자료구조 과제 4 – 정렬 보고서

20222663 권기영

**1. 대량의 데이터 생성하기**

**<문제 내용>**

학생의 학번, 이름, 주소의 정보를 담고 있는 데이터를 20000개 생성한다. 생성할 때, 정렬되어 있으면 안된다. 동일한 학번을 가진 학생은 없어야 하고, 이를 검증하여야 한다.

**<문제 해결 - python>**

**(1) 데이터 생성하기**

1) 학번, 이름, 주소 세 가지의 정보를 담을 수 있는 학생 클래스를 선언한다.

2) 알파벳이 들어있는 배열(대문자 + 소문자 총 52개) letters와, 0~9까지의 숫자가 들어있는 배열 digits을 생성한다(전부 char 형으로 저장)

3) 각 학번(2016 ~ 2023)마다 2500번씩 반복하면서, random.sample 메소드를 활용하여 배열 요소를 꺼내 임시 배열에 넣는다.. 학번은 digits 배열에서 4개를 꺼내고, 이름은 letters 배열에서 10개를 꺼내고, 주소는 letters 배열에서 50개를 꺼낸다.

4) join메소드를 활용하여 임시 배열을 문자열로 바꾸어 준다. 이 때, 학번은 4개만 꺼냈기 때문에, 문자열로 바꿀 때는 년도(2016 + i) + 4자리로 바꾸어서 저장한다. 또한 이름과 주소는 대문자만 가능하므로 upper함수를 활용하여 대문자로 바꾸어 저장한다.

5) 이렇게 만든 문자열 3개를 받아서 학생 객체를 만들고, student\_list에 학생 객체 20000개를 저장한다.

**6)** 학번은 겹치면 안되기 때문에, check\_wrap 배열에 학번을 저장하고, 새로 만들어지는 학번 in check\_wrap 인 경우에는 학번을 새로 생성한다.

**7)** 이렇게 만든 학생 정보는 학번을 기준으로 정렬되어 있다. 따라서 random.shuffle 메소드를 활용하여 배열을 무작위로 한번 섞어준다.

8) 파일에 20000개의 학생 객체의 정보를 학번, 이름, 주소 순서대로 쓴다.

**(2) 동일한 학번 없음 검증하기**

1) 20000개의 학생 데이터를 저장한 텍스트 파일을 열고, 한 줄씩 읽으면서 temp에 학번 8자리 문자열을 저장한다.

**2)** check\_list 배열을 만들고, temp가 check\_list에 있는지 없는지를 확인하면서, 없다면 temp를 check\_list에 넣고, 있다면 “ERROR!”를 출력한다.

3) 모든 학번이 다 다르다면 “all the information is different”를 출력한다.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

**2. 내장된 정렬 방법과 비교하기**

**<문제 내용>**

1번에서 만든 데이터를 내장 정렬 함수를 사용하여 정렬한다. 학번 순, 이름 순으로 정렬(오름차순)한다. 이때 걸리는 시간을 측정한다.

**<문제 해결 – python>**

**(1) 텍스트 파일에서 정보 읽고 저장하기**

1) 학번, 이름, 주소의 정보를 담을 수 있는 학생 클래스를 만든다.

2) f.readlines 메소드를 활용하여 텍스트 파일의 모든 정보를 개행을 기준으로 information\_list 배열에 모두 저장한다. Information\_list의 원소 각각은 텍스트 파일 한 줄이다.

**3)** 한 줄을 적절히 슬라이싱 하여 학생 객체의 멤버로 전달해준다. 학번 -> [9 : 17] 이름 -> [25 : 35] 주소 -> [46 : 96]

4) 이 과정을 information\_list 원소 전부(20000개)에 대해서 반복하면 텍스트 파일의 모든 정보가 학생 객체 20000개에 저장된다.

**(2) 저장한 정보 정렬하기**

**1)** lambda를 활용하여, 정렬할 때의 key를 학생 객체의 멤버로 설정해준 뒤, 오름차순 정렬을 수행한다..

2) key를 학생 객체의 학번으로 설정하고 정렬 수행, key를 학생 객체의 이름으로 설정하고 정렬 수행, 총 2번을 수행한다.

**(3) 정렬 시간 측정하기**

1) time.time 메소드를 활용하여 정렬 시작 지점의 시간(start)와 정렬 끝난 지점의 시간(end)을 저장한 뒤, 정렬에 걸린 시간(end – start)을 출력한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

측정할 때 마다 값이 조금씩 달라지긴 하지만, 기준이 학번인 것과 이름인 것의 정렬 모두 평균적으로 0.006~0,007초의 정렬 시간이 걸렸다.

**<문제 해결 – C>**

**(1) 텍스트 파일에서 정보 읽고 저장하기**

1) 학번, 이름, 주소의 정보를 담을 수 있는 학생 구조체를 만든다.

**2)** fgets 함수를 활용하여 한 줄을 str에 담고, for문을 활용하여 구조체 멤버 변수(문자열)에 직접 str의 문자 하나 하나씩 저장한다. 예를 들어, 학번의 경우 str[9] ~ str[17]까지 구조체 멤버에 저장한다.

3) 이 과정을 20000번 반복해서 학생 구조체 20000개에 텍스트 파일의 정보를 모두 저장한다.

**(2) 저장한 정보 정렬하기**

**1)** qsort 함수를 사용하여 학생 구조체 배열을 정렬한다.

2) 어떠한 기준으로 요소들을 비교할지에 대한 함수인 compare 함수를 학번 기준, 이름 기준으로 2가지를 만든다. compare\_number, compare\_name 이다.

**3)** 문자열을 비교할 것이기 때문에 strcmp 함수를 통해 두 요소를 비교하고, strcmp가 두 요소의 순서를 반환해주기 때문에 compare 함수의 반환값 또한 strcmp의 반환값으로 한다.

4) qsort로 정렬할 배열을 학생 구조체 배열로 전달하고, 학생 학번과 이름을 기준으로 비교하는 함수를 전달하여 2번 정렬을 수행한다.

**(3) 정렬 시간 측정하기**

1) time.h의 clock 함수를 사용하여 정렬 끝 시간 – 정렬 시작 시간 을 측정한다.

텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

학번을 기준으로 정렬했을 때는 평균적으로 0.007~0.008초 정도 소요되고, 이름을 기준으로 정렬했을 때는 0.006~0.007초 정도 소요되었다

**3. 선택 정렬/퀵 정렬/힙 정렬**

**<문제 내용>**

1번에서 생성한 데이터에 대해서, 선택 정렬, 퀵 정렬, 힙 정렬을 직접 만들어 수행한 후 결과를 비교한다. 데이터는 동일해야 하고, 정렬을 검증하는 수단을 사용해야 한다. 각 정렬에 소요되는 시간을 비교한다.

**<문제 해결 – python>**

**(1) 정렬 구현**

선택 정렬, 퀵 정렬, 힙 정렬 모두 교재의 방법과 동일하게 구현했으나, 사용하는 자료형과 기준이 달라지므로 몇 가지 추가된 사항이 있다.

**1)** 학번, 이름 총 두 가지 기준으로 정렬을 시행해야 하므로, 함수의 매개변수에 추가로 기준이 무엇인지를 전달하는 check 인자를 추가하였다. check == “number” 이면 학번으로 정렬, check == “name” 이면 이름으로 정렬한다.

**2)** 전달하는 자료형이 정수 배열에서 구조체로 바뀌었기 때문에, 정렬 또한 구조체의 멤버를 기준으로 정렬을 한 뒤, 구조체 전체를 swap해주는 식으로 바꾸었다.

3) 힙 정렬의 경우, 1번째 인덱스부터 자료를 저장하도록 구현하였다.

**(2) 정렬 수행 및 시간 측정**

1) 학번을 기준으로, 이름을 기준으로 총 두 번 정렬을 수행하였다. 이 때, 동일한 데이터를 가지고 정렬을 수행해야 하므로 학생 객체 배열을 복사하여 정렬을 수행해주었다.

2) 정렬에 걸리는 시간을 밀리 초 단위로 측정하였다(time 함수).

**(3) 정렬 여부 확인**

**1)** 파이썬에서 제공하는 all 함수를 활용하여, 정렬한 배열의 처음 요소부터 마지막 요소까지 인접한 요소끼리 차례대로 비교하면서, 하나라도 오름차순을 만족하지 못하면 False를 반환, 모든 요소가 오름차순을 만족하면 True를 반환하도록 하였다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

**<문제 해결 – C>**

**(1) 정렬 구현**

선택 정렬, 퀵 정렬, 힙 정렬 모두 교재의 방법과 동일하게 구현했으나, 사용하는 자료형과 기준이 달라지므로 몇 가지 추가된 사항이 있다.

1) 파이썬 에서와 마찬가지로, 정렬 기준을 전달받는 check 인자를 추가하였다.

**2)** C언어에서는 인자로 전달받은 값을 바꾸기 위해서는 주소 값을 전달받아야 하므로, 구조체 배열의 포인터를 인자로 전달하였다.

**3)** 구조체 배열의 멤버가 모두 문자열이기 때문에, 파이썬 에서와는 달리 strcmp 함수를 통해 순서를 비교하였다.

**(2) 정렬 수행 및 시간 측정**

1) 동일한 데이터에 대해서 정렬을 수행하기 위해, 구조체 배열을 복사하여 복사한 배열을 정렬하였다.

2) 정렬에 걸리는 시간을 밀리 초 단위로 측정하였다(clock 함수).

**(3) 정렬 여부 확인**

1) 구조체 배열 요소의 처음부터 끝까지 인접한 요소끼리 차례대로 비교하면서, 하나라도 오름차순을 만족하지 못하면 1을 반환하고, 모두 만족한다면 0을 반환하는 check\_sort함수를 만들어 정렬 여부를 확인하였다.

텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

<정렬 시간 정리 표>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 시간(초) | 선택, 학번 | 선택, 이름 | 퀵, 학번 | 퀵, 이름 | 힙, 학번 | 힙, 이름 |
| Python | 14.059 | 13.197 | 0.049 | 0.04 | 0.079 | 0.072 |
| C | 1.344 | 0.535 | 0.013 | 0.008 | 0.012 | 0.008 |

**<예측과 결과>**

**1)** 이름이 학번보다 더 길고, 정렬 함수 내에서도 학번이 아니면(else if) 이름을 기준으로 정렬하도록 구현했기 때문에 당연히 이름을 통해 정렬하는 수행 시간이 더 길 것이라고 예측했다.

-> 결과는 모든 정렬에서 이름을 기준으로 하는 것이 학번을 기준으로 하는 것 보다 수행 시간이 짧았다. 예측과 맞지 않는다.

-> 이에 대한 이유는, 데이터의 특성 및 문자열 비교 알고리즘과 연관이 있다고 생각한다. 우선 데이터의 특성상, 이름은 10개의 랜덤한 알파벳 문자열인데 반해, 학번은 초반 4자리(년도)에 규칙성이 있는 문자열이다. 그런데 문자열을 비교 알고리즘은, 순서가 결정되었다면 나머지 남은 문자열은 비교하지 않고 끝낸다. 따라서 학번은 초반 4자리가 비슷하고, 심지어 앞의 두 자리(20XX)는 무조건 똑같기 때문에 이름에 비해서 비교하는 횟수가 많아진 것이라고 생각한다.

2) C언어에서 구현한 정렬이 python에서 구현한 정렬보다 무조건 빠를 것이라고 예측했다.

-> 결과는 모든 정렬에서 C언어 속도 > python 속도였다. 예측과 일치한다.

-> C언어가 python보다 더 저급 언어이고, 파이썬은 객체 지향, 인터프리터 방식 등 느려질 수 있는 요소가 충분하기 때문에 그렇게 예측했다.

**4. 정렬 개선**

**<문제 내용>**

2번, 3번의 결과를 보고 정렬을 개선한다. 정렬 방법을 개선할 수도 있고, 저장 방법을 개선할 수도 있다. 모듈을 사용할 수도 있다.

**<문제 해결 - Python>**

**(1) 저장 자료형 바꾸기 – 학번**

학번은 숫자임에도 불구하고 문자열로 저장했고, 숫자라면 한 번의 연산에 끝날 비교가 여러번에 걸쳐 수행되었기 때문에 연산 속도가 느려지고, 심지어 이름을 기준으로 한 정렬보다도 더 느려지게 되었다.

->문자열로 저장되었던 학번을 정수형으로 바꿔서 저장한다.

텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |
| --- | --- | --- | --- |
| sort함수 | 선택 정렬 | 퀵 정렬 | 힙 정렬 |
| 0.003(-0.004) | 12.291(-1.768) | 0.042(-0.007) | 0.069(-0.01) |

학번을 정수형으로 바꿔서 저장했더니, 유의미한 시간 감소를 보였다. 퀵 정렬과 힙 정렬은 시행에 따라 큰 차이가 없었긴 하지만, sort함수와 선택 정렬에서는 확실히 일정한 감소를 보였다. 또한, 원래 학번을 기준으로 하는 정렬이 이름을 기준으로 하는 정렬보다 느렸는데, 정수로 바꾼 뒤에는 학번을 기준으로 하는 것이 더 빨라졌다.

**(2) 저장 방법 바꾸기 – 딕셔너리**

기준이 되는 값을 키로 가지는 딕셔너리로 학생 정보를 저장하고, 키 만을 저장하는 배열을 따로 만들어서 키 배열에 대해서만 정렬을 수행하게 되면, 딕셔너리를 직접 정렬하지 않고도 정렬된 구조를 얻을 수 있다. 키 값을 통해서 나머지 학생 정보를 알 수 있기 때문이다.

텍스트, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | sort함수 | 선택 정렬 | 퀵 정렬 | 힙 정렬 |
| 학번 기준 | 0.002(-0.005) | 6.959(-7.1) | 0.028(-0.021) | 0.049(-0.03) |
| 이름 기준 | 0.003(-0.003) | 7.917(-5.28) | 0.029(-0.011) | 0.052(-0.02) |

학번을 정수형으로 저장하는 것도 유지한 상태로 구현하였다.

원래는 구조체 전체를 정렬 했는데, 이를 키만 정렬하는 것으로 바꾸었다. 구조체 전체가 아닌 키 만을 정렬하는 방법을 사용하니 확실히 원래 정렬 시간보다 많이 개선됨을 알 수 있었다.