허프만 코드를 이용한 복호화 구현 및 실험

2016706018 박건희

본 최종보고서는 과제1의 압축프로그램을 해제하는 것을 중심으로 작성되었습니다.

본 과제는 c++언어를 이용하여 수행됐습니다.

**압축해제 알고리즘**

본 실험에서 압축해제 프로그램은 과제1에 작성된 방식으로 인코딩된 hfb 파일을 원본 텍스트 파일로 복원하였기 때문에, 과제1의 인코딩 체계를 기반으로 작성되었습니다.

허프만트리를 이용한 압축 해제 알고리즘의 핵심은, 원본파일의 텍스트가 가변길이 바이너리로 치환이 되었기 때문에, 반드시 해당 허프만트리를 이용해 생성된 바이너리 정보를 알아야 합니다.

인코딩시 텍스트를 바이너리로 변환한 부분의 앞에 헤더로서 인코딩 체계를 입력하였고, 디코딩시 인코딩 체계를 알아야 하기 때문에 해당 헤더만큼 불러옵니다.

헤더의 첫 바이트에는 마지막 바이트의 유효bit, 두 번째 바이트는 문자 종류의 개수, 세 번째 바이트부터 각 문자에 대한 정보를 저장했습니다.

두번째 바이트에서 문자 종류의 개수를 알기 때문에, 문자 종류 개수만큼 문자에 대한 정보를 프로세스에서 읽어오면, 헤더의 바이트를 전부 읽는 과정입니다.

세번째 바이트부터 작성된 각 문자에 대한 정보는 바이트 순서대로, 문자의 아스키코드, 문자의 길이, 문자에 해당하는 허프만코드입니다.

이후 바이트부터 마지막 바이트까지 각 문자가 바이너리로 치환되어 파일에 입력되었고, 마지막 바이트에 도달하면 첫번째 바이트에 저장해 놓았던 유효 bit를 이용하여 마지막 바이트에서 쓸모 있는 bit가 얼마나 되는지 알아냅니다.

**입력 파일 복원 실험**

과제물의 실행 방법은

Comprss.exe가 압축프로그램 Decompress.exe가 압축해제 프로그램입니다.

각 실행파일은 이름의 입력을 따로 지정하는게 아닌 디폴트로 셋팅되어있고, 처음에 테스트케이스의 개수를 입력 받습니다.

테스트케이스 이름의 형식은, compress.exe의 경우 순서대로 testcase0.txt, testcase1.txt, testcase2.txt, …으로 “testcase” + 번호 + “.txt”의 형태로 파일을 입력으로 받으며, 과제의 경우는 테스트케이스가 5개이기 때문에 5를 입력시 testcase0.txt ~ testcase4.txt의 5개 파일로부터 인코딩을 거쳐 hfb 파일과, 그에 따른 정보를 담고 사람이 볼 수 있도록 한 hft파일을 생성합니다.

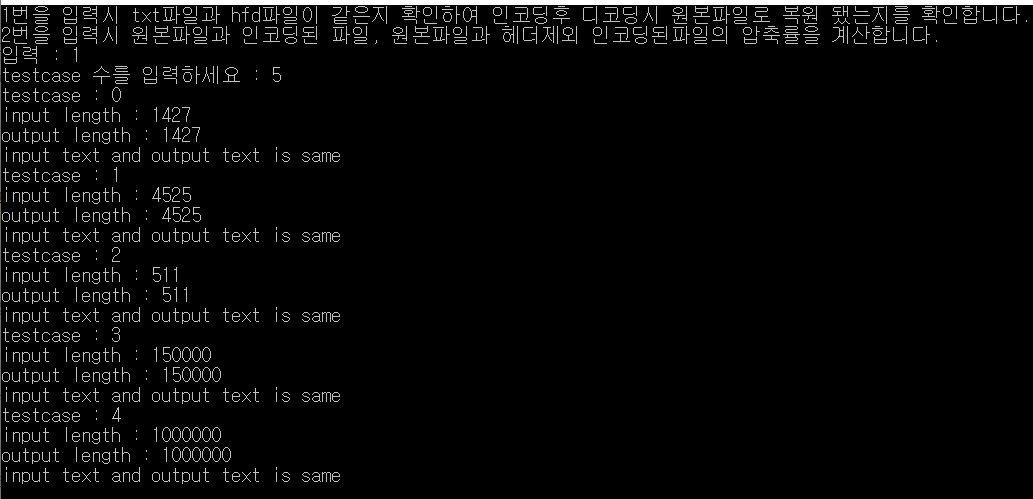
Decompress.exe의 경우 compress.exe와 같이 “testcase” + 번호 + “.hfb”의 형태로 파일을 입력받으며, 5를 입력시 testcase0.hfb ~ testcase4.hfb 파일을 입력으로 하여 testcase0.hfd~testcase4.hfd 파일을 출력으로 디코딩 합니다.

과제 1에서 생성한 testcase0.hfb~ testcase5.hfb 파일을 입력으로 출력물을 생성하여 원본 텍스트파일인 testcas0.txt~testcas5.txt와 비교한 결과 모두 동일하게 결과가 생성된 것을 확인했습니다.

제대로 복원이 됐는지 확인을 위해, 텍스트 파일을 직접 열어 비교하는 것과, 프로그램을 만들어서 확인하는 방법을 사용했고, 정확히 원본 파일로 복원이 된 것을 확인했습니다.

프로그램의 경우 처음에 두 입력파일의 크기를 측정한 후, 각 문자 위치별로 같은 지 확인하는 방식으로 코딩을 진행했고 보고서의 하단에 검사에 사용한 프로그램의 소스코드를 따로 첨부합니다.

복원시 testcase5.hfb의 경우는 원본 텍스트 파일의 크기가 1,000,000바이트이기 때문에 시간이 다른 파일보다 더 걸렸습니다



프로그램을 통하여 5개 테스트케이스를 확인한 결과 오차 없이 정확히 원본 파일로 복원되었습니다.

본 테스트케이스 5개 이외에도, 다른 여러가지 랜덤한 테스트케이스를 작성하여 인코딩 및 디코딩을 확인한 결과 모두 정상적으로 복원이 된 것을 확인했습니다.

검사에 사용된 프로그램의 경우 인코딩된 프로그램의 압축률 확인과, 두개의 입출력 파일이 같은지 확인 가능하는 2가지 기능을 넣었습니다.

파일의 일치 여부 확인의 경우, 2개 파일의 바이트를 검사 후에 다른 문자가 존재하는지 전 영역을 검사합니다. 프로그램 또한 testcase0.hfd의 경우 testcase0.txt와 비교하는 방식으로 testcase 수를 n개 입력시 testcase0~testcase(n-1)의 입출력 파일 일치 여부를 확인합니다.

\*\* 검사 프로그램 소스코드 첨부 \*\*

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

int sw;

cout << "1번을 입력시 txt파일과 hfd파일이 같은지 확인하여 인코딩후 디코딩시 원본파일로 복원 됐는지를 확인합니다." << endl;

cout << "2번을 입력시 원본파일과 인코딩된 파일, 원본파일과 헤더제외 인코딩된파일의 압축률을 계산합니다. " << endl;

cout << "입력 : ";

cin >> sw;

int testcase;

cout << "testcase 수를 입력하세요 : ";

cin >> testcase;

for (int casenum = 0; casenum < testcase; casenum++) {

if (sw == 1) {

fstream read\_input("testcase" + to\_string(casenum) + ".txt");

fstream read\_output("testcase" + to\_string(casenum) + ".hfd");

cout << "testcase : " << casenum << endl;

char chi;

char cho;

int checki;

int checko;

read\_input.seekg(0, ios::end);

checki = read\_input.tellg();

cout << "input length : " << read\_input.tellg() << endl;

read\_input.seekg(0, ios::beg);

read\_output.seekg(0, ios::end);

checko = read\_output.tellg();

cout << "output length : " << read\_output.tellg() << endl;

read\_output.seekg(0, ios::beg);

if (checki != checko) {

cout << "length is different" << endl;

}

else {

int count = 0;

while (((chi = read\_input.get()) != EOF) && ((cho = read\_output.get()) != EOF)) {

// read.get(ch);

if (chi != cho)

count++;

}

if (count == 0)cout << "input text and output text is same" << endl;

else cout << "different character " << count << endl;

}

}

else if (sw == 2) {

int checki;

int binsize;

int binsize2;

fstream read\_input("testcase" + to\_string(casenum) + ".txt");

fstream read\_outputbin("testcase" + to\_string(casenum) + ".hfb", ios::in | ios::binary);

fstream read\_outputbin2("testcase" + to\_string(casenum) + "(no\_heaer).hfb", ios::in | ios::binary);

read\_input.seekg(0, ios::end);

checki = read\_input.tellg();

read\_input.seekg(0, ios::beg);

read\_outputbin.seekp(0, ios::end);

binsize = read\_outputbin.tellp();

read\_outputbin.seekp(0, ios::beg);

read\_outputbin2.seekp(0, ios::end);

binsize2 = read\_outputbin2.tellp();

read\_outputbin2.seekp(0, ios::beg);

cout << "testcase" << casenum << " : "<<endl;

cout << "원본 파일 크기 : " << checki << " Bytes" << endl;

cout << "압축 파일 크기(헤더제외) : " << binsize2 << " Bytes" << endl;

cout << "압축률(헤더제외) : " << (1 - (float)binsize2 / checki) \* 100 << "%" << endl;

cout << "압축 파일 크기(헤더포함) : " << binsize << " Bytes" << endl;

cout << "압축률(헤더포함) : " << (1 - (float)binsize / checki)\*100 << "%" << endl;

cout << endl;

}

}

char ch;

cin >> ch;

return 0;

}