

시간복잡도, 완전탐색, STL

들어가기 전에 : 컴퓨터처럼 생각하기

- Problem Solving은 주어진 문제를 제한된 메모리와 시간안에 해결하는 것
- 그 과정에서 여러 알고리즘을 활용해 최적화
- 내가 생각한 풀이가 해당 문제의 제한과 맞을지 생각하는 것이 매우 중요
- 무작정 제출 했다가는 **시간 초과, 메모리 초과** 등 오답을 받을 수 있음
- 어떻게 나의 코드, 풀이가 해당 문제의 제한 안에 돈다는 것을 보장할까?

시간복잡도

연습 문제

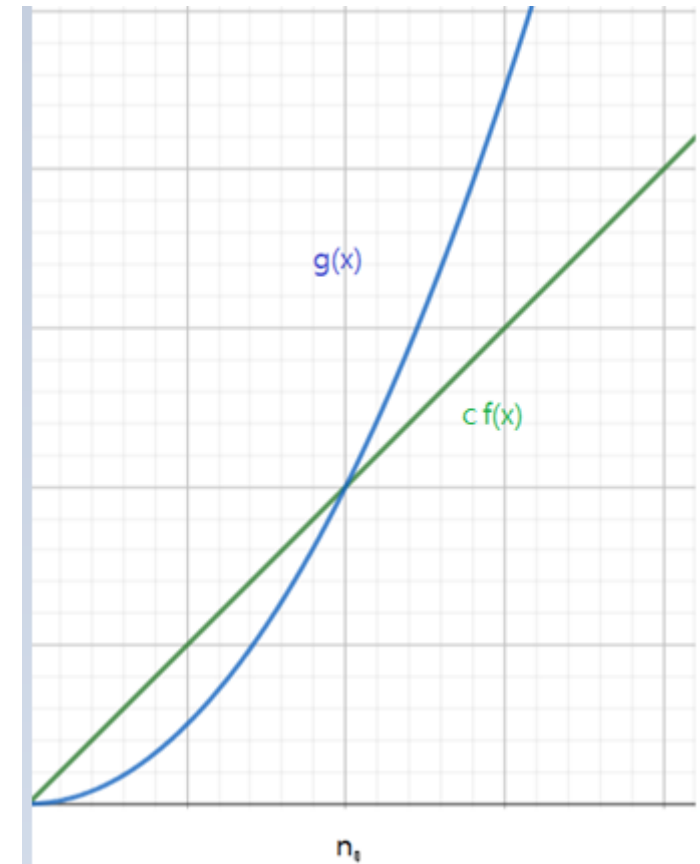
- [24262](#) : 알고리즘 수업 – 알고리즘 수행 시간1
- [24263](#) : 알고리즘 수업 – 알고리즘 수행 시간2
- [24264](#) : 알고리즘 수업 – 알고리즘 수행 시간3

시간복잡도

- 정의: 문제를 해결하는데 걸리는 시간과 **입력 크기의 관계**를 나타내는 함수
- PS에서는 네트워크 통신, CPU 점유 등은 신경 안 씀
- 연산을 몇 번 수행하는지 확인하면 수행 시간을 대략적으로 유추할 수 있음
- 최악의 경우가 중요함

O-Notation

- 정의: $O(f(x)) = \{ g(x) \leq cf(x) \ (x \geq n_0) \}$ 을 만족하는 양의 상수 c 와 n_0 가 존재 }
- $f(x) = x, g(x) = x^2$
- Let $c = 1, n_0 = 1 \rightarrow 1 \times f(1) = g(1)$
- $x > 1, f(x) < g(x)$ 정의를 만족하지 않으므로,
- 따라서 $O(x) \neq x^2$

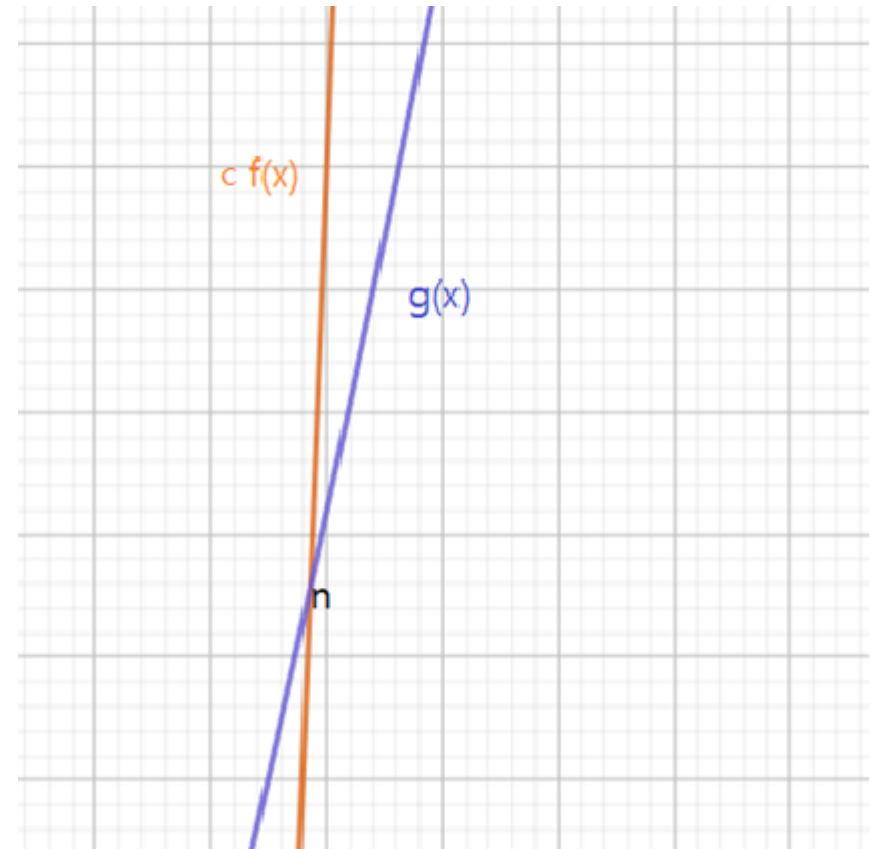


O-Notation

- 정의: $O(f(x)) = \{ g(x) \leq cf(x) \ (x \geq n_0) \}$ 을 만족하는 양의 상수 c 와 n_0 가 존재 }
- $f(x) = 1, g(x) = 10$
- $c = 10, n_0 = 1$
- $x \geq n_0, 10f(x) \geq g(x)$ 는 정의를 만족하므로
- $O(1) = 10$

O-Notation

- 정의: $O(f(x)) = \{ g(x) \leq cf(x) \ (x \geq n_0) \}$ 을 만족하는 양의 상수 c 와 n_0 가 존재 }
- $f(x) = x^2$, $g(x) = 10x^2 + 10000x + 2$
- Let $c = 1000$, $n_0 = 10^4$
- 대입하면, $10^{11} \geq 2 \times 10^9 + 2$
- $x > n_0$, $cf(x) \geq g(x)$ 를 만족하니
 $O(n^2) = g(n)$ 을 만족



O-Notation

- 쉽게 생각하면 함수 $g(x)$ 의 최고차항 x^n 은 $O(x^n) = g(x)$ 를 만족함
- 중요한 것은 최고차항보다 낮은 항들을 무시하는 것
- $g(x)$ 가 지수 함수일 경우라면 밑이 중요함
- $O(2^x) = 2^x + 10x$

질문?

쉬운 문제

- 1부터 n 까지의 합을 구하여라

$$1 + 2 + 3 + 4 + \dots + n$$

쉬운 문제 – 간단한 풀이

- 반복문을 통해서 변수 S 에 1부터 n 까지 다 더한다
- 이 풀이의 시간 복잡도는 어떻게 될까? (S 가 변하는 횟수에 집중)

```
int solve(int n){  
    int S = 0;  
    for(int i=1;i<=n;i++){  
        S += i;  
    }  
    return S;  
}
```

쉬운 문제 – 수학적 풀이

- 등차수열 합 공식을 사용한다.
- 이 풀이의 시간 복잡도는 어떻게 될까?

```
int solve(int n){  
    int S = n*(n+1)/2;  
    return S;  
}
```

알고리즘 수행 시간 1 BOJ 24262

```
MenOfPassion(A[], n) {  
    i = [n / 2];  
    return A[i];  
}
```

- **문제:** 다음과 같은 코드의 반복횟수와 시간복잡도 최고차항의 차수를 출력하라
- **풀이:** 어떤 값이든 변수 i 에 1번 할당. 이는 시간복잡도로 $O(1)$ 이고 최고차항의 차수는 0
- **정답:** 1번, 0

알고리즘 수행 시간 2 BOJ 24263

```
MenOfPassion(A[], n) {  
    sum <- 0;  
    for i <- 1 to n sum <- sum + A[i];  
    return sum;  
}
```

- 풀이: sum이 총 n 번 갱신되므로, 총 횟수는 n 번,
시간복잡도는 $O(n)$, 최고차항은 1
- 정답: 1번, 0

알고리즘 수행 시간 3 BOJ 24264

```
MenOfPassion(A[], n) {  
    sum <- 0;  
    for i <- 1 to n  
        for j <- 1 to n  
            sum <- sum + A[i] × A[j];  
    return sum;  
}
```

- 풀이: sum이 $n \times n$ 번 갱신되고 이는 시간복잡도로 $O(n^2)$
- 정답: n^2 번, 2

질문?

Bruteforcing

연습 문제

- [2798](#) : 블랙잭
- [19532](#) : 수학은 비대면 강의입니다
- [2309](#) : 일곱 난쟁이
- [1145](#) : 적어도 대부분의 배수
- [1977](#) : 완전 제곱수
- [1436](#) : 영화감독 숨
- [2839](#) : 설탕 배달

브루트포스 (완전탐색)

- 정의: 모든 경우를 다 탐색해보는 것
- 제일 컴퓨팅적인 문제 풀이 방식이라고 할 수 있음
- 보통 1초 \approx 1억이라고 가정하기 때문에 시간복잡도를 잘 계산해야 함
- ex) $N \leq 10000$, 시간제한이 1초라면, $O(N^2)$ 알고리즘은 사용 가능

블랙잭 BOJ 2798

- 문제: N 개의 카드 중에서 M 을 넘지 않는 3장의 최대합을 출력 ($3 \leq N \leq 100$), ($10 \leq M \leq 300000$), 시간제한 1초

예제 입력 1 복사

```
5 21
5 6 7 8 9
```

예제 출력 1 복사

```
21
```

블랙잭 BOJ 2798

- 풀이: 3겹의 반복문으로 3장의 카드를 뽑을 경우의 수를 다 탐색
- 시간복잡도는 $O(n^3)$
- n 은 최대 100 이고, $100^3 = 1000000$ 은 1억보다 작음
- 따라서 시간 제한인 1초 내에 돌 수 있음(1초 \approx 1억)

블랙잭 BOJ 2798

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int n,m; int a[100];
6     cin>>n>>m;
7     int MAX = 0;
8     for(int i=0;i<n;i++){
9         cin>>a[i];
10    }
11    for(int i=0;i<n;i++){
12        for(int j=0;j<n;j++){
13            if(i==j) continue;
14            for(int k=0;k<n;k++){
15                if(k==i || k==j) continue;
16                const int val = a[i] + a[j] + a[k];
17                if(val <= m && MAX < val) MAX = val;
18            }
19        }
20    }
21    cout<<MAX;
22
```

일곱 난쟁이 BOJ 2309

- **문제:** 아홉 명의 난쟁이들이 존재, 각각의 키가 존재할 때 7명의 합이 100인 난쟁이들을 출력. 시간제한 2초

예제 입력 1 복사

```
20
7
23
19
10
15
25
8
13
```

예제 출력 1 복사

```
7
8
10
13
19
20
23
```


일곱 난쟁이 BOJ 2309

- 풀이 1: 아홉 명 중 7명을 뽑는 모든 경우를 탐색 $O(n^7)$
- $O(9^7) = 4,782,969$
- 풀이 2: 아홉 명 전체의 키를 합한 뒤,
 - 2명씩 뽑아서 전체 합 - 2명의 키 == 100 이 되는지를 확인
 - 전체 합을 구하는 과정 $O(n)$, 2명을 고르는 과정 $O(n^2)$

일곱 난쟁이 BOJ 2309

```
4 int main(void){
5     int a[9];
6     int S = 0;
7     for(int i =0;i<9;i++){
8         cin>>a[i];
9         S += a[i];
10    }
11    sort(a,a+9);
12    for(int i =0;i<9;i++){
13        for(int j=0;j<9;j++){
14            if(S-a[i]-a[j] == 100){
15                a[i] = a[j] = 100;
16                i = j = 10;
17            }
18        }
19    }
20    for(int i=0;i<9;i++){
21        if(a[i]==100) continue;
22        cout<<a[i]<<'\\n';
23    }
24    return 0;
25 }
```

영화감독 숨 BOJ 1436

- **문제:** 제일 작은 종말의 수는 666이고, 그 다음으로 큰 수는 1666, 2666, 3666, 이다. N 번째 종말의 수를 구하여라 ($1 \leq N \leq 10000$)
- 시간제한 2초

예제 입력 1 복사

2

예제 출력 1 복사

1666

영화감독 숨 BOJ 1436

- 풀이: 1만 번째 종말의 수를 예상해보자
- 생각해보면 대략 $666 * 10000$ 쯤일 것이다
- 666부터 6660000 까지 반복문을 돌리고 666이 들어간다면 카운트
- 카운트 된 값이랑 N 이랑 같다면 출력
- $O(6660000 - 666) \leq O(10^8)$

영화감독 소름 BOJ 1436

```
3 int check(int a){
4     int c= 0;
5     while(a>0){
6         if(a%10 == 6) c++;
7         else c = 0;
8         if(c==3) return 1;
9         a /= 10;
10    }
11    return 0;
12 }
13 int main(void){
14     int i,c,N;
15     c=0;
16     cin>>N;
17     for(i=666;i<66600001;i++){
18         if(check(i)){
19             c++;
20         }
21         if(c==N) break;
22     }
23     cout<<i;
24
25     return 0;
26 }
```

질문?

Standard Template Library(STL)

연습 문제

15552

: 빠른 $A+B$

Standard Template Library

- C++ 에서 제공하는 표준 템플릿 라이브러리
- 컨테이너: pair, vector, tuple, queue, priority_queue, stack, set, map
- 알고리즘: sort, lower_bound, ...
- Iterator: begin, end, rbegin, rend, ...
- Template 자료형 사용으로 호환이 장점.

vector<T>

- 여러 개의 자료를 담는 가변 길이의 컨테이너
- push_back을 통해 배열의 끝에 값을 추가할 수 있다
- emplace_back을 통해 해당 객체의 생성자를 부를 수 있다
- 삽입 시간복잡도는 amortized $O(1)$

```
vector<int> v;  
v.push_back(10);
```

```
vector<pair<int, int>> points;  
points.push_back(make_pair(2, 3));  
points.push_back({2, 3})  
points.emplace_back(2, 3);
```

pair<T1,T2>

- 두 개의 자료를 담는 컨테이너
- 가져올 때는 first, second로 가져온다.

```
pair<int, int> p1 = {1, 5};  
pair<int, int> p2 = make_pair(1, 5);  
auto p3 = make_pair(1, 5);  
cout << p1.first << " " << p2.second << '\n';
```

tuple<T1,T2,T3>

- 세 개의 자료를 담는 컨테이너
- 가져올 때에는 get<index>(tuple)로 가져온다

```
tuple<int, int, string> t = {1, 2, "123"};  
cout << get<1>(t) << '\n';
```

```
vector<tuple<string, int, int>> v;  
v.emplace_back("abc", 100, 123);
```

Iterator

- 컨테이너의 위치를 나타내는 클래스, 포인터와 같이 동작한다
- begin, end는 각각 시작 위치, 끝 뒤 위치를 가리킨다
- 끝 위치를 가리키고 싶다면 end() - 1이다
- 포인터처럼 참조연산이 가능하다

```
vector<int> v;  
v.push_back(1);  
v.push_back(2);  
v.push_back(3);  
cout << *v.begin() << '\n';  
cout << *(v.end() - 1) << '\n';
```

sort

- [first,last) 범위를 비교 기준에 따라 정렬
- 기본적인 비교 함수는 operator<
- 비교함수를 원하는 대로 작성할 수 있다
- 비교함수 `bool compare(T a, T b)`
- `sort(시작주소, 끝 뒤 주소, 비교함수)`

```
vector<int> v;  
v.push_back(4);  
v.push_back(1);  
v.push_back(2);  
sort(v.begin(), v.end(), cmp);
```

빠른 $A+B$ BOJ 15552

- **문제:** 테스트 케이스 수 T 가 주어질 때, 각 케이스마다 두 수의 입력을 받고 합을 출력하라. ($0 < T \leq 1,000,000$), ($0 < A, B \leq 1000$)
- 시간 제한 1초

예제 입력 1 복사

```
5
1 1
12 34
5 500
40 60
1000 1000
```

예제 출력 1 복사

```
2
46
505
100
2000
```

빠른 A+B BOJ 15552

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a, b, t;
    cin >> t;
    for(int i = 0; i < t; i++){
        cin >> a >> b;
        cout << a + b << endl;
    }
    return 0;
}
```

- 다음과 같은 코드는 **시간 초과**를 받게 된다
- 왜 **시간 초과**가 발생 할까?

빠른 $A+B$ BOJ 15552

- 프로그램 입출력 시간은 수행 시간에 포함됨
- 입출력은 굉장히 오래 걸리는 작업
- 특히 `endl`은 버퍼를 지우는 작업이 느림
- 입/출력이 방대한 경우(1만 정도), 그 시간만으로 시간 초과할 수 있다.



빠른 $A+B$ BOJ 15552

- 입출력 싱크를 끊고, 한번에 출력하도록 설정
- 아래 세 줄을 main 바로 아래 추가
- 이 경우 **cin**, **cout** 만을 사용하고, **scanf**, **printf**와 혼용 금지
- **endl**은 버퍼를 지우므로 개행문자 ('\n')를 사용하자
- 입출력 양이 10,000줄을 넘기면 사용을 고려하자

```
cin.tie(nullptr);  
cout.tie(nullptr);  
ios::sync_with_stdio(false);
```

빠른 A+B BOJ 15552

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     cin.tie(nullptr);
6     cout.tie(nullptr);
7     ios::sync_with_stdio(0);
8     int a,b,t; cin>>t;
9     while(t--){
10         cin>>a>>b;
11         cout<<a+b<<'\\n';
12     }
13
14 }
```

제출 번호	아이디	문제	결과
74730498	 s6xybr8in	 15552	맞았습니다!!

빠른 A+B BOJ 15552

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int a,b,t; scanf("%d",&t);
6     while(t--){
7         scanf("%d %d",&a,&b);
8         printf("%d\n",a+b);
9     }
10
11 }
```

60530896

 s6xybr8in

 15552



맞았습니다!!

빠른 A+B BOJ 15552

- 파이썬을 사용하는 경우, 아래와 같이 사용하자
- readline의 경우 개행문자까지 받아온다
- 문자열을 받고 싶다면 rstrip을 통해 개행문자를 제거하자

```
import sys
s = sys.stdin.readline().rstrip()
```

빠른 A+B BOJ 15552

제출 번호	아이디	문제	결과
74730841	 s6xybr8in	 15552	맞았습니다!!

```
import sys
for case in range(int(sys.stdin.readline().rstrip())):
    a,b = map(int,sys.stdin.readline().rstrip().split())
    print(a+b)
```

Reference

- <https://github.com/justiceHui/SSU-SCCC-Study/blob/master/2023-summer-basic/slide/04-3-cpp-sort.pdf>
- <https://github.com/KU-AIKon/study?tab=readme-ov-file>