21/1/19

1. 괄호 변환

**문제 설명**

카카오에 신입 개발자로 입사한 **콘**은 선배 개발자로부터 개발역량 강화를 위해 다른 개발자가 작성한 소스 코드를 분석하여 문제점을 발견하고 수정하라는 업무 과제를 받았습니다. 소스를 컴파일하여 로그를 보니 대부분 소스 코드 내 작성된 괄호가 개수는 맞지만 짝이 맞지 않은 형태로 작성되어 오류가 나는 것을 알게 되었습니다.  
수정해야 할 소스 파일이 너무 많아서 고민하던 콘은 소스 코드에 작성된 모든 괄호를 뽑아서 올바른 순서대로 배치된 괄호 문자열을 알려주는 프로그램을 다음과 같이 개발하려고 합니다.

**용어의 정의**

**'('** 와 **')'** 로만 이루어진 문자열이 있을 경우, '(' 의 개수와 ')' 의 개수가 같다면 이를 **균형잡힌 괄호 문자열**이라고 부릅니다.  
그리고 여기에 '('와 ')'의 괄호의 짝도 모두 맞을 경우에는 이를 **올바른 괄호 문자열**이라고 부릅니다.  
예를 들어, "(()))("와 같은 문자열은 균형잡힌 괄호 문자열 이지만 올바른 괄호 문자열은 아닙니다.  
반면에 "(())()"와 같은 문자열은 균형잡힌 괄호 문자열 이면서 동시에 올바른 괄호 문자열 입니다.

'(' 와 ')' 로만 이루어진 문자열 w가 균형잡힌 괄호 문자열 이라면 다음과 같은 과정을 통해 올바른 괄호 문자열로 변환할 수 있습니다.

1. 입력이 빈 문자열인 경우, 빈 문자열을 반환합니다.

2. 문자열 w를 두 "균형잡힌 괄호 문자열" u, v로 분리합니다. 단, u는 "균형잡힌 괄호 문자열"로 더 이상 분리할 수 없어야 하며, v는 빈 문자열이 될 수 있습니다.

3. 문자열 u가 "올바른 괄호 문자열" 이라면 문자열 v에 대해 1단계부터 다시 수행합니다.

3-1. 수행한 결과 문자열을 u에 이어 붙인 후 반환합니다.

4. 문자열 u가 "올바른 괄호 문자열"이 아니라면 아래 과정을 수행합니다.

4-1. 빈 문자열에 첫 번째 문자로 '('를 붙입니다.

4-2. 문자열 v에 대해 1단계부터 재귀적으로 수행한 결과 문자열을 이어 붙입니다.

4-3. ')'를 다시 붙입니다.

4-4. u의 첫 번째와 마지막 문자를 제거하고, 나머지 문자열의 괄호 방향을 뒤집어서 뒤에 붙입니다.

4-5. 생성된 문자열을 반환합니다.

**균형잡힌 괄호 문자열** p가 매개변수로 주어질 때, 주어진 알고리즘을 수행해 **올바른 괄호 문자열**로 변환한 결과를 return 하도록 solution 함수를 완성해 주세요.

**매개변수 설명**

* p는 '(' 와 ')' 로만 이루어진 문자열이며 길이는 2 이상 1,000 이하인 짝수입니다.
* 문자열 p를 이루는 '(' 와 ')' 의 개수는 항상 같습니다.
* 만약 p가 이미 올바른 괄호 문자열이라면 그대로 return 하면 됩니다.

**입출력 예**

| **p** | **result** |
| --- | --- |
| "(()())()" | "(()())()" |
| ")(" | "()" |
| "()))((()" | "()(())()" |

* 내 답

#include <string>

#include <vector>

#include <algorithm>

#include <iostream>

#include <utility>

#include <stack>

using namespace std;

void change(char \* c) {

if(\*c=='(')

\*c = ')';

else if(\*c==')')

\*c = '(';

}

bool is\_match(char c, char t) {

if(c=='(' && t==')')

return true;

else

return false;

}

bool is\_correct(string s) {

stack<char> st;

char temp = 'a';

if(s[0]!='(')

return false;

st.push(s[0]);

for(int i=1;i<s.length();i++) {

temp = 'a';

if(!st.empty())

temp = st.top();

if(is\_match(temp,s[i])) {

st.pop();

continue;

}

st.push(s[i]);

}

if(st.empty())

return true;

else

return false;

}

string solution(string p) {

string answer = "";

int l\_num=0;

int r\_num=0;

if(p.length()==0)

return "";

string u,v;

for(int i=0;i<p.length();i++) {

if(l\_num==r\_num && l\_num!=0) {

v = p.substr(i,p.length()-u.length());

break;

}

u+=p[i];

if(p[i] == '(')

l\_num++;

else if(p[i] == ')')

r\_num++;

}

if(is\_correct(u))

return u+=solution(v);

else {

string temp = "(";

temp+=solution(v);

temp+=")";

for(int i=1;i<u.length()-1;i++)

change(&u[i]);

temp+=u.substr(1,u.length()-2);

return temp;

}

return answer;

}

조건 연산자를 활용하고, 스택을 사용하지 않아도 체크가 가능했다.

* 다른 사람의 답

#include <bits/stdc++.h>

using namespace std;

bool check(const string &a) {

int r = 0;

for (char ch : a) {

if (ch == '(') ++r;

else --r;

if (r < 0) return false;

}

return r == 0;

}

string solution(string p) {

if (p == "") return "";

if (check(p)) return p;

int i, t = 0;

for (i = 0; i < p.size(); ++i) {

if (p[i] == '(') ++t;

else --t;

if (t == 0) break;

}

string u = p.substr(0, i + 1);

string v = p.substr(i + 1);

if (check(u)) return u + solution(v);

for (char &ch : u) ch = ch == '(' ? ')' : '(';

return string("(") + solution(v) + ")" + u.substr(1, u.size() - 2);

}

1. 더 맵게

###### 문제 설명

매운 것을 좋아하는 Leo는 모든 음식의 스코빌 지수를 K 이상으로 만들고 싶습니다. 모든 음식의 스코빌 지수를 K 이상으로 만들기 위해 Leo는 스코빌 지수가 가장 낮은 두 개의 음식을 아래와 같이 특별한 방법으로 섞어 새로운 음식을 만듭니다.

섞은 음식의 스코빌 지수 = 가장 맵지 않은 음식의 스코빌 지수 + (두 번째로 맵지 않은 음식의 스코빌 지수 \* 2)

Leo는 모든 음식의 스코빌 지수가 K 이상이 될 때까지 반복하여 섞습니다.  
Leo가 가진 음식의 스코빌 지수를 담은 배열 scoville과 원하는 스코빌 지수 K가 주어질 때, 모든 음식의 스코빌 지수를 K 이상으로 만들기 위해 섞어야 하는 최소 횟수를 return 하도록 solution 함수를 작성해주세요.

##### **제한 사항**

* scoville의 길이는 2 이상 1,000,000 이하입니다.
* K는 0 이상 1,000,000,000 이하입니다.
* scoville의 원소는 각각 0 이상 1,000,000 이하입니다.
* 모든 음식의 스코빌 지수를 K 이상으로 만들 수 없는 경우에는 -1을 return 합니다.

##### **입출력 예**

| **scoville** | **K** | **return** |
| --- | --- | --- |
| [1, 2, 3, 9, 10, 12] | 7 | 2 |

* 내 답

#include <string>

#include <vector>

#include <queue>

using namespace std;

int solution(vector<int> scoville, int K) {

int answer = 0;

priority\_queue<int,vector<int>,greater<int>> q;

int a,b;

for(int i=0;i<scoville.size();i++)

q.push(scoville[i]);

while(q.top()<K) {

if(q.size()==1)

return -1;

a = q.top();

q.pop();

b = q.top();

q.pop();

q.push(a+2\*b);

answer++;

}

return answer;

}

다른 사람의 답도 비슷하다. 사실 힙이 아니라 문자열을 그때그때 정렬하여 문제를 푸려했으나, 우선순위 큐를 이용해서 푸는게 훨씬 간단하고 효율적이었다. 우선순위 큐는 <queue>헤더파일에 내장되어 있다. 기본적으로는 최대 원소가 q.top()에 배치된다.

최소 원소가 대상이 되게 하려면

priority\_queue<int,vector<int>,greater<int>> q; 와 같이 선언해주어야 한다.

1. H-Index

###### 문제 설명

H-Index는 과학자의 생산성과 영향력을 나타내는 지표입니다. 어느 과학자의 H-Index를 나타내는 값인 h를 구하려고 합니다. 위키백과[1](https://programmers.co.kr/learn/courses/30/lessons/42747#fn1)에 따르면, H-Index는 다음과 같이 구합니다.

어떤 과학자가 발표한 논문 n편 중, h번 이상 인용된 논문이 h편 이상이고 나머지 논문이 h번 이하 인용되었다면 h의 최댓값이 이 과학자의 H-Index입니다.

어떤 과학자가 발표한 논문의 인용 횟수를 담은 배열 citations가 매개변수로 주어질 때, 이 과학자의 H-Index를 return 하도록 solution 함수를 작성해주세요.

##### **제한사항**

* 과학자가 발표한 논문의 수는 1편 이상 1,000편 이하입니다.
* 논문별 인용 횟수는 0회 이상 10,000회 이하입니다.

##### **입출력 예**

| **citations** | **return** |
| --- | --- |
| [3, 0, 6, 1, 5] | 3 |

##### **입출력 예 설명**

이 과학자가 발표한 논문의 수는 5편이고, 그중 3편의 논문은 3회 이상 인용되었습니다. 그리고 나머지 2편의 논문은 3회 이하 인용되었기 때문에 이 과학자의 H-Index는 3입니다.

※ 공지 - 2019년 2월 28일 테스트 케이스가 추가되었습니다.

1. <https://en.wikipedia.org/wiki/H-index> 위키백과 [↩](https://programmers.co.kr/learn/courses/30/lessons/42747#fnref1)

* 내 답

#include <string>

#include <algorithm>

#include <vector>

#include <iostream>

using namespace std;

int solution(vector<int> citations) {

int answer = 0;

int a,b;

int all\_zero = 0;

vector<int> ordered\_c;

ordered\_c.assign(citations.begin(),citations.end());

sort(ordered\_c.begin(),ordered\_c.end(),greater<int>());

for(int i=0;i<ordered\_c.size();i++)

if(ordered\_c[i])

all\_zero = 1;

if(!all\_zero)

return 0;

for(int i=0;i<ordered\_c.size();i++)

if(i+1>=ordered\_c[i])

if(ordered\_c[i]==0)

a=0;

else {

a = ordered\_c[i];

break;

}

else

b = i+1;

answer = max(a,b);

return answer;

}

* 문제 지문이 애매해서 좀 해매다가, 다른 사람의 힌트를 보고 풀게 되었다. 또한 마지막 테스트 케이스가 안맞았는데, 내 코드에서 돌릴때 전부 0인 케이스가 0이 나왔지만 웹에서는 안그래서, all\_zero 변수를 추가해줬는데 왜 이게 다르게 나오는지 모르겠다. 문제를 정확히 잘 파악해야 할 것 같다. 하지만 문제도 불분명하게 나왔다. 참고링크를 잘 보고 풀었어야 했던 것 같다.
* 남의 답

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int solution(vector<int> citations) {

sort(citations.begin(), citations.end(), greater<int>());

for (int i = 0; i < citations.size(); ++i) {

if (citations[i] < i + 1) {

return i;

}

}

return citations.size();

}

그냥 이렇게 생각해도 되었을 것을…

더 간단하게 표현하는 것을 연습해야 할 것 같다. 무작정 답을 내기보다는 간단하게 표현하는 것도 생각해보자.

1. 전화번호 목록

###### 문제 설명

전화번호부에 적힌 전화번호 중, 한 번호가 다른 번호의 접두어인 경우가 있는지 확인하려 합니다.  
전화번호가 다음과 같을 경우, 구조대 전화번호는 영석이의 전화번호의 접두사입니다.

* 구조대 : 119
* 박준영 : 97 674 223
* 지영석 : 11 9552 4421

전화번호부에 적힌 전화번호를 담은 배열 phone\_book 이 solution 함수의 매개변수로 주어질 때, 어떤 번호가 다른 번호의 접두어인 경우가 있으면 false를 그렇지 않으면 true를 return 하도록 solution 함수를 작성해주세요.

##### **제한 사항**

* phone\_book의 길이는 1 이상 1,000,000 이하입니다.
* 각 전화번호의 길이는 1 이상 20 이하입니다.

##### **입출력 예제**

| **phone\_book** | **return** |
| --- | --- |
| [119, 97674223, 1195524421] | false |
| [123,456,789] | true |
| [12,123,1235,567,88] | false |

##### **입출력 예 설명**

입출력 예 #1  
앞에서 설명한 예와 같습니다.

입출력 예 #2  
한 번호가 다른 번호의 접두사인 경우가 없으므로, 답은 true입니다.

입출력 예 #3  
첫 번째 전화번호, “12”가 두 번째 전화번호 “123”의 접두사입니다. 따라서 답은 false입니다.

* 내 답

#include <string>

#include <vector>

using namespace std;

bool solution(vector<string> phone\_book) {

bool answer = true;

for(int i=0;i<phone\_book.size();i++) {

for(int j=0;j<phone\_book.size();j++) {

if(j==i)

continue;

if(phone\_book[j].find(phone\_book[i])==0)

return false;

}

}

return answer;

}

* 남의 답

|  |  |
| --- | --- |
| 2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include <string>  #include <vector>  #include <algorithm>  #include <iostream>  using namespace std;  bool solution(vector<string> phoneBook) {  bool answer = true;  sort(phoneBook.begin(), phoneBook.end());  for ( int i = 0 ; i < phoneBook.size() - 1 ; i++ )  {  if ( phoneBook[i] == phoneBook[i+1].substr(0, phoneBook[i].size()) )  {  answer = false;  break;  }  }  return answer;  } |

이게 조금 더 깔끔하고 시간복잡도가 적다. 먼저 정렬을 해주면 접두사를 포함하는 문자열은 반드시 인접해있기 때문에 바로 다음 항만 비교해주면 된다.

* 남의 답 2

#include <string>

#include <vector>

#include <unordered\_map>

using namespace std;

bool solution(vector<string> phone\_book) {

bool answer = true;

unordered\_map<string, int> hash\_map;

for(int i = 0; i < phone\_book.size(); i++)

hash\_map[phone\_book[i]] = 1;

for(int i = 0; i < phone\_book.size(); i++) {

string phone\_number = "";

for(int j = 0; j < phone\_book[i].size(); j++) {

phone\_number += phone\_book[i][j];

if(hash\_map[phone\_number] && phone\_number != phone\_book[i])

answer = false;

}

}

return answer;

}

unordered\_map 은 해시기반으로, 원소에 접근하는데 O(1) 시간이 걸린다. 따라서 다음과 같이 풀어줄 수도 있다. string.find()보다 빠를 수 있음.

1. 구명보트

###### 문제 설명

무인도에 갇힌 사람들을 구명보트를 이용하여 구출하려고 합니다. 구명보트는 작아서 한 번에 최대 **2명**씩 밖에 탈 수 없고, 무게 제한도 있습니다.

예를 들어, 사람들의 몸무게가 [70kg, 50kg, 80kg, 50kg]이고 구명보트의 무게 제한이 100kg이라면 2번째 사람과 4번째 사람은 같이 탈 수 있지만 1번째 사람과 3번째 사람의 무게의 합은 150kg이므로 구명보트의 무게 제한을 초과하여 같이 탈 수 없습니다.

구명보트를 최대한 적게 사용하여 모든 사람을 구출하려고 합니다.

사람들의 몸무게를 담은 배열 people과 구명보트의 무게 제한 limit가 매개변수로 주어질 때, 모든 사람을 구출하기 위해 필요한 구명보트 개수의 최솟값을 return 하도록 solution 함수를 작성해주세요.

##### **제한사항**

* 무인도에 갇힌 사람은 1명 이상 50,000명 이하입니다.
* 각 사람의 몸무게는 40kg 이상 240kg 이하입니다.
* 구명보트의 무게 제한은 40kg 이상 240kg 이하입니다.
* 구명보트의 무게 제한은 항상 사람들의 몸무게 중 최댓값보다 크게 주어지므로 사람들을 구출할 수 없는 경우는 없습니다.

##### **입출력 예**

| **people** | **limit** | **return** |
| --- | --- | --- |
| [70, 50, 80, 50] | 100 | 3 |
| [70, 80, 50] | 100 | 3 |

* 내 답

#include <string>

#include <vector>

#include <algorithm>

#include <iostream>

#include <deque>

using namespace std;

int solution(vector<int> people, int limit) {

int answer = 0;

deque<int> q;

q.assign(people.begin(),people.end());

sort(q.begin(),q.end(),greater<int>());

while(!q.empty()) {

if(q[0]+q[q.size()-1]<=limit) {

answer++;

if(q.size()!=1)

q.pop\_back();

q.pop\_front();

}

else {

answer++;

q.pop\_front();

}

}

return answer;

}

별 다르게 어려울 것은 없었다. 예외를 잘 생각하자!