

2022-1학기 스터디 #6

나정휘

<https://justicehui.github.io/>

목차

- 스택
- 스택 예시 (3문제)
- 큐
- 큐 예시 (2문제)

주의

- 문제 풀이에 사용할 수 있을 정도로만 간단하게 다룸
 - 직접 구현하는 방법은 다루지 않음
 - C++ 표준 라이브러리에서 제공하는 거 사용하면 됨
- 자세한 내용이 궁금하면 2학년 1학기 자료구조 과목에서...
 - 화요일 12시 ~ 3시 사이에 001 와서 듣는 방법도 있음
 - 화요일에 3시간 공강이라서 오프라인으로 알려드릴 수 있어요
 - 전날(월요일)까지 슬랙 DM으로 미리 연락주세요

스택

스택

- 스택
 - 가장 마지막에 넣은 원소가 가장 먼저 빠져나오는 구조
 - push x: x를 스택에 넣음
 - pop : 가장 마지막에 넣은 원소를 스택에서 제거함
- 프링글스 통
 - 통의 맨 위에만 과자를 넣을 수 있음
 - 맨 위에 있는 과자만 꺼낼 수 있음

스택

- `std::stack`
 - C++ 표준 라이브러리에 이미 스택이 구현되어 있음
 - 헤더: `<stack>`
 - 클래스: `stack<Type>`
 - `int`형 자료를 관리하는 스택: `std::stack<int>`
 - 두 번째 파라미터도 있는데 지금은 신경 쓸 필요 없음
 - 멤버 함수
 - `push(value)` : `value`를 스택에 넣음
 - `pop()` : 스택에서 원소를 하나 제거
 - `top()` : 가장 최근에 넣은 원소를 반환
 - `empty()` : 스택이 비어 있으면 `true` 반환
 - `size()` : 원소의 개수 반환

스택

- std::stack
 - 예시



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

int main(){
    stack<int> stk;
    for(int i=1; i<=5; i++) stk.push(i);
    cout << stk.top() << "\n";
    stk.pop();
    cout << stk.top() << "\n";
    cout << stk.size() << " " << stk.empty() << "\n";
    while(!stk.empty()) stk.pop();
    cout << stk.empty() << "\n";
}
```

질문?

스택 예시 - 1

- BOJ 9012 괄호
 - 괄호 문자열이 주어지면 올바른 괄호 문자열인지 판별
 - ')' 는 이전에 매칭되지 않은 가장 가까운 '(' 와 매칭되어야 함
 - ((())())
 - '(' 가 등장할 때마다 스택에 넣고
 - ')' 가 등장하면 스택에서 하나 제거
 - 모든 문자를 다 처리했을 때 스택이 비어 있어야 함

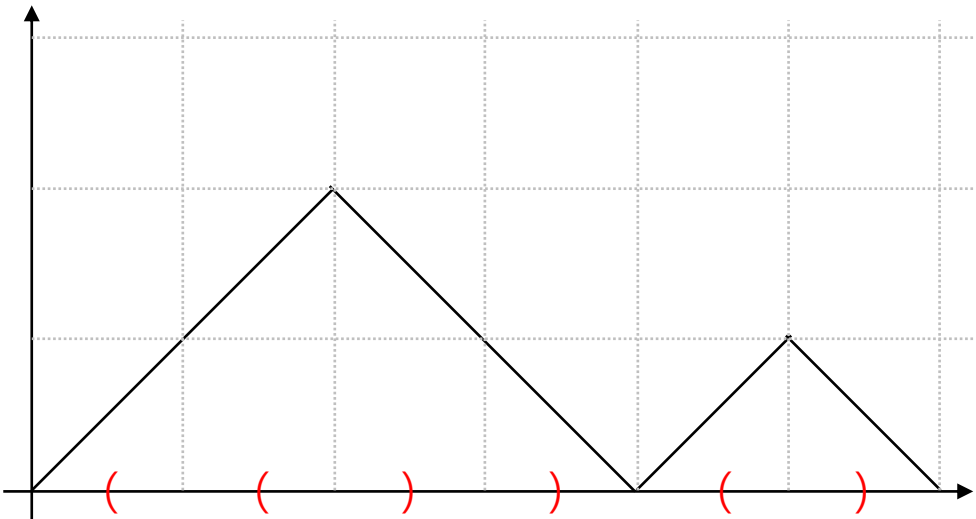
```
#include <bits/stdc++.h>
using namespace std;

void Solve(){
    string str; cin >> str;
    stack<char> stk;
    for(auto i : str){
        if(i == '(') stk.push(i);
        else{
            if(!stk.empty()) stk.pop();
            else{ cout << "NO\n"; return; }
        }
    }
    if(stk.empty()) cout << "YES\n";
    else cout << "NO\n";
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int TC; cin >> TC;
    for(int tc=1; tc<=TC; tc++) Solve();
}
```

스택 예시 - 1

- BOJ 9012 괄호
 - 스택의 크기만 중요하기 때문에 스택을 명시적으로 관리하지 않아도 됨
 - '(' 를 +1, ')' 를 -1이라고 생각하면, 누적 합이 모두 0 이상이고 전체 합이 0이면 됨



```
#include <bits/stdc++.h>
using namespace std;

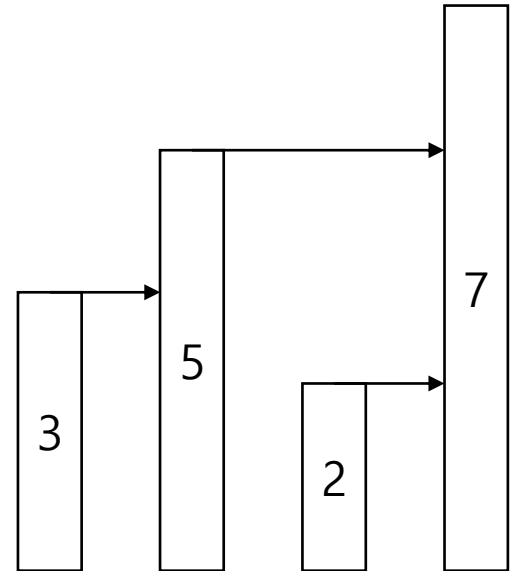
void Solve(){
    string str; cin >> str;
    int sum = 0, mn = 0;
    for(auto i : str){
        if(i == '(') sum += 1;
        else sum -= 1;
        mn = min(mn, sum);
    }
    if(sum == 0 && mn == 0) cout << "YES\n";
    else cout << "NO\n";
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int TC; cin >> TC;
    for(int tc=1; tc<=TC; tc++) Solve();
}
```

질문?

스택 예시 - 2

- BOJ 17298 오큰수
 - 자신보다 오른쪽에 있으면서 자신보다 큰 수 중 가장 왼쪽에 있는 수를 구하는 문제
 - 3 5 2 7 \rightarrow 5 7 7 -1
 - 막대 그래프로 그렸을 때, 각 막대의 시야를 차단하는 막대를 구하는 문제



스택 예시 - 2

- BOJ 17290 오큰수
 - 스택을 사용한 풀이
 - 가장 오른쪽에 있는 원소부터 차례대로 보면서
 - 지금까지 본 원소 중 $A[i]$ 보다 큰 가장 왼쪽 원소를 구하면 됨
 - $i < j$ 이고 $A[i] \geq A[j]$ 이면 $A[j]$ 는 더 이상 오큰수가 될 수 없음
 - $A[i+1..N]$ 중 $A[1..i]$ 의 오큰수가 될 수 있는 후보는 오름차순
 - $A[i]$ 의 오큰수는 후보 중 $A[i]$ 보다 큰 첫 번째 원소
 - $A[i]$ 를 후보 집합에 넣을 때, $A[i]$ 이하인 원소를 모두 제거해야 함
 - $A[i]$ 의 오큰수를 구하는 방법
 - 스택에서 $A[i]$ 보다 작거나 같은 수를 모두 제거한 뒤
 - 스택의 맨 위에 있는 원소가 오큰수
 - 오큰수를 구한 뒤 $A[i]$ 를 스택에 삽입

스택 예시 - 2

- BOJ 17290 오큰수
 - 시간 복잡도: $O(N)$
 - 각 원소는 최대 한 번 스택에 들어가고
 - 최대 한 번 스택에서 빠져나옴
- 이런 식으로 스택의 내용물의 단조성을 유지하는 것을 **monotone stack** 기법이라고 부름

```
#include <bits/stdc++.h>
using namespace std;

int N, A[1010101], B[1010101];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> A[i];

    stack<int> S;
    for(int i=N; i>=1; i--){
        while(!S.empty() && S.top() <= A[i]) S.pop();
        B[i] = S.empty() ? -1 : S.top();
        S.push(A[i]);
    }
    for(int i=1; i<=N; i++) cout << B[i] << " ";
}
```

스택 예시 - 3

- BOJ 1725 히스토그램
 - 히스토그램에서 가장 큰 직사각형의 넓이를 구하는 문제
 - 분할 정복에서 했던 그 문제
 - 스택을 사용하면 $O(N)$ 에 해결할 수 있음
 - $L[i]$: $A[i]$ 의 왼쪽에 있으면서 $A[i]$ 보다 작으면서 가장 오른쪽에 있는 원소의 위치
 - $R[i]$: $A[i]$ 의 오른쪽에 있으면서 $A[i]$ 보다 작으면서 가장 왼쪽에 있는 원소의 위치
 - $A[i]$ 를 포함하면서 높이가 $A[i]$ 인 직사각형의 최대 너비 = $R[i] - L[i] - 1$
 - $res = \max(res, A[i] * (R[i] - L[i] - 1));$

질문?

큐

큐

- 큐
 - 가장 먼저 넣은 원소가 가장 먼저 빠져나오는 구조
 - enqueue x : x를 큐에 넣음
 - dequeue : 가장 먼저 넣은 원소를 큐에서 제거함
 - 대기열
 - 새로 들어온 사람은 줄의 맨 뒤로 감
 - 줄의 맨 앞에 있는 사람부터 처리

큐

- `std::queue`
 - C++ 표준 라이브러리에 이미 큐가 구현되어 있음
 - 헤더: `<queue>`
 - 클래스: `queue<Type>`
 - int형 자료를 관리하는 큐: `std::queue<int>`
 - 두 번째 파라미터도 있는데 지금은 신경 쓸 필요 없음
 - 멤버 함수
 - `push(value)` : value를 큐에 넣음
 - `pop()` : 큐에서 원소를 하나 제거
 - `front()` : 가장 처음에 넣은 원소를 반환
 - `empty()` : 큐가 비어 있으면 true 반환
 - `size()` : 원소의 개수 반환

큐

- std::queue
 - 예시



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

int main(){
    queue<int> que;
    for(int i=1; i<=5; i++) que.push(i);
    cout << que.front() << "\n";
    que.pop();
    cout << que.front() << "\n";
    cout << que.size() << " " << que.empty() << "\n";
    while(!que.empty()) que.pop();
    cout << que.empty();
}
```

큐

- `std::deque`
 - double ended queue: 삽입/삭제가 앞/뒤에서 모두 가능한 자료구조
 - 헤더: `<deque>`
 - 클래스: `deque<Type>`
 - int형 자료를 관리하는 큐: `std::deque<int>`
 - 두 번째 파라미터도 있는데 지금을 신경 쓸 필요 없음
 - 멤버 함수
 - `push_front(value), push_back(value)`
 - `pop_front(), pop_back()`
 - `front(), back()`
 - `empty(), size()`

질문?

큐 예시 - 1

- BOJ 2003 수들의 합 2
 - $A[i] + A[i+1] + \dots + A[j] = M$ 이 되는 구간 $[i, j]$ 의 개수를 구하는 문제
 - $A[1..N]$ 은 양의 정수
 - 각각의 j 마다 $A[i] + \dots + A[j] = M$ 인 i 를 찾는 방식
 - j 가 증가할 때마다 조건을 만족하는 i 도 증가함
 - $A[j]$ 를 맨 뒤에 넣고 $A[i..]$ 를 앞에서 빼는 방식으로 큐를 이용해 구간을 관리할 수 있음
 - 먼저 $A[j]$ 를 큐의 맨 뒤에 넣고
 - 큐에 있는 원소의 합이 M 보다 크면 앞에 있는 원소를 제거
 - 큐에 있는 원소의 합이 정확히 M 이라면 정답 += 1

큐 예시 - 1

- BOJ 2003 수들의 합 2
 - 시간 복잡도: $O(N)$
 - 각 원소는 최대 한 번 큐에 들어가고
 - 최대 한 번 큐에서 빠져나옴
 - 큐에 있는 원소의 합은 따로 관리

```
#include <bits/stdc++.h>
using namespace std;

int N, M, A[10101];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=N; i++) cin >> A[i];

    int sum = 0, cnt = 0;
    queue<int> que;
    for(int i=1; i<=N; i++){
        sum += A[i];
        que.push(A[i]);
        while(!que.empty() && sum > M){
            sum -= que.front();
            que.pop();
        }
        if(sum == M) cnt++;
    }
    cout << cnt;
}
```


큐의 예시 - 1

- BOJ 2003 수들의 합 2
 - 구간의 끝점만 알고 있으면 되므로 큐를 명시적으로 관리하지 않아도 됨
 - 단조 증가하는 구간의 끝점을 관리하는 기법을 **투 포인터** 또는 **슬라이딩 윈도우**라고 부름

```
#include <bits/stdc++.h>
using namespace std;

int N, M, A[10101];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=N; i++) cin >> A[i];

    int cnt = 0;
    for(int i=1, j=1, sum=0; j<=N; j++){
        sum += A[j];
        while(i <= j && sum > M) sum -= A[i++];
        if(sum == M) cnt++;
    }
    cout << cnt;
}
```

```
#include <bits/stdc++.h>
using namespace std;

int N, M, A[10101];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=N; i++) cin >> A[i];

    int l = 1, r = 1, sum = A[1], cnt = 0;
    while(true){
        if(sum == M) cnt++;
        if(l == N) break;
        if(r == N) sum -= A[l++];
        else if(l == r) sum += A[++r];
        else if(sum < M) sum += A[++r];
        else sum -= A[l++];
    }
    cout << cnt;
}
```

질문?

큐의 예시 - 2

- BOJ 11003 최솟값 찾기
 - 고정된 L 에 대해, 모든 i 에 대해 $A[i-L+1..i]$ 의 최솟값을 구하는 문제
 - 오큰수 문제에서 했던 것처럼, 최솟값이 될 수 있는 원소를 관리
 - 인덱스가 $i-L+1$ 보다 작은 원소는 더 이상 필요 없음
 - $x < y$ 이면서 $A[x] > A[y]$ 이면 x 는 더 이상 최솟값이 될 수 없음
 - $x < y$ 이면서 $A[x] = A[y]$ 이면 y 가 더 오래 살아있으므로 x 는 신경 쓰지 않아도 됨
 - 오큰수 문제에서 인덱스가 $i-L+1$ 미만인 원소를 제거하는 것만 추가됨
 - +) $A[i]$ 를 넣을 때 $A[i]$ 이상인 원소 모두 제거

큐의 예시 - 2

- BOJ 11003 최솟값 찾기
 - $A[i-L+1..i]$ 의 최솟값을 구하는 과정
 - 인덱스가 $i-L+1$ 보다 작은 후보를 모두 제거
 - $A[i]$ 보다 크거나 같은 후보 모두 제거
 - $A[i]$ 를 후보에 삽입
 - deque로 관리하면 위 3가지 작업은 `pop_front`, `pop_back`, `push_back`으로 할 수 있음
 - deque의 내용물은 증가하는 상태
 - deque의 맨 앞에 있는 원소가 최솟값

큐의 예시 - 2

- BOJ 11003 최솟값 찾기
 - 시간 복잡도: $O(N)$
 - monotone stack과 비슷하게 **monotone queue**라고 부름

```

#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int N, L; cin >> N >> L;

    deque<pair<int,int>> dq;
    for(int i=1; i<=N; i++){
        int t; cin >> t;
        while(!dq.empty() && dq.front().second < i-L+1) dq.pop_front();
        while(!dq.empty() && dq.back().first >= t) dq.pop_back();
        dq.push_back(make_pair(t, i));
        cout << dq.front().first << " ";
    }
}
```

질문?

과제

- 필수

- 10808 스택 / 10845 큐 / 10866 덱
- 9012 괄호
- 2003 수들의 합 2
- 17298 오큰수
- 11003 최솟값 찾기

- 심화

- 10799 쇠막대기
- 2800 괄호 제거
- 1725 히스토그램
- 15678 연세워터파크