
알고리즘분석 실습 자료
15주차
Photo by Piotr Guzik on Unsplash

실습 소요 시간 100분

8장 계산복잡도: 검색 문제(2)

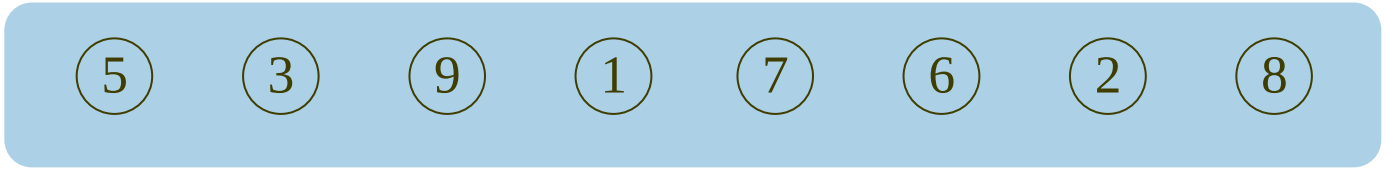
실습프로그램

- ✓ 최소키, 최대키 찾기
- ✓ 차대키 찾기
- ✓ k번째 작은 키 찾기
- ✓ 문자열 매칭

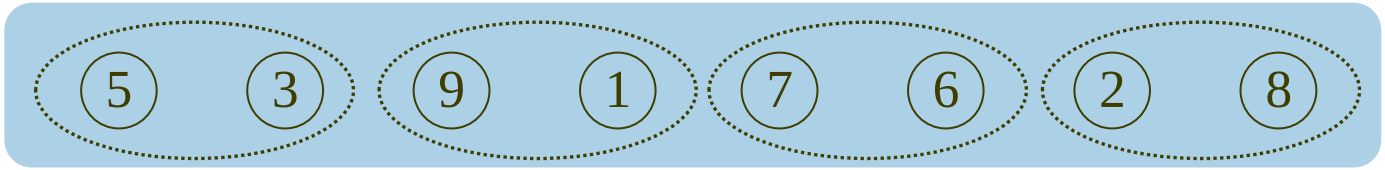
선택문제(selection problem)

- 키가 n개인 리스트에서 k번째로 큰(또는 작은) 키를 찾는 문제
- 키가 정렬되어 있지 않다고 가정
- 사례
  - ✓ k=1
  - ✓ 최대, 최소
  - ✓ k=2

키를 짝 지워서 최소키와 최대키 찾기



짝을 지어 L, S를 찾음

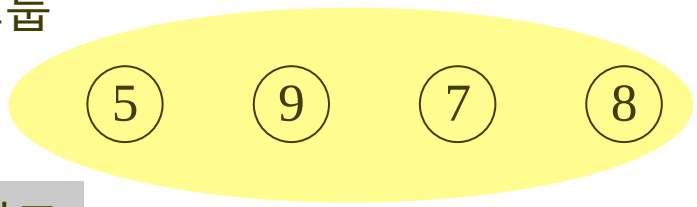


n/2 비교

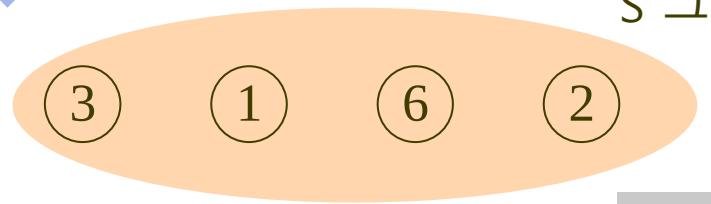
L S L S L S S L

L, S 그룹 생성

L 그룹



S 그룹



n/2-1 비교

Find MAX

Find MIN

n/2-1 비교

총 O(3n/2) 비교

키를 짝 지워서 최소키와 최대키 찾기

n: even number

```
void find_both2t(int n, const
keytype S[], keytype& small,
keytype& large){
      index i;

      if (S[1] < S[2]) {
          small = S[1];
          large = S[2];
          }
      else  {
          small = S[2];
          large = S[1];
          }
```

비교

```
 for (i=3; i<= n-1; i=i+2){
        if (S[i] < S[i+1]){
            if ( S[i] < small)
                 small = S[i];
            if ( S[i+1] > large)
                 large = S[i+1];
            }
        else  {
            if ( S[i+1] < small)
                 small = S[i+1];
            if ( S[i] > large)
                 large = S[i];
            }
      }
}
```

(n-2)/2회 반복

비교

비교

비교

# of repetitions

$$T(n) \approx \frac{n-2}{2} \times 3$$

# of comparisons

[실습프로그램] 키를 짝 지워서 최소키와 최대키 찾기

```
s=[2,6,4,8,10,3,7,1,5,9]
n= len(s)

     구현
```

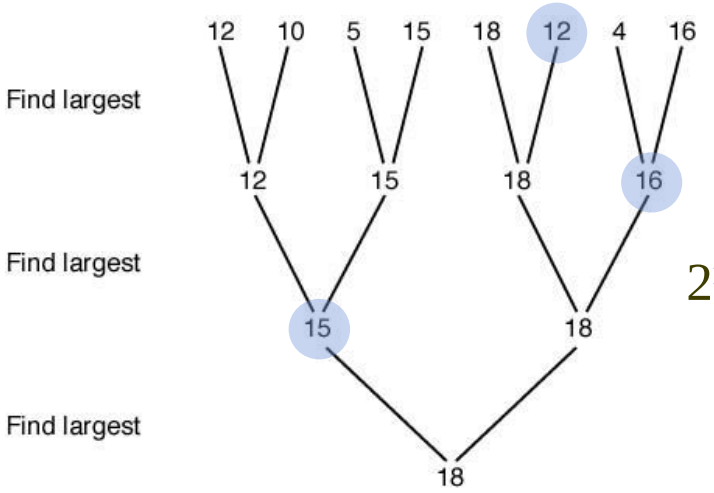
2,6,4,8,10,3,7,1,5,9

```
1 10
>>>
```

차대키 (second largest key) 찾기

1. 단순한 방법: 먼저 최대키를 찾고(n-1회 비교), 이후 다음 최대키를 찾음 (n-2회 비교). 총 $2n$-3회 비교

2. Tournament Method:
   (단계1) 토너먼트를 시행하여 최대 우승자가 최대키이다.
   (단계2) 각 시합의 진 팀을 이긴 팀의 리스트로 만든다.
   (단계3) 우승팀의 리스트에서 최대값을 찾으면, 이것이 차대키이다.



2^{nd}=MAX{12,16,15}

[실습프로그램] 차대키 찾기

2,6,4,8,10,3,7,1

```
s=[2,6,4,8,10,3,7,1]
a={}
while(len(s)>1):
  t=list()



   구현



winner = t[:]
print(winner[0])
print(a[winner[0]])
print("second =", max(a[winner[0]]))
```

```
{8: [4], 10: [3], 6: [2], 7: [1]}
[6, 8, 10, 7]
{8: [4, 6], 10: [3, 7], 6: [2], 7: [1]}
[8, 10]
{8: [4, 6], 10: [3, 7, 8], 6: [2], 7: [1]}
[10]
10
[3, 7, 8]
second = 8
>>>
```

k번째 작은 키 찾기

(1) 단순방법 : 정렬 후 k번째 선택 – $\Theta(n \lg n)$

(2) partition 사용: selection(1,n,k)

- ✓ quick sort의 partition을 사용
- ✓ $W(n) = n(n-1)/2$, $A(n) \approx 3n$

(3) $O(n)$ 방법

3. O(n) **방법**

$T(n) \le T(n/5)+T(3n/4)+cn$

```
procedure SELECT(k,S)
  if |S| < 50 then
      sort S
      return k^th smallest element in S
  else
      divide S into ⌊|S|/5⌋ sequences of 5 elements
      sort each 5-element sequence
      let M be the sequence of medians of the 5-element sets
      m ← SELECT(⌈|M|/2⌉, M)
      let S1, S2, and S3 be the sequences of elements in S less
        than, equal to, and greater than m, respectively.
      if |S1| ≥ k, then return SELECT(k,S1)
      else
          if |S1| +|S2|≥ k, then return m
          else return SELECT(k-|S1|-|S2|, S3)
```

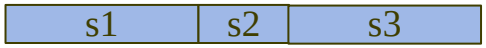
O(n)

T(n/5)

O(n)

T(3n/4)

뒤에 설명

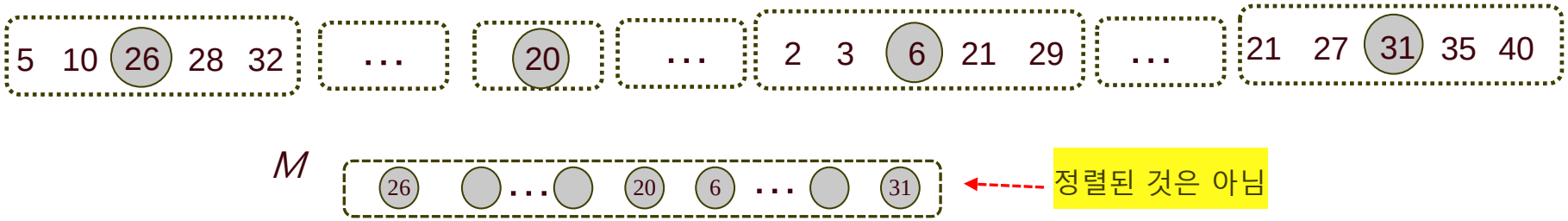


SELECT(k,S) 알고리즘의 수행 단계 설명

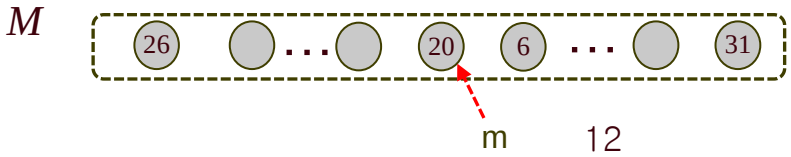
1. 데이터를 5개의 묶음으로 만듦

2. 5개의 묶음 내에서 정렬

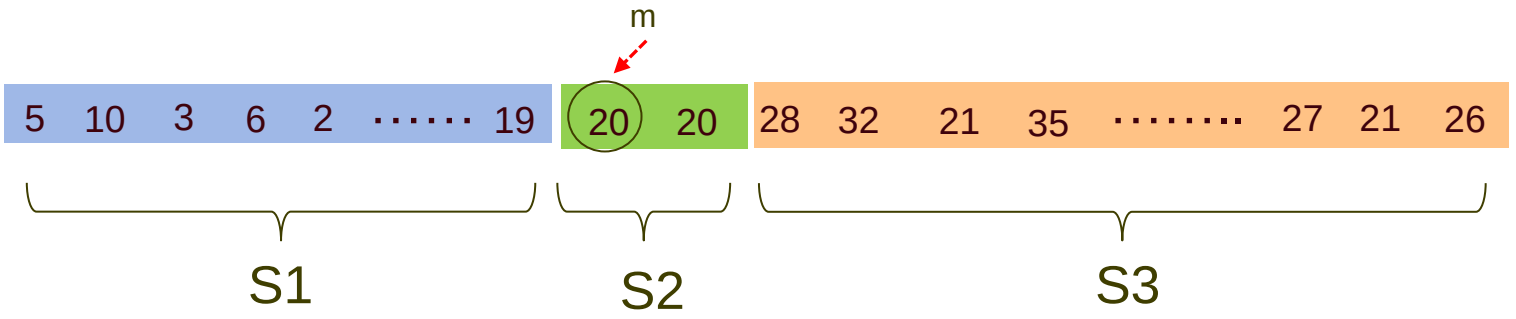
3. 5개의 묶음의 가운데 값들을 모아 M 생성



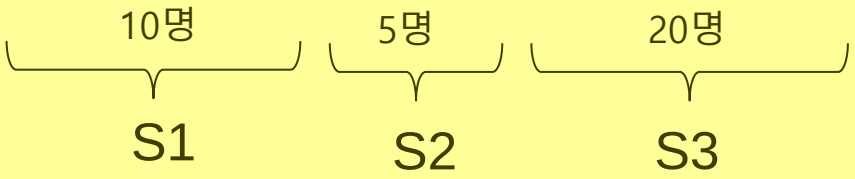
4. M에서 $\lceil |M|/2 \rceil$ 번째(가운데)를 찾음 : $m \leftarrow \text{SELECT}(\lceil |M|/2 \rceil, M)$



5. m보다 작은 데이터 S1, 같은 데이터 S2, 큰 데이터 S3를 생성



6. if |S1| ≥ k, then return SELECT(k,S1)
   else
      if |S1| +|S2|≥ k, then return *m*
      else return SELECT(k-|S1|-|S2|, S3)



(1) 5등을 찾는다면
(2) 12등을 찾는다면
(3) 23등을 찾는다면

7. 작아진 문제에 대해 select 수행

[실습프로그램] k번째 작은 키 찾기


```
import random

def insertion_sort(s):
    n = len(s)
    for i in range (1,n):
        x = s[i]
        j=i-1
        while j>=0 and s[j] > x:
          s[j+1]=s[j]
          j -= 1
        s[j+1]=x
    return s
```


```
def select(k,s):
  tempList=5*[0]
  mList =[]
  s1=[]
  s2=[]
  s3=[]
  if len(s) <50:
      insertion_sort(s)
      return s[k-1]
  else:
      for j in range(0,int(len(s)/5)):
         for p in range(0,5):
            tempList[p]=s[j*5+p]
         tempList = insertion_sort(tempList)
         mList.append(tempList[2])
      insertion_sort(mList)
      m = select( int(len(mList)/2),mList)
      for i in range(0,len(s)):
          if s[i]< m:
              s1.append(s[i])
          elif s[i]== m:
              s2.append(s[i])
          else:
              s3.append(s[i])

      if len(s1) >= k:
          return select(k,s1)
      elif len(s1)+len(s2) >= k:
          return m
      else:
          return select(k-len(s1)-len(s2), s3)
```

2/3

```
N=125   # 간단히 하기 위해 N은 5의 배수로 설정
s=N*[0]
k=120

for i in range(0,N):
     s[i]=random.randint(1,1000)

print()
print("original")
print(s)
print()
print(k,"th element = ", select(k,s))

print()
print("sorted")
print(insertion_sort(s))
```


```
original
[161, 154, 802, 280, 22, 292, 426, 72, 492, 395, 137, 371, 209, 449, 722,
798, 243, 264, 680, 82, 573, 948, 862, 683, 906, 552, 548, 930, 276, 681,
382, 8, 372, 280, 65, 126, 450, 105, 984, 151, 818, 176, 621, 962, 598,
226, 701, 691, 993, 118, 145, 397, 365, 838, 394, 308, 486, 333, 29, 41,
230, 525, 990, 880, 386, 356, 314, 163, 405, 782, 234, 17, 16, 844, 725,
112, 661, 384, 452, 921, 561, 690, 327, 844, 961, 391, 865, 536, 519, 852,
699, 589, 740, 27, 111, 105, 797, 421, 196, 803, 964, 639, 866, 277, 373,
642, 116, 559, 795, 777, 393, 381, 835, 277, 665, 156, 282, 56, 946, 415,
276, 342, 305, 543, 306]

120 th element =  961

sorted
[8, 16, 17, 22, 27, 29, 41, 56, 65, 72, 82, 105, 105, 111, 112, 116, 118,
126, 137, 145, 151, 154, 156, 161, 163, 176, 196, 209, 226, 230, 234, 243,
264, 276, 276, 277, 277, 280, 280, 282, 292, 305, 306, 308, 314, 327, 333,
342, 356, 365, 371, 372, 373, 381, 382, 384, 386, 391, 393, 394, 395, 397,
405, 415, 421, 426, 449, 450, 452, 486, 492, 519, 525, 536, 543, 548, 552,
559, 561, 573, 589, 598, 621, 639, 642, 661, 665, 680, 681, 683, 690, 691,
699, 701, 722, 725, 740, 777, 782, 795, 797, 798, 802, 803, 818, 835, 838,
844, 844, 852, 862, 865, 866, 880, 906, 921, 930, 946, 948, 961, 962, 964,
984, 990, 993]
```

```
naiveMatching(A,p) {

  i = 1;
  while(i ≤ n-m+1) {
    if(p[1..m] == A[i..i+m-1])
        a matching is found at A[i]
    i=i+1
  }
}
```

```
BoyerMooreHorspool(A,p) {
  computeJump(p, jump);
  i = 1;
  while(i ≤ n-m+1) {
    j = m;
    k=i+m-1;
    while(j>0 and p[j]==A[k]){
       j--;
       k--;}
    if(j==0)
        a matching is found at A[i]
    i= i + jump[A[i+m-1]];
  }
}
```

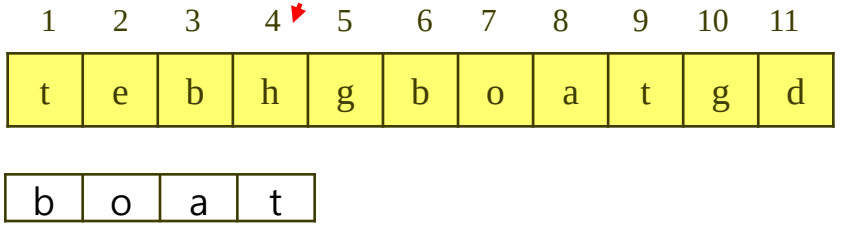
jump 정보를 계산 θ(m)

O(n-m+1)

O(m)

단순 방법

Boyer Moor Horspool 방법



- jump 정보 (입력문자열의 현재 매칭 확인하는 구역의 오른쪽 끝 문자에 따라)

오른쪽끝문자	b	o	a	t	etc
jump	3	2	1	4	4

오른쪽끝문자	s	h	o	o	l	etc
jump	4	3	2	1	5	5

오른쪽끝문자	s	h	o	l	etc
jump	4	3	1	5	5

- ✓ 같은 문자가 있을 경우는 작은 jump 값으로 설정
- ✓ 총 O(mn)
- ✓ Boyer-Moore 방법은 이 방법보다 정교한 jump 판단 방식채용

[실습프로그램] 문자열 매칭

1/2

```
def compJump(p):
    jump = {}
    m = len(p)

    for i in range(0,m-1):
        jump[p[i]]=m-i-1
    return jump

def BMH(a, p):
    jump = compJump(p)
    m= len(p)
    n = len(a)
    i = 0
    while(i<= n-m):
        j=m-1
        k=i+m-1
        while(j>=0 and p[j]==a[k]):
              j-=1
              k-=1
        if j == -1:
              print("matching is found at ", i)
        if a[i+m-1] in jump :
            i += jump[a[i+m-1]]
        else:
            i+=m
```

```
p="school"

f = open('textfile2.txt','r')
# f.read는 파일 전체를 읽어 온다.
text = f.read()
BMH(text, p)
f.close()

# line 단위로 check. 파일의 모든 줄을 lines로 읽어 온다.
f = open('textfile2.txt','r')
lines = f.readlines()
f.close()
jj=1
for line in lines:
    print(jj)
    jj+=1
    BMH(line,p)
```

```
this is a sample
course school nation
good meal
```

입력 textfile2.txt

```
matching is found at  24
1
2
matching is found at  7
3
>>>
```