**알고리즘 설계와 분석**

**CSE3081-MP3**

2017xxxx 000

* Experiment environment

Window 10 Home

프로세서 : Intel® Core™ i5-7200U CPU @ 2.50GHz 2.71GHz

RAM : 8GB

시스템 종류 : 64비트 운영체제, x64 기반 프로세서

* Experiment setup

파일을 압축하기 위해 Huffman 알고리즘을 이용하였고, 알고리즘은 min heap을 이용하여 구현하였다. Heap의 노드들에는 구조체 타입을 이용하여 문자의 종류와 빈도수, leaf node인지에 대한 정보 등을 저장하였다. Heapsort를 사용하여 빈도수가 낮은 순으로 정렬한 후 Huffman 알고리즘을 통해 하나의 트리로 만들어 각 leaf node(즉, 문자의 종류)에 0과 1로 이루어진 코드를 부여하였다. 입력의 문자들을 0과 1로 이루어진 코드로 바꾼 후, 이를 6개씩 잘라 6자리의 이진수를 십진수로 표현하여 해당하는 아스키코드 문자를 부여하여 최종적으로 출력(압축) 파일에 .txt로 저장하였다. 이 때, 압축 파일의 앞쪽에 압축 해제를 위한 입력파일에 대한 정보(문자의 종류 수, 입력 파일의 총 문자 수, 문자의 종류와 빈도수에 대한 정보를 담고 있는 구조체 등)를 저장하였다.

압축한 파일을 풀 때는 압축할 때와는 반대의 방식으로 구현하였다. 압축 파일의 문자를 한 개씩 읽어 들여 아스키코드에 해당하는 십진수를 6자리 이진수로 표현하고, 이를 다시 압축할 때 huffman 알고리즘을 통해 만든 트리를 이용하여 대응하는 문자를 찾아 출력(압축해제) 파일에 .txt로 저장하였다.

* Observation

|  |  |  |  |
| --- | --- | --- | --- |
|  | 아스키 코드 문자만 | 한글 포함 | 몇 개의 문자로만(약 10개) |
| Original file size | 6.04 | 5.74 | 5.73 |
| Compressed file size | 4.48 | 4.74 | 3.03 |
| Compression ratio | 1.35 | 1.21 | 1.89 |

한글을 포함한 파일보다는 아스키코드의 문자만 포함한 파일이, 그리고 이 보다는 몇 개의 문자로만 이루어진 파일의 압축률이 더 높았다. 즉, 문자의 종류가 적을수록 압축률이 좋다는 것을 알 수 있었다. 이는 문자의 종류가 늘어나면 필요한 0과 1의 수가 늘어나는 허프만 코드의 특징을 잘 반영하고 있다.