알고리즘 HW\_Huffman\_Compressor

2016706033 이동준

1.개발환경

본 과제는 visual studio 2019에서 개발되었습니다.

2.Huffman 인코딩 체계

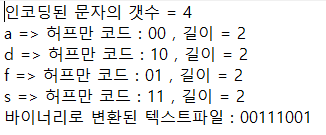
텍스트 파일에 대한 압축을 위해서 각 symbol에 대한 Huffman code를 만들고 트리를 만듭니다.

먼저 .hfb의 헤더부분에 이 symbol에 대한 인코딩 체계가 있습니다.제일 먼저 텍스트파일에 포함된 symbol의 개수가 가장 맨앞에 나오고 symbol Huffman code의 체계가 나옵니다.

하나의 symbol에 총 2바이트가 부여됩니다. 1바이트는 symbol을 표현하고 나머지 1바이트는 Huffman code의 길이입니다.

Ex)

입력 텍스트 “asdf”에 대한 .hft와 인코더 체계입니다.

 (.txt를 통한 .hft 확인입니다.)

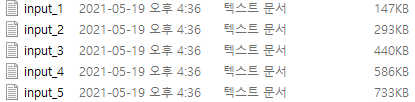


 (Binary viewer을 통한 .hfb확인입니다.)

1. 가장 먼저 symbol의 갯수가 4이므로 첫번째 바이트는 4가 저장됩니다.
2. 다음 ‘a’의 ascii code 61이 저장이되고 다음 바이트에 허프만 코드의 길이 2가 저장됩니다. (a d f g 순으로 헤더에 저장이 되있습니다.)
3. 다음 바이트에 각 symbol에 대한 Huffman code가 저장되있습니다. 저장된 값은  
   00100111(2)으로 각각 2개씩 a(00) d(10) f(01) s(11)를 의미합니다.
4. 다음으로 본 텍스트의 압축된 바이너리가 저장이 돼 있습니다.  
   현재 00111001(2)로 저장된 바이너리는 a(00) s(11) d(10) f(01)로 잘 저장이 돼 있습니다.

3.테스트 케이스

랜덤한 입력값을 받는 크기가 다른 텍스트파일 5개를 만듭니다.



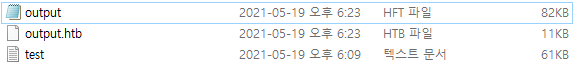
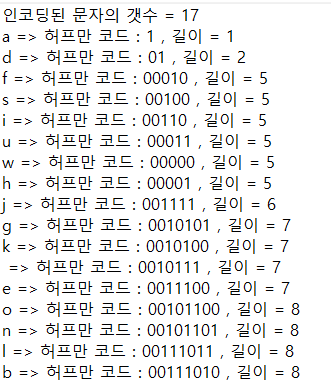
그 이후 각각 압축을 합니다

|  |  |  |
| --- | --- | --- |
| 입력 파일 | 출력 파일 | 압축률 |
| 147kb | 96 | 0.653 |
| 293 | 190 | 0.648 |
| 440 | 285 | 0.647 |
| 586 | 380 | 0.648 |
| 733 | 475 | 0.648 |

이를 통해 완전 랜덤 입력에 대한 압출력은 약 65%인 것을 확인할수있습니다.

3-1)특이 케이스

만약 랜덤 입력이 아니라 특정 입력이 많은 파일에 대해서 압축률을 구해보겠습니다



a가 압도적으로 많고 그다음으로 d가 많은 텍스트파일에 대해서는 압축률이 거의 18%가 되는것을 확인할수있습니다.

4.결론

압축률은 정해지는 것이 아니라 입력 텍스트의 유형에 따라 달라질수있다.

5.개선사항

뒷부분의 텍스트파일에 대한 압축은 허프만을 이용할 경우 더 이상 압축할 방법은 없지만, 헤더 같은 부분에서 symbol하나에 대한 Huffman code에 관한 데이터 크기를 2바이트나 주는 것은 너무 과하게 준 것 같지만 통상 크게 차이나도 1kb에 한참 못미치므로 무시할만 하다.