Lecture 7 : 분할정복법 알고리즘 설계의 실제

앞서 말한 바와 같이 좋은 알고리즘을 개발하기 위한 일반론은 없다. 즉 어떤 문제에 대한 가장 좋은 알고리즘 개발법은 있을 수 없기 때문에 여러 다양한 문제를 활용하여 다양한 실전 경험을 쌓는 것이 가장 좋은 현실적 방법이다. 한가지 유의할 점은 데이터의 입력이 유사하다고 해서 이전에 사용한 방법을 그대로 적용하는 것이다. 많은 경시대회에서 이런 식의 함정 문제를 자주 출제하곤 한다.

예를 들어 배열에서 특정한 원소를 찾는 문제인데 마치 이분법을 사용하면 쉽게 해결될 듯한 구조의 문제이다. 앞 장에서 보인 구글의 "항아리" 문제가 그 전형적인 예이다. 이 문제를 접한 대부분의 초보 연구자들의 전체 건물의 중간 높이를 기준으로 문제를 접근한다. 그러나 이 문제의 핵심은 만일 중간 높이에서 떨어뜨린 항아리가 깨어지지 않았을 경우에는 2개의 항아리를 사용할 수 있고, 만일 깨어진 경우에는 남은 1개의 항아리만 사용할 수 있다는 서로 다른 조건이다.

분할정복으로 해결할 수 있는, 정확히 말하자면 분할정복으로 접근했을 때 최적의 알고리즘이 나오는 다양한 문제가 인터넷에 제시되어 있다. 문제의 수는 매우 많지만 크게 구분하면 이 장에서 제시하는 문제의 여러 변형이 불과하다. 따라서 여러분이 이 장에서 강사가 설명하는 문제에 집중하고, 여유가 있으면 문제의 조건을 조금씩 바꾸어 풀어보는 것을 제안한다.

7.1 분할정복으로 쉽게 해결되는 문제의 예

최대 합 구간(maximal sum interval) : 문제) 어떤 일차원 배열 X[n]이 있다. 이 배열에는 음수와 양수로 표시된 정수($-K \leq X[i] \leq +K$) 가 들어있다. 우리는 어떤 특정 구간의 합이 가장 큰, 그 해당 구간을 구하고자 한다.

0	1	2	3	4	5	6	7	8	9	10	11
6	-11	9	7	-20	9	11	4	-6	2	-5	8

(a) 가장 무식한 알고리즘 (모든 구간의 합을 고려하는 방법)
기본동작은 두 배열항목을 더하는 일 X[i]+X[j]

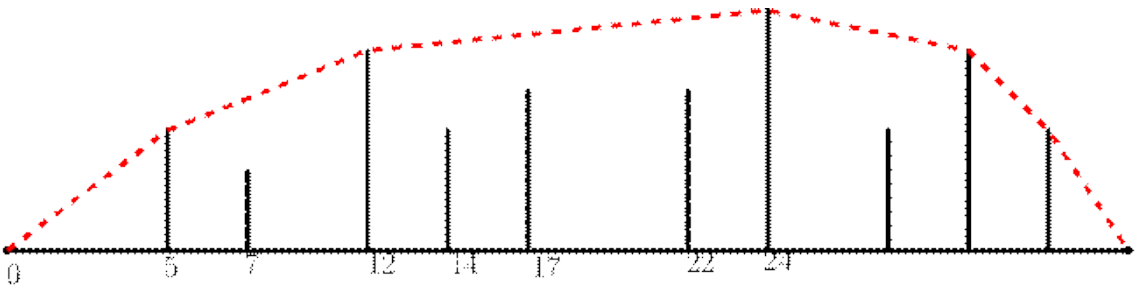
(b) Pre-computing을 이용해서 모든 구간을 조사해보는 방법
즉 $O(N^2)$개의 가능한 모든 구간에 대하여 미리 답을 마련해둔다.
계산 시간은 O(1)으로 제일 빠르지만.
Alas...이 답을 저장해두는 공간이 $O(N^2)$

(c) Divide and Conquer를 이용해서 $O(n \log n)$에 구하는 방법

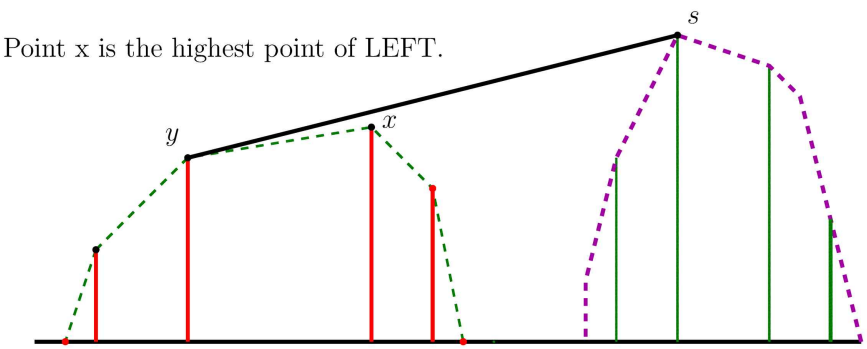
7.2 텐트 문제[1]:

어떤 세로 기둥이 일렬로 세워져 있다. 이 기둥을 모두 둘러씌우기 위하여 텐트 막을 치려고 한다. 이 때 텐트 막에 걸리는 모든 세로 기둥을 왼쪽부터 차례대로 출력하시오. 시작은 x축으로 0이며 끝 점은 n으로 고정되어 있다. 각

기둥은 그 x축 위치와 그 높이가 쌍으로 주어진다. 예를 들면 (5,10), (7,5), (12,13), (14,6)... 같은 식이며 출력은 폴에 걸리는 기둥의 x좌표를 출력한다. 이 문제는 divide and conquer로 해결해보자. 일단 전체를 반으로 나눈 다음에 각각에서 텐트를 구한 다음에 이 둘을 합치는 방식으로 하면 된다.

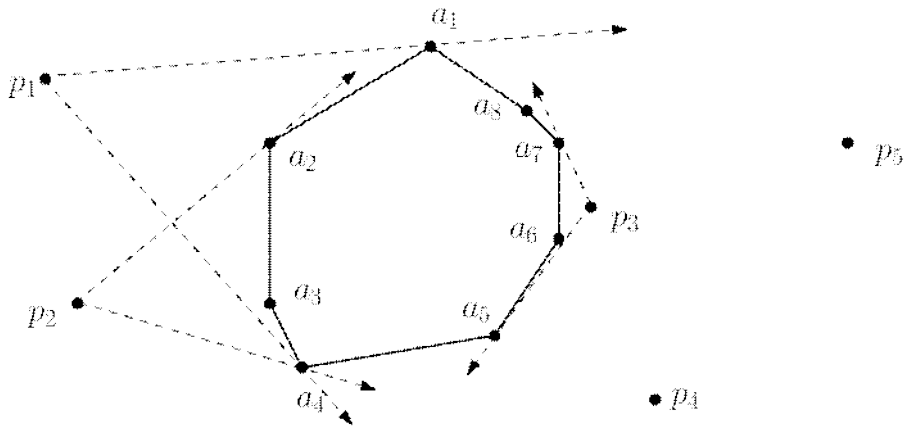


질문) 양쪽으로 나눠진 텐트에서 가장 높은 2점을 연결하면 전체의 답이 되지 않나요?
답) 아니요. 그렇지 않습니다. 그림 참조. (s,x) 선분은 답이 안됨.



볼록 다각형(convex hull)가 있을 때 두 개의 support line을 구하는 문제와 같다. support line은 한 점에서 다각형의 꼭지점 중 하나와 연결되는 선을 그었을 때 다각형의 모든 점이 한쪽에 몰려있게 만들어주는 선이다.

1) 이 문제는 점 집합에서 convex hull(볼록 다각형)을 구하는 문제로 요약된다.



Support Line for Convex Polygon

위 그림에서 점 p_1의 두 support line은 선분 (p_1,a_1), (p_1,a_4)이다

Q) 같은 방식으로 p_2의 두 support line을 구해보시오.

Q) 만일 p_5와 convex polygon 사이의 support line을 구하는 방법을 Divide and Conquer 방식으로 제안해보고, 그 때의 시간복잡도를 제시하시오.

Hint) p_x와 polygon의 꼭지점 한 점 v_i와 선을 연결한다. 이 경우 v_i와 인접한 두 개의
꼭지점 v_{i-1}와 v_{i+1}이 있을 때 이 두 꼭지점이 선분 (p_x, v_i)의 양쪽으로 나뉘어져 있으면 Support line이 될 수 없다. 즉 하나는 위에, 다른 하나는 아래에.

Lecture 8: 분할정복법 알고리즘 설계의 실제 2[2)]

8.1 n개의 원소가 있는 일차원 배열 D[n] 에서 가장 큰 원소를 찾아내는 알고리즘을 분할정복 방법으로 구하시오. (제한 조건. 코드에서 for loop과 같은 iterating loop을 사용하면 안됨.)

Sol) 전체를 반으로 나누어서 각각에서 가장 큰 원소를 구하고 그 두 원소의 큰 값을 전체의 답으로 보고함.

```
array_max(i,j) {
    if i == j : return D[i] ;
    leftmax  = array_max(i, (i+j)/2) ;
    rightmax = array_max((i+j)/2+1, j) ;
    if leftmax > rightmax : return(leftmax);
    else return(rightmax) ;
}
```

위 알고리즘의 복잡도 계산: $D(N) = D(N/2) + D(N/2-1) + 1$

8.2 n개의 원소가 있는 일차원 배열 D[n] 에서 두 번째로 큰 원소(second largest element)를 찾아내는 알고리즘을 분할정복 방법으로 구하시오. 이 문제는 약간 Tricky하다.[3)] 아래와 같이 위 array_max( )를 수정하면 될까요 ?

```
second_max(i,j) {
    if i == j : return D[i] ;
    lmax1, lmax2= second_max(i, (i+j)/2) ;
    rmax1, rmax2= second_max((i+j)/2+1, j) ;


    /* FILL THIS PART  */


}
```

2) 어떤 문제를 주고 분할정복법으로 해결하라고 하면 쉽게 해결할 수 있지만 이 문제가 분할정복법으로 잘 해결될 수 있을지를 예감하는 것은 어려운 일이다. 콜롬부스의 달걀과 같이 간단한 것이라도 풀고난 것을 보면 이해가 되지만 그 방법을 처음 생각해내는 것은 쉽지 않다. 따라서 많은 문제를 실제 풀어보고 경험을 쌓는 수밖에 없다.

3) tricky한 문제는 알고리즘 코딩 인터뷰의 단골 질문이므로 주의 깊게 이해하고 있어야 한다.

8.3 각각 sorting된 원소를 n, m개 담고 있는 배열 X[n]과 Y[m]이 있다. 이들 전체 { X[n] ∪ Y[m] }중에서 k번째 원소를 찾아내는 복잡도 $O(\log m \cdot \log n)$의 알고리즘을 분할정복법으로 설계하시오.

아래 2개의 배열에서 21번째 원소를 찾아라. 단 두 배열을 "털어서" 하나의 배열로 만들어 다시 merge sorting하는 방법을 사용하면 안 된다. 이 방법의 complexity는 O(n+m)인데, 우리는 그 이하의 시간에 k번째 원소를 찾아야 한다.

3	***14***	***21***	***34***						***231***										***634***

2	***15***	***31***	***55***	***76***							***133***				***542***				***723***

8.4 유명인사 찾아내기 (Celebrity Problem)

n명 회원들로 구성된 동호회 집합 $\{p_i\}$이 있다. 각 사람들은 알거나 모르거나 한다. 두 사람과의 관계는 서로 알거나, 서로 모르거나 한 사람이 다른 사람을 아는 2가지 경우, 즉 4가지 경우가 존대한다. 그런데 유명인(celebrity)는 자신은 모든 사람을 모르고, 나머지 사람들은 자신을 모두 아는 사람으로 정의된다.

어떤 두 사람 h_i, h_j에 대해서 h_i가 h_j가 잘 알고 있는 경우에는 다음으로 표시된다. 이것을 $h_i \rightarrow h_j$로도 표시할 수 있다.

$$know(h_i, h_j) = 1$$

그러나 모를 경우에는 $know(h_i, h_j) = 0$ 이다. 이 결과는 $H_{i,j}$ 행렬(matrix)에 저장되어 있다. 어떤 경우에는 Celebrity가 없는 경우도 있다.[4)]

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

이 유명인은 찾아내는 가장 단순한 알고리즘은 $O(N^2)$의 알고리즘이 있다. 즉 모든 행렬을 다 살펴보아야 한다. 이것을 개선할 수 있는 알고리즘이

4) 우리 알고리즘 Class에서는 아무도 유명인이 될 수 없다. 그러나 이런 우리 반에 "빌 게이츠"가 등장하면 이 사람은 유명인이 될 수밖에 없다. 그는 우릴 아무도 모르지만 우리는 모두 그를 알기 때문이다.

있으면 설명하고 그 복잡도를 계산하시오.

8.5 Binary Search는 전체 구간을 2개로 나눠서 한다. 만일 전체 구간을 3개 혹은 k개로 나눠 탐색하면 더 효율적일까 ? 어떤 사람은 그것이 효율적이라고 주장한다. 왜냐하면 잘라진 구간의 크기가 n/2, n/3.. n/k로 줄어들기 때문이다. 이 말이 맞는지 $k-$ary Searching에 대하여 그 시간복잡도를 분석하시오.

$S(k, N)$ 는 N개의 구간을 k개의 "거의 같은(nearly equal)" 너비의 구간으로 나눠 탐색하는 것이다. 이 알고리즘의 시간복잡도를 구하시오.

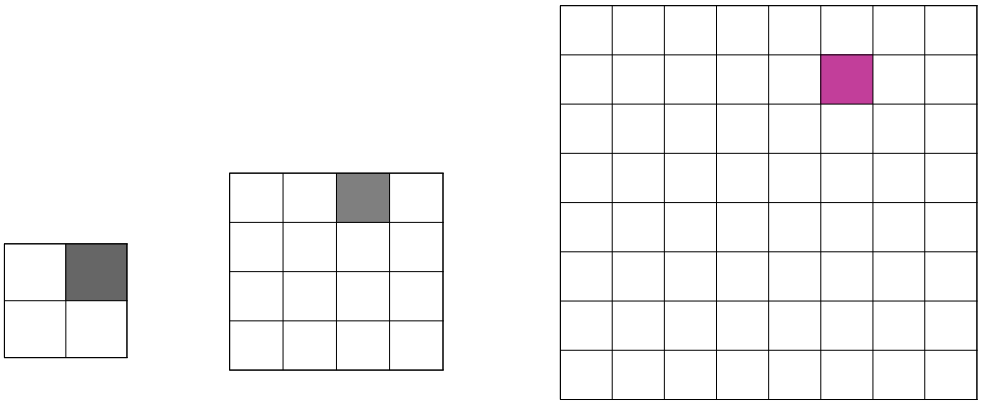
$S(1, N) = O(N)$

$S(2, N) = O((2-1)\log_2 N) = \Theta(\log_2 N)$

$S(3, N) = 2 \cdot \log_3 N = \frac{2}{\log_2 3} \log_2 N = 1.2618 \cdot \log_2 N$

$S(k, N)$? $= (k-1)\ \log_k \mathrm{N}$

8.6 한 변의 길이가 2^k인 격자공간이 있다. 이 공간에는 하나의 격자에 장애물이 있다. 즉 이 공간에는 타일을 놓을 수 없다. 여러분은 이 공간을 3개의 정사각형이 'ㄱ'자 모양으로 연결된 tromino로 항상 cover할 수 있음을 보이시오.



hint) $N = 2^k$ 일 때 가능함을 증명한 뒤에 $N = 2^{k+1}$일 때를 증명한다.

8.7 [연구문제] Counting the Number of Inversion

See https://www.youtube.com/watch?v=Vj5IOD7A6f8

Inversion: 두 수 $x < y$ 가 나열된 순서로 볼 때 y가 x 앞에 있는 경우
수열이 주어질 때 모든 inversion의 개수를 계산하시오.

무식한 알고리즘: 마구 계산할 경우 $O(N^2)$ algorithm
효율적인 알고리즘: $O(N \log N)$ algorithm

(15, 4, 11, 1, 7, 12, 8, 9, 21, 5, 10, 3, 16)

Hint) 이 문제를 분할정복법으로 접근해보자.

- 전체를 반으로 나눠 생각해 봅시다.

L	R

- Inversion은 다음 3가지 중 하나가 된다.

case 1) L에만 있는 경우 (5,2)

5, 2	

case 2) R에만 있는 경우 (11, 6)

	11, ... 6

case 3) L과 R에 있는 경우 (32, 6)

32, ...	 9

merge sort를 이용하면 된다. 주어진 배열을 sorting 한 후에 inversion을 출력
만일 L과 R이 각각 sorting되어 있다고 가정하면 (L,R) inversion 수를 쉽게 계산할 수 있다.

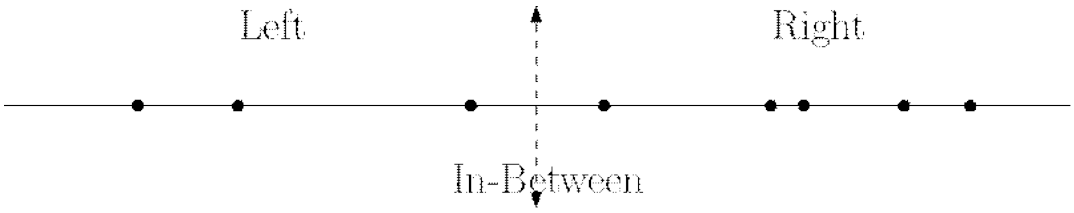
예)

3 , 5, 11, 15, 32, 54	2 13 22 29 45 60

Sorted List	추가 Inversion	합계
2	+ 6	6
2 3	+ 1	7
2 3 5	+ 1	8
2 3 5 11	+ 1	9
2 3 5 11 13	+ 3	12
2 3 5 11 13 **15**	+ 2	14
2 3 5 11 13 **15 22**	+ 2	
2 3 5 11 13 **15 22**		

8.8 [분할정복법의 오용]

수직선 상에서 가장 가까운(closest) 두 점의 쌍 찾기



1) 양쪽으로 반으로 나눠서
2) 각각에서 가장 가까운 점을 구하고
3) 분할선을 끼고 있는 가장 가까운 pairs를 탐색

8.9 가짜 동전(fake coin) 문제 [5]

N개의 금화 중에서 약간 가벼운 가짜 동전이 1개 있다. 양팔 저울만을 이용해서 이것을 찾아내야 하는데 이 과정에서 양팔저울의 사용횟수를 최소화해야 한다. 이 경우 N에 대하여 살펴보자. 이 최소 횟수를 B(N)으로 표시하자. 단 이 경우에는 항상 worst case를 고려해야 한다.

1) $N=2$, 한번이면 충분하다. 따라서 $B(2)=1$
2) $N=3$, (1,1,1)로 나눠서 첫 2개를 비교한다. 만일 한 쪽이 내려가면 그것. 만일 같으면 3번째 동전이 가짜이므로 $B(3)=1$ 이다.
3) $N=4$, (2,2,0)으로 처리하면 2번이 필요하다. 즉 한 쪽이 반드시 내려가므로 그 내려간 쪽의 2개를 선택해서 다시 비교해야 한다.
4) $N=5$, (2,2,1)로 처리하면 이 역기 $B(5)=2$ 이다.
5) $N=6$이라고 하자. 만일 (2,2,2)로 비교해보자. 한쪽이 내려갈 경우 2번 비교가 필요. 만일 같은 무게라면 나머지 2개에서 가려내야하므로 이 역시 $B(6)=2$ 이다.
6) $N=7$이라고 한다면 (3,3,1)로 나누어보자. 만일 1번의 비교 무게가 같아지만을 확인하면 나머지 1개가 가짜. 만일 한쪽이 기울면 이 문제는 $B(3)$+1 비교가 필요. 따라서 최악의 경우에는 2번이므로 $B(7)=B(3)+1=1+1=2$ 이다.

7) $N=3k+1,\ 3k+2$ 이라고 가정하자. $3k+1$일 경우 $(k,k,k+1)$로 나눠서 측정을 한다. 이 경우 만일 k,k가 같으면 나머지 $k+1$에서 가짜를 찾아야 하므로 이 경우 비교횟수는 $B(k+1)+1$ 이 된다. 만일 한쪽이 기울면 그 기울어진 쪽에서 찾아야 하므로 이 경우에 비교횟수는 $B(k)+1$이다. 따라서 이 둘 경우에는 우리는 최악을 경우를 가정해야 하므로 이 두 경우의 max를 찾아야 하고 이렇게 정리된다.

5) 2023년 추가함.

$$B(3k+1) = \max\begin{cases} B(k+1)+1, & (k,k,k+1) \\ B(k)+1 \end{cases}$$

만일 $3k+2$일 때 $(k,k,k+2)$라고 하자. 처음 비교한 k개 뭉치의 무게가 같으면나머지 $k+2$ 묶음에서 찾아야 하므로 총 $B(k+2)+1$ 번 필요하다. 만일 처음 비교한 한 쪽으로 내려간다면 다시 그 쪽에서 가짜를 찾아야 하므로 $B(k)+1$ 이다. 따라서 다음과 같이 정리된다.

$$B(3k+2) = \max\begin{cases} B(k+2)+1 \\ B(k)+1 \end{cases}$$

만일$N=2k,\ 2k+1$ 인 경우도 생각해볼 수 있다. 이 경우 $(k,\ k,\ 1)$로 나눠 측정을 해본다. 만일 올린 (k,k)가 같다면 k개 묶음의 한 쪽에 답이 있으므로 이 경우 $B(k)+1$된다. 만일 양쪽의 무게가 같아지면 나머지 제껴둔 1개가 가짜이므로 2번이면 충분하다. 단 이 경우는 거의 best에 해당된다. 따라서 $(k,k,\ \{1,2\})$로 분할할 경우 최악의 경우를 가정한다면 $B(2k)=B(k)+1,\ B(2k+1)=B(k)+1$가 된다. 이 식을 중심으로 문제를 "분할정복"을 해야 한다. 따라서 (k,k)로 나눌 경우에는 다음과 같이 정리된다.

$$B(2k) = B(2k+1) = B(k)+1$$

이 식을 중심으로 $3k+1$로 고려할 경우와 $2k+$로 계산해서 그 minimum을 선택해야 한다.

아래 표를 이용해서 B(N)을 실제 구해보자.

N	1	2	3	4	5	6	7	8	9	10	11	12	13
$B(N)$	0	1	1	2	2	2	2						

이제 $N=8$을 계산해보자.

8은 $3k+2,\ (k=2)$로도 볼 수 있고 4*2로도 볼 수 있으므로 앞서의 식에 적용하면 다음과 같다.

$$B(3k+2) = \max\begin{cases} B(k+2)+1 \\ B(k)+1 \end{cases}$$

$$B(3\cdot 2+2) = \max\begin{cases} B(2+2)+1 & =3 \\ B(2)+1 & =2 \end{cases} \quad = 3$$

또는 $8=4\cdot 2$로도 볼 수 있으므로 $B(8)=B(4)+1=3$ 이다. 이제 $N=9$인 경우를 살펴보자. 이것을 $(3,3,3)$으로 나누면 $B(3)+1=2$ 만에 가능하다. 만일 홀짝으로 본다면 $(4,4,1)$로 나는 것으로 이 경우에는 $B(4)+1$=3 이므로 $B(3)+1$ 보다 큰 값이다. 따라서 $B(9)=3$이다. 3묶음으로 나눗 처리한다.

$N=10$인 경우를 보자. 만일 $(3,3,4)$로 나누면 아래 식을 적용할 수 있다.

$$B(3k+1)=\max\begin{cases}B(k+1)+1, & (k,k,k+1)\\ B(k)+1\end{cases}$$

$$B(3*3+1)=\max\begin{cases}B(3+1)+1, & =3\\ B(3)+1 & =2\end{cases} \quad = 3$$

만일 $(4,4,2)$ 또는 $(5,5,0)$으로 한다면 각각 $\min\{B(4)+1,\ B(5)+1\}$ 이므로 그 값은 3이 된다. 따라서 $B(10)=3$ 이다. 방법은 (4,4,2), (5,,5,0)로 측정을 하면 된다.