POSCAT Seminar 3-1 : Brute Force

yougatup @ POSCAT

Topic

- Topic today
  - Brute Force
    - Time Complexity
    - Algorithm Design
    - Implementation

Brute Force

- 모든 경우를 고려하여 답을 찾자
  - 모든 경우를 고려하는 알고리즘을 생각하고 // **Verification**
  - 그 경우의 수가 실제로 얼마나 되는지를 계산 // **Efficiency**
- 가장 기본적인 접근
  - 어떤 문제를 받던 **가장 먼저** Brute Force하게 접근
  - 경우의 수가 **충분히 적다면** 단순히 Brute Force하게 풀면 된다
  - 단순하게 생각 !

Brute Force

- Example

  N개의 숫자가 주어졌을 때, 연속부분 최대합을 구하라

2	-4	1	2	-1	3	-7	2	1	-4	-2

Brute Force

- Example

  N개의 숫자가 주어졌을 때, 연속부분 최대합을 구하라

Brute Force

- Example

  가장 먼저 Brute Force하게 생각을 해 보자

2	-4	1	2	-1	3	-7	2	1	-4	-2

Brute Force

- Example

  가장 먼저 Brute Force하게 생각을 해 보자

  → 모든 경우를 고려한다는 뜻은 무엇인가?

2	-4	1	2	-1	3	-7	2	1	-4	-2

Brute Force

- Example

  가장 먼저 Brute Force하게 생각을 해 보자

  → 모든 경우를 고려한다는 뜻은 무엇인가?
  → 모든 연속부분의 합을 구해보고 그 중에 최댓값 선택 !

2	-4	1	2	-1	3	-7	2	1	-4	-2

Brute Force

- Example

  모든 연속부분의 합은 어떻게 구할 것인가 ?

2	-4	1	2	-1	3	-7	2	1	-4	-2

Brute Force

- Example

  모든 연속부분의 합은 어떻게 구할 것인가 ?

  연속 부분의 시작점과 끝점을 정하면 구간이 정해짐

Brute Force

- Example

  모든 연속부분의 합은 어떻게 구할 것인가 ?

  연속 부분의 시작점과 끝점을 정하면 구간이 정해짐

  가능한 모든 시작점과 끝점을 선택하여 합을 구하면 되는군

2	-4	1	2	-1	3	-7	2	1	-4	-2

Brute Force

- Example

  모든 연속부분의 합은 어떻게 구할 것인가 ?

  연속 부분의 시작점과 끝점을 정하면 구간이 정해짐

  가능한 모든 시작점과 끝점을 선택하여 합을 구하면 되는군

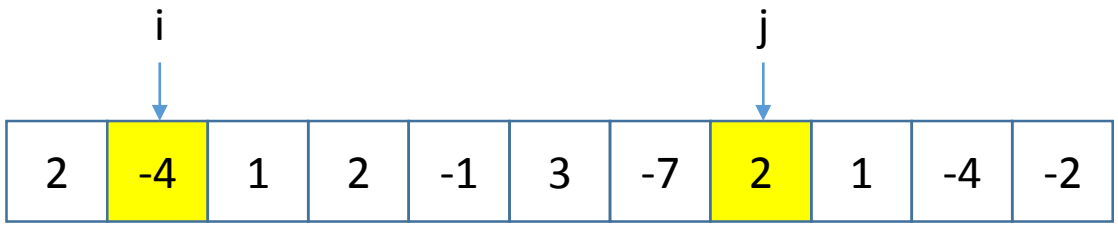
2	-4	1	2	-1	3	-7	2	1	-4	-2

More concretely !

Brute Force

- Example

  i를 시작점이라 하고, j를 끝점이라 하자

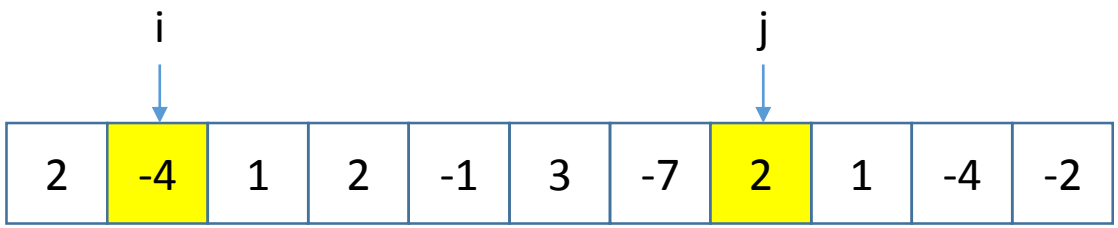


Brute Force

- Example

  i를 시작점이라 하고, j를 끝점이라 하자

  모든 가능한 i와 j에 대하여 D[i] ~ D[j] 의 합을 구하면 됨

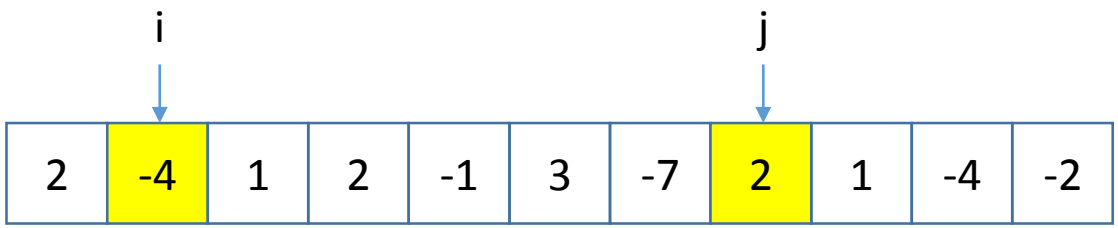


Brute Force

- Example

  i를 시작점이라 하고, j를 끝점이라 하자

  모든 가능한 i와 j에 대하여 D[i] ~ D[j] 의 합을 구하면 됨



GOOD !

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ?

Efficiency : How long does it **take** ?

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?

Rough Criteria

- 약 1억 번의 연산이 1초라고 생각
  - Time Complexity 를 계산했을 때 '1억'이면 대략 1초
  - O-notation 자체가 **근사**이기 때문에 **완전히 믿기는 힘듦**
  - $O(n^3)$ VS $O(99999999n^2)$
- 의심되면 좀 더 정확하게 계산
  - Constant factor 까지 좀 더 **세밀하게** 고려하여 계산
  - 어느 정도의 **감**이 있는 것이 굉장히 중요
- 보통 의도한 솔루션으로 풀면 ACCEPT
  - 계산 결과가 아슬아슬하면 대부분의 경우 **방법이 틀림**
  - 알고 보니 **해법 자체**가 아슬아슬한 문제라면 **굉장히 어려운 문제**!!

out of scope ☺

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 100$ and Time Limit = 1s

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 100$ and Time Limit = 1s

$$n^3 = 1{,}000{,}000 \leq 100{,}000{,}000$$

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 100$ and Time Limit = 1s

$$n^3 = 1{,}000{,}000 \leq 100{,}000{,}000$$

GOOD!!

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 1000$ and Time Limit = 1s

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 1000$ and Time Limit = 1s

$$n^3 = 1{,}000{,}000{,}000 > 100{,}000{,}000$$

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 1000$ and Time Limit = 1s

$$n^3 = 1{,}000{,}000{,}000 > 100{,}000{,}000$$

BAD, this solution is wrong

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^3 = 103{,}823{,}000 > 100{,}000{,}000$$

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

BAD?

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^3 = 103{,}823{,}000 > 100{,}000{,}000$$

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

BAD?
No. n^3 is over-estimated

In fact, It takes less than $\frac{n^3}{2}$

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^3 = 103{,}823{,}000 > 100{,}000{,}000$$

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

BAD?
No. n^3 is over-estimated

In fact, It takes less than $\frac{n^3}{2}$

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$\frac{n^3}{2} = 51{,}911{,}500 \leq 100{,}000{,}000$$

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

GOOD ENOUGH!

BAD?
No. n^3 is over-estimated

In fact, It takes less than $\frac{n^3}{2}$

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$\frac{n^3}{2} = 51{,}911{,}500 \leq 100{,}000{,}000$$

It is hard to calculate perfectly

- 완벽하게 계산하는 것은 굉장히 어려움
  - 당장 이전 슬라이드의 3중 for문이 $\frac{n^3}{2}$ 보다는 확실히 덜 걸림
  - 문제를 많이 풀다가 보면 감이 생깁니다
- 아슬아슬하면 다른 솔루션을 찾자
  - 아까도 이야기 했지만, 아슬아슬하면 솔루션이 틀린 경우
  - 다른 방법이 없을지를 생각하는 것이 좋다

out of scope ☺

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^3 = 103{,}823{,}000 > 100{,}000{,}000$$

BAD?

Brute Force

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^3 = 103{,}823{,}000 > 100{,}000{,}000$$

BAD?
Hmm… Is there another solution ?

Brute Force

We can remove this loop !

```
1 int result = -inf;   // -infinity
2 for(int i=0;i<n;i++){
3     for(int j=i;j<n;j++){
4         int sum = 0;
5         for(int k=i;k<=j;k++) sum += D[k];
6         result = max(result, sum);
7     }
8 }
```

Verification : Is this algorithm **true** ? We consider all the cases

Efficiency : How long does it **take** ? O(n^3)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^3 = 103{,}823{,}000 > 100{,}000{,}000$$

BAD?
Hmm… Is there another solution ?

Brute Force

```
 1 S[0] = D[0];
 2 for(int i=1;i<n;i++) S[i] = S[i-1] + D[i];
 3
 4 int result = -inf;
 5
 6 for(int i=0;i<n;i++){
 7     for(int j=i;j<n;j++){
 8         int sum = S[j] - S[i-1]; // sum of D[i] ~ D[j]
 9         result = max(result, sum);
10     }
11 }
```

Verification : Is this algorithm **true** ?

Efficiency : How long does it **take** ?

Brute Force

```
1 S[0] = D[0];
2 for(int i=1;i<n;i++) S[i] = S[i-1] + D[i];
3
4 int result = -inf;
5
6 for(int i=0;i<n;i++){
7     for(int j=i;j<n;j++){
8         int sum = S[j] - S[i-1]; // sum of D[i] ~ D[j]
9         result = max(result, sum);
10    }
11 }
```

Verification : Is this algorithm **true** ? Still, we consider all the cases

Efficiency : How long does it **take** ? O(n^2)

Brute Force

```
1 S[0] = D[0];
2 for(int i=1;i<n;i++) S[i] = S[i-1] + D[i];
3
4 int result = -inf;
5
6 for(int i=0;i<n;i++){
7     for(int j=i;j<n;j++){
8         int sum = S[j] - S[i-1]; // sum of D[i] ~ D[j]
9         result = max(result, sum);
10    }
11 }
```

Verification : Is this algorithm **true** ? Still, we consider all the cases

Efficiency : How long does it **take** ? O(n^2)

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^2 = 220{,}900 \leq 100{,}000{,}000$$

Brute Force

```
1 S[0] = D[0];
2 for(int i=1;i<n;i++) S[i] = S[i-1] + D[i];
3
4 int result = -inf;
5
6 for(int i=0;i<n;i++){
7     for(int j=i;j<n;j++){
8         int sum = S[j] - S[i-1]; // sum of D[i] ~ D[j]
9         result = max(result, sum);
10    }
11 }
```

Verification : Is this algorithm **true** ? Still, we consider all the cases

Efficiency : How long does it **take** ? O(n^2)

Completely Right !

Is this algorithm efficient to solve this problem ?
Suppose $n \leq 470$ and Time Limit = 1s

$$n^2 = 220{,}900 \leq 100{,}000{,}000$$

Brute Force

- 어떤 문제를 잡건 **시작은 Brute Force !!!!!**
  - **문제를 이해**하는 것을 도와주고
  - **전체 경우의 수**가 얼마나 되는지를 계산하게 하며
  - **솔루션**의 **모양**, **특성**을 찾는 것을 도와준다
- Recursion이 합쳐지면 완벽한 Brute Force
  - **Backtracking** 이 모든 시도를 다 해보는 것
  - Recursion 없이는 모든 경우를 탐색하지 못하는 경우가 있습니다
  - n개의 숫자 중에서 m개 뽑기 → We need n for loops
    How can you generate n for loops ?

Question ?

- To-do List
  - Basic implementation problems