파이썬으로 배우는 알고리즘 기초

Chap 2. 분할정복

2.3-2.4 분할정복과 퀵 정렬

2.3 분할정복 설계방법

- 분할정복 설계 전략
  - 분할: 문제의 입력사례를 둘 이상의 작은 입력사례로 분할
  - 정복: 작은 입력사례들을 각각 정복
    작은 입력사례들이 충분히 작지 않으면 재귀 호출
  - 통합: 필요하면, 작은 입력사례의 해답을 통합하여 원래 입력사례의 해답을 도출

2.3 분할정복 설계방법

- **분할정복 알고리즘**
  - **분할정복** .vs. **동적계획**
    - **하향식**(Top-Down) .vs. **상향식**(Bottom-Up) **문제풀이 방식**
  - **분할정복** .vs. **탐욕법**
    - **탐욕법은 가장 비효율적인 분할정복 알고리즘?**

2.4 퀵 정렬 (분할 교환 정렬)

- **퀵 정렬**: **분할 정복**(Divide-and-Conquer)
  - **내부**(*in-place*) **정렬**: **추가적인 리스트를 사용하지 않는 정렬**
  - **추가적인 리스트를 생성하지 않고 정렬할 수 없을까**?
    - Hoare(1962), Quick Sort Algorithm

  - QUICK-SORT
    - **[Divide] 기준 원소**(*pivot*)**를 정해서 기준원소를 기준으로 좌우로 분할**
    - **[Conquer] 왼쪽의 리스트와 오른쪽의 리스트를 각각 재귀적으로 퀵 정렬**
    - **[Obtain] 정렬된 리스트를 리턴**

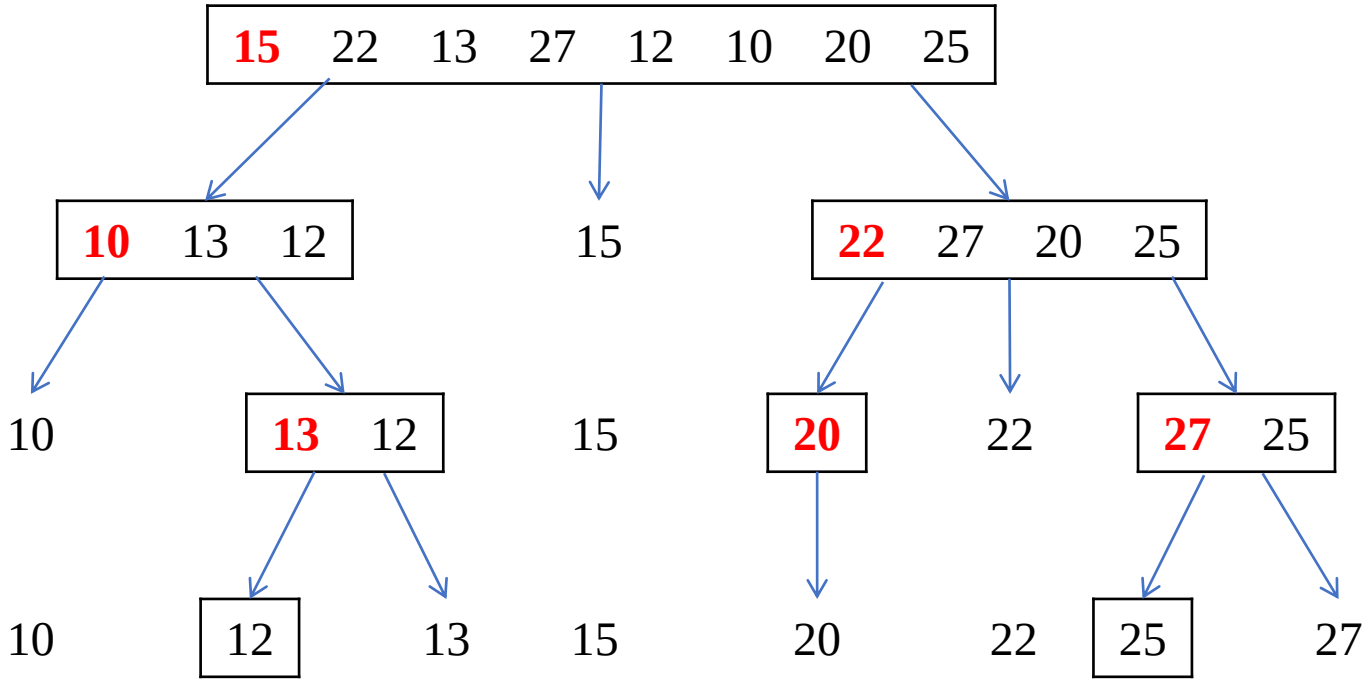
2.4 퀵 정렬 (분할 교환 정렬)

Algorithm 2.6: Quick Sort

```
def quicksort (S, low, high):
    if (high > low):
        pivotpoint = partition(S, low, high)
        quicksort(S, low, pivotpoint - 1)
        quicksort(S, pivotpoint + 1, high)
```

2.4 퀵 정렬 (분할 교환 정렬)

- 기준 원소(pivot)는 어떻게 정할까?
  - 편의상, 일단 리스트의 첫 원소를 기준원소로 정하도록 하자



2.4 퀵 정렬 (분할 교환 정렬)

- 기준 원소로 어떻게 리스트를 나눌 수 있을까?
  - 두 개의 인덱스(i, j)를 이용해서 비교$(compare)$와 교환$(swap)$

```
pivotpoint = partition(S, low = 0, high = 7)
```

i	j	$S[0]$	$S[1]$	$S[2]$	$S[3]$	$S[4]$	$S[5]$	$S[6]$	$S[7]$
-	-	15	22	13	27	12	10	20	25
1	0	**15**	**22**	13	27	12	10	20	25
2	0	**15**	22	**13**	27	12	10	20	25
3	1	**15**	13	22	**27**	12	10	20	25
4	1	**15**	13	22	27	**12**	10	20	25
5	2	**15**	13	12	27	22	**10**	20	25
6	3	**15**	13	12	10	22	27	**20**	25
7	3	**15**	13	12	10	22	27	20	**25**
-	-	10	13	12	15	22	27	20	25

2.4 퀵 정렬 (분할 교환 정렬)

Algorithm 2.7: Partition (for Quick Sort)

```
def partition (S, low, high):
    pivotitem = S[low]
    j = low
    for i in range(low + 1, high + 1):
        if (S[i] < pivotitem):
            j += 1;
            S[i], S[j] = S[j], S[i] # swap
    pivotpoint = j
    S[low], S[pivotpoint] = S[pivotpoint], S[low] # swap
    return pivotpoint
```

2.4 퀵 정렬 (분할 교환 정렬)


```
S = [15, 22, 13, 27, 12, 10, 20, 25]
print('Before:', S)
quicksort(S, 0, len(S) - 1)
print(' After:', S)
```

```
S = [15, 22, 13, 27, 12, 10, 20, 25]
print('Before:', S)
partition(S, 0, len(S) - 1)
print(' After:', S)
```

2.4 퀵 정렬 (분할 교환 정렬)

- partition() 함수의 다른 구현 방법

```
S = [26, 5, 37, 1, 61, 11, 59, 15, 48, 19]

    [26, 5, 37, 1, 61, 11, 59, 15, 48, 19]: 2, 9

    [26, 5, 19, 1, 61, 11, 59, 15, 48, 37]: 4, 7

    [26, 5, 19, 1, 15, 11, 59, 61, 48, 37]: 6, 5

    [11, 5, 19, 1, 15, 26, 59, 61, 48, 37]
```

2.4 퀵 정렬 (분할 교환 정렬)


```
def partition2 (S, low, high):
    pivotitem = S[low]
    i = low + 1
    j = high
    while (i <= j):
        while (S[i] < pivotitem):
            i += 1
        while (S[j] > pivotitem):
            j -= 1
        if (i < j):
            S[i], S[j] = S[j], S[i] # swap
    pivotpoint = j
    S[low], S[pivotpoint] = S[pivotpoint], S[low] # swap
    return pivotpoint
```

2.4 퀵 정렬 (분할 교환 정렬)


```
def quicksort2 (S, low, high):
    if (high > low):
        pivotpoint = partition2(S, low, high)
        # print(S)
        quicksort2(S, low, pivotpoint - 1)
        quicksort2(S, pivotpoint + 1, high)

print('Before:', S)
quicksort2(S, 0, len(S) - 1)
print(' After:', S)

print('Before:', S)
partition2(S, 0, len(S) - 1)
print(' After:', S)
```

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 **구독**과 **좋아요**는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: **구글 드라이브**에서 다운로드
  (다운로드 주소는 영상 하단 설명란 참고)

  https://bit.ly/3fN0q8t