|  |  |  |  |
| Kmp | 15 | 30 | 45 |
| 10000 | 230 | 340 | 283 |
| 100000 | 699 | 821 | 796 |
| 1000000 | 5674 | 5780 | 5644 |
| 10000000 | 47037 | 56033 | 62519 |

탐색시간의 결과를 표로 나타내면 위와 같다. 의외였던 부분은 라빈카프 알고리즘이 무조건 탐색시간이 가장 짧을 것이라 생각하였는데, 브루트 포스가 더 짧게 나오는 경우도 있었고, 라빈카프 알고리즘이 셋 중 가장 긴 경우도 있었다. 알고리즘 상 비교 횟수는 라빈카프 알고리즘이 가장 작으나 전체 걸리는 시간도 짧지는 않게 나왔다. 위와 같은 결과가 나온 이유로 예상 해 볼 수 있는 것은, 라빈카프 알고리즘이 해싱 기반으로 하는 문자열 탐색 알고리즘 이기 때문에 해시 함수가 어떠냐에 따라 비교횟수와 무관하게 탐색까지의 시간이 달라지는 것으로 보인다. 해당 알고리즘 구현에서 해시 함수를 단순한 모듈러 연산을 채택했기 때문에 모듈러 연산에 사용되는 숫자의 영향도 있을 것으로 보인다. 해시 함수를 모듈러 연산보다 효율이 좋고 성능이 좋은 함수로 택할 경우 라빈카프 알고리즘의 탐색시간은 더욱 많이 줄어들 것으로 보인다.

-소스코드

브루트 포스

텍스트, 사람, 가장이(가) 표시된 사진

자동 생성된 설명

Pattern의 처음부터 끝까지 비교하고 틀릴경우 반복문 탈출한다. 마지막에 패턴 길이와 비교횟수가 같을경우 문자열이 일치하는것으로 본다.

KMP

s텍스트이(가) 표시된 사진

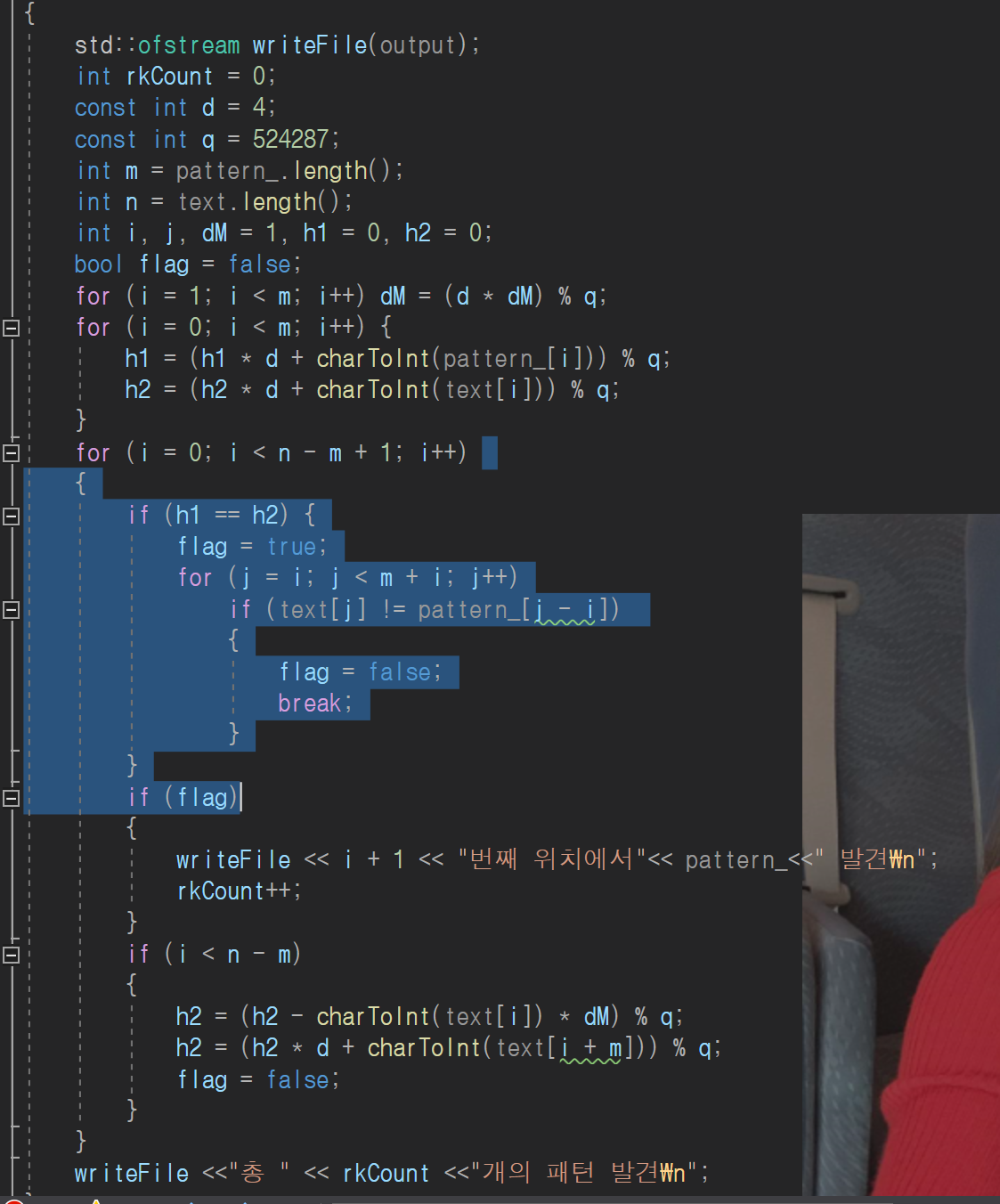
자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

불필요한 비교 횟수를 줄이기 위해 suffix와 prefix의 일치 부분을 찾아야 하고 이를 위해 최대 접두부 테이블을 생성해야한다. 이를 위해 initSP함수를 따로 이용하였다. 이후 Kmp함수에서 테이블을 이용하여 탐색을 진행한다.

Rabin Karp



해시 함수를 이용한 문자열 탐색 알고리즘으로 해시값이 동일한 경우에만 비교를 하여 비교횟수를 크게 줄일 수 있다. 해시 함수는 mod연산을 통해 구현하였다.