[PROGRAMMERS – HASH MAP]

// 완주하지 못한 선수

#include <string>

#include <vector>

#include <unordered\_map>

using namespace std;

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

string answer = "";

unordered\_map<string, int> strMap;

for(auto elem : completion){

if(strMap.end() == strMap.find(elem))

strMap.insert(make\_pair(elem, 1));

else

strMap[elem]++;

}

for(auto elem : participant){

if(strMap.end() == strMap.find(elem)){

answer = elem;

break;

}

else{

strMap[elem]--;

if(strMap[elem] < 0){

answer = elem;

break;

}

}

}

return answer;

}

// 잠입

|  |  |
| --- | --- |
|  | #include <string>  #include <vector>  #include <unordered\_map>  using namespace std;  int solution(vector<vector<string>> clothes) {  int answer = 1;  unordered\_map <string, int> attributes;  for(int i = 0; i < clothes.size(); i++)  attributes[clothes[i][1]]++;  for(auto it = attributes.begin(); it != attributes.end(); it++)  answer \*= (it->second+1);  answer--;  return answer;  } |

// 베스트앨범

#include <string>

#include <vector>

#include <map>

using namespace std;

vector<int> solution(vector<string> genres, vector<int> plays)

{

vector<int> answer;

map<string, int>music;

map<string, map<int, int>> musiclist;

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

{

music[genres[i]] += plays[i];

musiclist[genres[i]][i] = plays[i];

}

while(music.size() > 0)

{

string genre{};

int max{0};

for(auto mu : music)

{

if(max < mu.second)

{

max = mu.second;

genre = mu.first;

}

}

for(int i = 0; i < 2; i++)

{

int val = 0, ind = -1;

for(auto ml : musiclist[genre])

{

if(val < ml.second)

{

val = ml.second;

ind = ml.first;

}

}

if(ind == -1) break;

answer.push\_back(ind);

musiclist[genre].erase(ind);

}

music.erase(genre);

}

return answer;

}

[STACK / QUEUE]

// 탑

#include <string>

#include <vector>

using namespace std;

vector<int> solution(vector<int> heights) {

vector<int> answer;

vector<int> temp;

for(int i = heights.size() - 1; i >= 0; i--) {

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

if(heights[j] > heights[i]){

temp.push\_back(j + 1);

break;

}

if(j == 0){

temp.push\_back(0);

}

}

if(i == 0){

temp.push\_back(0);

}

}

for(int i = temp.size() - 1; i >= 0; i--){

answer.push\_back(temp[i]);

}

return answer;

}

[SORT]

//K번째 수

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

vector<int> solution(vector<int> array, vector<vector<int>> commands) {

vector<int> answer;

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

vector<int> temp;

for(int j = commands[i][0] - 1; j < commands[i][1]; j++){

temp.push\_back(array[j]);

}

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

answer.push\_back(temp[commands[i][2] - 1]);

}

return answer;

}

//가장 큰 수

#include <algorithm>

#include <string>

#include <vector>

using namespace std;

bool compare(const string &a, const string &b)

{

if (b + a < a + b)

return true;

return false;

}

string solution(vector<int> numbers) {

string answer = "";

vector<string> strings;

for (int i : numbers)

strings.push\_back(to\_string(i));

sort(strings.begin(), strings.end(), compare);

for (auto iter = strings.begin(); iter < strings.end(); ++iter)

answer += \*iter;

if (answer[0] == '0')

answer = "0";

return answer;

}

// H Index

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int solution(vector<int> citations) {

int answer = 0;

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

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

if(citations[i] >= citations.size() - i)

{

answer = citations.size() - i;

break;

}

return answer;

}

// 다리를 지나는 트럭

#include <string>

#include <vector>

#include <queue>

using namespace std;

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

int answer = 0;

int onWeight = 0;

queue<int> trucks, left;

while(-1){

int size = left.size();

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

int len = left.front();

left.pop();

if(len <= 1){

onWeight -= trucks.front();

trucks.pop();

continue;

}

left.push(len - 1);

}

if(truck\_weights.size() > 0 && onWeight + truck\_weights.at(0) <= weight){

trucks.push(truck\_weights.at(0));

onWeight += truck\_weights.at(0);

left.push(bridge\_length);

truck\_weights.erase(truck\_weights.begin());

}

answer++;

if(truck\_weights.size() == 0 && trucks.size() == 0){

break;

}

}

return answer;

}

// 기능개발

#include <string>

#include <vector>

#include <queue>

#include <iostream>

using namespace std;

vector<int> solution(vector<int> progresses, vector<int> speeds) {

vector<int> answer;

queue<int> days;

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

int temp = (100 - progresses[i]) / speeds[i];

if((100 - progresses[i]) % speeds[i] != 0) temp++;

days.push(temp);

}

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

int cnt = 0;

int d = days.front();

days.pop();

cnt++;

if(days.empty()){

answer.push\_back(cnt);

break;

}

while(-1){

if(days.front() > d || days.empty()) break;

else{

days.pop();

cnt++;

i++;

}

}

answer.push\_back(cnt);

}

return answer;

}

// 프린터

include <string>

#include <vector>

#include <queue>

using namespace std;

bool flag = false;

int solution(vector<int> priorities, int location) {

int answer = 0;

queue<int> q;

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

q.push(i);

while(!q.empty()){

int cand = q.front();

q.pop();

queue<int> canPrint;

int n = q.size();

for(int i = 0; i < n; i++){

int temp = q.front();

q.pop();

q.push(temp);

canPrint.push(temp);

}

n = canPrint.size();

if(n > 0){

for(int i = 0; i < n; i++){

int temp = canPrint.front();

canPrint.pop();

if(priorities[cand] < priorities[temp]){

q.push(cand);

break;

}

if(i == (n - 1)) flag = true;

}

if(flag == true){

answer++;

if(cand == location){

return answer;

}

}

}

else if(n == 0){

return answer + 1;

}

flag = false;

}

}

// 쇠막대기

#include <string>

#include <vector>

#include <stack>

using namespace std;

int solution(string a) {

int answer = 0;

stack<char> s;

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

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

s.push(a[i]);

else{

s.pop();

if(a[i-1] == '(') answer += s.size();

else answer ++;

}

}

return answer;

}

// 주식가격

#include <string>

#include <vector>

#include <queue>

using namespace std;

bool flag = false;

vector<int> solution(vector<int> prices) {

vector<int> answer;

queue<int> temp;

int index = -1;

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

temp.push(prices[i]);

}

while(!temp.empty()){

int now = temp.front();

temp.pop();

index++;

if(temp.size() == 0){

answer.push\_back(0);

}

else{

for(int i = index + 1; i < prices.size(); i++){

if(now > prices[i]){

answer.push\_back(i - index);

flag = true;

break;

}

}

if(flag == false){

answer.push\_back(temp.size());

}

}

flag = false;

}

return answer;

}

[HEAP]

// 더 맵게

#include <string>

#include <vector>

#include <queue>

using namespace std;

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

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

int answer = 0;

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

pq.push(scoville[i]);

while(-1){

if(pq.top() >= K){

return answer;

}

else if(pq.size() <= 1){

return -1;

}

else{

int first = pq.top(); pq.pop();

int second = pq.top(); pq.pop();

pq.push(first + second \* 2);

answer++;

}

}

}

// 라면공장

#include <string>

#include <vector>

#include <queue>

#include <iostream>

using namespace std;

struct Data{

int date;

int supply;

Data(int a, int b): date(a), supply(b) {}

};

struct cmp{

bool operator()(Data a, Data b){

return a.supply < b.supply;

}

};

int solution(int stock, vector<int> dates, vector<int> supplies, int k) {

int answer = 0;

int current = stock;

priority\_queue<Data, vector<Data>, cmp> pq;

queue<Data> q;

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

q.push(Data(dates[i], supplies[i]));

while(current < k){

while(!q.empty() && current >= q.front().date){

pq.push(q.front());

q.pop();

}

current += pq.top().supply;

pq.pop();

answer++;

}

return answer;

}

// 디스크 컨트롤러

#include <string>

#include <vector>

#include <set>

#include <algorithm>

using namespace std;

bool cmp(vector<int> a, vector<int> b){

return a[0] == b[0] ? a[1] < b[1] : a[0] < b[0];

}

int solution(vector<vector<int>> jobs) {

int answer = 0;

int num = jobs.size();

int workTime = 0;

set< pair<int, int> > pq;

sort(jobs.begin(), jobs.end(), cmp);

while(!jobs.empty()){

for(size\_t i = 0; i < jobs.size(); i++){

if(jobs[i][0] <= workTime){

pq.insert(make\_pair(jobs[i][1], i));

}

}

if(pq.empty()){

workTime = jobs[0][0] + jobs[0][1];

answer += jobs[0][1];

jobs.erase(jobs.begin());

continue;

}

auto val = pq.begin();

workTime += val->first;

answer += workTime - jobs[val->second][0];

jobs.erase(jobs.begin() + val->second);

pq.clear();

}

return answer / num;

}

//이중우선순위큐

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

bool cmp(int a, int b){

return a > b;

}

vector<int> solution(vector<string> operations) {

vector<int> answer;

vector<int> pq;

int N = operations.size();

for(int i = 0; i < N; i++){

string a;

a.assign(operations[i], 0, 1);

if(a == "I"){

operations[i].erase(0, 2);

int insert = stoi(operations[i]);

pq.push\_back(insert);

}

else if(!pq.empty() && a == "D"){

operations[i].erase(0, 2);

int del = stoi(operations[i]);

if(del == 1){

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

pq.pop\_back();

}

else if(del == -1){

sort(pq.begin(), pq.end(), cmp);

pq.pop\_back();

}

}

}

if(pq.empty()){

answer.push\_back(0); answer.push\_back(0);

}

else{

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

answer.push\_back(pq[pq.size() - 1]);

answer.push\_back(pq[0]);

}

return answer;

}

[완전탐색]

// 모의고사

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

vector<int> solution(vector<int> answers) {

vector<int> answer;

int std1[6] = {0, 1, 2, 3, 4, 5};

int std2[9] = {0, 2, 1, 2, 3, 2, 4, 2, 5};

int std3[11] = {0, 3, 3, 1, 1, 2, 2, 4, 4, 5, 5};

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

if(answers[i] == (std1[i % 5 + 1])) std1[0]++;

if(answers[i] == (std2[i % 8 + 1])) std2[0]++;

if(answers[i] == (std3[i % 10 + 1])) std3[0]++;

}

vector<int> count = {std1[0], std2[0], std3[0]};

int max = \*max\_element(count.begin(), count.end());

for(int i = 0; i < 3; i++){

if(max == count[i]) answer.push\_back(i + 1);

}

return answer;

}

// 소수 찾기

#include <string>

#include <vector>

#include <iostream>

#include <algorithm>

#include <set>

#define MAX 9999999999

using namespace std;

bool isPrime(int number){

if (number == 1) return false;

if (number == 2) return true;

if (number % 2 == 0) return false;

bool isPrime = true;

for (int i = 2; i <= sqrt(number); i++){

if (number% i == 0) return false;

}

return isPrime;

}

bool compare(char a, char b){

return a > b;

}

int solution(string numbers) {

int answer = 0;

string temp;

temp = numbers;

sort(temp.begin(), temp.end(),compare);

vector<bool> prime(std::stoi(temp)+1);

//cout << stoi(temp) << endl;

prime[0] = false;

for (long long i = 1; i < prime.size(); i++)

{

prime[i] = isPrime(i);

}

string s, sub;

s = numbers;

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

set<int> primes;

int l = s.size();

do {

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

sub = s.substr(0, i);

if (prime[std::stoi(sub)]){

primes.insert(std::stoi(sub));

}

}

} while (next\_permutation(s.begin(), s.end()));

answer = primes.size();

return answer;

}

// 숫자야구

#include <string>

#include <vector>

using namespace std;

bool check(vector<vector<int>> baseball, int cmp){

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

int S = 0, B = 0;

int temp = baseball[i][0];

if((cmp / 100) == (temp / 100)) S++;

if(((cmp / 10) % 10) == ((temp / 10) % 10)) S++;

if((cmp % 10) == (temp % 10)) S++;

if((temp / 100) == ((cmp / 10) % 10) || (temp / 100) == (cmp % 10)) B++;

if(((temp / 10) % 10) == (cmp / 100) || ((temp / 10) % 10) == (cmp % 10)) B++;

if((temp % 10) == (cmp / 100) || (temp % 10) == ((cmp / 10) % 10)) B++;

if(S != baseball[i][1] || B != baseball[i][2]) return false;

}

return true;

}

int solution(vector<vector<int>> baseball) {

int answer = 0;

for(int i = 1; i <= 9; i++){

for(int j = 1; j <= 9; j++){

for(int k = 1; k <= 9; k++){

if(i == j || j == k || k == i) continue;

int cmp = 100 \* i + 10 \* j + k;

if(check(baseball, cmp)) answer++;

}

}

}

return answer;

}

// 카펫

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

vector<int> solution(int brown, int red) {

vector<int> answer;

for(int div = 1; div <= sqrt(red); div++){

if((red % div) == 0){

int width = max(div + 2, (red / div) + 2);

int height = min(div + 2, (red / div) + 2);

if(brown == ((2 \* width) + (2 \* height) - 4)){

answer.push\_back(width);

answer.push\_back(height);

break;

}

}

}

return answer;

}

[GREEDY ALGORITHM]

// 체육복

#include <string>

#include <vector>

using namespace std;

int solution(int n, vector<int> lost, vector<int> reserve) {

int answer = 0;

vector<int> cloth(n + 2, 1);

cloth[0] = 0;

cloth[n + 1] = 0;

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

cloth[lost[i]]--;

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

cloth[reserve[i]]++;

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

if(cloth[i] == 0){

if(cloth[i - 1] == 2){

cloth[i - 1]--;

cloth[i]++;

answer++;

}

else if(cloth[i + 1] == 2){

cloth[i + 1]--;

cloth[i]++;

answer++;

}

}

else if(cloth[i] == 1) answer++;

else if(cloth[i] == 2) answer++;

}

return answer;

}

// 큰 수 만들기

#include <string>

#include <vector>

#include <algorithm>

#include <iostream>

using namespace std;

int flag = 1;

string solution(string number, int k) {

string answer = "";

int result = number.size() - k;

while(k > 0){

string temp;

temp.assign(number, 0, k);

char allDelete;

allDelete = number[k];

if(allDelete > \*max\_element(temp.begin(), temp.end())){

for(int i = 0; i < k; i++){

number.erase(number.begin());

}

k = 0;

}

else{

char max = \*max\_element(temp.begin(), temp.end());

int index = temp.find(max);

int transMax = (int)max - '0';

if(index == 0){

answer.append(to\_string(transMax));

number.erase(number.begin());

if(answer.length() == result){

flag = 0;

break;

}

continue;

}

else{

k -= index;

for(int i = 0; i < index; i++){

number.erase(number.begin());

}

}

}

}

if(flag == 1){

answer.append(number);

}

return answer;

}

// 구명보트

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

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

int answer = 0;

int N = people.size() - 1;

int M = 0;

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

while(true){

if(N <= M){

if(N == M) answer++;

break;

}

if(people[N] + people[M] <= limit){

answer++;

N--;

M++;

}

else{

answer++;

N--;

}

}

return answer;

}

// 단속카메라

#include <bits/stdc++.h>

using namespace std;

bool cmp(vector<int> a, vector<int> b) { return a[1] < b[1]; }

int solution(vector<vector<int>> routes) {

int answer = 0;

int limit = -30001;

sort(routes.begin(), routes.end(), cmp);

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

if(limit < routes[i][0]){

answer++;

limit = routes[i][1];

}

}

return answer;

}

// 섬 연결하기

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

bool cmp(vector<int> a, vector<int> b){

return a[2] < b[2];

}

int getParent(vector<int>& parent, int x){

if(parent[x] == x) return x;

return parent[x] = getParent(parent, parent[x]);

}

void unionParent(vector<int>& parent, int a, int b){

a = getParent(parent, a);

b = getParent(parent, b);

if(a < b) parent[b] = a;

else parent[a] = b;

}

int findParent(vector<int>& parent, int a, int b){

a = getParent(parent, a);

b = getParent(parent, b);

if(a == b) return 1;

else return 0;

}

int solution(int n, vector<vector<int>> costs) {

int answer = 0;

vector<int> parent(n, 0);

for(int i = 0; i < n; i++)

parent[i] = i;

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

if(costs[i][0] > costs[i][1]){

int temp = costs[i][0];

costs[i][0] = costs[i][1];

costs[i][1] = temp;

}

}

sort(costs.begin(), costs.end(), cmp);

for(auto a : costs){

if(findParent(parent, a[0], a[1]) == 0){

answer += a[2];

unionParent(parent, a[0], a[1]);

}

}

return answer;

}

// 저울

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int solution(vector<int> weight) {

int sum = 0;

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

if(weight[0] > 1) return 1;

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

sum += weight[i];

if(i == (weight.size() - 1)){

return sum + 1;

}

if(sum < weight[i + 1] && (weight[i + 1] - sum) != 1)

return sum + 1;

}

}

[Dynamic Programming]

// N으로 표현

In DFS

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int answer = 9;

void dfs(int N, int number, int count, int currentNumber){

if(count >= 9) return;

if(currentNumber == number){

answer = min(answer, count);

return;

}

int tempNumber = 0;

for(int i = 0; i < 9 + count; i++){

tempNumber = tempNumber \* 10 + N;

dfs(N, number, count + 1 + i, currentNumber + tempNumber);

dfs(N, number, count + 1 + i, currentNumber - tempNumber);

dfs(N, number, count + 1 + i, currentNumber \* tempNumber);

dfs(N, number, count + 1 + i, currentNumber / tempNumber);

}

}

int solution(int N, int number) {

dfs(N, number, 0, 0);

if(answer == 9) return -1;

return answer;

}

In DP

#include <string>

#include <vector>

#include <unordered\_set>

using namespace std;

int N;

unordered\_set<int> cache[10];

unordered\_set<int> solve(int n) {

if (!cache[n].empty()) return cache[n];

int num = 0;

for (int i = 0; i < n; i++) num = 10 \* num + N;

unordered\_set<int> res;

res.insert(num);

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

int j = n - i;

auto s1 = solve(i);

auto s2 = solve(j);

for (int n1 : s1) {

for (int n2 : s2) {

res.insert(n1 + n2);

res.insert(n1 - n2);

res.insert(n1 \* n2);

if (n2 != 0) res.insert(n1 / n2);

}

}

}

return cache[n] = res;

}

int solution(int \_N, int number) {

N = \_N;

for (int i = 1; i <= 8; i++) {

solve(i);

if (cache[i].find(number) != cache[i].end()) return i;

}

return -1;

}

// 타일 장식물

#include <string>

#include <vector>

using namespace std;

long long tiles[81];

long long dp(int N){

if(N == 1 || N == 2) return tiles[N] = 1;

else{

if(tiles[N] == 0) return tiles[N] = dp(N - 1) + dp(N - 2);

else return tiles[N];

}

}

long long solution(int N) {

return 4 \* dp(N) + 2 \* dp(N - 1);

}

// 정수 삼각형

\*\* 주석 : 효율성 문제 발생

#include <string>

#include <vector>

#include <algorithm>

#include <iostream>

using namespace std;

/\*

vector< vector<int> > maxTriangle(500);

vector<int> dp(vector<vector<int>> triangle, int depth){

if(depth == 0){

if(maxTriangle[0].empty())

maxTriangle[0].push\_back(triangle[0][0]);

return maxTriangle[0];

}

else{

if(maxTriangle[depth - 1].empty())

maxTriangle[depth - 1] = dp(triangle, depth - 1);

for(int i = 0; i < depth + 1; i++){

if(i == 0)

maxTriangle[depth].push\_back(maxTriangle[depth - 1][0] + triangle[depth][0]);

else if(i == depth)

maxTriangle[depth].push\_back(maxTriangle[depth - 1][depth - 1] + triangle[depth][depth]);

else

maxTriangle[depth].push\_back(max(maxTriangle[depth - 1][i - 1], maxTriangle[depth - 1][i]) + triangle[depth][i]);

}

}

return maxTriangle[depth];

}

int solution(vector<vector<int>> triangle) {

vector<int> result = dp(triangle, triangle.size() - 1);

return \*max\_element(result.begin(), result.end());

}

\*/

int solution(vector<vector<int>> triangle) {

int answer = 0;

int size = triangle.size();

vector<vector<int>> maxTriangle(size, vector<int>(size));

maxTriangle[0][0] = triangle[0][0];

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

for(int j = 0; j <= i; j++){

if(j == 0)

maxTriangle[i][0] = maxTriangle[i - 1][0] + triangle[i][0];

else if(j == i)

maxTriangle[i][j] = maxTriangle[i - 1][j - 1] + triangle[i][j];

else

maxTriangle[i][j] = max(maxTriangle[i - 1][j - 1], maxTriangle[i - 1][j]) + triangle[i][j];

if(i == size - 1) answer = max(maxTriangle[i][j], answer);

}

}

return answer;

}

// 등굣길

#include <string>

#include <vector>

using namespace std;

int solution(int m, int n, vector<vector<int>> puddles) {

long long cases[201][201];

for(int i = 0; i < 201; i++){

for(int j = 0; j < 201; j++){

if(i == 0 || j == 0){

cases[i][j] = 0;

}

else cases[i][j] = 1;

}

}

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

cases[puddles[i][0]][puddles[i][1]] = 0;

for(int d = 2; d <= m + n; d++){

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

int j = d - i;

if((i == 1 && j == 1) || cases[i][j] == 0) continue;

cases[i][j] = (cases[i - 1][j] + cases[i][j - 1]) % 1000000007;

if(i == m && j == n) break;

}

}

return cases[m][n];

}

// 카드 게임

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int solution(vector<int> left, vector<int> right) {

int LS = left.size(), RS = right.size();

vector< vector<int> > dp(LS + 1, vector<int>(RS + 1, 0));

for(int i = 0; i <= RS; i++)

dp[0][i] = -1;

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

for(int j = 1; j <= RS; j++){

dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - 1]);

if(dp[i][j - 1] != - 1 && left[i - 1] > right[j - 1])

dp[i][j] = dp[i][j - 1] + right[j - 1];

}

}

return dp[LS][RS];

}

// 도둑질

// 서울에서 경산까지

[BFS, DFS]

// 타켓 넘버

#include <string>

#include <vector>

#include <iostream>

using namespace std;

int answer = 0;

void BF(vector<int> numbers, int target, int index, int sum){

if(index > numbers.size()) return;

else if(sum == target && index == numbers.size()){

answer++;

}

else{

BF(numbers, target, index + 1, sum - numbers[index]);

BF(numbers, target, index + 1, sum + numbers[index]);

}

}

int solution(vector<int> numbers, int target) {

BF(numbers, target, 0, 0);

return answer;

}

// 네트워크

#include <string>

#include <vector>

using namespace std;

bool dfs(vector< vector<int> > & com, int n){

if(!com[n][n]) return false;

com[n][n] = 0;

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

if(com[n][i]) dfs(com, i);

}

return true;

}

int solution(int n, vector<vector<int>> computers) {

int answer = 0;

for(int i = 0; i < n; i++){

if(computers[i][i] && dfs(computers, i))

answer++;

}

return answer;

}

// 단어 변환

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int answer{ 100 };

bool dfs(string begin, string target, vector<string>& words, vector<bool>& useCheck, int cnt = 0){

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

int count{ 0 };

for(int j = 0; j < words[j].length(); j++)

if(!useCheck[i] && begin[j] != words[i][j]) count++;

if(count == 1){

if(target == words[i] && answer > cnt + 1){

answer = cnt + 1;

return true;

}

useCheck[i] = true;

dfs(words[i], target, words, useCheck, cnt + 1);

useCheck[i] = false;

}

}

}

int solution(string begin, string target, vector<string> words) {

vector<bool> useCheck(words.size(), false);

dfs(begin, target, words, useCheck);

if(answer == 100) return 0;

return answer;

}

// 여행경로

#include <string>

#include <vector>

#include <iostream>

using namespace std;

vector<string> answer;

void dfs(vector<vector<string>>& tickets, vector<bool>& used, string location, vector<string>& route){

if(answer.size() != 0 && route.size() == tickets.size() + 1){

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

if(answer[i].compare(route[i]) < 0){

return;

}

else if(answer[i].compare(route[i]) > 0){

for(auto k : route){

answer.erase(answer.begin());

answer.push\_back(k);

}

return;

}

}

}

else if(answer.size() == 0 & route.size() == tickets.size() + 1){

for(auto k : route)

answer.push\_back(k);

return;

}

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

if(!used[i] && tickets[i][0] == location){

used[i] = true;

route.push\_back(tickets[i][1]);

dfs(tickets, used, tickets[i][1], route);

used[i] = false;

route.pop\_back();

}

}

}

vector<string> solution(vector<vector<string>> tickets) {

vector<bool> used(tickets.size(), false);

vector<string> route;

route.push\_back("ICN");

dfs(tickets, used, "ICN", route);

return answer;

}

[이분탐색]

// 예산

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int solution(vector<int> budgets, int M) {

int num = budgets.size();

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

for(auto k : budgets){

if(k > (M / num)) return M / num;

else{

M -= k;

num--;

}

}

return budgets[budgets.size() - 1];

}

// 입국심사

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

long long solution(int n, vector<int> times) {

long long answer = 0;

long long min = 1, avg, humans;

long long max = \*max\_element(times.begin(), times.end()) \* (long long) n;

while(min <= max){

avg = (max + min) / 2;

for(auto t : times)

humans += avg / t;

if(n <= humans){

answer = avg;

max = avg - 1;

}

else

min = avg + 1;

humans = 0;

}

return answer;

}

// 징검다리

[그래프]

// 가장 먼 노드

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int INF = 1000000000;

int solution(int n, vector<vector<int>> edge) {

int answer = 0;

vector<vector<int>> arr(n, vector<int>(n, INF));

for(int i = 0; i < n; i++)

arr[i][i] = 0;

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

arr[edge[i][0] - 1][edge[i][1] - 1] = 1;

arr[edge[i][1] - 1][edge[i][0] - 1] = 1;

}

vector<int> d(n);

for(int i = 0; i < n; i++)

d[i] = arr[0][i];

vector<bool> v(n, false);

v[0] = true;

for(int i = 0; i < n - 2; i++){

int min = INF;

int index = 0;

for(int j = 0; j < n; j++){

if(d[j] < min && !v[j]){

min = d[j];

index = j;

}

}

v[index] = true;

for(int j = 0; j < n; j++){

if(!v[j]){

if(d[index] + arr[index][j] < d[j]){

d[j] = d[index] + arr[index][j];

}

}

}

}

int max = \*max\_element(d.begin(), d.end());

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

if(d[i] == max) answer++;

}

return answer;

}

// 순위

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

int solution(int n, vector<vector<int>> results) {

int answer = 0;

vector< vector<bool> > graph(n + 1, vector<bool>(n + 1, false));

for(auto k : results) graph[k[0]][k[1]] = true;

for(int k = 1; k <= n; k++){

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

for(int j = 1; j <= n; j++){

if(graph[i][k] && graph[k][j])

graph[i][j] = true;

}

}

}

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

int cnt = 0;

for(int j = 1; j <= n; j++){

if(graph[i][j] || graph[j][i])

cnt++;

}

if(cnt == n - 1)

answer++;

}

return answer;

}

// 사이클 제거

// 방의 개수

[2020카카오공채]

// 문자열 압축

#include <string>

#include <vector>

#include <cmath>

#include <iostream>

using namespace std;

int solution(string sa) {

int answer = 10000000;

for(int i = 1; i <= sa.length() / 2 + 1; i++){

int cnt = 1;

int result = 0;

string s = sa;

while(!s.empty()){

string temp1, temp2;

if(s.length() >= i){

temp1.assign(s, 0, i);

s.erase(0, i);

}

else{

temp1 = s;

s.clear();

}

if(s.length() >= i){

temp2.assign(s, 0, i);

}

else{

temp2 = s;

}

if(temp1.compare(temp2) == 0){

cnt++;

}

else{

if(cnt == 1) result += temp1.size();

else{

string numLength = to\_string(cnt);

result += numLength.size() + temp1.size();

}

cnt = 1;

}

}

answer = min(result, answer);

}

return answer;

}