21/1/6

1. 124 나라의 숫자

**문제 설명**

124 나라가 있습니다. 124 나라에서는 10진법이 아닌 다음과 같은 자신들만의 규칙으로 수를 표현합니다.

1. 124 나라에는 자연수만 존재합니다.
2. 124 나라에는 모든 수를 표현할 때 1, 2, 4만 사용합니다.

예를 들어서 124 나라에서 사용하는 숫자는 다음과 같이 변환됩니다.

| **10진법** | **124 나라** | **10진법** | **124 나라** |
| --- | --- | --- | --- |
| 1 | 1 | 6 | 14 |
| 2 | 2 | 7 | 21 |
| 3 | 4 | 8 | 22 |
| 4 | 11 | 9 | 24 |
| 5 | 12 | 10 | 41 |

자연수 n이 매개변수로 주어질 때, n을 124 나라에서 사용하는 숫자로 바꾼 값을 return 하도록 solution 함수를 완성해 주세요.

제한사항

* n은 500,000,000이하의 자연수 입니다.

입출력 예

| **n** | **result** |
| --- | --- |
| 1 | 1 |
| 2 | 2 |
| 3 | 4 |
| 4 | 11 |

* 내 답

#include <string>

#include <vector>

#include <cmath>

using namespace std;

string solution(int n) {

string answer = "";

int a,b,t;

int l;

int k;

for(int i=1;;i++) {

a = (pow(3,i)-3)/2;

b = (pow(3,i+1)-3)/2;

if(a<n && n<=b) {

t = i;

break;

}

}

k = n-a-1;

for(int i=t-1;i>=0;i--) {

int po = pow(3,i);

l = k/po;

switch(l) {

case 0:

answer += "1";

break;

case 1:

answer += "2";

break;

case 2:

answer += "4";

break;

}

k -= l\*po;

}

return answer;

}

처음에 몇번 오류를 만들다가 n에서 a를 빼주는게아니라 a+1을 빼줘야 한다는 것을 알고 고쳤다.

* 다른 사람 코드

#include<iostream>

#include<vector>

using namespace std;

string change124(int no)

{

string answer = "";

int a;

while(no > 0){

a = no % 3;

no = no / 3;

if (a == 0){

no -= 1;

}

answer = "412"[a] + answer;

}

return answer;

}

int main()

{

int testNo = 6;

string testAnswer = change124(testNo);

cout<<testAnswer;

}

switch를 안쓰고 “412”[a] 식으로 해결한 것이 놀라웠다. 나머지는 3진법의 풀이 비슷하게 한 것 같다. 그리고 나머지를 이용하여 문제를 해결했다. 이 방법이 조금 더 효율적인 것 같다.

2. 다리를 지나는 트럭

**문제 설명**

트럭 여러 대가 강을 가로지르는 일 차선 다리를 정해진 순으로 건너려 합니다. 모든 트럭이 다리를 건너려면 최소 몇 초가 걸리는지 알아내야 합니다. 트럭은 1초에 1만큼 움직이며, 다리 길이는 bridge\_length이고 다리는 무게 weight까지 견딥니다.  
※ 트럭이 다리에 완전히 오르지 않은 경우, 이 트럭의 무게는 고려하지 않습니다.

예를 들어, 길이가 2이고 10kg 무게를 견디는 다리가 있습니다. 무게가 [7, 4, 5, 6]kg인 트럭이 순서대로 최단 시간 안에 다리를 건너려면 다음과 같이 건너야 합니다.

| **경과 시간** | **다리를 지난 트럭** | **다리를 건너는 트럭** | **대기 트럭** |
| --- | --- | --- | --- |
| 0 | [] | [] | [7,4,5,6] |
| 1~2 | [] | [7] | [4,5,6] |
| 3 | [7] | [4] | [5,6] |
| 4 | [7] | [4,5] | [6] |
| 5 | [7,4] | [5] | [6] |
| 6~7 | [7,4,5] | [6] | [] |
| 8 | [7,4,5,6] | [] | [] |

따라서, 모든 트럭이 다리를 지나려면 최소 8초가 걸립니다.

solution 함수의 매개변수로 다리 길이 bridge\_length, 다리가 견딜 수 있는 무게 weight, 트럭별 무게 truck\_weights가 주어집니다. 이때 모든 트럭이 다리를 건너려면 최소 몇 초가 걸리는지 return 하도록 solution 함수를 완성하세요.

제한 조건

* bridge\_length는 1 이상 10,000 이하입니다.
* weight는 1 이상 10,000 이하입니다.
* truck\_weights의 길이는 1 이상 10,000 이하입니다.
* 모든 트럭의 무게는 1 이상 weight 이하입니다.

입출력 예

| **bridge\_length** | **weight** | **truck\_weights** | **return** |
| --- | --- | --- | --- |
| 2 | 10 | [7,4,5,6] | 8 |
| 100 | 100 | [10] | 101 |
| 100 | 100 | [10,10,10,10,10,10,10,10,10,10] | 110 |

* 내 답

#include <string>

#include <vector>

#include <deque>

#include <map>

#include <algorithm>

using namespace std;

int solution(int bridge\_length, int weight, vector<int> truck\_weights) {

int answer = 0;

int bridge\_w = 0;

deque<pair<int,int>> q;

deque<int> t\_q;

//sort(truck\_weights.begin(),truck\_weights.end());

t\_q.assign(truck\_weights.begin(),truck\_weights.end());

while(!t\_q.empty() || !q.empty()) {

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

q[i].second++;

answer++;

if(!q.empty())

if(q[0].second == bridge\_length+1) {

bridge\_w -= q[0].first;

q.pop\_front();

}

if(!t\_q.empty())

if(t\_q[0] + bridge\_w <= weight ) {

pair<int,int> m;

m = make\_pair(t\_q[0],1);

q.push\_back(m);

bridge\_w += t\_q[0];

t\_q.pop\_front();

}

}

return answer;

}

처음에는 배열에 들어가있는 ‘순서대로’ 라는 말을 못보고 배열안에 있는 트럭들을 내가 순서를 조절해서 최소 값을 구하는 것으로 알고, sort를 이용했는데, 아니었다. 문제를 잘 읽어야 할 것 같다. Dequeue 2개를 만들어서 구해주었다.

* pair 클래스 정리

 **pair<[type1], [type2]> p**  
사용할 데이터 타입 1, 2를 넣고 그 타입의 pair 클래스인 p를 만듭니다.

 **p.first**: p의 첫번째 인자를 반환해 줍니다.

 **p.second :**p의 두번째 인자를 반환해 줍니다.

 **make\_pair(변수1, 변수2) :**변수1과 변수2가 들어간 pair를 만들어줍니다.

 operator로 **(==, !=, <, >, <=, >=)가 정의**되어있어서, 사용이 가능합니다.

 **sort 알고리즘에 의해 정렬이 가능합**니다.  
**(대소 비교 및 sort에의한 정렬에서 : 첫번째 인자 기준, 첫번째가 같으면 두번째인자로 판단)**  
  
출처: <https://blockdmask.tistory.com/64> [개발자 지망생]

* 다른 사람의 풀이

#include <string>

#include <vector>

#include <queue>

using namespace std;

int solution(int bridge\_length, int weight, vector<int> truck\_weights) {

int answer = 0;

int tot\_w = 0;

int t\_front = 0;

int t\_cur = 0;

int sec = 0;

queue <int> q;

while (t\_front != truck\_weights.size()) {

if (!q.empty() && (sec - q.front() == bridge\_length)) {

tot\_w -= truck\_weights[t\_front];

++t\_front;

q.pop();

}

if (t\_cur != truck\_weights.size() && tot\_w + truck\_weights[t\_cur] <= weight) {

tot\_w += truck\_weights[t\_cur];

++t\_cur;

q.push(sec);

}

++sec;

}

answer = sec;

return answer;

}

생각해보니, 순서대로라는 것을 인지하고 풀었으면, 코드가 더 간단해졌을 것 같다. 이 사람의 풀이에서도 처음과 끝 트럭만 Tracking 하면서 문제를 해결하였다.

1. 스킬트리

**문제 설명**

선행 스킬이란 어떤 스킬을 배우기 전에 먼저 배워야 하는 스킬을 뜻합니다.

예를 들어 선행 스킬 순서가 스파크 → 라이트닝 볼트 → 썬더일때, 썬더를 배우려면 먼저 라이트닝 볼트를 배워야 하고, 라이트닝 볼트를 배우려면 먼저 스파크를 배워야 합니다.

위 순서에 없는 다른 스킬(힐링 등)은 순서에 상관없이 배울 수 있습니다. 따라서 스파크 → 힐링 → 라이트닝 볼트 → 썬더와 같은 스킬트리는 가능하지만, 썬더 → 스파크나 라이트닝 볼트 → 스파크 → 힐링 → 썬더와 같은 스킬트리는 불가능합니다.

선행 스킬 순서 skill과 유저들이 만든 스킬트리[1](https://programmers.co.kr/learn/courses/30/lessons/49993" \l "fn1)를 담은 배열 skill\_trees가 매개변수로 주어질 때, 가능한 스킬트리 개수를 return 하는 solution 함수를 작성해주세요.

제한 조건

* 스킬은 알파벳 대문자로 표기하며, 모든 문자열은 알파벳 대문자로만 이루어져 있습니다.
* 스킬 순서와 스킬트리는 문자열로 표기합니다.
  + 예를 들어, C → B → D 라면 CBD로 표기합니다
* 선행 스킬 순서 skill의 길이는 1 이상 26 이하이며, 스킬은 중복해 주어지지 않습니다.
* skill\_trees는 길이 1 이상 20 이하인 배열입니다.
* skill\_trees의 원소는 스킬을 나타내는 문자열입니다.
  + skill\_trees의 원소는 길이가 2 이상 26 이하인 문자열이며, 스킬이 중복해 주어지지 않습니다.

입출력 예

| **skill** | **skill\_trees** | **return** |
| --- | --- | --- |
| "CBD" | ["BACDE", "CBADF", "AECB", "BDA"] | 2 |

입출력 예 설명

* BACDE: B 스킬을 배우기 전에 C 스킬을 먼저 배워야 합니다. 불가능한 스킬트립니다.
* CBADF: 가능한 스킬트리입니다.
* AECB: 가능한 스킬트리입니다.
* BDA: B 스킬을 배우기 전에 C 스킬을 먼저 배워야 합니다. 불가능한 스킬트리입니다.
* 내 답

include <string>

#include <vector>

#include <iostream>

using namespace std;

int solution(string skill, vector<string> skill\_trees) {

int answer = 0;

for(int i=0;i<skill\_trees.size();i++) {

string sub;

for(int j=0;j<skill\_trees[i].size();j++)

if(skill.find(skill\_trees[i][j]) != string::npos)

sub += skill\_trees[i][j];

if(skill.find(sub) == 0)

answer++;

}

return answer;

}

string 클래스 함수를 이용하여 해결했는데, 효율성은 떨어질 것 같다. skill에 존재하지 않는 문자는 다 제거 후 그 문자열이 skill 문자열의 substring 이면서 skill의 첫 문자열부터 시작하는지를 체크하였다.

string.find() 함수 관련 정리

**str1.find(....)**

**함수 원형** : size\_t find (const string& str, size\_t index = 0) const;

**함수 원형** : size\_t find (const char\* arr, size\_t index = 0) const;

**함수 설명** : 매개변수로 들어온 문자열과, 내 문자열중에 일치하는 게 있는지 확인하는 함수입니다.

  만약에 일치하는게 있다면, 일치하는 부분의 첫번째 순서(index)를 반환합니다.

  두번째 매개변수로 들어온 index는 어느 위치에서 부터 찾을까 입니다.

**함수 예시** : str2.find("Blog");    // "BlogBlogBlogBlog" -> 0 을 반환합니다.

**함수 예시**: str2.find("Blog", 5);    // 5번째 인자부터 blog를 찾게되니 BlogB**l**og**B**logBlog 빨간색 l에서부터 찾게되므로 그 다음에 나오는 b의 위치인 8을 반환할 것 입니다.

찾지 못한 경우에 std::string::npos 를 반환한다.  
출처: <https://blockdmask.tistory.com/338> [개발자 지망생]

* 다른 사람 코드

시간 복잡도는 비슷한 것 같고, 내 코드가 더 깔끔한 것 같다.

1. 완주하지 못한 선수

**문제 설명**

수많은 마라톤 선수들이 마라톤에 참여하였습니다. 단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주하였습니다.

마라톤에 참여한 선수들의 이름이 담긴 배열 participant와 완주한 선수들의 이름이 담긴 배열 completion이 주어질 때, 완주하지 못한 선수의 이름을 return 하도록 solution 함수를 작성해주세요.

제한사항

* 마라톤 경기에 참여한 선수의 수는 1명 이상 100,000명 이하입니다.
* completion의 길이는 participant의 길이보다 1 작습니다.
* 참가자의 이름은 1개 이상 20개 이하의 알파벳 소문자로 이루어져 있습니다.
* 참가자 중에는 동명이인이 있을 수 있습니다.

입출력 예

| **participant** | **completion** | **return** |
| --- | --- | --- |
| [leo, kiki, eden] | [eden, kiki] | leo |
| [marina, josipa, nikola, vinko, filipa] | [josipa, filipa, marina, nikola] | vinko |
| [mislav, stanko, mislav, ana] | [stanko, ana, mislav] | mislav |

입출력 예 설명

예제 #1  
leo는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #2  
vinko는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #3  
mislav는 참여자 명단에는 두 명이 있지만, 완주자 명단에는 한 명밖에 없기 때문에 한명은 완주하지 못했습니다.

* 내 답

#include <string>

#include <vector>

#include <algorithm>

#include <iostream>

using namespace std;

string solution(vector<string> participant, vector<string> completion) {

string answer;

vector<string> s(participant.size());

sort(participant.begin(),participant.end());

sort(completion.begin(),completion.end());

set\_difference(participant.begin(),participant.end(),completion.begin(),completion.end(),s.begin());

cout<<s[1]<<endl;

answer = s[0];

return answer;

}

정렬한 후에 그 차이를 비교했다.

set\_difference를 사용하는 법을 배웠다.

set\_difference( 기존의 vector의 시작위치, 기존의 vector의 끝위치, 새로운 vector의 시작위치  
                   , 새로운 vector의 끝위치, 결과를 담을 vector의 시작위치)  
  
출처: <https://yongho1037.tistory.com/470> [개발 노트]