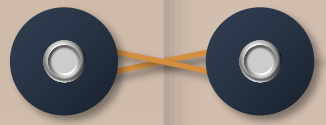


알고리즘 세미나 OT

PoolC 2021 1학기

1. 세미나 진행방식

1. 알고리즘 개념 설명
2. 알고리즘 문제 풀이
3. 지난 시간 도전문제 풀이
4. 자유로운 질문 및 토의
5. 오늘의 도전문제 공유(백준 그룹)

	진행 내용
1주차	OT, 재귀
2주차	완전탐색, 백트래킹
3주차	다이나믹 프로그래밍, 이분탐색
4주차	깊이 우선 탐색
5주차	너비 우선 탐색
6주차	다익스트라, 플로이드 워셜
7주차	그리디, 비트마스킹
8주차	유니온 파인드, 최소 스패닝 트리

2. 알고리즘에 대해서

알고리즘이란? → 주어진 문제를 논리적으로 해결하기 위한 방법

왜 중요한 것일까? → 단순 코드 구성 차이만으로 최적의 효율을 꾀할 수 있다.

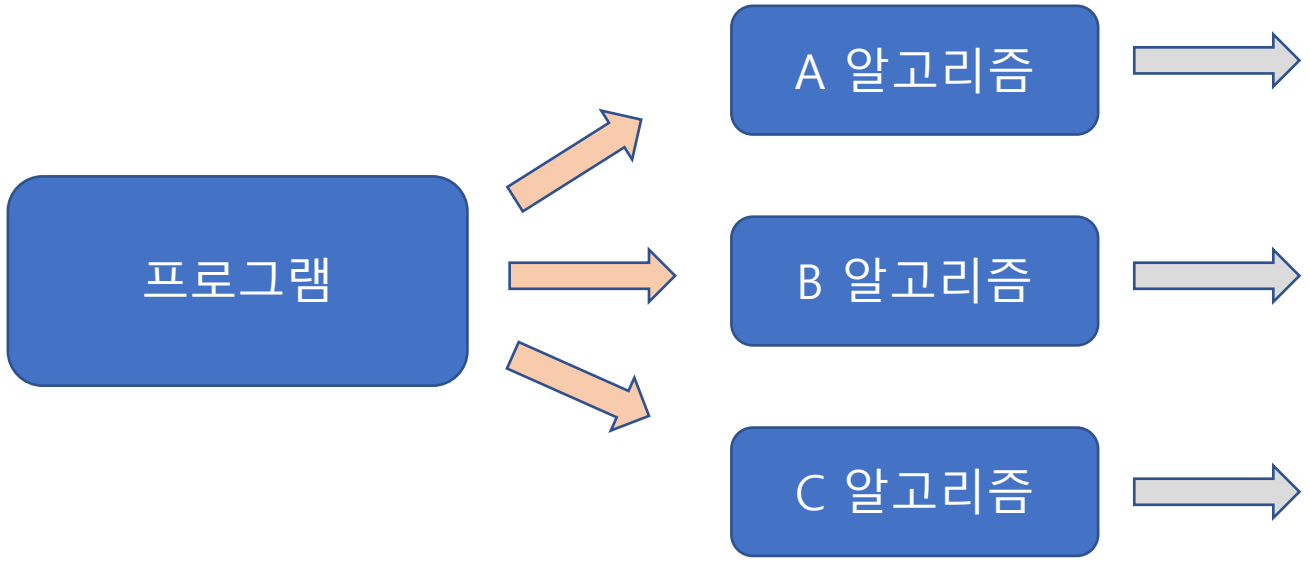
어떻게 접근해야 할까? → 간단한 문제에서부터 출발해서 요리조리 생각해보자!

3. 알고리즘의 3요소

시간

공간

원하는 결과



시간	공간	원하는 결과
10시간	100MB	O
1초	1TB	O
1초	100MB	X

4. 백준? 프로그래머스?

백준(BOJ) : 온갖 알고리즘 문제들을 모아 놓은 사이트! https://www.acmicpc.net/

Solved.ac : 백준의 모든 문제들을 난이도를 매겨주는 사이트! https://solved.ac/

경험치 올리는 재미를 느끼게 해준다

프로그래머스 : 문제 수는 적지만 기출문제들만 모아 놓은 사이트! https://programmers.co.kr/

5. 본격적인 PS에 들어가기에 앞서..

1. 빠른 입출력	https://www.acmicpc.net/board/view/22716
2. 개행 문자	endl -> "₩n"
3. 언어 선택	자신 있는 언어로 하되, 만약 없다면 c++이나 python 추천!
4. 전역 변수	전역변수의 장점을 최대한 활용하자.
5. 각종 에러	런타임 에러, 오버플로우, … https://www.acmicpc.net/help/rte
6. 인풋 입력	freopen OR 명령인수전달

(선택 사항)

6. 시간복잡도

느낌적인 느낌만 가지고 가자!

상수는 무시하고, 시간 복잡도에서 제일 큰 영향을 주는 것만 가져온다!

Big-O 표기법 : O(N), O(N^2), O(NlogN), O(2^N), ...

7. 시간복잡도

O(N)

```
for (int i = 1; i <= N; i++)
{
    cout << i << endl;
}
```

O(logN)

```
int i = N;
while (i>0)
{
    cout << "Hi" << endl;
    i /= 2;
}
```

O(N^2)

```
for (int i = 1; i <= N; i++)
{
    for (int j = 1; j <= N; j++)
    {
        cout << i*j << endl;
    }
}
```

O(NlogN)

```
int i = N;
while (i>0)
{
    for (int i = 0; i < N; i++)
    {
        cout << "Hi" << endl;
    }
    i /= 2;
}
```

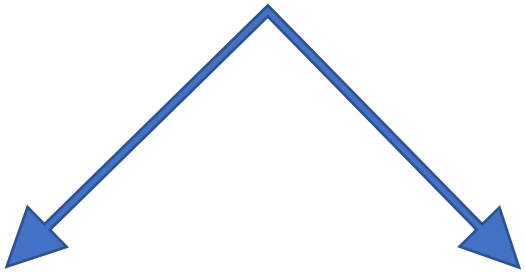
8. 시간과 공간의 상관관계

int arr[10]	4	6	2	-1	5	6	9	3	2	3

2번째 원소부터 6번째 원소까지의 합을 구하자!

```
int result = 0;
for (int i = 1; i < 6; i++)
{
    result += arr[i];
}
```

어쨌든 N개



result = arr[1] + arr[2] + … + arr[5]

시간복잡도 : O(N)

9. 시간과 공간의 상관관계

int arr[10]	4	6	2	-1	5	6	9	3	2	3

2번째 원소부터 6번째 원소까지의 합을 구하자!

int sum[10]	4	10	12	11	16	22	31	34	36	39

추가 메모리

```
int result = 0;
result = sum[5] - sum[0];
```

시간복잡도 : O(1)

10. 재귀(Recursion)

재귀함수

- 종료조건이 꼭 필요하다!

- 쓰이는 곳이 정말 많다!

- 단독적으로 쓰일 수도 있지만 대부분 특정 알고리즘을 구현하는데 기본이 됨

```
int main()
{
    Recur(5);
    return 0;
}
```

```
void Recur(int x)
{
    if (x == 0)
        return;
    cout << "Hi\n";
    Recur(x - 1);
}
```

```
Hi
Hi
Hi
Hi
Hi
계속하려면 아무 키나 누르십시오 . . .
```

11. 재귀의 동작 과정

종료조건 설정이 중요!

```
int main()
{
    Recur(1);
    return 0;
}
```

(x == 1)

```
void Recur(int x)
{
    if (x == 4)
        return;
    cout<<"Hi "<<x<<endl;
    Recur(x + 1);
    cout<<"Bye "<<x<<endl;
}
```

(x == 2)

```
void Recur(int x)
{
    if (x == 4)
        return;
    cout<<"Hi "<<x<<endl;
    Recur(x + 1);
    cout<<"Bye "<<x<<endl;
}
```

(x == 3)

```
void Recur(int x)
{
    if (x == 4)
        return;
    cout<<"Hi "<<x<<endl;
    Recur(x + 1);
    cout<<"Bye "<<x<<endl;
}
```

(x == 4)

```
void Recur(int x)
{
    if (x == 4)
        return;
    cout<<"Hi "<<x<<endl;
    Recur(x + 1);
    cout<<"Bye "<<x<<endl;
}
```

```
Hi 1
Hi 2
Hi 3
Bye 3
Bye 2
Bye 1
계속하려면 아무 키나 누르십시오 . . .
```

12. 반복문으로도 할 수 있지 않을까?

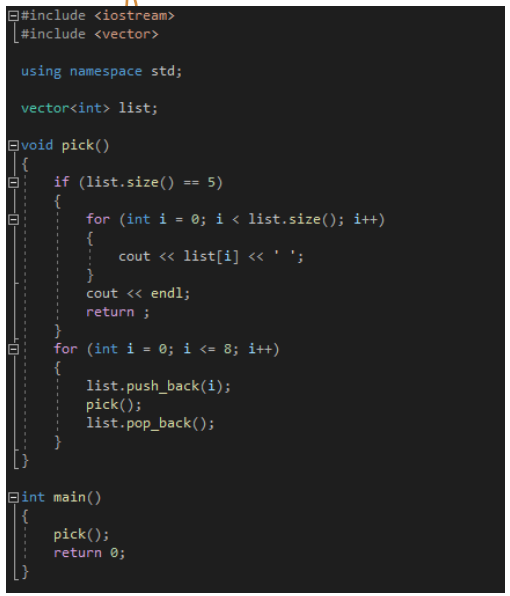
반복문 이용

```cpp
#include <iostream>

using namespace std;

int main()
{
    for (int i = 0; i <= 8; i++)
    {
        for (int j = 0; j <= 8; j++)
        {
            for (int k = 0; k <= 8; k++)
            {
                for (int l = 0; l <= 8; l++)
                {
                    for (int m = 0; m <= 8; m++)
                    {
                        cout <<i<<' '<<j<<' '<<k<<' '<<l<<' '<<m<<endl;
                    }
                }
            }
        }
    }

    return 0;
}
```

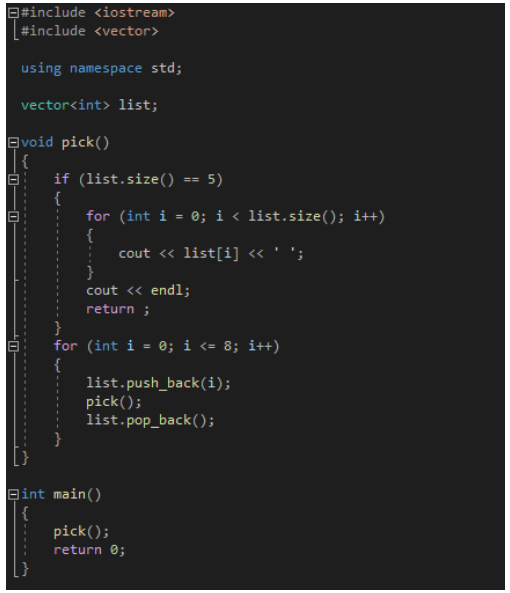
재귀 이용

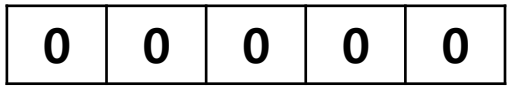
```cpp
#include <iostream>
#include <vector>

using namespace std;

vector<int> list;

void pick()
{
    if (list.size() == 5)
    {
        for (int i = 0; i < list.size(); i++)
        {
            cout << list[i] << ' ';
        }
        cout << endl;
        return ;
    }
    for (int i = 0; i <= 8; i++)
    {
        list.push_back(i);
        pick();
        list.pop_back();
    }
}

int main()
{
    pick();
    return 0;
}
```

13. 재귀의 이해

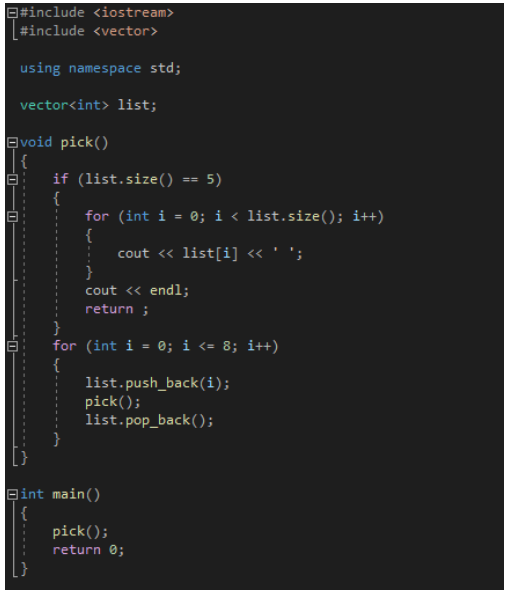


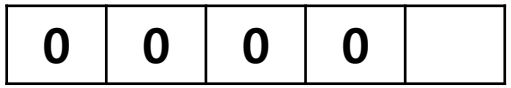
Depth : 0



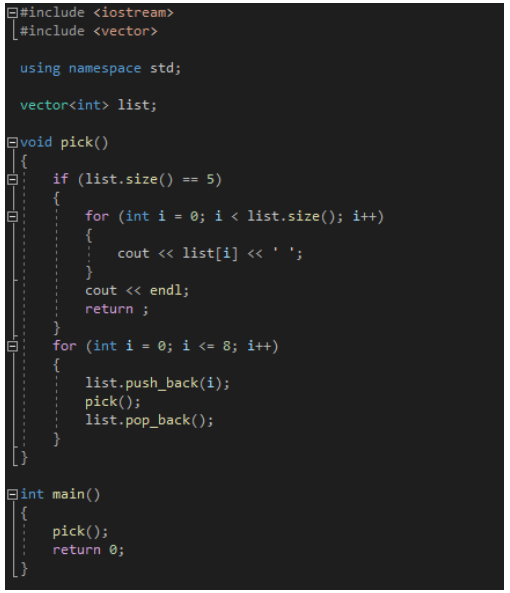


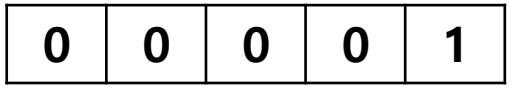
Depth : 5





Depth : 4





Depth : 5

14. 재귀를 이용한 문제 소개

1. 합 : https://www.acmicpc.net/problem/8393

2. 팩토리얼 : https://www.acmicpc.net/problem/10872

3. 재귀함수가 뭔가요? : https://www.acmicpc.net/problem/17478

4. 피보나치 수 : https://www.acmicpc.net/problem/2747

15. 도전 문제!

1. 3의배수 : https://www.acmicpc.net/problem/1769

2. 종이의 개수 : https://www.acmicpc.net/problem/1780

3. 색종이 만들기 : https://www.acmicpc.net/problem/2630

4. Z : https://www.acmicpc.net/problem/1074

16. 자유로운 질문 및 토의!