

오픈소스SW기초 (2024-2)

## 6. Linux 기초 명령어 II

국립금오공과대학교

컴퓨터공학과 / 인공지능공학과

김 경 수

# 목차

- ① 텍스트 정렬과 중복 처리 명령어
- ② 다양한 텍스트 처리 명령어

# 학습 목표

- ① 리눅스에서 제공하는 기초적인 텍스트 처리 명령들을 확인하고 각 기능들의 활용 방법을 숙달할 수 있다.
- ② 리눅스의 다양한 텍스트 명령을 조합하여 실행하는 방법을 이해하고 이를 실제로 활용할 수 있다.

# 텍스트 정렬과 중복 처리 명령어

# sort

- 텍스트 파일의 내용을 행 단위로 정렬하여 결과를 출력함.
- 사용법: **sort [옵션] 파일명**
- 주요 옵션
  - -n: 숫자 값으로 정렬
  - -r: 역순으로 정렬
  - -u: 데이터 내 중복 제거

```
GNU nano 6.4  sort_example1.txt  Modified
Gumi
Daegu
Seoul
Busan
Ulsan
Incheon
Pohang
Gwangju
Daejeon
Chuncheon|

^G Help      ^O Write Ou^W Where Is^K Cut
^X Exit      ^R Read Fil^_ Replace ^U Paste
```

- “sort”명령어 실습을 위해서, “./documents/” 디렉토리에  
“sort\_example1.txt” 파일을 만들고 위와 같이 도시명을 입력한 후  
저장한다. (※ nano를 사용하여 편집)

**nano sort\_example1.txt**

# sort 명령어 활용 방법 1 – 일반 문자 정렬

- 다음과 같이 sort\_example1.txt 파일에 “sort” 명령어를 적용하면 해당 파일의 내용들이 오름차순으로 정렬됨.

➤ (예제) **sort sort\_example1.txt**

```
Gumi  
Daegu  
Seoul  
Busan  
Ulsan  
Incheon  
Pohang  
Gwangju  
Daejeon  
Chuncheon|
```

```
Busan  
Chuncheon  
Daegu  
Daejeon  
Gumi  
Gwangju  
Incheon  
Pohang  
Seoul  
Ulsan
```

# sort 명령어 활용 방법 1 – 일반 문자 정렬

- 만약 sort\_example1.txt 파일의 내용을 내림차순으로 정렬하기 위해서는 “sort” 명령어의 옵션에 “-r”을 추가하면 됨.

➤ (예제) `sort -r sort_example1.txt`

```
Gumi  
Daegu  
Seoul  
Busan  
Ulsan  
Incheon  
Pohang  
Gwangju  
Daejeon  
Chuncheon|
```

```
Ulsan  
Seoul  
Pohang  
Incheon  
Gwangju  
Gumi  
Daejeon  
Daegu  
Chuncheon  
Busan
```

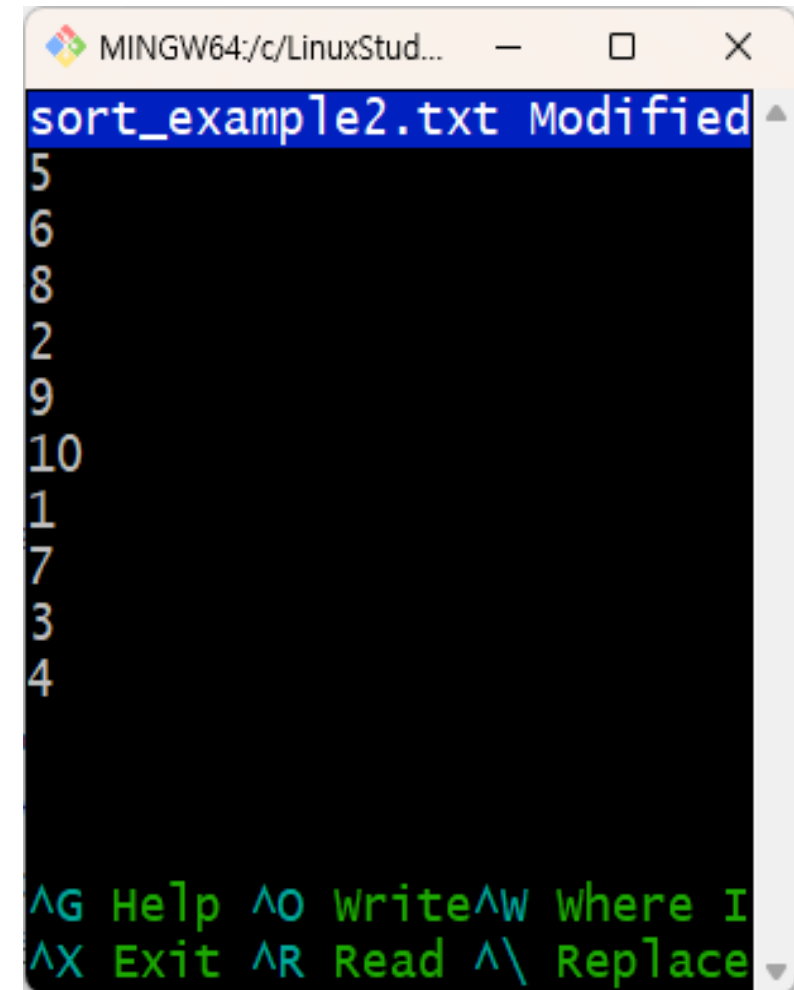
# sort 명령어 활용 방법 2 - 숫자 정렬

- “sort” 명령어의 옵션으로 “-n”을 지정하면, 문자열을 숫자로 인식하여 정렬함.

➤ 사용법: **sort -n 파일명**

- 실습을 위해 아래와 같이 “nano” 에디터를 사용하여 오른쪽 그림과 같이 10개의 숫자를 갖는 텍스트 파일 “sort\_example2.txt”를 “./documents/” 디렉토리에 생성함.

**nano sort\_example2.txt**



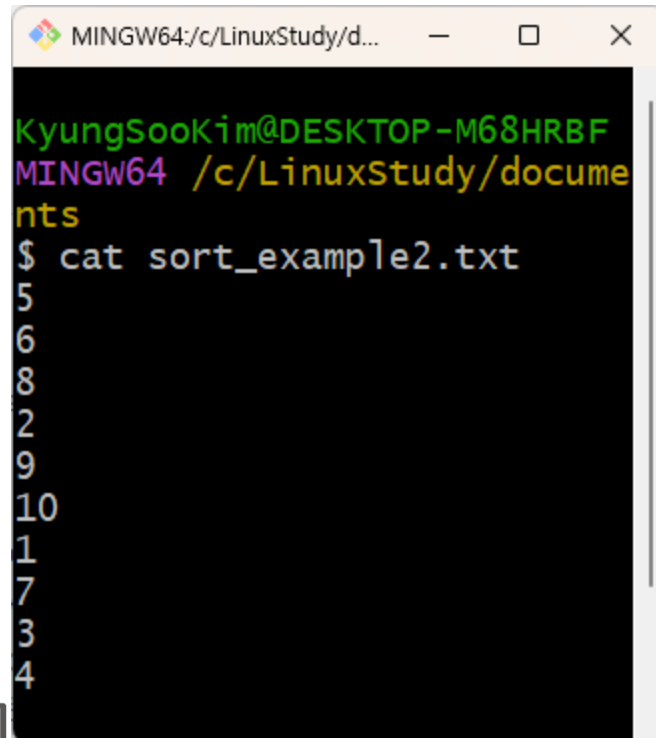
```
MINGW64:/c/LinuxStud...
sort_example2.txt Modified
5
6
8
2
9
10
1
7
3
4

^G Help ^O Write ^W Where I
^X Exit ^R Read ^\ Replace
```

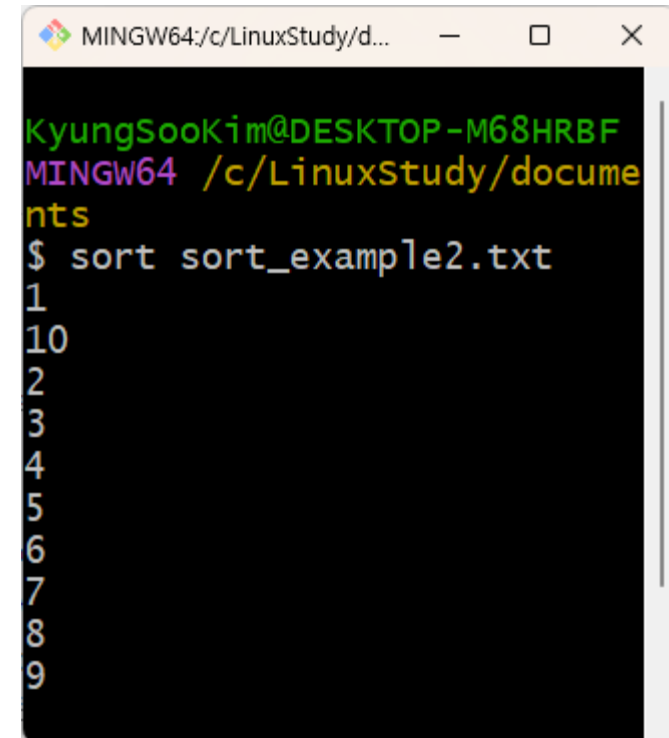


## sort 명령어 활용 방법 2 – 숫자 정렬

- “sort\_example2.txt” 파일의 내용을 “-n” 옵션 없이 정렬한 결과
  - **sort sort\_example2.txt**
  - 우리가 생각한 것처럼 숫자들이 정렬되지 않는다.



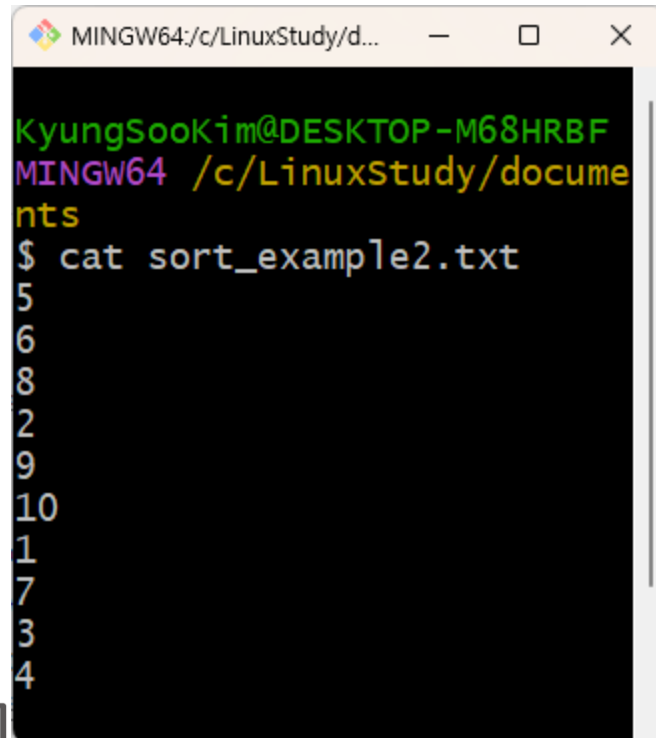
```
MINGW64:/c/LinuxStudy/d...  
KyungSooKim@DESKTOP-M68HRBF  
MINGW64 /c/LinuxStudy/docume  
nts  
$ cat sort_example2.txt  
5  
6  
8  
2  
9  
10  
1  
7  
3  
4
```



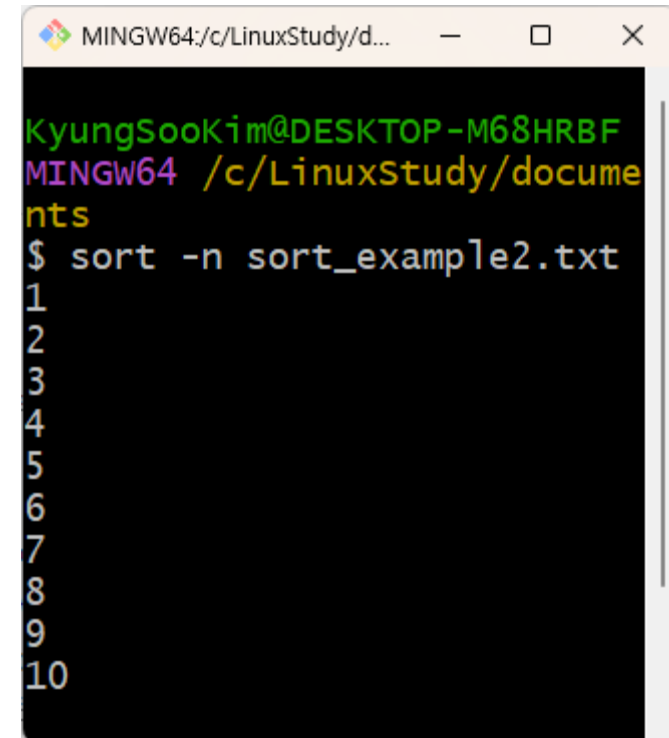
```
MINGW64:/c/LinuxStudy/d...  
KyungSooKim@DESKTOP-M68HRBF  
MINGW64 /c/LinuxStudy/docume  
nts  
$ sort sort_example2.txt  
1  
10  
2  
3  
4  
5  
6  
7  
8  
9
```

# sort 명령어 활용 방법 2 - 숫자 정렬

- “-n” 옵션을 적용하여 “sort\_example2.txt” 파일의 내용을 정렬한 결과
  - **sort -n sort\_example2.txt**
  - 각 행의 숫자들이 정상적인 숫자 데이터로 인식되어 정렬된다.



```
MINGW64:/c/LinuxStudy/d...  
KyungSooKim@DESKTOP-M68HRBF  
MINGW64 /c/LinuxStudy/docume  
nts  
$ cat sort_example2.txt  
5  
6  
8  
2  
9  
10  
1  
7  
3  
4
```

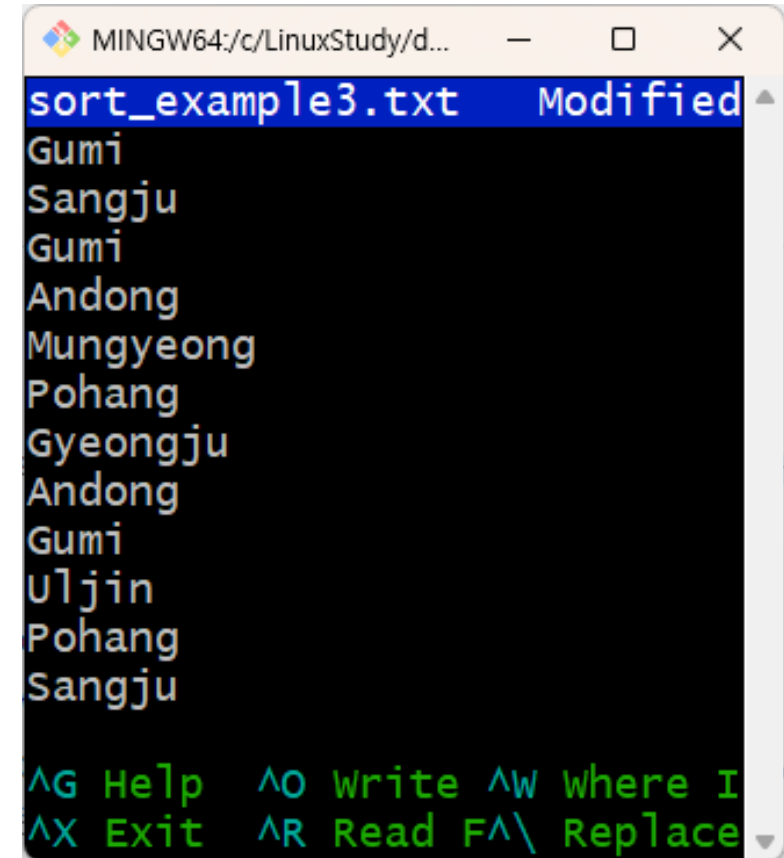


```
MINGW64:/c/LinuxStudy/d...  
KyungSooKim@DESKTOP-M68HRBF  
MINGW64 /c/LinuxStudy/docume  
nts  
$ sort -n sort_example2.txt  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

# sort 명령어 활용 방법 3 – 데이터 내 중복 제거

- “sort” 명령을 활용하여 데이터를 정렬할 때, 중복 데이터가 존재하는 경우 중복을 제거하고 정렬을 수행하는 옵션 → “-u” 활용.
- 실습을 위해 오른쪽과 같이 “./documents/” 디렉토리에 “sort\_example3.txt” 파일을 작성하고 저장함.

**nano sort\_example3.txt**



```
MINGW64:/c/LinuxStudy/d... - □ X
sort_example3.txt Modified
Gumi
Sangju
Gumi
Andong
Mungyeong
Pohang
Gyeongju
Andong
Gumi
Uljin
Pohang
Sangju

^G Help ^O Write ^W Where I
^X Exit ^R Read F^ Replace
```

# sort 명령어 활용 방법 3 – 데이터 내 중복 제거

- “sort” 명령어를 이용하여 “sort\_example3.txt” 파일의 내용을 오름차순과 내림차순으로 각각 정렬함.

➤ `sort sort_example3.txt`

➤ `sort -r sort_example3.txt`

중복된 내용이  
제거되지 않음.

```
/c/LinuxStudy/documents
$ sort sort_example3.txt
Andong
Andong
Gumi
Gumi
Gumi
Gyeongju
Mungyeong
Pohang
Pohang
Sangju
Sangju
Uljin
```

```
/c/LinuxStudy/documents
$ sort -r sort_example3.txt
Uljin
Sangju
Sangju
Pohang
Pohang
Mungyeong
Gyeongju
Gumi
Gumi
Gumi
Andong
Andong
```

# sort 명령어 활용 방법 3 – 데이터 내 중복 제거

- 중복된 내용의 제거를 위해 “sort” 명령어에 옵션으로 “-u”를 지정하고 “sort\_example3.txt” 파일의 내용을 정렬함.

➤ `sort -u sort_example3.txt`

➤ `sort -ru sort_example3.txt`

중복된 내용이 제거됨.

```
/c/LinuxStudy/documents
$ sort -u sort_example3.txt
Andong
Gumi
Gyeongju
Mungyeong
Pohang
Sangju
Uljin
```

```
/c/LinuxStudy/documents
$ sort -ru sort_example3.txt
Uljin
Sangju
Pohang
Mungyeong
Gyeongju
Gumi
Andong
```

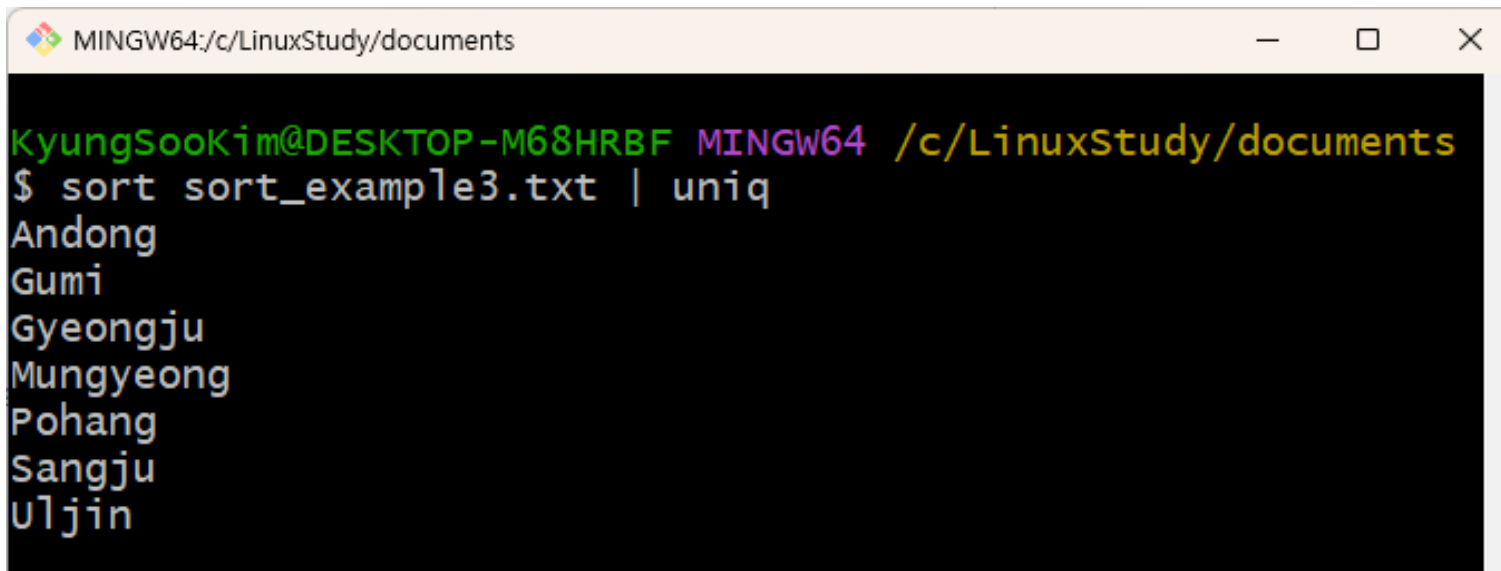
# uniq

- 연속된 중복 데이터를 제거하여 출력하는 명령어
- 사용법: **uniq 파일명**
- 유의사항
  - 같은 내용이 연속되어 있는 경우에만 중복 제거
  - 즉, 연속되지 않은 중복 데이터는 제거되지 않음.

```
MINGW64:/c/LinuxStudy/documents
KyungSooKim@DESKTOP-M68HRBF
/c/LinuxStudy/documents
$ uniq sort_example3.txt
Gumi
Sangju
Gumi
Andong
Mungyeong
Pohang
Gyeongju
Andong
Gumi
Uljin
Pohang
Sangju
```

# uniq

- “sort” 명령을 수행한 후 중복을 제거하면 중복된 모든 데이터를 바로 삭제할 수 있음.
- 사용법: **sort 파일명 | uniq**

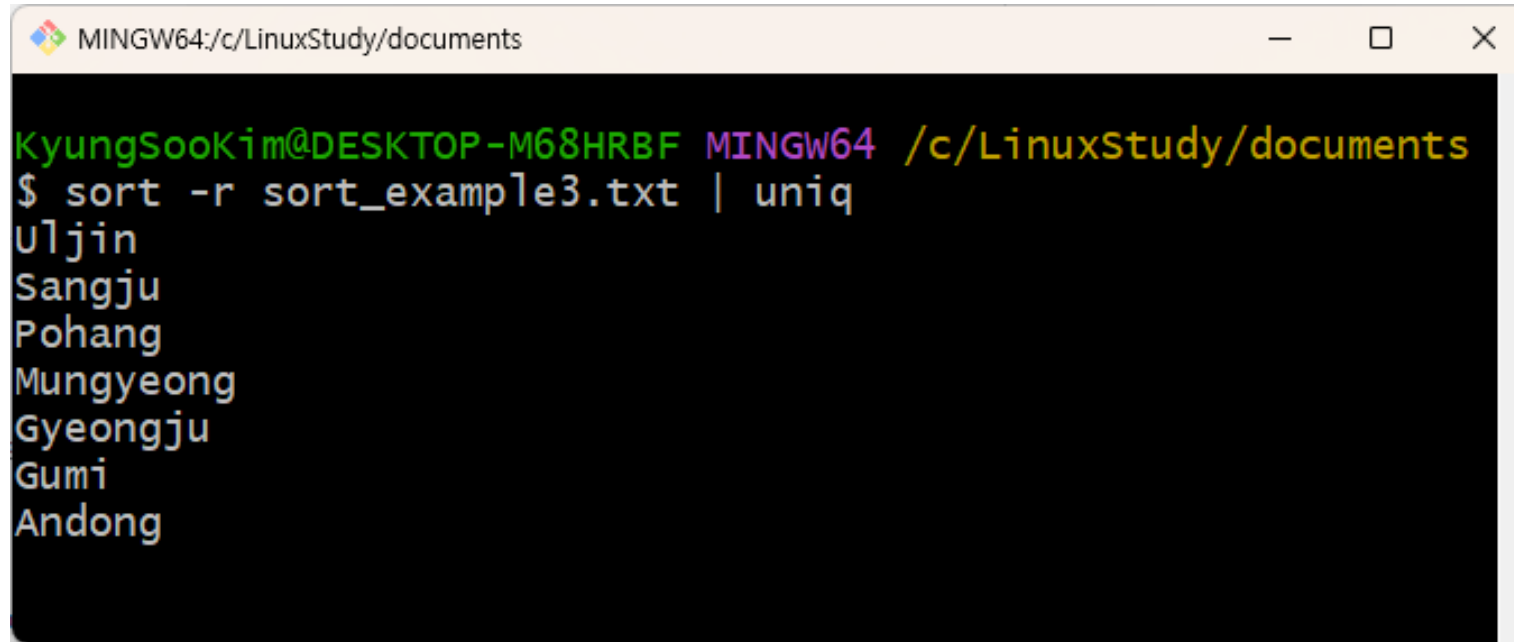


```
MINGW64:/c/LinuxStudy/documents  
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents  
$ sort sort_example3.txt | uniq  
Andong  
Gumi  
Gyeongju  
Mungyeong  
Pohang  
Sangju  
Uljin
```

(예제) **sort sort\_example3.txt | uniq**

# uniq

- “sort” 명령을 수행한 후 중복을 제거하면 중복된 모든 데이터를 바로 삭제할 수 있음.
- 사용법: **sort 파일명 | uniq**



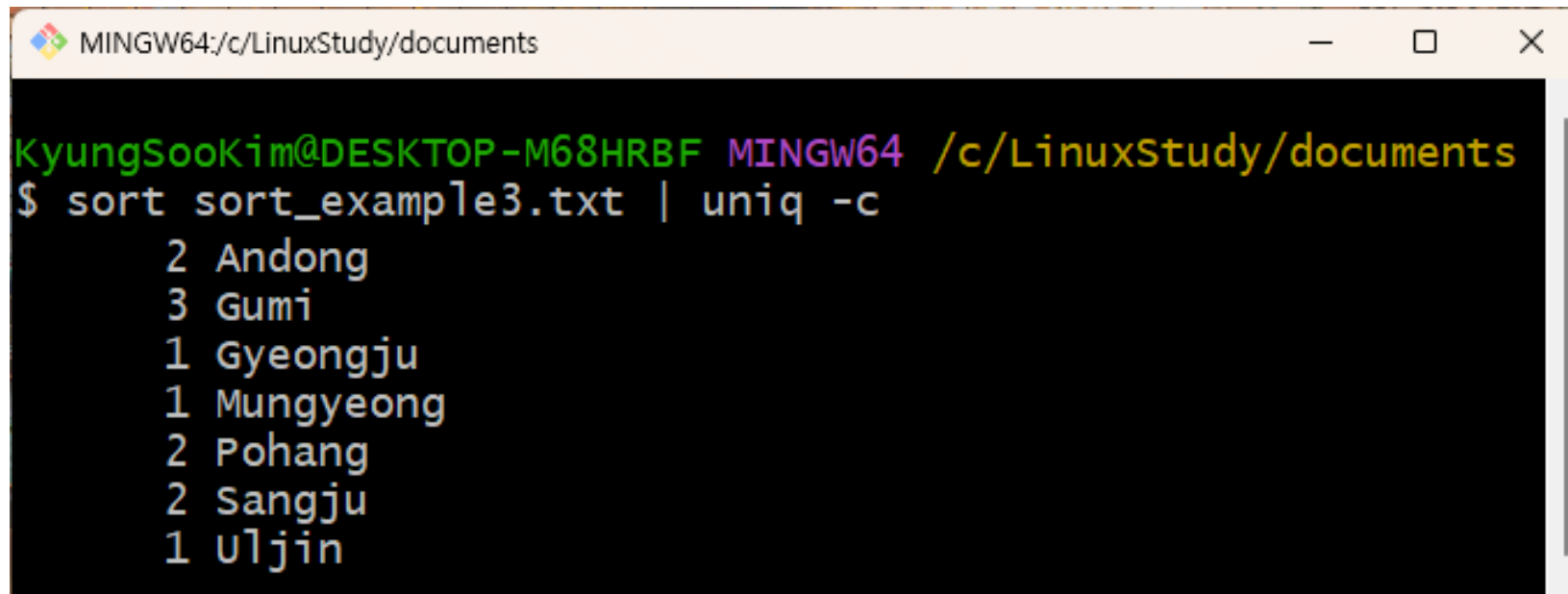
```
MINGW64:/c/LinuxStudy/documents
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ sort -r sort_example3.txt | uniq
Uljin
Sangju
Pohang
Mungyeong
Gyeongju
Gumi
Andong
```

(예제) **sort -r sort\_example3.txt | uniq**



# uniq 명령에서 중복 데이터의 개수 세기

- 사용법: uniq 명령어의 옵션에 “-c”를 추가함.
- (예제 1) `sort sort_example3.txt | uniq -c`



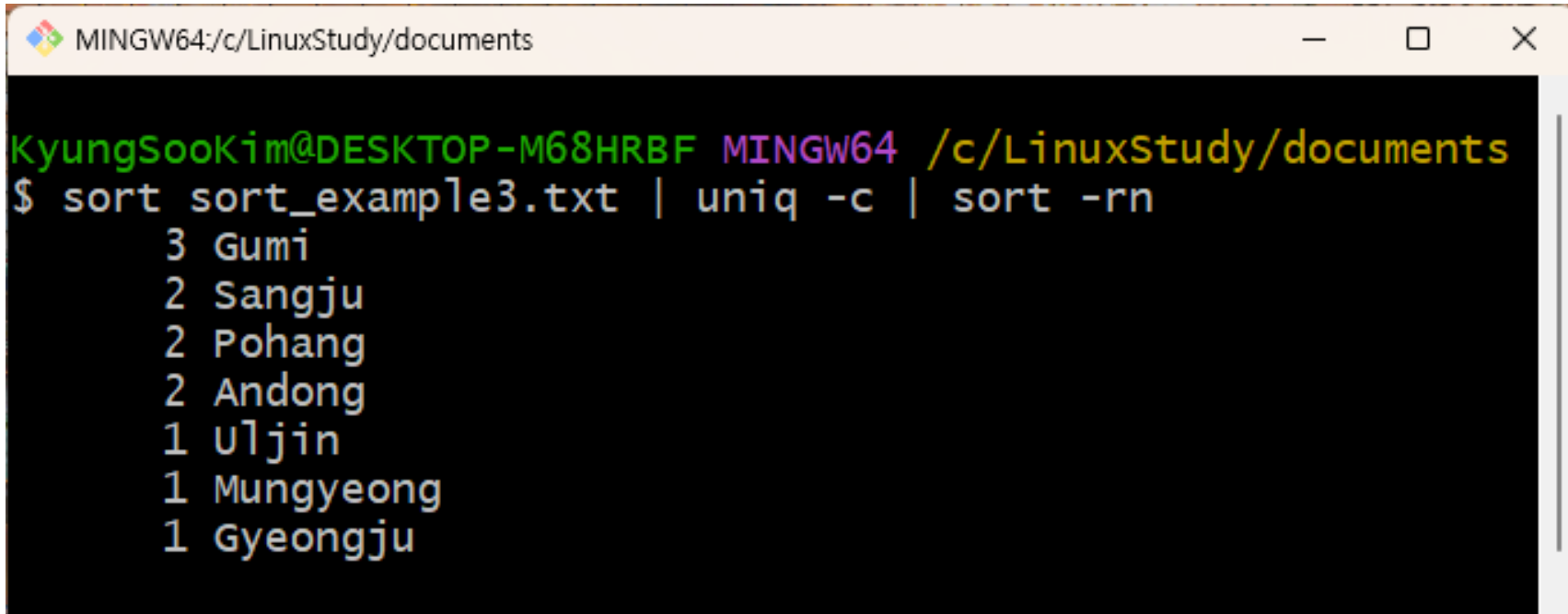
```
MINGW64:/c/LinuxStudy/documents  
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents  
$ sort sort_example3.txt | uniq -c  
  2 Andong  
  3 Gumi  
  1 Gyeongju  
  1 Mungyeong  
  2 Pohang  
  2 Sangju  
  1 Uljin
```

- “uniq” 명령어에 옵션으로 “-c” 또는 “--count”를 지정하면 중복 횟수(count)를 각 줄의 맨 앞부분에 출력함.

# uniq 명령에서 중복 데이터의 개수 세기

- (예제 2) “**uniq -c**”의 결과를 정렬하여 중복이 많은 순으로 출력

```
sort sort_example3.txt | uniq -c | sort -rn
```



```
MINGW64:/c/LinuxStudy/documents


KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ sort sort_example3.txt | uniq -c | sort -rn
      3 Gumi
      2 Sangju
      2 Pohang
      2 Andong
      1 Uljin
      1 Mungyeong
      1 Gyeongju
```

- “uniq” 명령어에 옵션으로 “-c” 또는 “--count”를 지정하면 중복 횟수(count)를 각 줄의 맨 앞부분에 출력함.

# uniq 명령에서 중복 데이터의 개수 세기

- (예제 3) “`uniq -c`”의 결과를 역순으로 정렬하여 중복이 적은 순으로 출력

```
sort sort_example3.txt | uniq -c | sort -n
```



```
MINGW64:/c/LinuxStudy/documents  
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents  
$ sort sort_example3.txt | uniq -c | sort -n  
1 Gyeongju  
1 Mungyeong  
1 Uljin  
2 Andong  
2 Pohang  
2 Sangju  
3 Gumi
```

# 다양한 텍스트 처리 명령어

# wc

- 바이트 수와 단어 수, 행의 개수를 카운트(count)함.
- 사용법: **wc [옵션] 파일명**
- 주요 옵션

- -l: 파일 내 행의 수 표시
- -w: 파일 내 단어의 수 표시
- -c: 파일의 바이트 수 표시

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ ls
hello.c      neural_types.txt  resnet.txt  theory.txt.save
learning.txt  neuralnet.txt     theory.txt
```

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ wc theory.txt
  9  453 3022 theory.txt
```

행의 수   단어의 수   바이트 수

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ wc -l theory.txt
9 theory.txt
```

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ wc -w theory.txt
453 theory.txt
```

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ wc -c theory.txt
3022 theory.txt
```

# cut

- 입력된 텍스트에서 특정 부분만 추출하여 출력함.
- 사용법: **cut -d <구분자> -f <필드번호> [파일명]**
  - <구분자>로 지정한 문자를 기준으로 입력 데이터를 분할함.
  - 이 중에서 <필드 번호>로 지정한 필드만 출력함.
  - CSV 파일에서 특정 컬럼만 선택하여 출력할 때 유용하게 사용됨.

“cut” 명령어 실습을 위해 LMS에 업로드되어 있는 “cut\_file.csv” 파일을  
“C:/LinuxStudy/documents/” 디렉토리에 복사한다.

# cut

- (예제 1) “cut\_file.csv” 파일의 내용을 쉼표(,)를 기준으로 나눈 후 3번째 필드를 출력하시오.

```
cut -d ',' -f 3 cut_file.csv
```

```
KyungSookKim@DESKTOP-M68HRBF MIN
$ cut -d ',' -f 3 cut_file.csv
LightTree
LightTree
LightTree
LightTree
LightTree
LightTree
LightTree
LightTree
LightTree
LightTree
```

	A	B	C	D	E	F	G	H
1	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
2	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
3	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
4	1E+08	OUTBOUN	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-1	2
5	1E+08	OUTBOUN	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-2	2
6	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
7	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
8	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
9	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
10	1E+08	OUTBOUN	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
11	1E+08	OUTBOUN	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
12	1E+08	OUTBOUN	LightTree	CALTRAN	LT-2	CALTRAN - San Diego	2017-07-2	2

# cut

- (예제 2) “cut\_file.csv” 파일의 내용을 쉼표(,)를 기준으로 나눈 후 2, 4, 5 번째 필드를 한꺼번에 출력하시오.

```
cut -d ',' -f 2,4,5 cut_file.csv
```

```
KyungSooKim@DESKTOP-M68HRBF MINGW64
$ cut -d ',' -f 2,4,5 cut_file.csv
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-2
OUTBOUND,AHTD,LT-2
OUTBOUND,AHTD,LT-2
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-1
OUTBOUND,AHTD,LT-1
OUTBOUND,CALTRAN,LT-1
OUTBOUND,CALTRAN,LT-1
```

	A	B	C	D	E	F	G	H
1	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
2	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
3	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
4	1E+08	OUTBOUND	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-1	2
5	1E+08	OUTBOUND	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-2	2
6	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
7	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
8	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
9	1E+08	OUTBOUND	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
10	1E+08	OUTBOUND	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
11	1E+08	OUTBOUND	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
12	1E+08	OUTBOUND	LightTree	CALTRAN	LT-2	CALTRAN - San Diego	2017-07-2	2



# cut

- (예제 3) “cut\_file.csv” 파일의 내용을 쉼표(,)를 기준으로 나눈 후, 4번째 필드의 값을 추출하여 내림차순으로 정렬하고 중복을 제거하시오.

```
cut -d ',' -f 4 cut_file.csv | sort -ru
```

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/Linux
$ cut -d ',' -f 4 cut_file.csv | sort -ru
QPWD
NYDOT
MAHD
LightTree
KYTC
FAA
COLA
CALTRAN
AHTD
```

	A	B	C	D	E	F	G	H
1	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
2	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
3	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
4	1E+08	OUTBOUN	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-1	2
5	1E+08	OUTBOUN	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-2	2
6	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
7	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
8	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
9	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
10	1E+08	OUTBOUN	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
11	1E+08	OUTBOUN	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
12	1E+08	OUTBOUN	LightTree	CALTRAN	LT-2	CALTRAN - San Diego	2017-07-2	2

# cut

- (예제 4) “cut\_file.csv” 파일의 내용을 쉼표(,)를 기준으로 나눈 후, 6번째 필드의 값을 추출하여 오름차순으로 정렬하고 중복을 제거하시오.

```
cut -d ',' -f 6 cut_file.csv | sort | uniq
```

```
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStu
$ cut -d ',' -f 6 cut_file.csv | sort | uniq
AHTD - Bentonville
CALTRAN - San Diego
COLA - Los Alamos
FAA - Washington
KYTC - Lexington
LT-1
LT-2
MAHD - Boston
NYDOT - New Rochelle
QPWD - Ville de Qu?ec
```

	A	B	C	D	E	F	G	H
1	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
2	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
3	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
4	1E+08	OUTBOUN	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-1	2
5	1E+08	OUTBOUN	LightTree	AHTD	LT-2	AHTD - Bentonville	2017-07-2	2
6	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
7	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
8	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
9	1E+08	OUTBOUN	LightTree	AHTD	LT-1	AHTD - Bentonville	2021-06-0	2
10	1E+08	OUTBOUN	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
11	1E+08	OUTBOUN	LightTree	CALTRAN	LT-1	CALTRAN - San Diego	2021-06-1	2
12	1E+08	OUTBOUN	LightTree	CALTRAN	LT-2	CALTRAN - San Diego	2017-07-2	2

# tr

- 지정된 문자를 치환 또는 삭제하는 명령어 (= translation)
- 사용법: **tr [옵션] <치환 이전 문자> <치환 이후 문자>**
  - “tr” 명령어는 파일을 입력으로 받지 않고, 표준 입력만을 받음.
  - 주로 “cat” 등 다른 명령어와 파이프를 연결하여 사용함.
  - 옵션으로 “-d”를 사용하면 치환이 아닌 지정된 문자를 삭제하는 기능을 수행함.

# tr

- (예제 1) “learning.txt” 파일의 모든 소문자를 대문자로 치환하여 출력하시오.

```
cat learning.txt | tr a-z A-Z
```

```
$ cat learning.txt | tr a-z A-Z
MACHINE LEARNING IS COMMONLY SEPARATED INTO THREE MAIN LEARNING PARADIGMS, SUPERVISED LEAR
NING UNSUPERVISED LEARNING AND REINFORCEMENT LEARNING.[62] EACH CORRESPONDS TO A PARTICULAR LEARNING TASK.
SUPERVISED LEARNING: SUPERVISED LEARNING USES A SET OF PAIRED INPUTS AND DESIRED OUTPUTS.
THE LEARNING TASK IS TO PRODUCE THE DESIRED OUTPUT FOR EACH INPUT. IN THIS CASE THE COST FUNCTION IS RELATED TO ELIMINATING INCORRECT DEDUCTIONS.[63] A COMMONLY USED COST IS THE MEAN-SQUARED ERROR, WHICH TRIES TO MINIMIZE THE AVERAGE SQUARED ERROR BETWEEN THE NETWORK'S OUTPUT AND THE DESIRED OUTPUT. TASKS SUITED FOR SUPERVISED LEARNING ARE PATTERN RECOGNITION (ALSO KNOWN AS CLASSIFICATION) AND REGRESSION (ALSO KNOWN AS FUNCTION APPROXIMATION). SUPERVISED
LEARNING IS ALSO APPLICABLE TO SEQUENTIAL DATA (E.G., FOR HAND WRITING, SPEECH AND GESTURE RECOGNITION). THIS CAN BE THOUGHT OF AS LEARNING WITH A "TEACHER", IN THE FORM OF A FUNCTION THAT PROVIDES CONTINUOUS FEEDBACK ON THE QUALITY OF SOLUTIONS OBTAINED THUS FAR.
UNSUPERVISED LEARNING: IN UNSUPERVISED LEARNING, INPUT DATA IS GIVEN ALONG WITH THE COST FUNCTION, SOME FUNCTION OF THE DATA  $\{x\}$  AND THE NETWORK
```

# tr

- (예제 2) “learning.txt” 파일의 모든 마침표를 : 기호로 변환하시오.

```
cat learning.txt | tr . :
```

```
$ cat learning.txt | tr . :  
Machine learning is commonly separated into three main learning paradigms, supervised learning  
unsupervised learning and reinforcement learning:[62] Each corresponds to a particular  
learning task:  
Supervised learning: Supervised learning uses a set of paired inputs and desired outputs:  
The learning task is to produce the desired output for each input: In this case the cost function  
is related to eliminating incorrect deductions:[63] A commonly used cost is the mean-squared  
error, which tries to minimize the average squared error between the network's output and the  
desired output: Tasks suited for supervised learning are pattern recognition (also known as  
classification) and regression (also known as function approximation): Supervised  
learning is also applicable to sequential data (e:g:, for hand writing, speech and gesture  
recognition): This can be thought of as learning with a "teacher", in the form of a function  
that provides continuous feedback on the quality of solutions obtained thus far:  
Unsupervised learning: In unsupervised learning, input data is given along with the cost function,  
some function of the data  $\text{\textstyle x}$  and the network
```

# tr

- “tr” 명령어에 “-d” 옵션을 추가하면, 해당 문자를 모두 삭제함.
- (예제 3) “theory.txt” 파일의 모든 소문자를 삭제하시오.

**cat learning.txt | tr -d a-z**

```
$ cat theory.txt | tr -d a-z
C
T
A      - ' (      - )      T , [129]      '      . F,      -T . [130]
C
A ' ""      . I      . T      . T      VC D. T      S D MK' [131]
T C. [132] T      ( )      [133]      . T      . T      , VC . VC D
. T ,      . A , [131] VC D      P. T VC D      M C. [134]
C
M
A      ,      ,      . S,      . T,      ,      .
T      ANN      . W      , ANN      T      ,      . [135] [136] A      ,      ANN
. T      ,      , . [137] [138] [139] [140] T      J . D      . [141]
```



# tr

- (예제 4) “theory.txt” 파일의 모든 내용을 한 줄로 이어 붙이시오.  
= 텍스트 내의 모든 개행 문자 “\n”을 삭제하시오.

```
cat learning.txt | tr -d “\n”
```

```
$ cat theory.txt | tr -d “\n”
The multilayer perceptron is a universal function approximator, as proven by the universal approximation theorem. However, the proof is not constructive regarding the number of neurons. A specific recurrent architecture with rational-valued weights (as opposed to full precision on real number-valued weights) has the power of a universal Turing machine,[129] using a finite number of neurons and standard linear connections. Further, the use of irrational values in a model's "capacity" property corresponds to its ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity. Two notions of capacity are known by the community. The information capacity and the VC Dimension. The information capacity of a perceptron is intensively discussed in Sir David MacKay's book[131] which summarizes work by Thomas Cover.[132] The capacity of a network of standard neurons (not convolutional) can be derived by four rules[133] that derive from understanding a neuron as an electrical element. The information capacity captures the functions modelable by the network given any data as input. The second notion, is the VC dimension. VC Dimension uses the principles of measure theory and finds the maximum capacity under the best possible circumstances. This is, given input data in a specific
```

# tail / head

- **tail**: 입력된 텍스트 파일에서 마지막  $n$ 개의 행을 출력 (\* Default  $n = 10$ )
- **head**: 입력된 텍스트 파일에서 처음  $n$ 개의 행을 출력 (\* Default  $n = 10$ )

```
MINGW64:/c/LinuxStudy

KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ tail mobilenet.txt
    return out

def mobilenetV3(mode="SMALL", num_of_classes=10):
    return MobileNetV3(model_mode=mode, num_classes=num_of_c
t_rate=0.0)

# temp = torch.zeros((1, 3, 224, 224))
# model = MobileNetV3(model_mode="LARGE", num_classes=1000,
# print(model(temp).shape)
# print(get_model_parameters(model))
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ |
```

```
MINGW64:/c/LinuxStudy

KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ head mobilenet.txt
import torch
import torch.nn as nn
import torch.nn.functional as F

def get_model_parameters(model):
    total_parameters = 0
    for layer in list(model.parameters()):
        layer_parameter = 1
        for l in list(layer.size()):
KyungSookKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ |
```



# tail / head

- 옵션으로 “-n”을 지정하여 출력되는 행의 수를 지정할 수 있음.
  - (예제 1) `tail -n 2 mobilenet.txt`
  - (예제 2) `head -n 5 mobilenet.txt`

```
MINGW64:/c/LinuxStudy
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ tail -n 2 mobilenet.txt
# print(model(temp).shape)
# print(get_model_parameters(model))
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ tail -n 4 mobilenet.txt
# temp = torch.zeros((1, 3, 224, 224))
# model = MobileNetV3(model_mode="LARGE", num_classes=1000,
tiplier=1.0)
# print(model(temp).shape)
# print(get_model_parameters(model))
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$
```

```
MINGW64:/c/LinuxStudy
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ head -n 5 mobilenet.txt
import torch
import torch.nn as nn
import torch.nn.functional as F

KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ head -n 1 mobilenet.txt
import torch

KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy
$ |
```

# nl

- 주어진 파일의 각 행에 행 번호를 추가하여 출력함.
- (예제 1) `nl sort_example1.txt`
- (예제 2) `nl sort_example3.txt`

```
$ nl sort_example1.txt
1  Gumi
2  Daegu
3  Seoul
4  Busan
5  Ulsan
6  Incheon
7  Pohang
8  Gwangju
9  Daejeon
10 Chuncheon
```

```
$ nl learning.txt
1 Machine learning is commonly separated into three main le
igms, supervised learning, unsupervised learning and reinforceme
62] Each corresponds to a particular learning task.
2 Supervised learning: Supervised learning uses a set of pa
and desired outputs. The learning task is to produce the desired
ach input. In this case the cost function is related to eliminati
deductions.[63] A commonly used cost is the mean-squared error,
to minimize the average squared error between the network's outpu
sired output. Tasks suited for supervised learning are pattern re
also known as classification) and regression (also known as functi
tion). Supervised learning is also applicable to sequential data
and writing, speech and gesture recognition). This can be thought
ing with a "teacher", in the form of a function that provides cor
back on the quality of solutions obtained thus far.
3 Unsupervised learning: In unsupervised learning, input da
along with the cost function, some function of the data {\display
```

# paste

- 주어진 두 파일을 열(column) 단위로 병합하여 출력함.
  - (예제 1) `paste sort_example1.txt sort_example3.txt`
  - (예제 2) `paste sort_example2.txt sort_example1.txt`

```
$ paste sort_example1.txt sort_example3.txt
Gumi      Gumi
Daegu     Sangju
Seoul     Gumi
Busan     Andong
Ulsan     Mungyeong
Incheon   Pohang
Pohang     Gyeongju
Gwangju   Andong
Daejeon   Gumi
Chuncheon      Ulsan
           Pohang
           Sangju
```

```
$ paste sort_example2.txt sort_example1.txt
5      Gumi
6      Daegu
8      Seoul
2      Busan
9      Ulsan
10     Incheon
1      Pohang
7      Gwangju
3      Daejeon
4      Chuncheon
```

# sed (stream editor)

- 텍스트 파일의 내용을 검색하거나 수정 또는 특정 패턴과 일치하는 부분을 치환할 때 사용하는 명령
  - (예제 1) `sed 's/the/EIN/g' learning.txt` → “the”를 “EIN”으로 치환
  - (예제 2) `sed 's/learning//g' learning.txt` → “learning”을 “”으로 치환(=삭제)

```
$ sed 's/the/EIN/g' learning.txt
Machine learning is commonly separated into three main learning
pervised learning, unsupervised learning and reinforcement learn
corresponds to a particular learning task.
Supervised learning: Supervised learning uses a set of paired in
red outputs. The learning task is to produce EIN desired output
t. In this case EIN cost function is related to eliminating incor
ons.[63] A commonly used cost is EIN mean-squared error, which t
ize EIN average squared error between EIN network's output and t
tput. Tasks suited for supervised learning are pattern recogniti
n as classification) and regression (also known as function app
upervised learning is also applicable to sequential data (e.g.,
ing, speech and gesture recognition). This can be thought of as
a "teacher", in EIN form of a function that provides continuous
EIN quality of solutions obtained thus far.
```

```
$ sed 's/learning//g' learning.txt
Machine is commonly separated into three main pa
ervised and reinforcement .[62] Each corresponds
Supervised : Supervised uses a set of paired inpu
task is to produce the desired output for each i
function is related to eliminating incorrect dedu
cost is the mean-squared error, which tries to mi
error between the network's output and the desired
upervised are pattern recognition (also known as
ion (also known as function approximation). Superv
sequential data (e.g., for hand writing, speech a
```

# sed (stream editor)

- 텍스트 파일의 내용을 검색하거나 수정 또는 특정 패턴과 일치하는 부분을 치환할 때 사용하는 명령
- 옵션으로 “-i”를 지정하면 변경된 사항을 해당 파일에 바로 저장할 수 있음.
- (예제 1) `sed -i 's/the/EIN/g' learning.txt` → “the”를 “EIN”으로 치환하여 저장

```
KyungSooKim@DESKTOP- MINGW64 /c/LinuxStudy/documents
$ sed -i 's/the/EIN/g' learning.txt

KyungSooKim@DESKTOP- MINGW64 /c/LinuxStudy/documents
$ cat learning.txt
Machine learning is commonly separated into three main learning paradigms, su
pervised learning, unsupervised learning and reinforcement learning.[62] Each
corresponds to a particular learning task.
Supervised learning: Supervised learning uses a set of paired inputs and desi
red outputs. The learning task is to produce EIN desired output for each inpu
t. In this case EIN cost function is related to eliminating incorrect deducti
ons [63] A commonly used cost is EIN mean-squared error, which tries to minim
ize EIN average squared error between EIN network's output and EIN desired ou
```



# sed (stream editor)

- 텍스트 파일의 내용을 검색하거나 수정 또는 특정 패턴과 일치하는 부분을 치환할 때 사용하는 명령
- 옵션으로 “-i”를 지정하면 변경된 사항을 해당 파일에 바로 저장할 수 있음.
- (예제 2) `sed -i 's/EIN/the/g' learning.txt` → “EIN”을 “the”로 치환하여 저장

```
KyungSooKim@DESKTOP- MINGW64 /c/LinuxStudy/documents
$ sed -i 's/EIN/the/g' learning.txt

KyungSooKim@DESKTOP- MINGW64 /c/LinuxStudy/documents
$ cat learning.txt
Machine learning is commonly separated into three main learning paradigms, supervised learning, unsupervised learning and reinforcement learning.[62] Each corresponds to a particular learning task.
Supervised learning: Supervised learning uses a set of paired inputs and desired outputs. The learning task is to produce the desired output for each input. In this case the cost function is related to eliminating incorrect deductions [63] A commonly used cost is the mean-squared error, which tries to minimize the average squared error between the network's output and the desired output
```

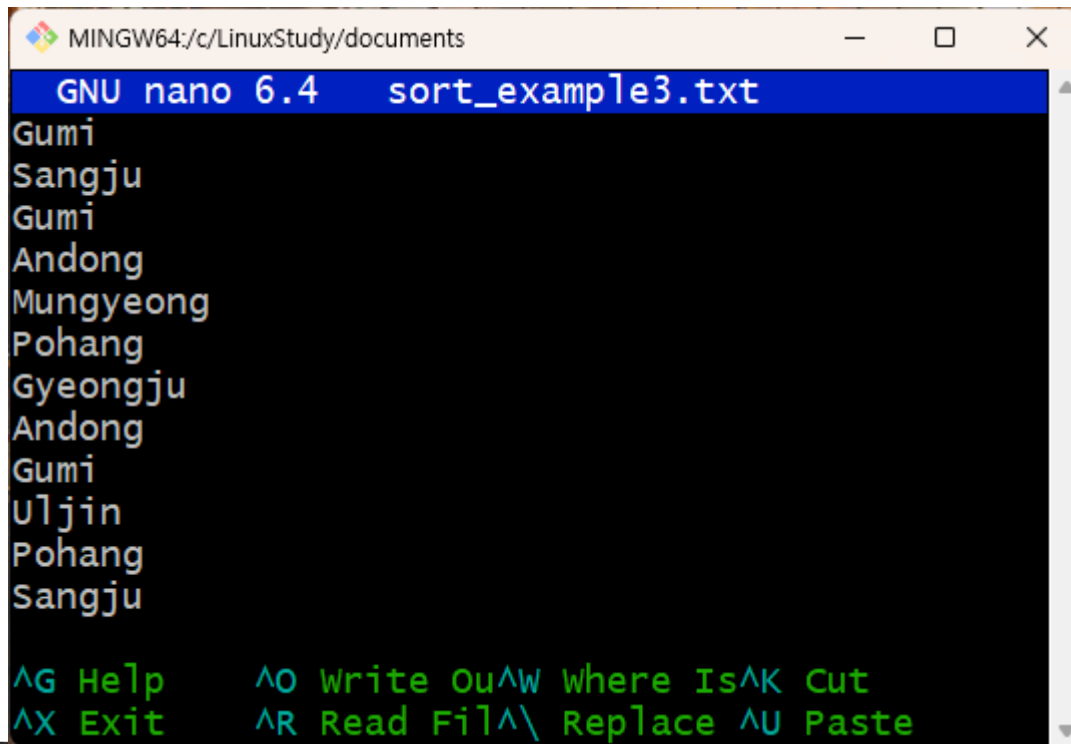
# diff

- 입력된 두 파일의 차이점을 비교하여 출력
- 사용법: **diff [옵션] <비교대상 파일 1> <비교대상 파일 2>**
- 주로 소스 코드나 설정 파일의 편집 전/후 변경사항(차이점)을 확인할 때 사용함.
- Git에서 사용되는 “diff” 명령과 유사한 기능을 수행함.  
[참고] Git의 “diff” 명령은 이후 수업시간에 자세히 다룰 예정임.

# diff

## • 실습을 위한 파일 준비

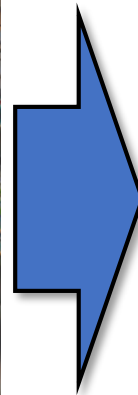
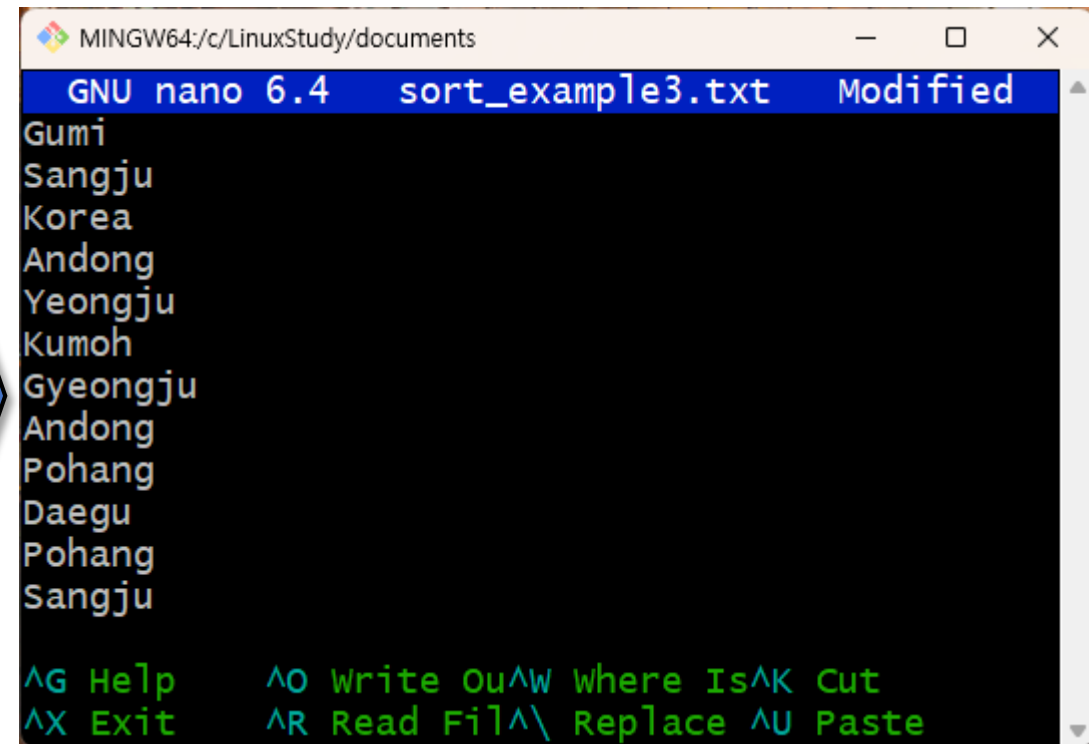
- nano 에디터로 sort\_example3.txt 파일을 열어서 일부 내용을 수정한 후 sort\_example4.txt 파일로 저장한다.



```

MINGW64:/c/LinuxStudy/documents
GNU nano 6.4 sort_example3.txt
Gumi
Sangju
Gumi
Andong
Mungyeong
Pohang
Gyeongju
Andong
Gumi
Uljin
Pohang
Sangju

^G Help      ^O Write Ou^W Where Is^K Cut
^X Exit      ^R Read Fil^_ Replace ^U Paste
  
```

```

MINGW64:/c/LinuxStudy/documents
GNU nano 6.4 sort_example3.txt Modified
Gumi
Sangju
Korea
Andong
Yeongju
Kumoh
Gyeongju
Andong
Pohang
Daegu
Pohang
Sangju

^G Help      ^O Write Ou^W Where Is^K Cut
^X Exit      ^R Read Fil^_ Replace ^U Paste
  
```



# diff

- 두 파일의 비교

➤ `diff sort_example3.txt sort_example4.txt` 명령을 이용하여 두 파일을 비교한다.

```
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/Linu
$ diff sort_example3.txt sort_example4.txt
3c3
< Gumi
---
> Korea
5,6c5,6
< Mungyeong
< Pohang
---
> Yeongju
> Kumoh
9,10c9,10
< Gumi
< Ulsan
---
> Pohang
> Daegu
```

# diff

- “diff” 명령의 출력 결과 해석
  - **3c3**: 첫 번째 파일의 3번째 행이 두 번째 파일의 3번째 행으로 변경됨(c).
  - **5,6c5,6**: 첫 번째 파일의 5,6번째 행이 두 번째 파일의 5,6번째 행으로 변경됨(c).
  - **< 지워진 행의 내용**  
 ---  
**> 추가된 행의 내용**

변경 범위  
= Hunk

```

MINGW64:/c/LinuxStudy/documents
KyungSooKim@DESKTOP-M68HRBF MINGW64 /c/LinuxStudy/documents
$ diff sort_example3.txt sort_example4.txt
3c3
< Gumi
---
> Korea
5,6c5,6
< Mungyeong
< Pohang
---
> Yeongju
> Kumoh
9,10c9,10
< Gumi
< Uljin
---
> Pohang
> Daegu
  
```

Gumi → Korea

Mungyeong → Yeongju  
Pohang → Kumoh

Gumi → Pohang  
Uljin → Daegu

- a : Add
- c : Change
- d : Delete

# diff

- nano 에디터로 sort\_example3.txt 파일을 열어서 아래와 같이 일부 내용을 수정한 후 sort\_example5.txt 파일로 저장한다.

```
MINGW64:/c/LinuxStudy/documents
GNU nano 7.2 sort_example5.txt
Seoul
Gumi
Sangju
Gumi
Andong
Mungyeong
Incheon
Busan
Andong
Jeju
Ulsan
```

# diff

## • 두 파일의 비교

- `diff sort_example3.txt sort_example5.txt` 명령을 이용하여 두 파일을 비교한다.

```
$ diff sort_example3.txt sort_example5.txt
0a1      } 두 번째 파일의 첫 번째 줄에 새로운 행 추가(a)
> Seoul  } ✓ → Seoul
6,7c7,8   } 첫 번째 파일의 6,7번째 행이
< Pohang } 두 번째 파일의 7,8번째 행으로 변경(c)
< Gyeongju } ✓ Pohang → Incheon
---      } ✓ Gyeongju → Busan
> Incheon
> Busan
9c10      } 첫 번째 파일의 9번째 행이
< Gumi    } 두 번째 파일의 10번째 행으로 변경
---      } ✓ Gumi → Jeju
> Jeju
11,12d11   } 첫 번째 파일의 11, 12번째 행이 11번째 행으로 삭제(d)
< Pohang  } ✓ Pohang → 삭제
< Sangju  } ✓ Sangju → 삭제
```

# diff -u

- “-u” 옵션을 사용하면, 두 파일의 비교 결과를 통일 포맷(unified format)으로 출력함.

- 기호 해석

- 처음 두 행의 내용: 비교 대상 파일의 이름과 변경 시각

- +, - : 추가/삭제된 행의 내용

- @@ ... @@ : 헤더(header)

- 변경된 위치와 범위를 표현

- 형식: @@ -시작 행 번호,수정된 행의 수 + 시작 행 번호,수정된 행의 수 @@

- (예) @@ -1,13 +1,13 @@

- ✓ 왼쪽 파일의 첫 번째 행부터 13개의 행이 포함된 영역이

- 오른쪽 파일의 첫 번째 행부터 13개의 행이 포함된 영역으로 변경됨.

```
$ diff -u sort_example3.txt sort_example4.txt
--- sort_example3.txt    2022-10-17 14:40:46.490989900 +0900
+++ sort_example4.txt    2022-10-17 19:57:16.507935200 +0900
@@ -1,13 +1,13 @@
    Gumi
    Sangju
-Gumi
+Korea
    Andong
-Mungyeong
-Pohang
+Yeongju
+Kumoh
    Gyeongju
    Andong
-Gumi
-Uljin
+Pohang
+Daegu
    Pohang
    Sangju
```

Gumi → Korea

Mungyeong → Yeongju  
Pohang → Kumoh

Gumi → Pohang  
Uljin → Daegu

# diff -u

- “-u” 옵션을 사용하면, 두 파일의 비교 결과를 통일 포맷(unified format)으로 출력함.

- 기호 해석

- 처음 두 행의 내용: 비교 대상 파일의 이름과 변경 시각

- +, - : 추가/삭제된 행의 내용

- @@ ... @@ : 헤더(header)

- 변경된 위치와 범위를 표현

- 형식: @@ -시작 행 번호,수정된 행의 수 + 시작 행 번호,수정된 행의 수 @@

- (예) @@ -1,12 +1,11 @@

- ✓ 왼쪽 파일의 첫 번째 행부터 12개의 행이 포함된 영역이

- 오른쪽 파일의 첫 번째 행부터 11개의 행이 포함된 영역으로 변경됨.

```
$ diff -u sort_example3.txt sort_example5.txt
--- sort_example3.txt      2024-11-04 22:22:54.192701900 +0900
+++ sort_example5.txt      2024-11-04 23:09:24.649041700 +0900
@@ -1,12 +1,11 @@
+Seoul  → Seoul 추가
Gumi
Sangju
Gumi
Andong
Mungyeong
-Pohang
-Gyeongju } Pohang → Incheon
+Incheon   } Gyeongju → Busan
+Busan
Andong
-Gumi
+Jeju } Gumi → Jeju
Uljin
-Pohang } Pohang → 삭제
-Sangju } Sangju → 삭제
```

# Q & A