알고리즘 스터디 풀이 - 5회차

김성진

- **이번 주차의 c++에서 핵심은 시간 초과가 많이 일어남!**
- **해결 법 1)**
  - ```
    ios_base::sync_with_stdio(false);
    ```
  - ```
    cin.tie(NULL);
    ```
  - ```
    cout.tie(NULL);
    ```
- **해결 법 2)**
  - Scanf와 printf 사용하기
- **해결 법 3)**
  - endl은 버퍼 초기화까지 하므로 "₩n"으로 사용

2493번 – 탑

탑들의 높이가 주어졌을 때 주어진 탑 왼쪽에 그 탑 보다 높은 위치를 출력하고 없으면 0을 출력하는 문제입니다.

- 첫 접근은 그냥 해당 탑 왼쪽부터 살피면서 높이가 더 크면 그 위치를 출력하고 for문을 나가는 식으로 가지치기를 하려고 했으나 시간 초과가 떴습니다.
- Cin과 cout의 속도 때문인가 싶어서 검색을 하다가 좀 다르게 생각해야 되는 것을 알았습니다.
- 최대 크기의 탑의 높이를 저장하고 주어진 탑보다 높은 탑의 위치와 높이만 저장하면 됩니다.
    ```

- 예를들어 5 2 3 4 2 이 주어진다 가정했을 때
  - 처음 5 높이의 탑을 기억합니다.
  - 다음에 2의 높이의 탑도 저장합니다.
  - 그리고 3 높이의 탑이 입력되면 2 높이의 탑에는 레이저가 걸릴 수 없으므로 5,3 만 기억합니다.
  - 그리고 4 높이의 탑이 입력되면 3을 pop하고 5,4만 저장합니다.
  - 다시 2 높이가 들어오면 5, 4, 2를 저장해야 합니다. 왜냐하면 1 높이의 탑이 들어오면 2에 걸리고 3높이가 들어오면 4에 걸리기 때문입니다.
  - 즉, 저장된 탑보다 높은 크기가 입력되면 저장된 탑의 정보를 제거하고 입력된 탑의 정보를 저장하고 낮은 크기가 입력되면 정보를 추가합니다.

이제 이 것을 구현한 코드를 보면 아래와 같습니다.

```
// 2019 03 25
// Kim Seoung Jin
// BaekJoon online judge 2493
// https://www.acmicpc.net/problem/2493

#include <iostream>
#include <vector>
#include <stack>
using  namespace std;

stack<pair<int, int>> st;

int main()
{
        int n;
        long long value, max = -9999;
        vector<long long> top;
        bool flag = false;

        // 이거 안 쓰면 시간 초과뜸.... 시부렝....
        ios_base::sync_with_stdio(false);
        cin.tie(NULL);
        cout.tie(NULL);

        cin >> n;

        for (int i = 0; i < n; i++)
        {
                cin >> value;
```

```
                while (!st.empty())
                {
                        if (st.top().second > value)
                        {
                                cout << st.top().first << " ";
                                break;
                        }
                        st.pop();
                }

                if (st.empty()) cout << "0 ";
                st.push(make_pair(i + 1, value));
        }
        cout << endl;

        return 0;
}
```

9012번 – 괄호

이번 주에서 가장 쉬웠던 문제인 것 같습니다. 괄호가 올바르게 만들어지는지 확인하는 문제입니다.

괄호 쌍이 안 맞거나, 괄호가)(이런식으로 입력되면 NO를 출력합니다.

"(" 가 입력되면 +1 ")" 가 입력되면 -1을 해서 음수가 되면 종료되게 짰습니다.

이제 코드로 구현해보면 아래와 같은 코드입니다.

```
// 2019 03 25
// Kim Seoung Jin
// BaekJoon online judge 9012
// https://www.acmicpc.net/problem/9012

#include <iostream>
#include <string>
using namespace std;

int main()
{
        int n, count;
        string bracket;
```

```
        cin >> n;

        for (int i = 0; i < n; i++)
        {
                count = 0;
                cin >> bracket;
                for (int j = 0; j < bracket.length(); j++)
                {
                        if (bracket[j] == '(')
                                count++;
                        else
                                count--;

                        if (count < 0)
                        {
                                break;
                        }
                }
                if (count == 0) cout << "YES" << endl;
                else cout << "NO" << endl;
        }
}
```

9012번 – 스택 수열

문제 알고리즘은 빠르게 생각했으나 시간초과가 계속 걸려서 답답해서 검색해보니 이번주차 풀이 앞에서 적은 거처럼 cin, cout의 문제였습니다.
그래서 비슷하지만 두 가지 방법으로 작성했습니다.
1번은 자료구조를 사용해서 작성해보고 싶었습니다.

1) 먼저 만들어야 하는 수열을 하나의 queue에 저장했습니다. 왜냐하면 만들어야 하는 수열은 처음에 입력된 것부터 처리해야 하기 때문입니다.
2) 다음은 스택에 1~n까지를 하나하나 저장하고 queue에 저장된 값과 같으면 stack과 queue의 값을 pop 합니다.
3) 만약 스택이 비어 있으면 다음 값을 받아서 비교해야 하므로 while문을 나갑니다.
4) Queue가 empty이면 수열을 만든 것 이므로 while 문을 나갑니다.
5) 스택에 저장된 값이 수열 값보다 크면 만들 수 없으므로 NO를 출력하고 while문을 종료합니다.

위 방식으로 구현하면

```
// 2019 03 25
// Kim Seoung Jin
// BaekJoon online judge 1874
// https://www.acmicpc.net/problem/1874

#include <iostream>
#include <vector>
#include <stack>
#include <queue>
#include <string>
using namespace std;


int n, value;
queue<int> sequence; // 만들 순열
vector<char> answer;

int main()
{
	stack<int> stack; // 빠지지 않고 저장된 값
	bool flag = true;

	ios_base::sync_with_stdio(false);
	cin.tie(NULL);
	cout.tie(NULL);

	cin >> n;

	// 입력받을 수열을 저장
	// 먼저 뽑아야하는 수가 아래에 저장되므로 queue를 사용
	for (int i = 0; i < n; i++)
	{
		cin >> value;
		sequence.push(value);
	}

	do
	{
		for (int i = 0; i < n; i++)
		{
			// 1~N까지 stack에 저장
			stack.push(i + 1);
			answer.push_back('+');

			// 뽑아야 하는 값이 더 큰 경우는 불가능 하므로 flag변수
false바꾸고 for문을 나갑니다.
			if (stack.top() > sequence.front())
			{
				flag = false;
				break;
			}
			else if (i + 1 == sequence.front())
			{
				do
				{
```

```
                                        stack.pop();
                                        sequence.pop();
                                        answer.push_back('-');
                                        if (sequence.empty()) break;
                                        if (stack.empty())
                                        {
                                                break;
                                        }
                                        if (sequence.front() < stack.top())
                                        {
                                                flag = false;
                                        }
                                } while (sequence.front() == stack.top());
                        }
                }
                if (!stack.empty() || !sequence.empty() || flag == false)
                {
                                cout << "NO" << "\n";
                                flag = false;
                                break;
                }
        } while (!sequence.empty() && !stack.empty() && flag != false);

        if (flag == true)
        {
                for (auto it = answer.begin(); it != answer.end(); it++)
                        cout << *it << "\n";
        }

        return 0;
}
```

2번은 배열을 이용해서 구현한 것 입니다.

1) 4가지 조건을 기준으로 작성했습니다.
2) 먼저 st(1~N까지 저장하는 스택)이 비어 있으면 수를 넣습니다.
3) 다음으로 st의 top 값과 수열의 값이 같으면 pop합니다.
4) St의 top 값이 수열의 값보다 크면 NO를 출력하고 프로그램을 종료합니다.
5) 아니면 st의 값에 수를 넣습니다.

위를 코드로 구현하면

```
// 시발 printf 짱짱맨...
// 주의 !이 문제는 출력이 많아 endl로 출력하면 시간초과가 걸립니다.대신 \n 로 개행해 주시면 해결됩니다.
// (endl은 개행 후 버퍼 제거까지 하기 때문에 시간이 더 걸립니다.)
#include <iostream>
#include <stack>
```

```cpp
#include <vector>
using namespace std;

int main()
{
	int n, value;
	stack<int> st;
	vector<int> seq;
	vector<char> answer;
	bool flag = true;
	int it = 1;

	cin >> n;

	for (int i = 0; i < n; i++)
	{
		cin >> value;
		seq.push_back(value);
	}

	for (int i = 0; i < n; i++)
	{
		while(1)
		{
			if (st.empty())
			{
				st.push(it);
				answer.push_back('+');
				it++;
			}
			else if (st.top() == seq[i])
			{
				st.pop();
				answer.push_back('-');
				break;
			}
			else if (st.top() > seq[i])
			{
				cout << "NO" << endl;
				return 0;
			}
			else
			{
				st.push(it);
				answer.push_back('+');
				it++;
			}
		}
	}
	if (flag == true)
	{
		for (auto it = answer.begin(); it != answer.end(); it++)
			printf("%c\n", *it);
	}

	return 0;
}
```

알고 스팟 – 조세푸스 문제

원형 연결 리스트를 사용해서 풀어야 하는 문제라고 생각합니다.
k 만큼 node를 다음 값으로 옮긴 후 삭제하고 남아있는 리스트의 데이터 값을 출력합니다.

```
#include <iostream>
using namespace std;

typedef struct Node
{
	int data;
	struct Node* link;
}Node;

// 연결 리스트의 시작점을 저장
Node* head = NULL;

void insert_Node(Node* pNode, int data)
{
	Node* new_node = (Node*)malloc(sizeof(Node));
	Node* restore_Node = pNode;

	// 처음 삽입할 때
	if (head == NULL)
	{
		head = new_node;
		new_node->data = data;
		new_node->link = new_node;
		return;
	}

	while(restore_Node->link != pNode)
	{
		restore_Node = restore_Node->link;
	}

	restore_Node->link = new_node;
	new_node->link = pNode;
	new_node->data = data;
}

void delete_Node(Node* pNode, int data)
{
	Node* tmp_node = pNode;
	Node* restore_node = NULL;

	while (tmp_node->link != pNode)
	{
		if (tmp_node->data == data && tmp_node != pNode)
		{
			restore_node->link = tmp_node->link;
			return;
		}

		restore_node = tmp_node;
```

```
                tmp_node = tmp_node->link;

        }

        if (tmp_node->data == data && tmp_node->link == pNode)
        {
                restore_node->link = tmp_node->link;
                return;
        }

        restore_node = tmp_node;
        tmp_node = tmp_node->link;

        if (tmp_node->data == data)
        {
                restore_node->link = tmp_node->link;
                head = tmp_node->link;
                return;
        }
}

void print(Node* pNode)
{
        Node* tmp_Node = pNode;

        while (tmp_Node->link != pNode)
        {
                cout << tmp_Node->data << " ";
                tmp_Node = tmp_Node->link;
        }
        cout << tmp_Node->data << endl;
}

int main()
{
        // n : 사람 수 , k : 번째 뒤가 자살
        int testCase;
        int n, k;
        int cnt = 0;
        int pos = 0;
        bool flag[1005] = { false };

        cin >> testCase;

        for (int T = 0; T < testCase; T++)
        {
                cin >> n >> k;
                cnt = n;
                head = NULL;

                for (int i = 1; i <= n; i++)
                {
                        insert_Node(head, i);
                }

                Node* tmp_node = head;

                for (int i = 1; i <= n - 2; i++)
                {
                        if (i != 1)
```

```
                        {
                                for (int j = 1; j <= k; j++)
                                {
                                        tmp_node = tmp_node->link;
                                }
                        }
                        delete_Node(head, tmp_node->data);
                }
                print(head);
        }
        return 0;
}
```