연결 리스트를 이용한 분산 텍스트 파일, 폴더 관리를 위한 프로그램

김민석, 최승호*
*광운대학교

Mskim020510@gmail.com, *jcn99250@naver.com

Program for distributed test file and folder management using linked list

Kim Min-seok, Choi Seoung-Ho *
*Kwangwoon Univ.

요 약

학교 수업이나 프로그램밍에 대해서 배우면서 그날 강연일지나 기타 자료, 학습 내용에 대해 텍스트 파일로 저장해 두는 일이 잦다. 하지만 분산되어 있는 텍스트파일을 전부 기억하고 관리하는 것은 쉬운 일이 아니었고, 기존의 프로그램은 자료가 유실되거나 검색기능이 마비되는 경우가 있다. 이러한 어려움을 경감하고자 자료구조와 파일 입출력을 이용하여 특정 파일과 폴더를 관리하고 검색하는데 집중하여 관리 목표가 아닌 다른 파일들을 생략하고 로딩하는 프로그램을 제공하였다. 기존의 텍스트 관리 프로그램과 제안 방법을 비교했을 때 제안 방법이 더 나은 성능을 보임을 확인했다. 이를 통해서 기존에 있는 어려움을 경감시켜줄 수 있음을 제안 방법을 통해서 확인했다.

Ⅰ. 서 론

대학교 수업이나 프로그래밍에 대해 배우면서 그날 강연일지나 기타 자료, 학습 내용에 대해 텍스트 파일로 저장해 두는 일이 많이 발생한다. 또한 대부분의 프로그램들은 텍스트 파일을 많이 사용한다. 하지만 분산되어 있는 텍스트 파일을 전부 기억하고 관리하는 것은 쉬운 일이 아니다. 왜냐하면 여러 마인드 맵이나 텍스트 파일 관리 방법을 찾아보고, 사용해 보았을 때 처음 사용할 때는 큰 불편함이 없었지만 수백개가 넘는 텍스트 파일과 각각의 텍스트파일 안의 수백줄의 글들을 로딩하는 데 기존의 프로그램은 엄청난 딜레이와 렉, 때로는 자료가 소실되는 경우가 발생한다.

이를 경감하고자 자료구조와 파일 입출력을 본논문에서 프로그램을 제공한다. 해당 프로그램의 효율성을 시험하기 위해서 비교방법으로 기존 텍스트 관리 프로그램과 프로젝트의 실행속도를 비교하여 검증한다.

실행하는데 필요한 텍스트파일의 데이터의 양의 많고 적음에 따른 실행 속도 차이와 텍스트파일에서 검색할 때 속도 차이를 비교 했을 때 두 케이스 모두 제안한 프로그램이 더 높은 효율성을 보임을 확인했다.

Ⅱ. 관련연구

장준혁 [1]에서는 대용량 텍스트 파일을 얼마나 빨리 정렬하고 인덱싱 하는 것에 대한 연구를 진행했다. 정렬, 인덱싱을 위한 방법으로 다양한 자료구조를 사용하였으며 링크드 리스트, 토큰 링크드 리스트, 분산 토큰 링크드 리스트, 그리고 CMSort 를 이용하고, 각 방식을 텍스트의 파일용량에 따라 실험하였으며, 분산 토큰 링크드 리스트,CMSORT 에서 정렬 그리고 병합에 따라 실험을 진행하였다. 실험의 결과로 분산 토큰 링크드 리스트가 CMSort 보다 20%가량 더 성능이 좋음을 밝혀 내었고, 이로써 분산 토큰 링크드 리스트 알고리즘이 타 알고리즘과 비교 했을 때 메모리 한계를 뛰어넘어 대용량 텍스트기반파일을 빠르고 안정적으로 정렬 할 수 있었음을 보여주었다.

Ⅲ. 제안방법

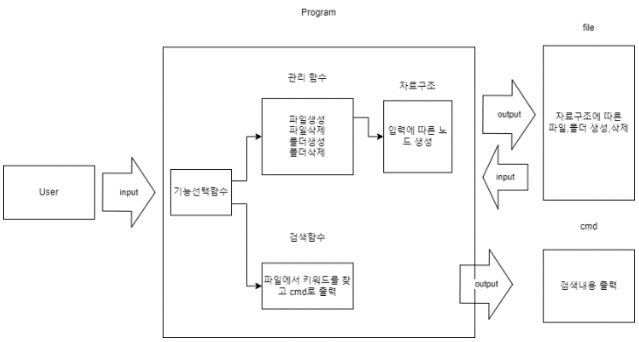

그림 1. 시스템 구성도

Figure 1. System Configuration

그림 1 은 본 논문에서 제안한 시스템 구성도이다. 제안한 방법은 두가지 구조로 구성되어 있다. 첫번째 내부동작 구조 그리고 파일 검색 내부동작 구조로 구성되어 있다.

첫번째 내부 동작 구조는 노드를 생성하고 제거하는 내부동작 구조이다. 프로그램의 자료구조에서 각 노드는 이전 링크를 지칭하는 pre_link, 다음 링크들을 가르키는 next_link 배열, 파일 노드를 가르키는 node 포인터, 파일의 경로에 ₩₩를 붙인 문자열을 가르키는 data 로 구성된 구조체이다. 또한 첫 번째 노드를 가르키는 head 포인터로 이루어진 ll_h 구조체로 구성되었다. 폴더생성, 삭제에선 프로그램에서 선언된 path_select 함수를 통해 사용자가 조작을 원하는 파일 위치를 경로 문자열로 반환한다. 이렇게 반환된 경로를 통해 노드 생성에선 헤드가 비어 있다면 path_select 에선 C:₩₩를 반환하고, 그 외엔 유저가 고른 경로를 통해 원하는 위치에서 새로운 노드와 새로운 파일을 생성하거나 삭제한다. 텍스트 파일 생성과정에선 폴더생성과 같이 원하는 경로를 받은 뒤 해당하는 위치의 노드의 node 포인터에 텍스트 파일을 가르켜 주는 노드를 순서 상관없이 연결한다. 삭제는 폴더 노드에 연결되어 있는 모든 파일 노드들의 데이터를 출력해주고 이중 선택한 노드를 삭제하는 방식이다.

두번째 파일 검색 내부동작 구조로 구성되어 있다. 해당 부분은 프로그램에서 선언된 path_select 함수를 통해 작업을 원하는 파일의 위치를 반환받는다.

그리고 검색을 원하는 문자열을 입력 받은 뒤 반환된 경로를 통해 자료구조의 원하는 위치로 이동한 뒤 해당 노드의 하위 노드와 노드의 node 포인터가 가르키는 파일을 열고 문자열을 받은 다음 일치하는 문자열이 있는지 확인한다음 파일을 닫기를 반복한다.

V. 실험결과

본논문에서는 먼저 파일 열기에 소요되는 시간에 대해서 실험 결과를 살펴보고자 한다.

표 1 은 데이터 양이 적을 때 프로그램을 여는데 필요한 시간을 의미한다. 데이터양이 적음에도 비교군인 GitMind는 긴 시간을 소모했고 제안 방법은 짧은 시간에 구동을 마침을 확인했다.

표 1. 적은 데이터 파일

Table 1. fewer data files

회차	GitMind (Sec)	제안 방법 (Sec)
1	3.35	0.34
2	2.31	0.46
3	2.41	1.08
4	2.89	0.87
5	3.55	0.34
6	2.21	0.41
7	3.18	0.47

표 2. 많은 데이터 파일

Table 2. Lots of data files

회차	GitMind (Sec)	제안 방법 (Sec)
1	4.19	1.02
2	3.36	0.89
3	4.08	0.75
4	4.04	0.94
5	4.38	0.54
6	4.14	0.72
7	4.19	0.92

표 3. 검색

Table 3. search

회차	GitMind (Sec)	제안 방법 (Sec)
1	Error	1.1
2	Error	0.97
3	Error	1.33
4	Error	1.28
5	Error	1.13
6	Error	2.1
7	Error	3.12

표 2 는 데이터 양이 많을 때 프로그램을 여는데 필요한 시간을 의미한다. 데이터 양이 많아 짐에 따라 두 대상 모두 로딩 시간이 상승했다. 비교적 제안 방법이 더 적은 시간이 걸림을 확인했다.

표 3 는 데이터 양이 많을 때 프로그램이 검색하는데 필요한 시간을 의미한다. GitMind 와 프로그램이 큰 차이를 보였는데 데이터 양이 늘어나 GitMind 프로그램이 검색시간 에서 큰 차이를 보였다. 데이터 양이 늘어나 GitMind 는 검색 검색을 실행하지 못하고 강제 종료되었지만 제안 방법은 시간이 소요되더라도 검색을 마침을 확인할 수 있다.

IV. 결론

프로그래밍이나 여러 파일들을 관리할 때 많은 텍스트 파일을 보관하고 이용하게 된다. 허나 기존의 프로그램은 디자인이나 다른 기능들로 인해 주 목적인 텍스트파일의 관리와 검색에선 효율성이 떨어지는 모습을 보였다. 이러한 문제를 해결하기위해 텍스트파일의 관리와 검색에 집중적으로 이용할 수 있는 프로그램을 만들고자 하였다. 빠른 접근과 원하는 경로에 제한적으로 집중하기위해 자료구조를 이용하여 프로그램을 제작하였고 실험을 통해 프로그램이 기존의 다른 프로그램에 비해 주어진 목표에 높은 성능을 보여주었다. 프로그램이 종료되면 내부에 저장하고 있던 자료구조는 저장되지 않기에 프로그램이 시작될 때 작업을 원하는 경로의 파일들을 반영하여 자료구조를 미리 재생성하거나 자료구조를 파일형식으로 저장할 필요가 있다. 또한 현재 작업하고 있는 경로를 화면을 통해 보여주거나 cmd 창을 이용해 작업하던 프로그램을 마우스를 이용해 조작 가능하도록 개선하는 방안으로 발전이 가능하다.

참 고 문 헌

[1] 장준혁. "대용량 텍스트 파일을 위한 분산 토큰 기반의 고속 정렬 알고리즘" 국내석사학위논문, 한양대학교 공학대학원, 2013.