

# 기초알고리즘, 분할정복 (divide and conquer)

EUnS

May 5, 2020

1 INSERTION-SORT(A)

2 **for**  $j = 2$  to  $A.length$

3      $key = A[j]$

4     // Insert  $A[j]$  into the sorted sequence  $A[1 \dots j - 1]$ .

5      $i = j - 1$

6     **while**  $i > 0$  **and**  $A[i] > key$

7          $A[i + 1] = A[i]$

8          $i = i - 1$

9      $A[i + 1] = key$

# 시간복잡도

- 최악의 경우 시간 복잡도

# 시간복잡도

- 최악의 경우 시간 복잡도
- 최선의 경우 시간 복잡도

$$\Theta(n^2)$$

# 시간복잡도

- 최악의 경우 시간 복잡도

$$\Theta(n^2)$$

- 최선의 경우 시간 복잡도

$$\Theta(n)$$

- 평균 시간 복잡도

$$\Theta(n^2)$$

# 시간복잡도

- if에 의해서 정확하게 구하기 어렵다.

# 시간복잡도

- if에 의해서 정확하게 구하기 어렵다.
- 입력 자료의 상태에 따라 시간 복잡도는 바뀐다.

# 시간복잡도

- if에 의해서 정확하게 구하기 어렵다.
- 입력 자료의 상태에 따라 시간 복잡도는 바뀐다.
- 최악은 for문만을 가지고 단순 계산해도 무방.



# 시간복잡도

- if에 의해서 정확하게 구하기 어렵다.
- 입력 자료의 상태에 따라 시간 복잡도는 바뀐다.
- 최악은 for문만을 가지고 단순 계산해도 무방.
- ps에선 1초에  $10^9$ (1억번)정도 계산한다고 생각하고 어림잡아 계산.

문제를 세가지 단계를 거치면서 재귀적으로 문제를 푼다.

- 1 분할 : 현재의 문제와 동일하되 입력의 크기가 더 작은 다수의 부분 문제로 분할한다.
- 2 정복 : 부분 문제를 재귀적으로 풀어서 정복. 부분 문제의 크기가 충분히 작으면 직접적인 방법으로 푼다.
- 3 결합 : 부분 문제의 해를 결합해 원래 문제의 해가 되도록 만든다.

# 분할정복 예시

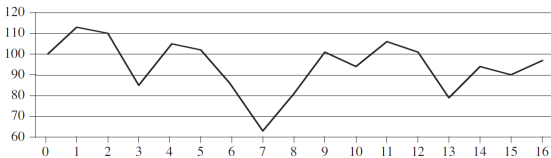
- 머지소트
- 최대부분 배열문제를 해결하는 알고리즘
- 퀵소트
- 행렬 곱
- FFT
- 카라추바 알고리즘

# 분할정복 예시

- 머지소트
- 최대부분 배열문제를 해결하는 알고리즘
- 퀵소트
- 행렬 곱
- FFT
- 카라츨바 알고리즘

# 최대 부분 배열문제(Maximum subarray problem)

- 배열값중 구간 끝값의 차가 가장 큰 구간 찾기



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

- 주먹구구 과제로 만들어올것.

# 분할정복을 이용한 방법

- 인덱스 low... high 사이의 임의의 mid를 잡고 이 중 최대 부분 배열이 어디에 속하는지 생각해보자.

# 분할정복을 이용한 방법

- 인덱스 low... high 사이의 임의의 mid를 잡고 이 중 최대 부분 배열이 어디에 속하는지 생각해보자.
- low ,mid 사이



# 분할정복을 이용한 방법

- 인덱스 low... high 사이의 임의의 mid를 잡고 이 중 최대 부분 배열이 어디에 속하는지 생각해보자.
- low ,mid 사이
- mid high 사이

# 분할정복을 이용한 방법

- 인덱스 low... high 사이의 임의의 mid를 잡고 이 중 최대 부분 배열이 어디에 속하는지 생각해보자.
- low ,mid 사이
- mid high 사이
- mid를 걸치는 low high 사이

# 분할정복을 이용한 방법

- 인덱스 low... high 사이의 임의의 mid를 잡고 이 중 최대 부분 배열이 어디에 속하는지 생각해보자.
- low ,mid 사이
- mid high 사이
- mid를 걸치는 low high 사이
- 이것이 분할의 세가지 케이스

# 분할정복을 이용한 방법

- 분할 : 일단 전체 길이를 반으로 쪼갬다.

# 분할정복을 이용한 방법

- 분할 : 일단 전체 길이를 반으로 쪼갬다.
- 정복 : 상수시간.

# 분할정복을 이용한 방법

- 분할 : 일단 전체 길이를 반으로 쪼갬다.
- 정복 : 상수시간.
- 결합 : 걸쳐있는 경우 좌우 길이를 새로 구하고 각각의 길이를 리턴한뒤 비교한다.

# 분할정복을 이용한 방법

```
1 FIND-MAXIMUM-SUBARRAY(A, low, high)
2   if high == low
3       return (low, high, A[low]) // base case: only one element
4   else mid = (low + high)/2
5       (left-low, left-high, left-sum)
6         = FIND-MAXIMUM-SUBARRAY(A, low, mid)
7       (right-low, right-high, right-sum)
8         = FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
9       (cross-low, cross-high, cross-sum)
10        = FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
11
12   if left-sum >= right-sum and left-sum >= cross-sum
13       return (left-low, left-high, left-sum)
14   elseif right-sum >= left-sum and right-sum >= cross-sum
15       return (right-low, right-high, right-sum)
16   else return (cross-low, cross-high, cross-sum)
17
```

## 분할정복을 이용한 방법

```
1 FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
2   left-sum = -inf
3   sum = 0
4   for i = mid downto low
5       sum = sum + A[i]
6       if sum > left-sum
7           left-sum = sum
8           max-left = i
9       right-sum = -inf
10  sum = 0
11  for j = mid + 1 to high
12      sum = sum + A[j]
13      if sum > right-sum
14          right-sum = sum
15          max-right = j
16  return (max-left, max-right, left-sum + right-sum)
```