UNISTUDY

파이썬

14강. 자료구조(자료구조복제, 알고리즘)

1. 자료구조 복제

- 자료구조 복제(copy)란 객체의 주소를 복사하는 것을 의미한다. 객체의 주소를 복사하는 방법에는 객 체의 주소를 그대로 넘겨주는 얕은 복사와 객체의 내용만 넘겨주는 깊은 복사가 있다.
- ‘깊은 복사’ 용어 는 copy 모듈에서 제공하는 deepcopy() 함수 이름에 의해서 붙여진 이름이다.

1. 자료구조 복제

chapter04.lecture.step06_copy.py

```
# (1) 얕은 복사 : 주소 복사(내용, 주소 동일)
name = ['홍길동', '이순신', '강감찬']
print('name address =', id(name))

name2 = name # 주소 복사
print('name2 address =', id(name2))

print(name)
print(name2)

# 원본 수정
name2[0] = "김길동" # 원본/사본 수정
print(name)
print(name2)

# (2) 깊은 복사 : 내용 복사(내용 동일, 주소 다름)
import copy
name3 = copy.deepcopy(name)
print(name)


print(name3)

print('name address =', id(name))
print('name3 address =', id(name3))

# 원본 수정
name[1] = "이순신장군"
print(name)
print(name3)
```

Python Console

```
name address = 63068520

name2 address = 63068520

['홍길동', '이순신', '강감찬']
['홍길동', '이순신', '강감찬']

['김길동', '이순신', '강감찬']
['김길동', '이순신', '강감찬']

['김길동', '이순신', '강감찬']

['김길동', '이순신', '강감찬']

name address = 63068520
name3 address = 62961024

['김길동', '이순신장군', '강감찬']
['김길동', '이순신', '강감찬']
```

1. 자료구조 복제

- 해설 객체 주소 복사
- (1) 얕은 복사 : 주소 복사(내용, 주소 동일)
- ‘name2=name’ 명령문은 name이 참조하고 있는 주소를 name2 참조 변수에 넘겨주는 형식으로 주소가 복사된다. 따라서 객체의 주소가 동일하고 객체의 내용도 동이하다. 얕은 복사를 하면 원본 객체의 원소가 수정되면 사본 객체도 함께 수정된다.이러한 현상은 당연한 결과이다. 사실 name과 name2는 같은 객체의 주소를 참조하기 때문이다.
- (2) 깊은 복사 : 내용 복사(내용 동일, 주소 다름)
- copy.deepcopy(name) 명령문은 copy 모듈에서 제공하는 deepcopy() 함수에 의해서 name이 참조 하고 있는 내용만 넘겨주는 형식으로 복사된다. 따라서 name3의 참조변수가 참조하는 객체의 내용은 동일하지만 객체의 주소는 서로 다르다. 깊은 복사를 하면 원본 객체의 원소가 수정되면 사본 객체의 원소는 변경되지 않다. 이유는 원본과 사본은 서로 다른 주소를 참조하기 때문이다.

2. 알고리즘(algorithm)

- 알고리즘이란 어떤 문제를 해결하기 위한 일련의 절차를 말한다. 프로그래밍의 논리적인 절차를 의미 하는 로직의 기초가 된다. 예를 들면 외부에서 자료가 입력되는 경우 논리적인 순서에 의해서 자료를 처리하고, 처리 결과를 출력하는 절차 등을 의미한다.

- 일반적으로 알고리즘은 다음과 같은 조건을 만족해야 한다.
  · 입력 : 외부에서 제공되는 자료가 있을 수 있다.
  · 출력 : 적어도 한 가지 이상의 결과가 나올 수 있다.
  · 명백성 : 각 명령들은 애매한 부분 없이 간단명료해야 한다.
  · 유한성 : 반드시 종료 조건이 있어야 한다.
  · 효과성 : 모든 명령들은 명백하고 실행 가능한 것이어야 한다.

3. 알고리즘(algorithm) 최댓값/최솟값(max/min)

- 전체 자료의 원소 중에서 가장 큰 값과 가장 작은 값을 선별하는 가장 기본적인 알고리즘이다.

3. 알고리즘(algorithm) 최댓값/최솟값(max/min)

chapter04.lecture.step07_algo.py

```
# (1) 입력 자료 생성
import random
dataset = []
for i in range(10) :
    r = random.randint(1,100)
    dataset.append(r)
print(dataset)

# (2) 변수 초기화
vmax = vmin = dataset[0]

# (3) 최댓값/최솟값 구하기
for i in dataset:
    if vmax < i:
        vmax = i
    if vmin > i:
        vmin = I

# (4) 결과 출력
print('max =', vmax, ' min =', vmin)
```

Python Console

```
[93, 45, 90, 16, 97, 12, 6
5, 92, 69, 12]

max = 97 min = 12
```

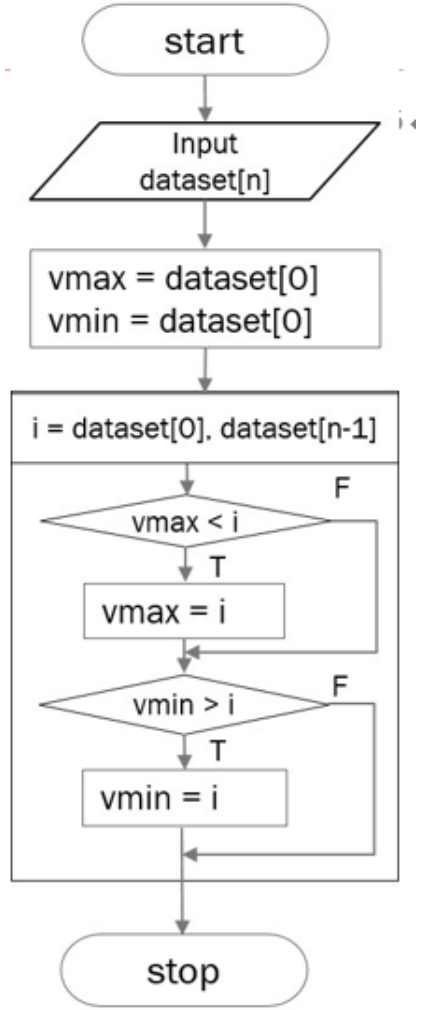
3. 알고리즘(algorithm) 최댓값/최솟값(max/min)

- 최댓값/최솟값 알고리즘 예
- (1) 입력 자료 생성
- 알고리즘에서 사용할 자료를 생성한다. 1~100 사이의 임의의 난수 정수를 10개를 생성하는 예문이다.
- (2) 변수 초기화
- 최댓값과 최솟값이 저장될 변수에 초기값으로 dataset의 첫 번째 원소를 두 변수에 똑같이 할당한다.
- (3) 최댓값/최솟값 구하기
- dataset의 전체 원소를 하나씩 비교하여 최댓값/최솟값을 구한다. 최댓값 변수(vmax) 보다 더 큰 원소가 나타나면 해당 값으로 교체하고, 최솟값 변수(vmin) 보다 더 작은 원소가 나타나면 해당 값으로 교체한다.
- (4) 결과 출력
- 최댓값과 최솟값을 출력하고 프로그램을 종료한다. [그림] 4-5는 최댓값/최솟값 알고리즘의 순서도이다.

3. 알고리즘(algorithm) 정렬(sort)

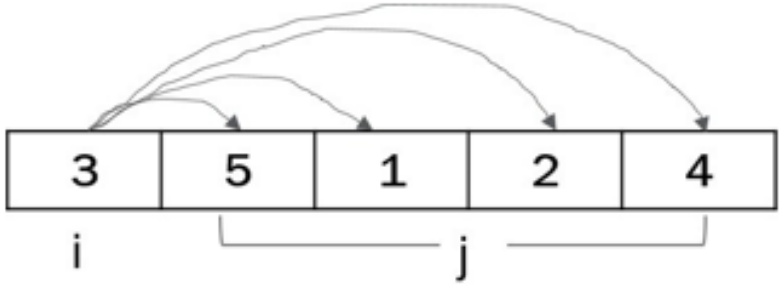
- 전체 자료의 원소를 일정한 순서로 나열하는 알고리즘이다. 예를 들면 숫자 자료를 대상으로 작은 수에서 큰 수 또는 큰 수에서 작은 수 순서대로 나열하는 논리적인 절차를 의미한다.
- 여기서 작은 수에서 큰 수로 정렬하는 방식을 오름차순(ascending sort)이라고 하고, 큰 수에서 작 은 수로 정렬하는 방식을 내림차순(descending sort)이라고 한다.
- 정렬 알고리즘은 선택정렬(selection sort), 버블정렬(bubble sort) 등 이 있다. 여기서는 특정 원소를 기준으로 나머지 모든 원소를 비교해가 면서 정렬하는 선택 정렬 알고리즘을 예문으로 알아본다.

3. 알고리즘(algorithm) 정렬(sort)



4. 알고리즘(algorithm) 선택정렬(sort)

- 선택정렬은 첫 번째 원소를 기준으로 나머지 모든 원소를 비교하여 정렬하는 방식으로 알고리즘 수행 가정이다.



- 첫 번째 원소를 변수 i로 지정하고, 나머지 원소를 변수 j로 지정할 때 i 변수는 기준 변수가 되고, j변수는 비교변수가 된다. 정렬 대상의 원소가 n개 이면 n-1 회전에 의해서 모든 원소가 정렬된다. 따라서 5개의 원소는 4회전에 의해서 모든 원소가 정렬된다.
- 선택 정렬 방 식으로 오름차순 정렬하는 과정을 각 회전별로 나타내고 있다.

4. 알고리즘(algorithm) 선택정렬(sort)

- 정렬과정을 살펴보면 각 회전이 종료될 때 마다 전체 원소 중에서 가장 작은 값이 왼쪽의 첫 번째로 옮 겨지는 것을 알 수 있다.
- 따라서 다음 회전에서는 기준변수의 위치가 오른쪽으로 한 칸씩 이동 된다. 기 준변수는 n-1이 될 때 까지 한 칸씩 이동한다.
- 한편 비교변수의 시작은 항상 기준변수 보다 하나 더 큰 위치(i+1)부터 시작해서 n까지 한 칸씩 이동한다.
- 여기서 기준변수가 가리키는 원소를 대상으로 나머 지 모든 원소와 비교하기 위해서는 중첩 반복문을 이용하면 된다.
- 즉 바깥쪽 반복문에서 기준변수를 만 들고, 안쪽 반복문에서 비교변수를 만들어서 두 변수가 가리키는 원소를 비교 판단하여 정렬한다.
- 다음 은 선택 정렬 알고리즘을 적용하여 오름차순과 내림차순으로 정렬하는 과정을 구현한 예문이다.

4. 알고리즘(algorithm) 선택정렬(sort)

회전	정렬 내용
1회전	[1, 5, 3, 2, 4]
2회전	[1, 2, 5, 3, 4]
3회전	[1, 2, 3, 5, 4]
4회전	[1, 2, 3, 4, 5]

4. 알고리즘(algorithm) 선택정렬(sort)

chapter04.lecture.step07_algo.py

```
# (1) 오름차순 정렬
dataset = [3, 5, 1, 2, 4]
n = len(dataset)
for i in range(0, n-1) : # 1 ~ n-1
    for j in range(i+1, n) : # i+1 ~ n
        if dataset[i] > dataset[j] :
            tmp = dataset[i]
            dataset[i] = dataset[j]
            dataset[j] = tmp
    print(dataset) # 각 회전 정렬내용

print(dataset) # [1, 2, 3, 4, 5]

# (2) 내림차순 정렬
dataset = [3, 5, 1, 2, 4]
n = len(dataset)
for i in range(0, n-1) : # 1 ~ n-1
    for j in range(i+1, n) : # i+1 ~ n
        if dataset[i] < dataset[j] :
            tmp = dataset[i]


            dataset[i] = dataset[j] # 3 5
            dataset[j] = tmp
    print(dataset) # 각 회전 정렬내용

print(dataset) # [5, 4, 3, 2, 1]
```

Python Console

```
[1, 5, 3, 2, 4]
[1, 2, 5, 3, 4]
[1, 2, 3, 5, 4]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[5, 3, 1, 2, 4]
[5, 4, 1, 2, 3]
[5, 4, 3, 1, 2]
[5, 4, 3, 2, 1]
[5, 4, 3, 2, 1]
```

4. 알고리즘(algorithm) 선택정렬(sort)

- 선택정렬 알고리즘 예
- (1) 오름차순 정렬
- 5개의 원소를 갖는 dataset 변수를 대상으로 오름차순 정렬하는 과정이다. dataset의 전체 원소의 길이를 변수 n에 할당하고, 이를 이용하여 바깥쪽 반복문에서 기준변수 i를 만들고, 안쪽 반복문에서 비교변수 j를 만든다. 오름차순 정렬이기 때문에 기준변수의 값과 비교변수의 값을 비교하여 기준변수 의 값 보다 더 작은 값이 나타나면 기준변수의 값과 비교변수의 값을 서로 교체한다.
- (2) 내림차순 정렬
- 내림차순 정렬은 오름차순 정렬에서 이용된 조건식만 if dataset[i] < dataset[j] 으로 변경하면 된다. 즉 기준변수의 값 보다 더 큰 비교변수의 값이 나타나면 기준변수의 값과 비교변수의 값을 서로 교체한다.

5. 검색(search)

- 검색 알고리즘은 크게 순차검색과 이진검색 방식이 있다. 알고리즘의 시간 복잡도를 비교할 때 순차 검색 알고리즘 보다 이진검색 알고리즘이 훨씬 좋다.
- 즉 이진검색 알고리즘이 검색 속도가 빠르다는 의 미이다. 하지만 이진검색 알고리즘은 자료가 정렬되어 있어야 한다는 전제 조건이 있다.
- 따라서 검색 속도만 보면 이진검색이 빠르지만 정렬에 필요한 시간 복잡도까지 고려해서 적합한 알고리즘을 선택해야한다.
- 여기서는 이진 검색 알고리즘에 대해서 알아본다.

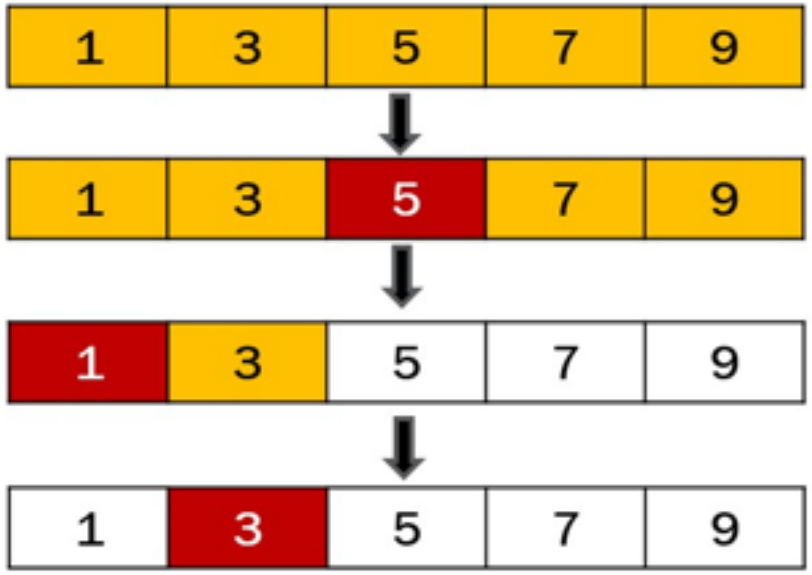
6. 이진검색(binary search)

- 이진검색은 전체 원소가 정렬된 상태에서 중앙 위치(mid)을 계산하여 절반은 버리고, 나머지 절반을 대상으로 검색을 수행하는 알고리즘이다.
- 5개의 원소가 오름차순 정렬된 상태에서 이진 검색으로 3의 위치를 찾는 과정이다. start 변수는 검색 대상의 첫 번째 원소의 위치이고, end는 마지막 원소의 위치를 갖는 변수이다.
- 알고리즘의 시작은 start와 end 변수를 이용하여 정 중앙 위치를 갖는 mid 변수를 계산한다. mid의 위치에는 5, 찾는 값은 3이기 때문에 조건식 (5 > 3)과 같다. 따라서 mid 위치를 기준으로 오른쪽 절반은 검색 대상에서 제외시키기 위해서 end=mid-1 수식으로 마지막 원소의 위치를 변경한다.

6. 이진검색(binary search)

- 두 번째 중앙 위치를 갖는 mid 변수는 (1+3/2=1.5)로 계산되는데, 색인은 실수를 허용하지 않기 때 문에 1로 정수화한다.
- mid 위치에는 1, 찾는 값은 3이기 때문에 조건식 (1 < 3)과 같다. 따라서 mid 위치를 기준으로 왼쪽 절반은 검색 대상에서 제외시키기 위해서 start=mid+1 수식으로 첫 번째 원소 의 위치를 변경한다.
- 세 번째 중앙 위치를 갖는 mid 변수는 (2+3/2=2.5)로 계산되어 2로 정수화한 다. 마침내 mid의 위치에 찾는 값 3을 발견할 수 있다.

6. 이진검색(binary search)



start=1, end = 5

mid = (start + end) / 2 →(3)

end=mid-1
mid = (start + end) / 2 →1(1.5)

start=mid+1
mid = (start + end) / 2 → 2(2.5)

6. 이진검색(binary search)

chapter04.lecture.step07_algo.py

```
dataset = [5, 10, 18, 22, 35, 55, 75, 103]
value = int(input("검색할 값 입력 : "))

low = 0 # start 위치
high = len(dataset) - 1 # end 위치
loc = 0
state = False # 상태 변수

while (low <= high):
    mid = (low + high) // 2


    if dataset[mid] > value: 중앙값이 큰 경우
        high = mid - 1
    elif dataset[mid] < value: # 중앙값이 작은 경우
        low = mid + 1
    else: # 찾은 경우
        loc = mid
        state = True
        break # 반복 exit


if state :
    print('찾은 위치 : %d 번째' %(loc+1))
else :
    print('찾는 값은 없습니다.')
```

Python Console

```
검색할 값 입력 : >? 55
찾은 위치 : 6 번째

검색할 값 입력 : >? 16
찾는 값은 없습니다.
```

6. 이진검색(binary search)

- 이진검색 알고리즘 예
- 오름차순 정렬된 8개의 원소를 갖는 dataset 변수를 대상으로 키보드로 입력한 값을 이진 검색 알고리즘으로 찾는 과정이다. low는 시작 위치, high는 끝 위치를 갖는 변수이다. loc 변수는 찾은 값의 색인을 저장할 변수이고, state는 값의 유무를 알려주는 상태변수이다.
- while은 시작 위치가 끝 위치보다 작거나 같으면 반복을 수행한다. 반복문에서 mid는 중앙 위치를 계산하는 변수이고, mid 변수를 색인으로 dataset에 찾을 값(value)이 있으면 loc에 색인을 넘겨주 고, state는 True 상태로 바꾼다. 만약 찾는 값(value) 보다 중앙값이 크거나 작으면 high와 low를 수정한다. 이러한 과정을 반복하다가 조건식이 거짓이면 반복을 멈춘다. 반복문이 종료된 이후 if문에 의해서 비교 판단한다. state가 True이면 찾는 값이 있다는 의미로 값의 위치를 출력하 고, False이면 찾는 값이 없다는 의미이다.

THANK YOU