>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 6

재귀와 반복 : 검색

검색

Searching

정보가 모여 있는 곳에서 원하는 정보를 찾는 작업

유일무이하게 구별할 수 있는 검색용 키key로 검색

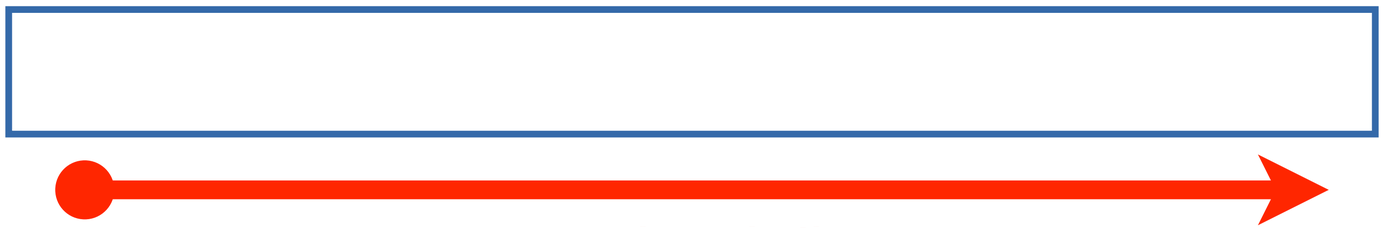
검색

Searching

정보가 모여 있는 곳에서 원하는 정보를 찾는 작업

유일무이하게 구별할 수 있는 검색용 키key로 검색

정렬 X



순차 검색

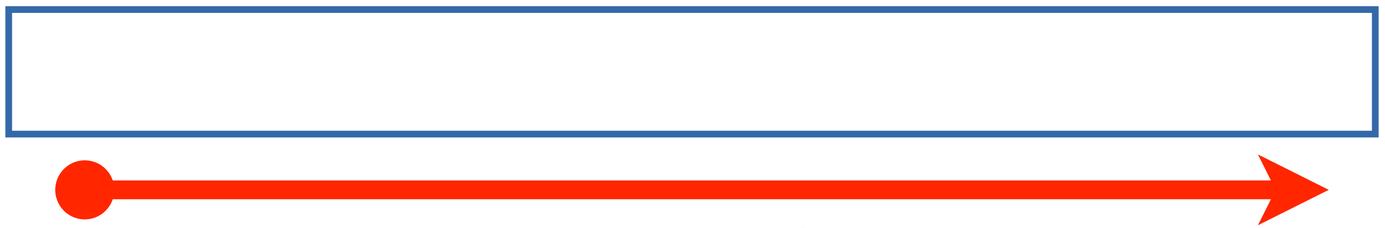
검색

Searching

정보가 모여 있는 곳에서 원하는 정보를 찾는 작업

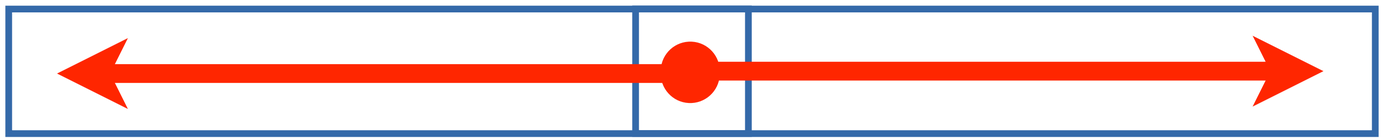
유일무이하게 구별할 수 있는 검색용 키key로 검색

정렬 X



순차 검색

정렬 O



이분 검색

검색

Searching

정보가 모여 있는 곳에서 원하는 정보를 찾는 작업

유일무이하게 구별할 수 있는 검색용 키key로 검색

시퀀스 검색

연산	의미
s.index(x)	시퀀스 s에서 가장 앞에 있는 키 x의 위치번호를 리턴

CHAPTER 6

재귀와 반복 : 검색

정렬 X

순차 검색

리스트 검색 : 존재 유무 확인

"키 x가 리스트 s에 있는가?"

객관식 OX 문제

입력 (파라미터) : 키의 리스트 s , 검색할 키 x

출력 (리턴) : x가 s에 있으면 `True`

없으면 `False`

순차 검색 알고리즘

Sequential Search

리스트 s에서 키 x를 검색하려면		
반복 조건	`s != []`	• s의 선두원소 s[0]가 x와 같으면, 찾았으므로 True를 리턴 • 그렇지 않으면, s의 후미리스트 s[1:]에서 키 x를 재귀로 검색
종료 조건	`s == []`	• 검색 대상이 없으므로 False를 리턴

리스트 s에서 키 x를 검색하려면		
반복 조건	`s != []`	• s의 선두원소 s[0]가 x와 같으면, 찾았으므로 True를 리턴 • 그렇지 않으면, s의 후미리스트 s[1:]에서 키 x를 재귀로 검색
종료 조건	`s == []`	• 검색 대상이 없으므로 False를 리턴

리스트 s에서 키 x를 검색하려면		
반복 조건	`s != []`	• s의 선두원소 s[0]가 x와 같으면, 찾았으므로 True를 리턴 • 그렇지 않으면, s의 후미리스트 s[1:]에서 키 x를 재귀로 검색
종료 조건	`s == []`	• 검색 대상이 없으므로 False를 리턴

리스트 s에서 키 x를 검색하려면		
반복 조건	`s != []`	• s의 선두원소 s[0]가 x와 같으면, 찾았으므로 True를 리턴 • 그렇지 않으면, s의 후미리스트 s[1:]에서 키 x를 재귀로 검색
종료 조건	`s == []`	• 검색 대상이 없으므로 False를 리턴

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],4)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],4)
=> seq_search_OX([5,4,2],4)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],4)
=> seq_search_OX([5,4,2],4)
=> seq_search_OX([4,2],4)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],4)
=> seq_search_OX([5,4,2],4)
=> seq_search_OX([4,2],4)
=> True
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],6)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],6)
=> seq_search_OX([5,4,2],6)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],6)
=> seq_search_OX([5,4,2],6)
=> seq_search_OX([4,2],6)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],6)
=> seq_search_OX([5,4,2],6)
=> seq_search_OX([4,2],6)
=> seq_search_OX([2],6)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],6)
=> seq_search_OX([5,4,2],6)
=> seq_search_OX([4,2],6)
=> seq_search_OX([2],6)
=> seq_search_OX([],6)
=>
```

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

```
   seq_search_OX([3,5,4,2],6)
=> seq_search_OX([5,4,2],6)
=> seq_search_OX([4,2],6)
=> seq_search_OX([2],6)
=> seq_search_OX([],6)
=> False
```

꼬리 재귀

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

while 루프

code : 6-2.py

```
1  def seq_search_OX(s,x):
2      while s != []:
3          if s[0] == x:
4              return True
5          else:
6              s = s[1:]
7      return False
```

code : 6-2.py

while 루프

```
1  def seq_search_OX(s,x):
2      while s != []:
3          if s[0] == x:
4              return True
5          else:
6              s = s[1:]
7      return False
```

while
루프

code : 6-2.py

```
1  def seq_search_OX(s,x):
2      while s != []:
3          if s[0] == x:
4              return True
5          else:
6              s = s[1:]
7      return False
```

for
루프

code : 6-3.py

```
1  def seq_search_OX(s,x):
2      for key in s:
3          if key == x:
4              return True
5      return False
```

꼬리 재귀

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

논리식

code : 6-4.py

```
1  def seq_search_OX(s,x):
2      return s != [] and (s[0] == x or seq_search_OX(s[1:],x))
```

리스트 검색 : 위치 인덱스 찾기

"키 x가 리스트 s의 어디에 있는가?"

주관식 문제

입력 (파라미터) : 키의 리스트 s , 검색할 키 x

출력 (리턴) : x가 s에 있으면 x의 위치 인덱스

없으면 None

존재유무 확인

code : 6-1.py

```
1  def seq_search_OX(s,x):
2      if s != []:
3          if s[0] == x:
4              return True
5          else:
6              return seq_search_OX(s[1:],x)
7      else:
8          return False
```

위치 인덱스 찾기

code : 6-5.py

```
1  def seq_search(s,x):
2      def loop(s,i):
3          if s != []:
4              if s[0] == x:
5                  return i
6              else:
7                  return loop(s[1:],i+1)
8          else:
9              return None
10     return loop(s,0)
```

위치 인덱스 찾기

code : 6-5.py

```
1  def seq_search(s,x):
2      def loop(s,i):
3          if s != []:
4              if s[0] == x:
5                  return i
6              else:
7                  return loop(s[1:],i+1)
8          else:
9              return None
10     return loop(s,0)
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
seq_search([3,5,4,2],4)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
    seq_search([3,5,4,2],4)
=>  loop(0)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],4)
=> loop(0)
=> loop(1)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],4)
=> loop(0)
=> loop(1)
=> loop(2)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],4)
=> loop(0)
=> loop(1)
=> loop(2)
=> 2
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=> loop(0)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=> loop(0)
=> loop(1)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=> loop(0)
=> loop(1)
=> loop(2)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=> loop(0)
=> loop(1)
=> loop(2)
=> loop(3)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=> loop(0)
=> loop(1)
=> loop(2)
=> loop(3)
=> loop(4)
=>
```

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

```
   seq_search([3,5,4,2],6)
=> loop(0)
=> loop(1)
=> loop(2)
=> loop(3)
=> loop(4)
=> None
```

꼬리재귀

code : 6-6.py

```
1  def seq_search(s,x):
2      def loop(i):
3          if i < len(s):
4              if s[i] == x:
5                  return i
6              else:
7                  return loop(i+1)
8          else:
9              return None
10     return loop(0)
```

while 루프

code : 6-7.py

```
1  def seq_search(s,x):
2      i = 0
3      while i < len(s):
4          if s[i] == x:
5              return i
6          else:
7              i = i + 1
8      return None
```

while 루프

code : 6-7.py

```
1  def seq_search(s,x):
2      i = 0
3      while i < len(s):
4          if s[i] == x:
5              return i
6          else:
7              i = i + 1
8      return None
```

for 루프

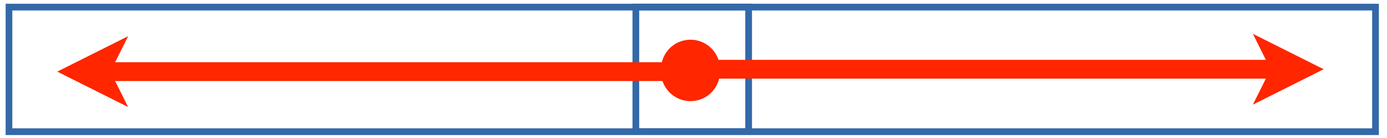
code : 6-8.py

```
1  def seq_search(s,x):
2      for i in range(len(s)):
3          if s[i] == x:
4              return i
5      return None
```

CHAPTER 6

재귀와 반복 : 검색

정렬 O



이분 검색

정렬된 리스트 검색 : 존재 유무 확인

"키 x가 리스트 s에 있는가?"

객관식 OX 문제

입력 (파라미터) : 키의 리스트 s , 검색할 키 x

출력 (리턴) : x가 s에 있으면 `True`

없으면 `False`

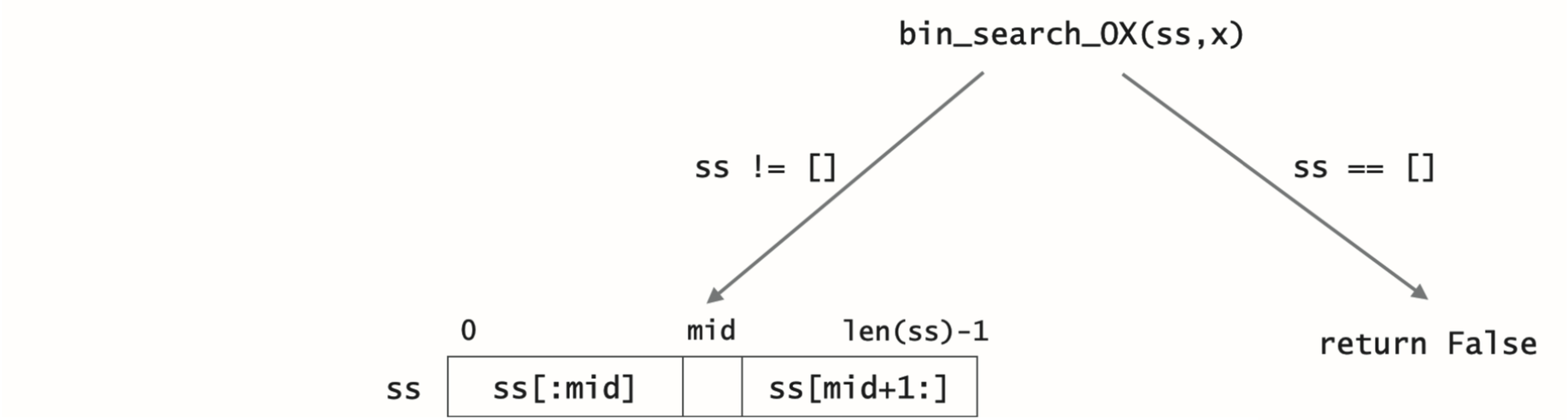
이분 검색 알고리즘

Binary Search

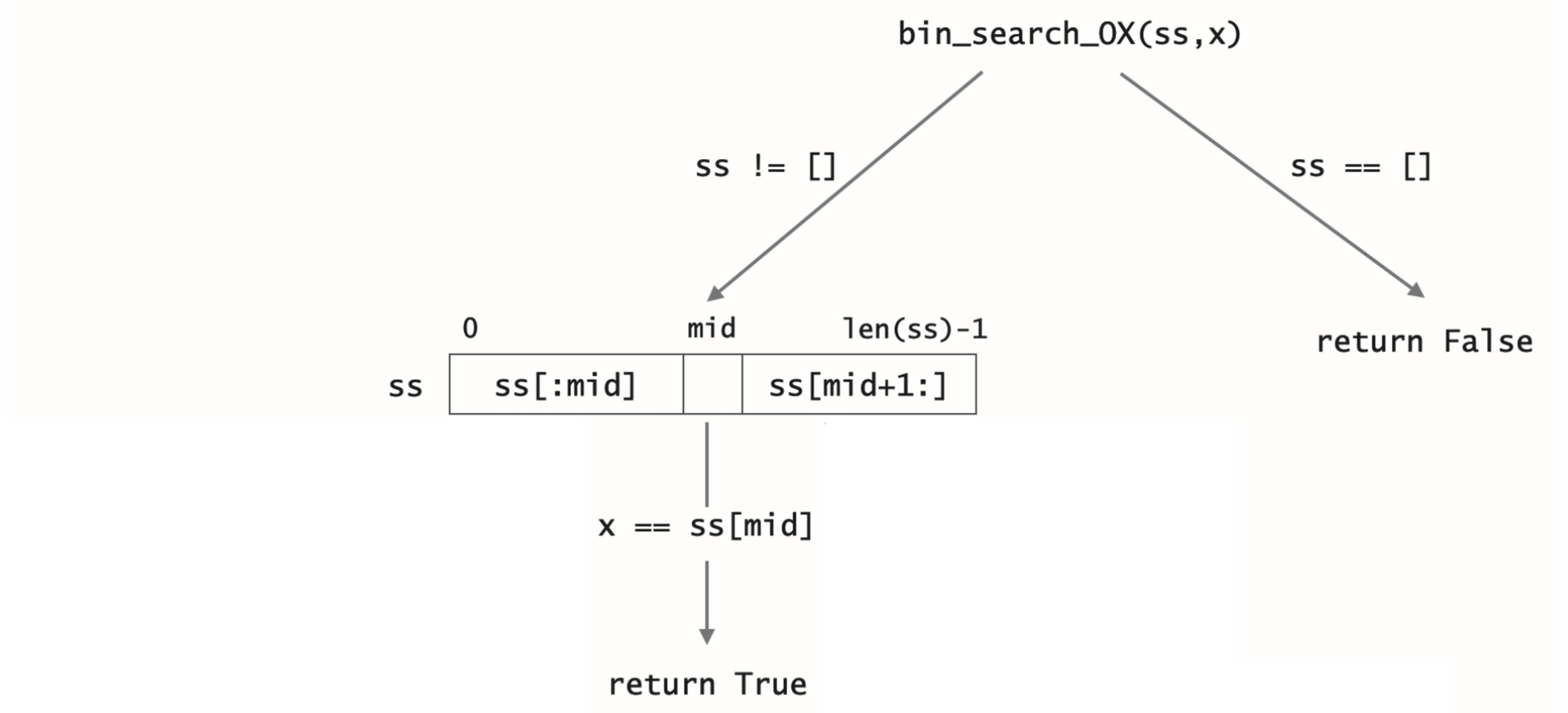
정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	`ss != []`	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	`ss == []`	• 검색 대상이 없으므로 False를 리턴

정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	ss != []	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	ss == []	• 검색 대상이 없으므로 False를 리턴

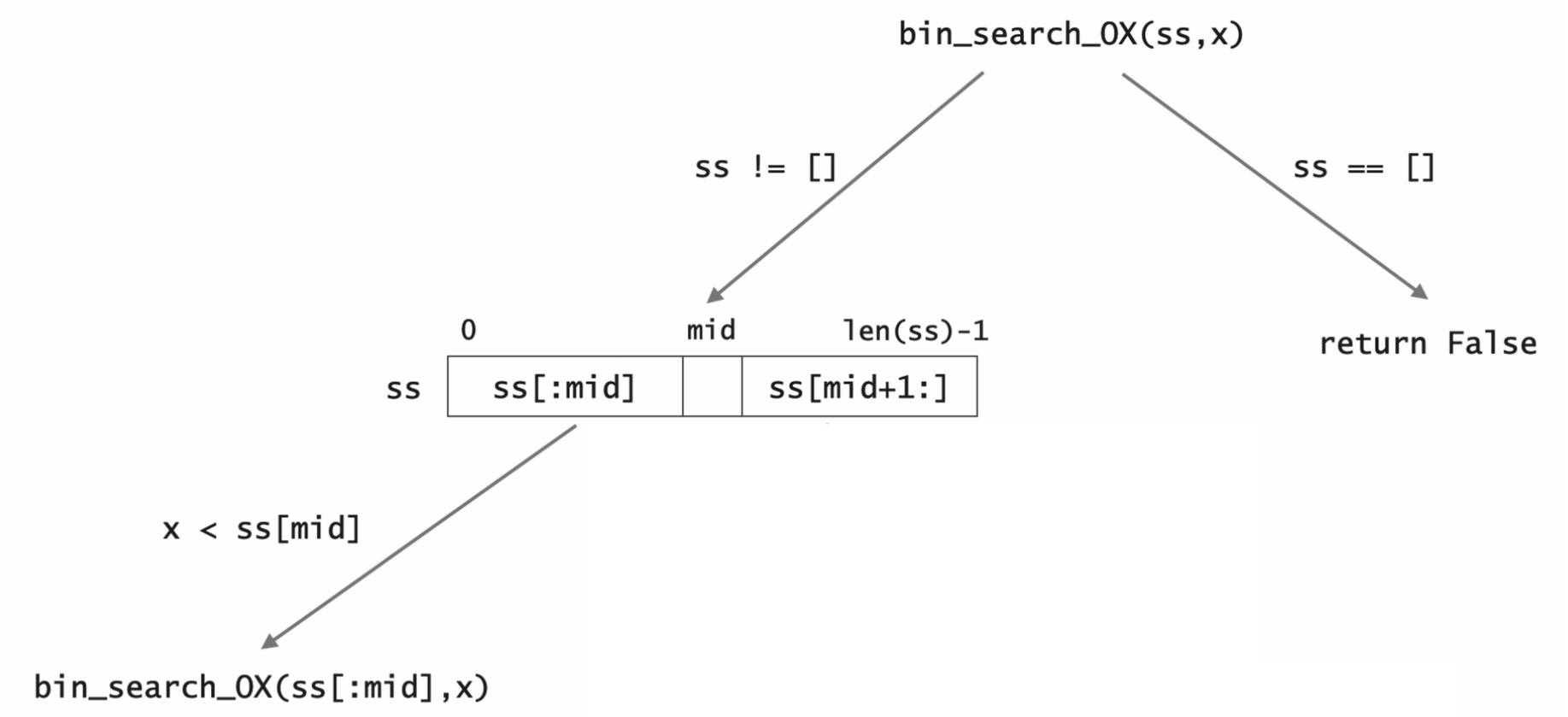
정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	ss != []	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	ss == []	• 검색 대상이 없으므로 False를 리턴



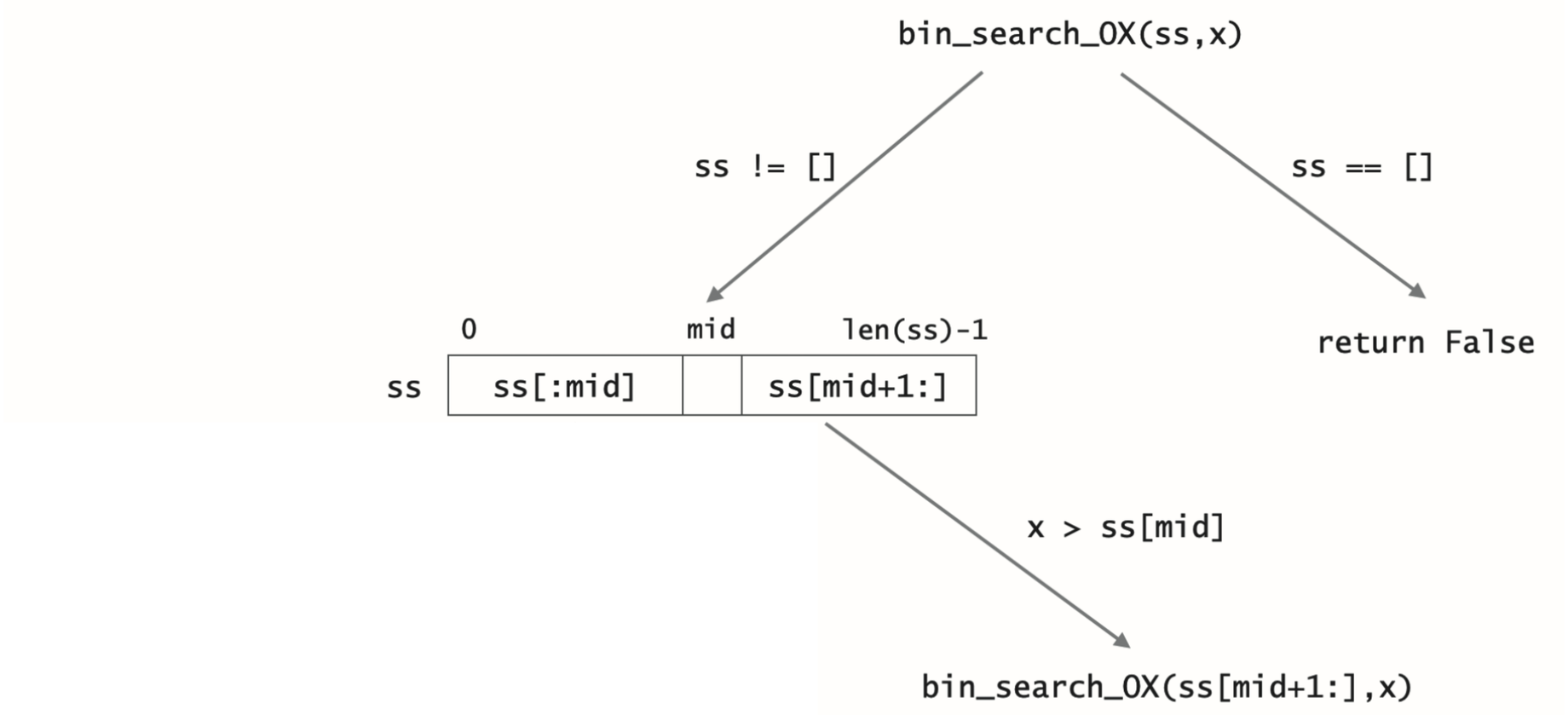
정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	ss != []	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	ss == []	• 검색 대상이 없으므로 False를 리턴



정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	ss != []	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	ss == []	• 검색 대상이 없으므로 False를 리턴



정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	ss != []	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	ss == []	• 검색 대상이 없으므로 False를 리턴



정렬된 리스트 ss에서 키 x를 검색하려면		
반복 조건	ss != []	• ss의 가운데 원소의 인덱스를 mid로 지정하고, • x가 ss[mid]와 같으면, 찾았으므로 True를 리턴 • x가 ss[mid]보다 작으면, ss[:mid]에서 x를 재귀로 검색 • x가 ss[mid]보다 크면, ss[mid+1:]에서 x를 재귀로 검색
종료 조건	ss == []	• 검색 대상이 없으므로 False를 리턴

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
bin_search_OX([1,2,3,4,5,6,7,8,9],5)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],5)
=> True
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
bin_search_OX([1,2,3,4,5,6,7,8,9],1)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],1)
=> bin_search_OX([1,2,3,4],1)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],1)
=> bin_search_OX([1,2,3,4],1)
=> bin_search_OX([1,2],1)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],1)
=> bin_search_OX([1,2,3,4],1)
=> bin_search_OX([1,2],1)
=> bin_search_OX([1],1)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],1)
=> bin_search_OX([1,2,3,4],1)
=> bin_search_OX([1,2],1)
=> bin_search_OX([1],1)
=> True
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
bin_search_OX([1,2,3,4,5,6,7,8,9],8)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],8)
=> bin_search_OX([6,7,8,9],8)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],8)
=> bin_search_OX([6,7,8,9],8)
=> True
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
bin_search_OX([1,2,3,4,5,6,7,8,9],11)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],11)
=> bin_search_OX([6,7,8,9],11)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],11)
=> bin_search_OX([6,7,8,9],11)
=> bin_search_OX([9],11)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],11)
=> bin_search_OX([6,7,8,9],11)
=> bin_search_OX([9],11)
=> bin_search_OX([],11)
=>
```

code : 6-9.py

```
1  def bin_search_OX(ss,x):
2      if ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              return bin_search_OX(ss[:mid],x)
8          else:
9              return bin_search_OX(ss[mid+1:],x)
10     else:
11         return False
```

```
   bin_search_OX([1,2,3,4,5,6,7,8,9],11)
=> bin_search_OX([6,7,8,9],11)
=> bin_search_OX([9],11)
=> bin_search_OX([],11)
=> False
```

code : 6-9.py

```python
def bin_search_OX(ss,x):
    if ss != []:
        mid = len(ss) // 2
        if x == ss[mid]:
            return True
        elif x < ss[mid]:
            return bin_search_OX(ss[:mid],x)
        else:
            return bin_search_OX(ss[mid+1:],x)
    else:
        return False
```

꼬리재귀

code : 6-10.py

```python
def bin_search_OX(ss,x):
    while ss != []:
        mid = len(ss) // 2
        if x == ss[mid]:
            return True
        elif x < ss[mid]:
            ss = ss[:mid]
        else:
            ss = ss[mid+1:]
    return False
```

while 루프

실습 6.1 논리 연산자로 흐름 제어

code : 6-10.py

```
1  def bin_search_OX(ss,x):
2      while ss != []:
3          mid = len(ss) // 2
4          if x == ss[mid]:
5              return True
6          elif x < ss[mid]:
7              ss = ss[:mid]
8          else:
9              ss = ss[mid+1:]
10     return False
```

code : 6-11.py

```
1  def bin_search_OX(ss,x):
2      mid = len(ss) // 2
3      return ss != [] and \
4             (x == ss[mid] or \
5                 _____________________________ or \
6                 _____________________________ )
```

정렬된 리스트 검색 : 위치 인덱스 찾기

"키 x가 리스트 s의 어디에 있는가?"

주관식 문제

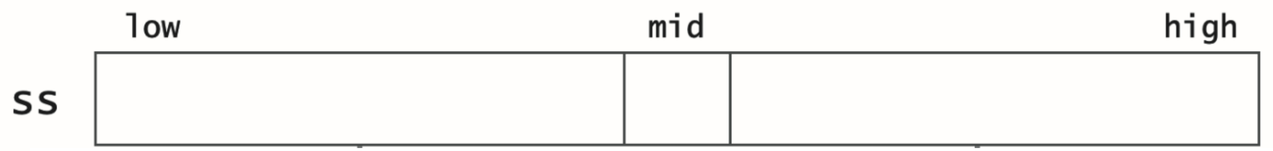
입력 (파라미터) : 키의 리스트 s , 검색할 키 x

출력 (리턴) : x가 s에 있으면 x의 위치 인덱스

없으면 None

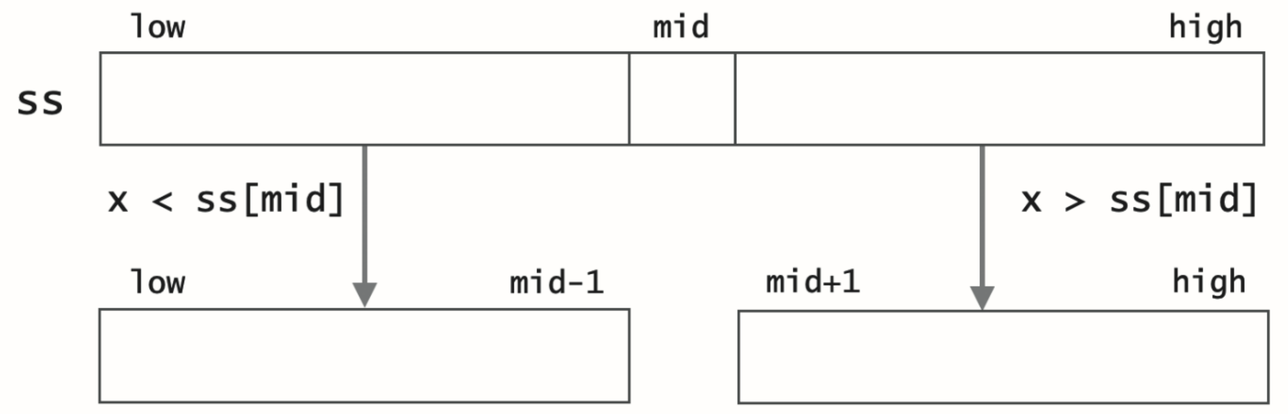
이분 검색 알고리즘
Binary Search

```
low = 0
high = len(ss) - 1

mid = (low + high) // 2
```



이분 검색 알고리즘
Binary Search

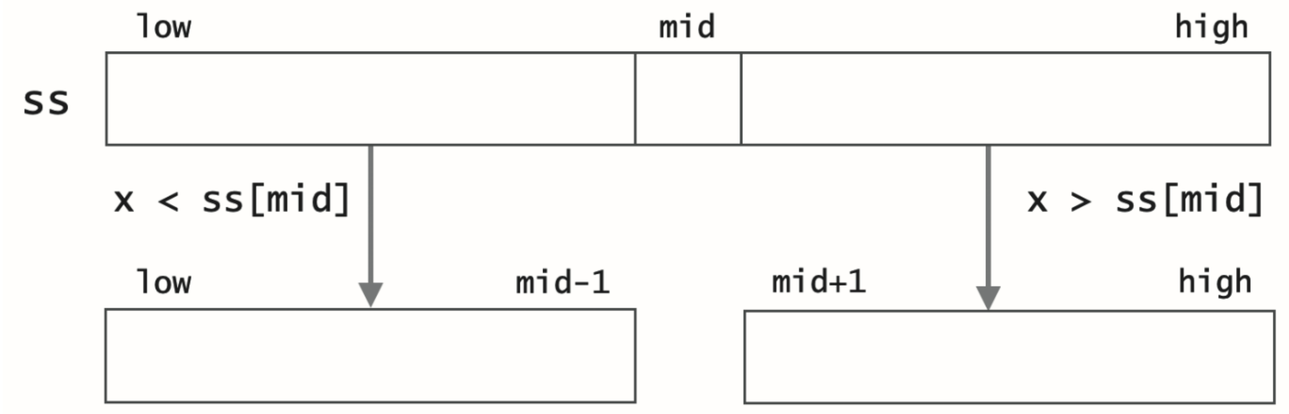
```
low = 0
high = len(ss) - 1

mid = (low + high) // 2
```



이분 검색 알고리즘
Binary Search

```
low = 0
high = len(ss) - 1

mid = (low + high) // 2
```

반복조건 `low <= high`



종료조건 `low > high`

CHAPTER 6

재귀와 반복 : 검색

순차 검색과 이분 검색의 성능 비교

리스트 길이	순차검색의 비교횟수	이분검색의 비교횟수
28	28	5
1,024	1,024	11
1,048,576	1,048,576	21
4,294,967,296	4,294,967,296	33
	n	n	$log_2 n$

성능 대결

`seq_search` **vs.** `bin_search`

random 모듈

연산	의미	용도
random.sample(population,k)	시퀀스 population에서 중복없이 k개를 무작위로 골라 리스트로 모아서 리턴한다.	검색 대상 리스트 만들기
random.randrange(n)	range(n) 정수범위에서 정수 하나 무작위로 골라서 리턴한다.	검색 키 뽑기

time 모듈

연산	의미	용도
time.perf_counter()	호출 시점의 절대 시각을 가능한 한 최대로 정밀하게 초 단위로 리턴	실행 시간 측정

code : 6-14.py

```
from random import sample, randrange
from time import perf_counter

print("Preparing data for seq_search. Please, wait a moment ...")
data = sample(range(10000000),8000000)

print("Testing seq_search function begins ...")
for _ in range(10):
    x = randrange(10000000)
    start = perf_counter()
    index = seq_search(data, x)
    finish = perf_counter()
    print(x, "is found at", index, "in", finish - start, "seconds")

print("Preparing data for bin_search. Please, wait a moment ...")
data.sort()

print("Testing bin_search function begins ...")
for _ in range(10):
    x = randrange(10000000)
    start = perf_counter()
    index = bin_search(data, x)
    finish = perf_counter()
    print(x, "is found at", index, "in", finish - start, "seconds")
```

CHAPTER 6

재귀와 반복 : 검색

파일

File

텍스트 파일

Text File

파일에 쓰기

```
t = open("sample.txt", "w")
```


```
t.close()
```

파일 접근 모드

접근 모드	의미
"r"	파일에서 읽음 (파일이 없으면 오류)
"w"	파일에 씀 (파일이 있으면 지우고 새로 씀, 없으면 새로 만듦)
"x"	새로 생성한 파일에 씀 (파일이 있으면 오류)
"a"	파일의 뒤에 이어서 씀 (파일이 없으면 새로 만듦)
"r+"	파일에서 읽고 씀 (파일이 없으면 오류)
"w+"	파일에 쓰고 읽음 (파일이 있으면 지우고 새로 씀, 없으면 새로 만듦)
"a+"	파일의 뒤에 이어서 쓰고 읽음 (파일이 없으면 새로 만듦)

파일 메소드

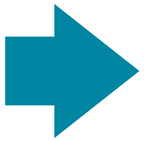
메소드	실행 의미
close()	파일을 닫는다. 일단 닫힌 파일은 다시 열기 전에는 읽거나 쓸 수 없다.
read(n)	파일의 현재 위치에서 문자 n개를 읽어서 문자열로 리턴한다.
read()	파일의 현재 위치에서 파일의 끝까지 모두 문자열로 리턴한다.
readline(n)	파일의 현재 위치에서 그 줄의 문자 n개를 읽어서 문자열로 리턴한다.
readline()	파일의 현재 위치에서 그 줄의 끝까지 모두 문자열로 리턴한다.
readlines()	파일의 현재 위치에서 한 줄씩 끝까지 읽어서 줄의 리스트로 리턴한다.
write(s)	문자열 s를 파일의 현재 위치에 쓴다.
writelines(ss)	문자열 리스트 ss에 있는 문자열을 모두 파일의 현재 위치에 쓴다.

CHAPTER 6

재귀와 반복 : 검색

문자열 검색용 메소드

문자열 메소드	실행 의미
str.find(sub)	str에서 맨 앞에 나타나는 sub의 시작 인덱스를 리턴, 없으면 -1을 리턴
str.index(sub)	str에서 맨 앞에 나타나는 sub의 시작 인덱스를 리턴, 없으면 ValueError 오류 발생
str.rfind(sub)	str에서 맨 뒤에 나타나는 sub의 시작 인덱스를 리턴, 없으면 -1을 리턴
str.startswith(prefix)	str이 prefix로 시작하면 True를 리턴, 그렇지 않으면 False를 리턴
str.endswith(suffix)	str이 suffix로 끝나면 True를 리턴, 그렇지 않으면 False를 리턴



문자열 검색 사례 학습

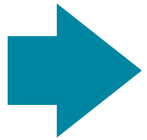
요구사항 : 텍스트 파일에서 처음 나타나는 문자열 x 찾기

```
def find_1st(filename, x):
```

- 텍스트 파일 전체를 문자열로 읽어오기
- 읽어온 문자열에서 처음 나타나는 문자열 x 찾기
- 찾은 문자열의 위치 인덱스를 “result.txt” 파일에 쓰기

code : 6-15.py

```
1  def find_1st(filename,x):
2      infile = open(filename,"r")
3      outfile = open("result.txt","w")
4      text = infile.read()
5      position = text.find(x)
6      if position == -1:
7          outfile.write(x + " is not found.\n")
8      else:
9          outfile.write(x + " is at " + str(position) + " the 1st
   time.\n")
10     outfile.close()
11     infile.close()
12     print("Done")
```



>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

사이드솔루션

pp.299~305

실습 6.2 find_1st 함수 테스트

```
find_1st('article.txt','computer')      # at 3269 the 1st time.
find_1st('article.txt','Whole Earth')   # at 10735 the 1st time.
find_1st('article.txt','Apple')         # at 4380 the 1st time.
find_1st('article.txt','apple')         # not found.
```

프로그래밍의 정석

파이썬

도경구 지음

pp.299~305

실습 6.3 find_2nd 함수 테스트

```
find_2nd('article.txt','computer')    # at 3357 the 2nd time.
find_2nd('article.txt','Whole Earth') # at 11280 the 2nd time.
find_2nd('article.txt','Apple')       # at 4455 the 2nd time.
find_2nd('article.txt','apple')       # not found.
```

>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

pp.299~305

실습 6.4 마지막 문자열 하나만 찾기

```
find_last('article.txt','computer')    # at 10975 the last time.
find_last('article.txt','Whole Earth') # at 11280 the last time.
find_last('article.txt','Apple')       # at 6604 the last time.
find_last('article.txt','apple')       # not found.
```

실습 6.5 모두 찾기

code : 6-18.py

```
1  def find_all(filename, x):
2      pass # Write your code here.
3
4  # Test code
5  # find_all('article.txt','computer')
6  # at 3269, 3357, 3601, 3725, 6209, 10975.
7  # find_all('article.txt','Whole Earth')
8  # at 10735, 11280.
9  # find_all('article.txt','Apple')
10 # at 4380, 4455, 4742, 5586, 5765, 6346, 6379, 6445, 6604.
11 # find_all('article.txt','apple')
12 # not found
```

실습 6.6 모두 찾고, 찾은 개수 세기

code : 6-19.py

```
1  def find_all_count(filename, x):
2      pass # Write your code here.
3
4  # Test code
5  # find_all_count('article.txt','computer')     # 6 time(s).
6  # find_all_count('article.txt','Whole Earth')  # 2 time(s).
7  # find_all_count('article.txt','Apple')        # 9 time(s).
8  # find_all_count('article.txt','commencement') # 1 time(s).
9  # find_all_count('article.txt','apple')        # 0 time(s).
```

실습 6.7 인용 문장 모두 찾기

find_quote_all("article.txt")

result.txt

```
"We have an unexpected baby boy; do you want him?"

"Of course."

"If you live each day as if it was your last, someday you'll most certainly be right."

"If today were the last day of my life, would I want to do what I am about to do today?"

"No"

"Stay Hungry. Stay Foolish."

There are 6 quotes in total.
```

>>>>>>>>>>> 제어 구조의 설계 원리를 중심으로 배우는 >>>>>>>>>>>

프로그래밍의 정석

파이썬

도경구 지음

생능출판

CHAPTER 6

재귀와 반복 : 검색