[[SWEA] 2819. 격자판의 숫자 이어 붙이기](https://dheldh77.tistory.com/entry/SWEA-2819-%EA%B2%A9%EC%9E%90%ED%8C%90%EC%9D%98-%EC%88%AB%EC%9E%90-%EC%9D%B4%EC%96%B4-%EB%B6%99%EC%9D%B4%EA%B8%B0)

2020. 3. 11. 14:06

[2819](https://dheldh77.tistory.com/tag/2819), [DFS](https://dheldh77.tistory.com/tag/DFS), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [깊이우선탐색](https://dheldh77.tistory.com/tag/%EA%B9%8A%EC%9D%B4%EC%9A%B0%EC%84%A0%ED%83%90%EC%83%89), [알고리즘](https://dheldh77.tistory.com/tag/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98)

**문제**

4×4 크기의 격자판이 있다. 격자판의 각 격자칸에는 0부터 9 사이의 숫자가 적혀 있다.  
  
격자판의 임의의 위치에서 시작해서, 동서남북 네 방향으로 인접한 격자로 총 여섯 번 이동하면서, 각 칸에 적혀있는 숫자를 차례대로 이어 붙이면 7자리의 수가 된다.  
  
이동을 할 때에는 한 번 거쳤던 격자칸을 다시 거쳐도 되며, 0으로 시작하는 0102001과 같은 수를 만들 수도 있다.  
  
단, 격자판을 벗어나는 이동은 가능하지 않다고 가정한다.  
  
격자판이 주어졌을 때, 만들 수 있는 서로 다른 일곱 자리 수들의 개수를 구하는 프로그램을 작성하시오.

**풀이방법**

DFS(깊이 우선 탐색)알고리즘으로 문제를 해결

방문했던 노드를 재방문할 수 있는 길이가 7인 모든 경로를 구한다.

이 때 중복되는 숫자를 처리하는 방법은 0부터 9999999까지 인덱스를 가지는 배열을 만들고

7자리 숫자를 찾을 때마다 이에 해당하는 인덱스에 값을 1로 표시한다.

최종적으로 배열의 원소 값 1의 개수가 격자판에서 만들 수 있는 경로의 개수가 된다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_2819 {

static String[][] a;

static int cnt;

static int[] dx = {-1, 0, 1, 0};

static int[] dy = {0, 1, 0, -1};

static int[] visit;

public static void dfs(int x, int y, int depth, String s) {

if(depth == 6) {

s+= a[x][y];

int num = Integer.parseInt(s);

if(visit[num] == 0) {

visit[num] = 1;

cnt++;

}

return;

}

s += a[x][y];

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < 4 && ny < 4) {

dfs(nx, ny, depth+1, s);

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

cnt = 0;

a = new String[4][4];

visit = new int[10000000];

for(int i = 0; i < 4; i++) {

for(int j = 0; j < 4; j++) {

a[i][j] = String.valueOf(sc.nextInt());

}

}

for(int i = 0; i < 4; i++) {

for(int j = 0; j < 4; j++) {

String s = "";

dfs(i, j, 0, s);

}

}

System.out.println("#" + t + " " + cnt);

}

}

}

[[SWEA] 1865. 동철이의 일 분배](https://dheldh77.tistory.com/entry/SWEA-1865-%EB%8F%99%EC%B2%A0%EC%9D%B4%EC%9D%98-%EC%9D%BC-%EB%B6%84%EB%B0%B0)

2020. 3. 10. 12:36

[1865](https://dheldh77.tistory.com/tag/1865), [Backtracking](https://dheldh77.tistory.com/tag/Backtracking), [DFS](https://dheldh77.tistory.com/tag/DFS), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [깊이우선탐색](https://dheldh77.tistory.com/tag/%EA%B9%8A%EC%9D%B4%EC%9A%B0%EC%84%A0%ED%83%90%EC%83%89), [되추적](https://dheldh77.tistory.com/tag/%EB%90%98%EC%B6%94%EC%A0%81)

**문제**

동철이가 차린 전자회사에는 N명의 직원이 있다.  
  
그런데 어느 날 해야할 일이 N개가 생겼다.  
  
동철이는 직원들에게 공평하게 일을 하나씩 배분하려고 한다.  
  
직원들의 번호가 1부터 N까지 매겨져 있고, 해야 할 일에도 번호가 1부터 N까지 매겨져 있을 때, i번 직원이 j번 일을 하면 성공할 확률이 Pi, j이다.  
  
여기서 우리는 동철이가 모든 일이 잘 풀리도록 도와주어야 한다.  
  
직원들에게 해야 할 일을 하나씩 배분하는 방법은 여러 가지다.  
  
우리는 여러 방법 중에서 생길 수 있는 “주어진 일이 모두 성공할 확률”의 최댓값을 구하는 프로그램을 작성해야 한다.

**풀이방법**

DFS(깊이 우선 탐색)알고리즘과 Backtracking(되추적)으로 해결한다.

모든 직원이 일이 중복 되지 않도록 일을 선택할 수 있는 모든 경우 중 최대값을 찾는다.

1번 직원이 선택할 수 있는 일은 n가지이다.

 1번 직원이 일을 선택한 후, 2번 직원이 선택할 수 있는 일은 n-1가지이다.

 2번 직원이 일을 선택한 후, 3번 직원이 선택할 수 있는 일은 n-2가지이다.

 ...

 n-1번 직원이 일을 선택한 후, n번 직원이 선택할 수 있는 일은 1가지이다.

즉, 1부터 n까지 n개의 수가 주어졌을 때, 중복이 허용되지 않는 모든 순열을 구하는 문제이다.

순열의 모든 경우의 수는 n개의 숫자가 주여졌을 때, n!이 된다.

 문제에서 n은 16까지 입력되므로 16!일 경우 시간초과가 발생하게 된다.

따라서 백트레킹으로 유망하지 않는 경우는 프루닝해서 탐색하는 노드의 수를 줄인다.

프루닝의 조건은 현재 탐색하는 경로에서 i번째 직원(1 <= i <= n)까지의 성공확률이 이전에 구했던 최대 성공확률보다 작을 때이다.

확률은 0 ~ 1사이인데, 0 ~ 1 사이의 숫자를 계속 곱하면 계속해서 작아지기만 할 뿐이다.

따라서 이미 이전의 최대 성공확률보다 작다면 최대 성공확률이 될 수 없으므로 백트레킹한다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_1865 {

static double[][] a;

static int n;

static double max;

static int[] visit;

public static void dfs(int x, int depth, double result) {

result \*= a[x][depth];

visit[x] = 1;

if(depth == n - 1) {

if(max < result)

max = result;

visit[x] = 0;

return;

}

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

if(visit[i] == 0 && result \* a[i][depth+1] > max) dfs(i, depth+1, result);

}

visit[x] = 0;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

n = sc.nextInt();

a = new double[n][n];

visit = new int[n];

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

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

a[i][j] = sc.nextDouble() / 100;

}

}

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

dfs(i, 0, 1);

}

System.out.printf("#%d %.6f\n", t, max\*100);

}

}

}

[[SWEA] 7699. 수지의 수지 맞는 여행](https://dheldh77.tistory.com/entry/SWEA-7699-%EC%88%98%EC%A7%80%EC%9D%98-%EC%88%98%EC%A7%80-%EB%A7%9E%EB%8A%94-%EC%97%AC%ED%96%89)

2020. 3. 10. 12:15

[7699](https://dheldh77.tistory.com/tag/7699), [DFS](https://dheldh77.tistory.com/tag/DFS), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [깊이우선탐색](https://dheldh77.tistory.com/tag/%EA%B9%8A%EC%9D%B4%EC%9A%B0%EC%84%A0%ED%83%90%EC%83%89)

**문제**

평소에 여행을 즐기는 수지는 이번에 새로운 섬에 도착했다.  
  
이 섬은 1행, 1열로 시작해서 R행, C열까지 있으며, 총 R\*C 칸으로 이루어져 있다.  
  
수지는 지금 1행 1열에 있으며 앞으로 있을 야근을 위하여 기회 있을 때 미리 여행을 떠나려고 한다.  
  
이 섬의 각 칸에는 알파벳이 적혀있다. 이 알파벳은 섬의 명물이고, 같은 알파벳은 같은 명물이다.  
  
이 섬에서는 명물을 볼 때마다 요금을 지급해야 하는데 각 알파벳 명물을 처음 볼 땐 무료로 볼 수 있다.  
  
그리고, 수지는 여행을 할 때 자신이 있는 지점의 명물을 본 후 4방향(상, 하, 좌, 우) 중 한 방향으로 1칸 이동 후 다음 명물을 보는 행동을 반복한다.  
  
  
올해 SGA사 1년 차인 수지는 현재 대출빚과 카드빚에 허덕이고 있다.  
  
따라서, 명물을 최대한 많이 보되, 요금을 지급하지 않는 방법을 택해야 한다.  
  
수지가 1행 1열에서 여행을 시작했을 때, 같은 알파벳 명물을 두 번 이상 보지 않도록 여행을 떠나는 방법 중에 볼 수 있는 명물의 최대 개수를 구하여라.

**풀이방법**

DFS(깊이 우선 탐색)알고리즘으로 해결한다.

한 정점을 시작으로 상, 하, 좌, 우에 인접한 노드 중 조건을 만족하는 모든 노드를 방문한다.

이 때의 조건은, 방문했던 노드의 알파벳과 같은 노드는 방문하지 않는다.

조건을 만족하는 노드를 방문해 나온 경로 중 가장 긴 경로를 출력한다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_7699 {

static int[][] map;

static int n;

static int m;

static int[] visit;

static int[] dx = {-1, 0, 1, 0};

static int[] dy = {0, 1, 0, -1};

static int max;

public static void dfs(int x, int y, int depth) {

visit[map[x][y]] = 1;

depth++;

if(max < depth)

max = depth;

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < n && ny < m) {

if(visit[map[nx][ny]] == 0) dfs(nx, ny, depth);

}

}

visit[map[x][y]] = 0;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

n = sc.nextInt();

m = sc.nextInt();

map = new int[n][m];

visit = new int[100];

max = 0;

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

String s = sc.next();

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

map[i][j] = s.charAt(j) - '0';

}

}

dfs(0, 0, 0);

System.out.println("#" + t + " " + max);

}

}

}

[[SWEA] 8659. GCD](https://dheldh77.tistory.com/entry/SWEA-8659-GCD)

2020. 3. 10. 12:11

[8659](https://dheldh77.tistory.com/tag/8659), [DP](https://dheldh77.tistory.com/tag/DP), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [동적계획법](https://dheldh77.tistory.com/tag/%EB%8F%99%EC%A0%81%EA%B3%84%ED%9A%8D%EB%B2%95)

**문제**

의석이는 종강 기념 피자 파티를 열기 위해서 피자를 사러 왔다.  
  
하지만 피자 가게 주인 동욱이는 피자를 순순히 판매하지 않는 사람이다.  
  
돈 보다 문제 내는 것을 더 좋아하는 이상한 동욱이는 피자를 사러 온 의석이에게 3개의 시련을 부여했고, 모두 통과해야만 거래를 시작한다.  
  
두 번째 관문에서는 GCD에 관련된 내용을 물어본다.  
   
  
           GCD(a,0) = a  
           GCD(a,b) = GCD(b, a%b)  
   
  
GCD는 위와 같은 성질을 가진 함수이다.  
   
동욱이가 원하는 것은 두 개의 숫자 A, B(A>B)를 찾는 것이다.  
   
단, 이 두 개의 숫자에 대해서 GCD(A,B)를 실행하면 % 연산자가 총 K 번 수행된다는 사실을 알고 있다.  
   
이러한 두 개의 숫자 조합 중에서 A가 가장 작으며 그런 조합이 여러 가지라면 B가 가장 작은 조합을 원한다.  
   
예를 들어서, % 연산자가 3번 수행되는 조합은 (5, 3), (8, 3), (10, 6) 등이 있다.  
   
그 중 원하는 조합은 (5, 3) 이 되는 것이다.  
   
의석이를 도와서 K가 주어졌을 때, 관문 2의 정답을 구하는 프로그램을 작성하라.

**풀이방법**

동적 계획법(DP, Dynamic Programming)을 이용해 해결한다.

GCD(a, 0) = a,  GCD(a, b) = GCD(b, a%b)일 때, k번 수행하는 가장 작은 조합을 찾아야하므로

1부터 k를 만족하는 가장 작은 a, b부터 찾는다.

(k = 0) GCD(1, 0) = 1

(k = 1) GCD(2, 1) = GCD(1, 2 % 1) = GCD(1, 0) = 1

(k = 2) GCD(3, 2) = GCD(2, 3 % 2) = GCD(2, 1) = GCD(1, 0) = 1

(k = 3) GCD(5, 3) = GCD(3, 5 % 3) = GCD(3, 2) = GCD(2, 1) = GCD(1, 0) = 1

(k = 3) GCD(8, 5) = GCD(5, 8 % 5) = GCD(5, 3) = GCD(3, 2) = GCD(2, 1) = GCD(1, 0) = 1

k+1일 때, GCD(a,b) 나머지 연산을 한 번했을 때, GCD(a, b)는 k일 때, GCD(a, b)의 a, b와 같다.

따라서 아래와 같은 점화식을 찾을 수 있다.

**a[k] = a[k-1] + b[k-1]**

**b[k] = a[k-1]**

**소스코드**

package samsung;

import java.util.\*;

public class s\_8659 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

int k = sc.nextInt();

long[][] dp = new long[k][2];

dp[0][0] = 2;

dp[0][1] = 1;

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

dp[i][0] = dp[i-1][0] + dp[i-1][1];

dp[i][1] = dp[i-1][0];

}

System.out.println("#" + t + " " + dp[k-1][0] + " " + dp[k-1][1]);

}

}

}

[[SWEA] 4050. 재관이의 대량 할인](https://dheldh77.tistory.com/entry/SWEA-4050-%EC%9E%AC%EA%B4%80%EC%9D%B4%EC%9D%98-%EB%8C%80%EB%9F%89-%ED%95%A0%EC%9D%B8)

2020. 3. 9. 11:29

**문제**

재관이의 옷 매장에서 대량 세일을 하고자 한다.  
  
세일에는 조건이 한가지 있는데, 세 벌의 옷을 사면 그 중 가장 저렴한 한 벌에 해당하는 값은 내지 않아도 된다는 조건이다.  
  
그리고 세 벌보다 많은 옷을 구매하는 경우에도 옷을 세 벌씩 나눠서 계산하면 같은 방식의 할인을 받을 수 있다.  
  
예를 들면 10, 3, 2, 4, 6, 4, 9원짜리 옷을 사고자 할 때 (3, 2, 4), (10, 4, 9), (6)원짜리로 묶어서 계산을 하게 된다면,  
  
첫 묶음에서 2원, 두 번째 묶음에서 4원 총 6원의 할인을 받게 되는 것이다.  
  
다만 세 번째 묶음은 한 벌만 구매하므로 할인이 적용되지 않는다.  
  
재관이네 매장에서 사고자 하는 옷이 있을 때 어떻게 묶어서 결제를 하면 가장 할인을 많이 받아서 구매를 할 수 있는지 계산하여라.



두 번째 테스트케이스를 예로 보면, {6,4,5,5,5,5}의 옷들이 있을 때 (6,4,5), (5,5,5)로 묶으면 가장 저렴한 조합이 된다.

**풀이방법**

그리디 알고리즘으로 해결한다.

가장 적은 액수를 내기 위해서는 가장 큰 금액을 할인받아야한다.

가장 큰 액수를 할인 받는 방법은 높은 가격 순으로 3개를 묶었을 때, 그 중에서 가장 작은 값을 할인받을 수 있게 된다.

예를 들어 {1, 2, 3, 4, 5, 6} 가격의 물건들을 구입했을 때,

가격이 높은 순으로 정렬하면 {6, 5, 4, 3, 2, 1}과 같다.

이 값들을 3개씩 묶으면 {6, 5, 4}, {3, 2, 1}이고, 따라서 4천원, 1천원, 총 5천원을 할인받을 수 있다.

입력받은 수들을 내림차순으로 정렬하고, i+2번째에 있는 값들을 제외한 합을 구한다.

단, 구입한 물건들을 3개씩 묶을 수 없는 경우는 3개씩 묶고 남은 물건은 가격이 제일 낮은 순으로 한다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_4050 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

int n = sc.nextInt();

ArrayList<Integer> a = new ArrayList<>();

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

int tmp = sc.nextInt();

a.add(tmp);

}

Collections.sort(a, Collections.reverseOrder());

int sum = 0;

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

sum += a.get(n-1-i);

}

int i = 0;

while(i + 2 < n) {

sum += a.get(i);

sum += a.get(i+1);

i += 3;

}

System.out.println("#" + t + " " + sum);

}

}

}

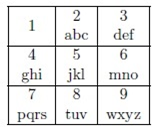
[[SWEA] 4261. 빠른 휴대전화 키패드](https://dheldh77.tistory.com/entry/SWEA-4261-%EB%B9%A0%EB%A5%B8-%ED%9C%B4%EB%8C%80%EC%A0%84%ED%99%94-%ED%82%A4%ED%8C%A8%EB%93%9C)

2020. 3. 9. 11:18

[4261](https://dheldh77.tistory.com/tag/4261), [SWEA](https://dheldh77.tistory.com/tag/SWEA)

**문제**

동한이는 창고에서 오래된 애니콜 휴대전화를 발견했다.  
  
이 휴대전화의 키패드는 아래와 같이 생겼다.



이 키패드는 각 키를 여러 번 눌러 영문을 입력할 수 있는데, a를 입력하려면 2를 한 번, b를 입력하려면 2를 두 번 누르는 식이다.  
  
그러나 동한이는 이 방식이 너무 느리다고 생각하여 문자열을 빠르게 타이핑할 수 있도록 키패드를 다음과 같이 바꿔 보려고 한다.  
  
사용할 단어들을 미리 휴대폰에 저장한 뒤, 해당 알파벳이 써있는 숫자를 한 번씩만 누르면 가능한 여러 단어 중에 사전에 저장된 단어를 찾아서 입력하는 것이다.  
  
예를 들면 car를 입력하려면 222, 2, 777을 입력하는 것이 정상이지만 이 자판의 경우 227을 입력하면 aap, aaq, …, ccs 등 3×3×4=36개의 단어 중에 사전에 존재하는 단어를 출력해준다.  
  
하지만 동한이는 이 시스템의 문제점을 발견하였다. 이 예시의 경우 car는 cap과 표현이 겹친다는 점을 확인할 수 있는데,  
  
이처럼 키 입력이 동일한 단어가 여럿 존재할 수 있다는 것이다.  
  
동한이는 얼마나 이런 경우가 자주 발생되는지 확인하기 위해, 사전과 키 입력이 주어지면 해당하는 키 입력과 대응되는 단어가 몇 개인지 구하는 프로그램을 원한다.  
  
이러한 프로그램을 작성하라.

**풀이방법**

각 문자 a, b, c, d, ... ,y, z를 int형으로 바꿔 키패드가 문자를 정수형으로 바꾼 범위안에 포함되는지 확인한다.

97(a) <= 키패드 2 <= 99(c)

97(d) <= 키패드 3 <= 99(f)

97(g) <= 키패드 4 <= 99(i)

97(j) <= 키패드 5 <= 99(l)

97(m) <= 키패드 6 <= 99(o)

97(p) <= 키패드 7 <= 99(s)

97(t) <= 키패드 8 <= 99(v)

97(w) <= 키패드 9 <= 99(z)

**소스코드**

package samsung;

import java.util.\*;

public class s\_4261 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

int[][] range = {{97, 99}, {100, 102}, {103, 105}, {106, 108}, {109, 111}, {112, 115}, {116, 118}, {119, 122}};

for(int t = 1; t <= tc; t++) {

String num = sc.next();

int n = sc.nextInt();

int cnt = 0;

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

String s = sc.next();

int flag = 0;

if(s.length() != num.length()) continue;

for(int j = 0; j < s.length(); j++) {

if((int)s.charAt(j) < range[num.charAt(j) - '0' - 2][0] || (int)s.charAt(j) > range[num.charAt(j) - '0' - 2][1]) {

flag = 1;

break;

}

}

if(flag == 0)

cnt++;

}

System.out.println("#" + t + " " + cnt);

}

}

}

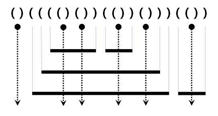
[[SWEA] 5432. 쇠막대기 자르기](https://dheldh77.tistory.com/entry/SWEA-5432-%EC%87%A0%EB%A7%89%EB%8C%80%EA%B8%B0-%EC%9E%90%EB%A5%B4%EA%B8%B0)

2020. 3. 9. 11:12

[5432](https://dheldh77.tistory.com/tag/5432), [SWEA](https://dheldh77.tistory.com/tag/SWEA)

**문제**

여러 개의 쇠막대기를 레이저로 절단하려고 한다.  
  
효율적인 작업을 위해서 쇠막대기를 아래에서 위로 겹쳐 놓고, 레이저를 위에서 수직으로 발사하여 쇠막대기들을 자른다.  
  
쇠막대기와 레이저의 배치는 다음 조건을 만족한다.  
  
λ - 쇠막대기는 자신보다 긴 쇠막대기 위에만 놓일 수 있다.  
  
λ - 쇠막대기를 다른 쇠막대기 위에 놓는 경우 완전히 포함되도록 놓되, 끝점은 겹치지 않도록 놓는다.  
  
λ - 각 쇠막대기를 자르는 레이저는 적어도 하나 존재한다.  
  
λ - 레이저는 어떤 쇠막대기의 양 끝점과도 겹치지 않는다.  
  
아래 그림은 위 조건을 만족하는 예를 보여준다.  
  
수평으로 그려진 굵은 실선은 쇠막대기이고, 점은 레이저의 위치, 수직으로 그려진 점선 화살표는 레이저의 발사 방향이다.



이러한 레이저와 쇠막대기의 배치는 다음과 같이 괄호를 이용하여 왼쪽부터 순서대로 표현할 수 있다.  
  
    1. 레이저는 여는 괄호와 닫는 괄호의 인접한 쌍 “()” 으로 표현된다. 또한, 모든 “()”는 반드시 레이저를 표현한다.  
  
    2. 쇠막대기의 왼쪽 끝은 여는 괄호 ‘(’ 로, 오른쪽 끝은 닫힌 괄호 ‘)’ 로 표현된다.  
  
위 예의 괄호 표현은 그림 위에 주어져 있다.  
  
쇠막대기는 레이저에 의해 몇 개의 조각으로 잘려지는데, 위 예에서 가장 위에 있는 두 개의 쇠막대기는 각각 3개와 2개의 조각으로 잘려지고,  
  
이와 같은 방식으로 주어진 쇠막대기들은 총 17개의 조각으로 잘려진다.  
  
쇠막대기와 레이저의 배치를 나타내는 괄호 표현이 주어졌을 때, 잘려진 쇠막대기 조각의 총 개수를 구하는 프로그램을 작성하라.

**풀이방법**

문자열로 입력받고 각 문자마다 경우를 생각해본다.

1) '('와 ')'가 연속으로 입력되는 경우 : 현재까지 쌓인 쇠막대기를 자르는 경우

    -> 잘려진 쇠막대기의 개수가 쌓인 쇠막대기만큼 증가한다.

2) 1)을 만족하지 않으면서 '('가 들어오는 경우 : 쇠막대기를 하나 더 쌓는 경우

    -> 현재 쌓여있는 쇠막대기에 하나 더 쌓는다.

3) 1)을 만족하지 않으면서 ')'가 들어오는 경우 : 쇠막대기의 길이가 다한 경우

    -> 쌓여있는 쇠막대기의 개수를 하나 감소시키고, 잘려진 쇠막대기의 수를 하나 증가시킨다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_5432 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

String s = sc.next();

int n = s.length();

int board = 0;

int cnt = 0;

for(int i = 0; i < s.length() - 1; i++) {

if(s.charAt(i) == '(' && s.charAt(i + 1) == ')') cnt += board;

else if(s.charAt(i) == ')' && s.charAt(i - 1) == '(') continue;

else if(s.charAt(i) == '(') board++;

else if(s.charAt(i) == ')') {

if(board > 0) {

board--;

cnt++;

}

}

}

cnt++;

System.out.println("#" + t + " " + cnt);

}

}

}

[[SWEA] 1861. 정사각형 방](https://dheldh77.tistory.com/entry/SWEA-1861-%EC%A0%95%EC%82%AC%EA%B0%81%ED%98%95-%EB%B0%A9)

2020. 3. 9. 11:04

[1861](https://dheldh77.tistory.com/tag/1861), [DFS](https://dheldh77.tistory.com/tag/DFS), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [깊이우선탐색](https://dheldh77.tistory.com/tag/%EA%B9%8A%EC%9D%B4%EC%9A%B0%EC%84%A0%ED%83%90%EC%83%89)

**문제**

N^2개의 방이 N×N형태로 늘어서 있다.  
  
위에서 i번째 줄의 왼쪽에서 j번째 방에는 1이상 N2 이하의 수 Ai,j가 적혀 있으며, 이 숫자는 모든 방에 대해 서로 다르다.  
  
당신이 어떤 방에 있다면, 상하좌우에 있는 다른 방으로 이동할 수 있다.  
  
물론 이동하려는 방이 존재해야 하고, 이동하려는 방에 적힌 숫자가 현재 방에 적힌 숫자보다 정확히 1 더 커야 한다.  
  
처음 어떤 수가 적힌 방에서 있어야 가장 많은 개수의 방을 이동할 수 있는지 구하는 프로그램을 작성하라.

**풀이방법**

DFS(깊이 우선 탐색)로 모든 경로를 찾는다.

A[i][j] = x일 때, 상하좌우의 원소로 방문할 수 있는 경우는, 방문하려는 원소의 값이 x + 1이면서 방문한 적이 없어야한다.

위의 조건을 만족하는 모든 경로 중 경로의 길이가 가장 긴 경우를 찾되, 길이가 같을 경우, 시작 원소가 가장 작을 값을 출력한다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_1861 {

static int[][] a;

static int n;

static int[][] visit;

static int idx;

static int max;

static int[] dx = {-1, 0, 1, 0};

static int[] dy = {0, 1, 0, -1};

public static void dfs(int start, int depth, int x, int y) {

visit[x][y] = 1;

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < n && ny < n) {

if(a[nx][ny] == (a[x][y] + 1) && visit[nx][ny] == 0)

dfs(start, depth + 1, nx, ny);

}

}

if(depth > max) {

max = depth;

idx = start;

}

if(depth == max) {

idx = Math.min(idx, start);

}

visit[x][y] = 0;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

n = sc.nextInt();

a = new int[n][n];

visit = new int[n][n];

idx = 0;

max = 0;

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

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

a[i][j] = sc.nextInt();

}

}

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

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

dfs(a[i][j], 1, i, j);

}

}

System.out.println("#" + t + " " + idx + " " + max);

}

}

}

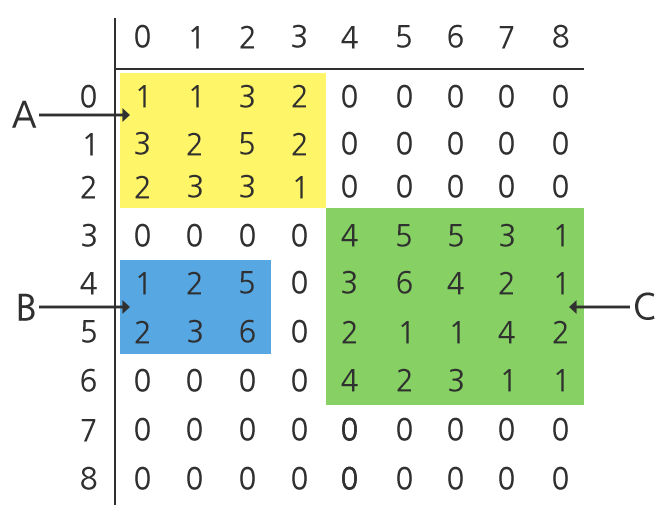
[[SWEA] 1258. [S/W 문제해결 응용] 7일차 - 행렬찾기](https://dheldh77.tistory.com/entry/SWEA-1258-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EC%9D%91%EC%9A%A9-7%EC%9D%BC%EC%B0%A8-%ED%96%89%EB%A0%AC%EC%B0%BE%EA%B8%B0)

2020. 3. 9. 10:54

[1258](https://dheldh77.tistory.com/tag/1258), [BFS](https://dheldh77.tistory.com/tag/BFS), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [너비우선탐색](https://dheldh77.tistory.com/tag/%EB%84%88%EB%B9%84%EC%9A%B0%EC%84%A0%ED%83%90%EC%83%89), [행렬](https://dheldh77.tistory.com/tag/%ED%96%89%EB%A0%AC)

**문제**

유엔 화학 무기 조사단이 대량 살상 화학 무기를 만들기 위해 화학 물질들이 저장된 창고를 조사하게 되었다.  
  
창고에는 화학 물질 용기 n2개가 n x n으로 배열되어 있었다.  
  
유엔 조사단은 각 용기를 조사하여 2차원 배열에 그 정보를 저장하였다.  
  
빈 용기에 해당하는 원소는 ‘0’으로 저장하고, 화학 물질이 들어 있는 용기에 해당하는 용기는 화학 물질의 종류에 따라 ‘1’에서 ‘9’사이의 정수를 저장하였다.  
  
다음 그림은 창고의 화학 물질 현황을 9x9 배열에 저장한 예를 보여준다.



유엔 조사단은 화학 물질이 담긴 용기들로부터 3가지 사항을 발견하였다.  
  
1. 화학 물질이 담긴 용기들이 사각형을 이루고 있다. 또한, 사각형 내부에는 빈 용기가 없다.  
  
예를 들어, 위의 그림에는 3개의 화학 물질이 담긴 용기들로 이루어진 사각형 A, B, C가 있다.  
  
2. 화학 물질이 담긴 용기들로 이루어진 사각형들은 각각 차원(가로의 용기 수 x 세로의 용기 수)이 다르다.  
  
예를 들어, 위의 그림에서 A는 3x4, B는 2x3, C는 4x5이다.  
  
3. 2개의 화학 물질이 담긴 용기들로 이루어진 사각형들 사이에는 빈 용기들이 있다.  
  
예를 들어, 위의 그림에서 A와 B 사이와 B와 C 사이를 보면, 빈 용기를 나타내는 ‘0’ 원소들이 2개의 사각형 사이에 있는 것을 알 수 있다.  
  
단, A와 C의 경우와 같이 대각선 상으로는 빈 용기가 없을 수도 있다.  
  
유엔 조사단은 창고의 용기들에 대한 2차원 배열에서 행렬(화학 물질이 든 용기들로 이루어진 사각형)들을 찾아내고 정보를 수집하고자 한다.  
  
찾아낸 행렬들의 정보를 출력하는 프로그램을 작성하시오.  
  
**[제약 사항]**  
  
n은 100 이하이다.  
  
부분 행렬의 열의 개수는 서로 다르며 행렬의 행의 개수도 서로 다르다.  
  
예를 들어, 3개의 부분행렬 행렬 (A(3x4), B(2x3), C(4x5))이 추출되었다면, 각 부분 행렬의 행의 수는 3, 2, 4로 서로 다르다.  
  
마찬가지로 각 부분 행렬의 열의 수도 4, 3, 5로 서로 다르다.  
  
테스트 케이스는 여러 개의 그룹으로 구성되며 아래와 같다.  
그룹 1. n <= 10 이고 sub matrix의 개수 5개 이하  
그룹 2. n <= 40 이고 5 < sub matrix <= 10  
그룹 3. 40 < n <=80 이고 5 < sub matrix <= 10  
그룹 4. 40 < n <=80 이고 10 < sub matrix <= 15  
그룹 5. 80 < n<=100 이고 15 < sub matrix <= 20

**풀이방법**

BFS(너비 우선 탐색)으로 각 영역의 수를 찾는다. 2차원 배열이 주어졌을 때, 원소의 값이 0 보다 크면서 원소를 방문하지 않았다면

큐에 담고 탐색을 시작한다.

이 때, 큐에 맨 처음에 담았던 x1, y1 좌표와 큐에서 맨 마지막에 나온 x2, y2좌표를 저장하고,

클래스를 만들어, 영역마다 행의 길이(x2 - x1), 열의 길이(y2 - y1), 영역의 크기((x2 - x1) \* (y2 - y1))를 저장한 객체를 생성한다.

영역의 크기 순으로 객체를 정렬하고, 행의 길이, 열의 길이, 크기를 출력한다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_1258 {

static int[][] a;

static int n;

static int[][] v;

static int[] dx = {-1, 0, 1, 0};

static int[] dy = {0, 1, 0, -1};

static ArrayList <Ret> rets;

public static class Ret implements Comparable<Ret>{

int row;

int col;

int size;

Ret(int row, int col){

this.row = row + 1;

this.col = col + 1;

this.size = this.row \* this.col;

}

@Override

public int compareTo(Ret r) {

if(this.size > r.size)

return 1;

else

return -1;

}

}

public static class Node{

int x;

int y;

Node(int x, int y){

this.x = x;

this.y = y;

}

}

public static void bfs(int x, int y) {

Queue <Node> q = new LinkedList<>();

v[x][y] = 1;

int start\_x = x;

int start\_y = y;

int end\_x = x;

int end\_y = y;

q.add(new Node(x, y));

while(!q.isEmpty()) {

Node node = q.poll();

x = node.x;

y = node.y;

end\_x = x;

end\_y = y;

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < n && ny < n) {

if(a[nx][ny] > 0 && v[nx][ny] == 0) {

v[nx][ny] = 1;

q.add(new Node(nx,ny));

}

}

}

}

rets.add(new Ret(end\_y - start\_y, end\_x - start\_x));

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

n = sc.nextInt();

a = new int[n+1][n+1];

v = new int[n+1][n+1];

rets = new ArrayList<>();

int cnt = 0;

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

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

a[i][j] = sc.nextInt();

}

}

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

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

if(a[i][j] > 0 && v[i][j] == 0) {

cnt++;

bfs(i, j);

}

}

}

Collections.sort(rets);

System.out.print("#" + t + " " + cnt + " ");

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

System.out.print(rets.get(i).col + " " + rets.get(i).row + " ");

}

System.out.println();

}

}

}

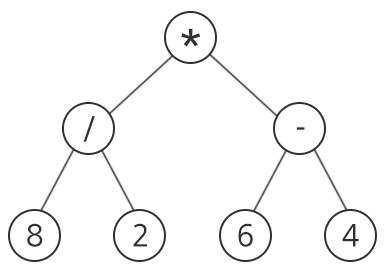
[[SWEA] 1233. [S/W 문제해결 기본] 9일차 - 사칙연산 유효성 검사](https://dheldh77.tistory.com/entry/SWEA-1233-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-9%EC%9D%BC%EC%B0%A8-%EC%82%AC%EC%B9%99%EC%97%B0%EC%82%B0-%EC%9C%A0%ED%9A%A8%EC%84%B1-%EA%B2%80%EC%82%AC)

2020. 3. 9. 10:48

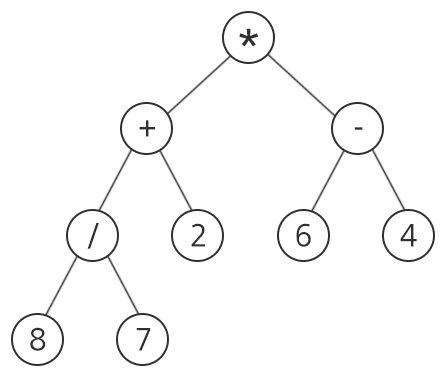
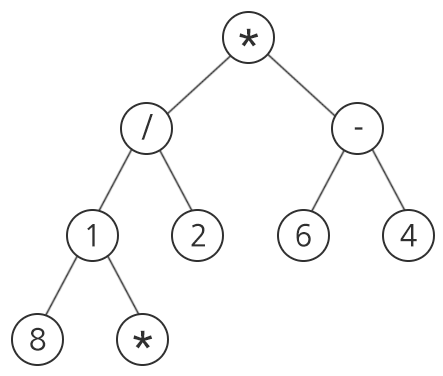
[1233](https://dheldh77.tistory.com/tag/1233), [DFS](https://dheldh77.tistory.com/tag/DFS), [SWEA](https://dheldh77.tistory.com/tag/SWEA), [깊이우선탐색](https://dheldh77.tistory.com/tag/%EA%B9%8A%EC%9D%B4%EC%9A%B0%EC%84%A0%ED%83%90%EC%83%89), [후위표기법](https://dheldh77.tistory.com/tag/%ED%9B%84%EC%9C%84%ED%91%9C%EA%B8%B0%EB%B2%95)

**문제**

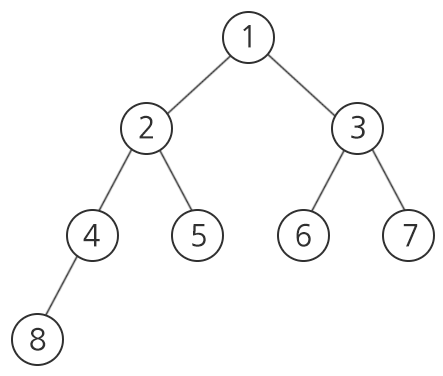
사칙연산으로 구성되어 있는 식은 이진 트리로 표현할 수 있다.  
  
아래는 식 “(8/2)\*(6-4)”을 이진 트리로 표현한 것이다.  
  
임의의 정점에 연산자가 있으면 해당 연산자의 왼쪽 서브 트리의 결과와 오른쪽 서브 트리의 결과를 사용해서 해당 연산자를 적용한다.



사칙연산 “+, -, \*, /”와 양의 정수로만 구성된 임의의 이진 트리가 주어질 때, 이 식의 유효성을 검사하는 프로그램을 작성하여라.  
  
여기서 말하는 유효성이란, 사칙연산 “+, -, \*, /”와 양의 정수로 구성된 임의의 식이 적절한 식인지를 확인하는 것으로, 계산이 가능하다면 “1”, 계산이 불가능할 경우 “0”을 출력한다.  
  
(단, 계산이 가능한지가 아닌 유효성을 검사하는 문제이므로 **0으로 나누는 경우는 고려하지 않는다.** )

연산이 가능한 경우연산이 불가능한 경우

**[제약 사항]**  
  
총 10개의 테스트 케이스가 주어진다.  
  
총 노드의 개수는 200개를 넘어가지 않는다.  
  
트리는 완전 이진 트리 형식으로 주어지며, 노드당 하나의 연산자 또는 숫자만 저장할 수 있다.  
  
노드는 아래 그림과 같은 숫자 번호대로 주어진다.



**풀이방법**

DFS(깊이 우선 탐색) 알고리즘으로 후위 순회를 하면서

노드의 값이 피연산자이면 스택에 담고

연산자이면 스택에 담겨진 두개의 피연산자를 뽑아 연산 후 다시 스택에 담는다.

이 때, 각 과정에서 연산이 유효한지 검토한다.

1) 중위 순회로 방문한 노드가 연산자일 때, 스택에 담겨진 피연산자가 2개 이상인지 확인한다.

2) 모든 노드를 방문했을 때, 스택의 크기가 1인지 확인한다.

3) 0으로 나누는 경우는 유효하지 않은 식으로 판단하지 않으므로 나누기나 나머지 연산자에서 0으로 나눌 경우 대비한다.

각 조건을 만족하는 식이 유효한 식이다. (식의 결과 값자체는 중요하지 않으므로)

**소스코드**

package samsung;

import java.util.\*;

import java.io.\*;

public class s\_1233 {

static int[] visit;

static int[][] edge;

static String[] node;

static int n;

static Stack <Integer> st;

static int flag;

public static void op(String s) {

if(s.equals("+") || s.equals("-") || s.equals("\*") || s.equals("/")) {

if(st.size() < 2) {

flag = 0;

return;

}

int a = st.pop();

int b = st.pop();

if(s.equals("+")) {

st.push(b+a);

}

if(s.equals("-")) {

st.push(b-a);

}

if(s.equals("\*")) {

st.push(b\*a);

}

if(s.equals("/")) {

if(a == 0) {

a = 1;

}

st.push(b/a);

}

}

else {

st.push(Integer.parseInt(s));

}

}

public static void dfs(int x) {

if(flag == 0)

return;

visit[x] = 1;

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

if(edge[x][i] == 1 && visit[i] == 0) {

dfs(i);

}

}

op(node[x]);

}

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

for(int t = 1; t <= 10; t++) {

n = Integer.parseInt(br.readLine());

node = new String[n+1];

edge = new int[n+1][n+1];

visit = new int[n+1];

st = new Stack<>();

flag = 1;

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

StringTokenizer tk = new StringTokenizer(br.readLine());

int from = Integer.parseInt(tk.nextToken());

node[i] = tk.nextToken();

for(int j = 0; j < 2; j++) {

if(tk.hasMoreTokens()) {

int to = Integer.parseInt(tk.nextToken());

edge[from][to] = 1;

}

}

}

dfs(1);

if(st.size() < 1)

flag = 0;

System.out.println("#" + t + " " + flag);

}

}

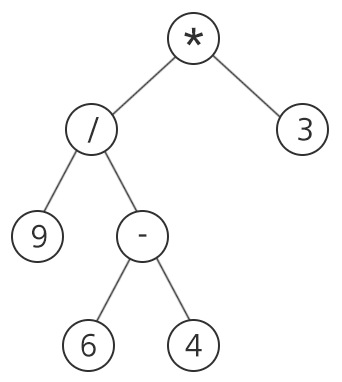
}

[[SWEA] 1232. [S/W 문제해결 기본] 9일차 - 사칙연산](https://dheldh77.tistory.com/entry/SWEA-1232-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-9%EC%9D%BC%EC%B0%A8-%EC%82%AC%EC%B9%99%EC%97%B0%EC%82%B0)

2020. 3. 7. 12:06

**문제**

사칙연산으로 구성되어 있는 식은 이진 트리로 표현할 수 있다. 아래는 식 “(9/(6-4))\*3”을 이진 트리로 표현한 것이다.  
  
임의의 정점에 연산자가 있으면 해당 연산자의 왼쪽 서브 트리의 결과와 오른쪽 서브 트리의 결과를 사용해서 해당 연산자를 적용한다.



사칙연산 “+, -, \*, /”와 양의 정수로만 구성된 임의의 이진트리가 주어질 때, 이를 계산한 결과를 출력하는 프로그램을 작성하라.  
  
단, 중간 과정에서의 연산은 실수 연산으로 하되, 최종 결과값이 정수로 떨어지지 않으면 정수부만 출력한다.  
  
위의 예에서는 최종 결과값이 13.5이므로 13을 출력하면 된다.

**풀이방법**

DFS(깊이 우선 탐색) 알고리즘으로 후위 순회를 하면서

노드의 값이 피연산자이면 스택에 담고

연산자이면 스택에 담겨진 두개의 피연산자를 뽑아 연산 후 다시 스택에 담는다.

탐색을 종료하고 스택에 남은 최종 값이 연산의 결과가 된다.

**소스코드**

package samsung;

import java.io.\*;

import java.util.\*;

public class s\_1232 {

static String[] vertex;

static int[][] edge;

static int n;

static int[] visit;

static Stack <Integer> nums;

public static void op(String s) {

if(s.equals("+")) {

int a = nums.pop();

int b = nums.pop();

nums.push(b+a);

}

else if(s.equals("-")) {

int a = nums.pop();

int b = nums.pop();

nums.push(b-a);

}

else if(s.equals("\*")) {

int a = nums.pop();

int b = nums.pop();

nums.push(b\*a);

}

else if(s.equals("/")) {

int a = nums.pop();

int b = nums.pop();

nums.push(b/a);

}

else {

int num = Integer.parseInt(s);

nums.push(num);

}

}

public static void dfs(int x) {

visit[x] = 1;

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

if(edge[x][i] == 1 && visit[i] == 0)

dfs(i);

}

op(vertex[x]);

}

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

for(int t = 1; t <= 10; t++) {

n = Integer.parseInt(br.readLine());

edge = new int[n + 1][n + 1];

vertex = new String[n+1];

visit = new int[n + 1];

nums = new Stack<>();

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

StringTokenizer tk = new StringTokenizer(br.readLine());

int from = Integer.parseInt(tk.nextToken());

vertex[from] = tk.nextToken();

for(int j = 0; j < 2; j++) {

if(tk.hasMoreTokens()) {

int to = Integer.parseInt(tk.nextToken());

edge[from][to] = 1;

}

}

}

dfs(1);

System.out.println("#" + t + " " + nums.pop());

}

}

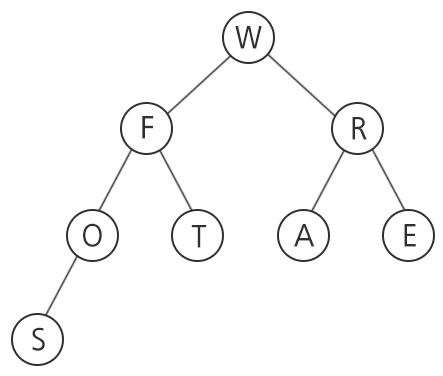
}

[[SWEA] 1231. [S/W 문제해결 기본] 9일차 - 중위순회](https://dheldh77.tistory.com/entry/SWEA-1231-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-9%EC%9D%BC%EC%B0%A8-%EC%A4%91%EC%9C%84%EC%88%9C%ED%9A%8C)

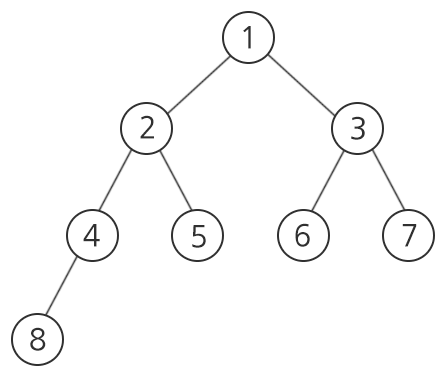
2020. 3. 7. 12:02

**문제**

다음은 특정 단어(또는 문장)를 트리 형태로 구성한 것으로, in-order 형식으로 순회하여 각 노드를 읽으면 원래 단어를 알 수 있다고 한다.



위 트리를 in-order 형식으로 순회할 경우 SOFTWARE 라는 단어를 읽을 수 있다.  
  
**[제약 사항]**  
  
총 10개의 테스트 케이스가 주어진다.  
  
총 노드의 개수는 100개를 넘어가지 않는다.  
  
트리는 완전 이진 트리 형식으로 주어지며, 노드당 하나의 알파벳만 저장할 수 있다.  
  
노드가 주어지는 순서는 아래 그림과 같은 숫자 번호대로 주어진다.



**풀이방법**

중위 순회 방식은 왼쪽의 자식노드 -> 부모노드 -> 오른쪽 자식노드 순으로 방문하는 알고리즘이다.

입력이 너비를 기준으로 1, 2, 3, 4,...로 입력되었기 때문에

노드에 자식노드를 가르키는 포인터 정보 없이 배열로 받아 인덱스와 재귀로만 해결할 수 있다.

n노드가 가르키는 왼쪽 자식노드는 n \* 2이고, 오른쪽 자식노드는 n\*2 + 1이 된다.

따라서 n\*2 노드 -> n노드 -> n\*2+1노드 순으로 방문하면 된다.

**소스코드**

package samsung;

import java.util.\*;

import java.io.\*;

public class s\_1231 {

static char[] a;

static int n;

public static void inOrder(int idx) {

if(idx > n) return;

inOrder(2 \* idx);

System.out.print(a[idx]);

inOrder(2 \* idx + 1);

}

public static void main(String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

for(int t = 1; t <= 10; t++) {

n = Integer.parseInt(br.readLine());

a = new char[n + 1];

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

StringTokenizer tk = new StringTokenizer(br.readLine());

tk.nextToken();

a[i] = tk.nextToken().charAt(0);

}

System.out.print("#" + t + " ");

inOrder(1);

System.out.println();

}

}

}

## [[SWEA] 4613. 러시아 국기 같은 깃발](https://dheldh77.tistory.com/entry/4613-%EB%9F%AC%EC%8B%9C%EC%95%84-%EA%B5%AD%EA%B8%B0-%EA%B0%99%EC%9D%80-%EA%B9%83%EB%B0%9C)

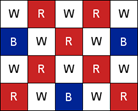
2020. 3. 7. 11:50

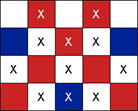
### 문제

2016년은 삼성전자가 러시아 현지법인을 설립한지 20주년이 된 해이다. 이를 기념해서 당신은 러시아 국기를 만들기로 했다.  
  
먼저 창고에서 오래된 깃발을 꺼내왔다. 이 깃발은 N행 M열로 나뉘어 있고, 각 칸은 흰색, 파란색, 빨간색 중 하나로 칠해져 있다.  
  
당신은 몇 개의 칸에 있는 색을 다시 칠해서 이 깃발을 러시아 국기처럼 만들려고 한다. 다음의 조건을 만족해야 한다.

* 위에서 몇 줄(한 줄 이상)은 모두 흰색으로 칠해져 있어야 한다.
* 다음 몇 줄(한 줄 이상)은 모두 파란색으로 칠해져 있어야 한다.
* 나머지 줄(한 줄 이상)은 모두 빨간색으로 칠해져 있어야 한다.

이렇게 러시아 국기 같은 깃발을 만들기 위해서 새로 칠해야 하는 칸의 개수의 최솟값을 구하여라.







첫 번째 예제이다. 왼쪽에 있는 그림이 입력이다. 중간에 있는 그림에 X가 적힌 칸들을 새롭게 색칠하여 오른쪽에 있는 그림과 같은 깃발을 만들면 최적이다.

### 풀이방법

동적계획법(DP, Dynamic Programming) 알고리즘을 활용해 문제를 해결한다.

흰색, 파란색, 빨간색 중에서 유동적으로 변하지 않는 값(색을 새로 칠해야하는 칸의 수)을 찾는다.

흰색의 같은 경우 0번째 행에서 시작하므로 행을 1개, 2개, 3개 차지할 때 값이 변하지 않는다.

빨간색의 경우도 마찬가지로 n-1번짹 행에서부터 칠해지므로 행을 1개, 2개, 3개 차지할 때 값이 유동적으로 변하지 않는다.

다만, 파란색의 경우 흰색의 행의 수에 따라 값이 유동적으로 변하게 된다.

파란색을 제외하고는 빨간색과 흰색은 여러 줄의 차지할 때 그 값이 고정이 된다.

따라서, 빨간색과 흰색이 각 각n, m줄일 때, 새로 칠해야하는 칸의 수를 미리 저장한다.

총 새로 칠해야하는 칸의 수는 흰색을  위에서부터 몇 줄 칠하고, 빨간색을 아래서부터 몇 줄 칠했을 때,

나머지 가운데 부분을 빨간색으로 칠했을 때 새로 칠하는 칸의 수를 구하면 된다.

이 때, 흰색의 몇줄, 빨간색의 몇줄은 미리 저장해서 알기 때문에, 파란색을 칠해야하는 수만 매 사이클마다 찾아주면 된다.

흰색과 빨간색이 1, 2, 3, ..., n-2행을 차지할 때의 값을 ws[m], rs[m]이라고 하고 파란색의 각 행을 새로 칠하는 수를 b[n]이라고 할 때,

총 새로 칠해야하는 칸의 개수는 ws[i] + rs[j] + r[i+1] + r[i+2] + ... + r[n-j]가 된다.

즉, 흰색을 i줄 칠하고, 빨간색을 j줄 칠했을 때, i+1부터 n-j줄까지는 파란색으로 칠하는 것이 된다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_4613 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

int n = sc.nextInt();

int m = sc.nextInt();

char[][] a = new char[n][m];

int[] w = new int[n];

int[] b = new int[n];

int[] r = new int[n];

int[] ws = new int[n - 2];

int[] rs = new int[n - 2];

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

String s = sc.next();

int w\_cnt = 0;

int b\_cnt = 0;

int r\_cnt = 0;

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

a[i][j] = s.charAt(j);

if(a[i][j] != 'W') w\_cnt++;

if(a[i][j] != 'B') b\_cnt++;

if(a[i][j] != 'R') r\_cnt++;

}

w[i] = w\_cnt;

b[i] = b\_cnt;

r[i] = r\_cnt;

}

ws[0] = w[0];

rs[0] = r[n-1];

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

ws[i] = ws[i - 1] + w[i];

rs[i] = rs[i - 1] + r[n - 1 - i];

}

int result = n \* m;

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

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

int tmp = 0;

tmp += ws[i];

tmp += rs[j];

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

tmp += b[k];

}

result = Math.min(result, tmp);

}

}

System.out.println("#" + t + " " + result);

}

}

}

## [[SWEA] 7854. 최약수](https://dheldh77.tistory.com/entry/SWEA-7854-%EC%B5%9C%EC%95%BD%EC%88%98)

2020. 3. 7. 11:10

### 문제

자연수 A가 자연수 B의 약수이기 위해서는 B가 A로 나누어 떨어져야 한다.  
  
초등학생인 정우는 오늘 배운 약수라는 개념에 홀딱 반해버렸다.  
  
집에 가는 길에 계속 약수만 떠올리던 정우는 엄청난 사실을 발견했다!  
  
숫자 2는 2 앞에 어떤 숫자를 붙여도 2를 약수로 가졌다.  
  
5 역시 5 앞에 어떤 숫자를 붙여도 5를 약수로 가졌다.  
  
예를 들어 숫자 3827를 5 앞에 붙이면 38275가 되며, 이는 5를 약수로 가진다.  
  
정우는 이런 특별한 성질을 가지는 자연수를 최약수 라고 명명했다.  
  
즉, 최약수란 앞에 어떤 숫자를 붙여도 자신을 약수로 가져야 한다.  
  
정우는 최약수의 종류가 궁금했다!  
  
정우를 도와서 주어지는 숫자 X 보다 작거나 같은 최약수의 개수를 구해주자.

### 풀이방법

문제에서 말하는 최약수는 한 자리 높은 수의 값을 1로 만들 수 있으면 된다.

한 자리 높은 수의 값을 1을 만든다는 말은 최약수 x가 있을 때,

최약수 x는 1x가 될 수 있다. 또한 최약수 x는 2x가 될 수도 있다. 같은 방식으로 3x, 4x, ..., 3827x까지 가능하다.

이러한 방식으로 10, 100, 1000, 10000을 만들 수 있는 최약수를 찾는다.

10을 만들 수 있는 최약수 x : {1, 2, 5} - 3개

100을 만들 수 있는 최약수 x : {10, 20, 25, 50} - 4개

1000을 만들 수 있는 최약수 x {100, 125, 200, 250, 500} - 5개

1000이상부터는 최약수의 패턴이 모두 같다.

10000은 1000, 1250, 2000, 2500, 5000,

100000은 10000, 12500, 20000, 25000, 50000...

즉, 주어진 입력 x가

1자리 일 때는 a

2자리일 때는 3 + a

3자리일 때는 3 + 4 + a

4자리일 때는 3 + 4 + 5 + a

n(n >= 3)자리일 때는 3 + 4 + 5(n - 3) + a이다.

a는 x가 n자리 일때, n자리 중 최약수의 개수이다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_7854 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int[] one = {1, 2, 5};

int[] two = {20, 25, 50, 100};

int[] three = {100, 125, 200, 250, 500};

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

int cnt = 0;

String s = sc.next();

int n = s.length();

if(n == 1) {

int num = Integer.parseInt(s);

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

if(num >= one[i]) cnt++;

else break;

}

}

else if(n == 2) {

cnt += 3;

int num = Integer.parseInt(s);

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

if(num >= two[i]) cnt++;

else break;

}

}

else {

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

if(i == 0) cnt += 3;

else if(i == 1) cnt += 4;

else cnt += 5;

}

String tmp = s.substring(0, 3);

int num = Integer.parseInt(tmp);

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

if(num >= three[i]) cnt++;

else break;

}

}

System.out.println("#" + t + " " + cnt);

}

}

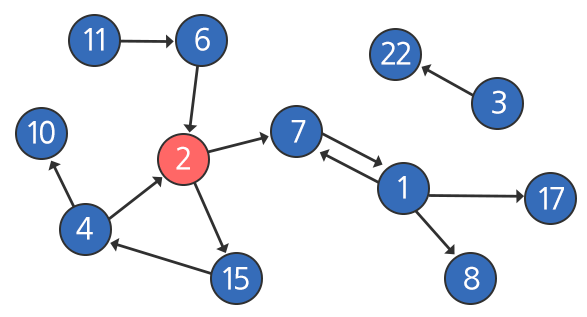
}

## [[SWEA] 1238. [S/W 문제해결 기본] 10일차 - Contact](https://dheldh77.tistory.com/entry/SWEA-1238-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-10%EC%9D%BC%EC%B0%A8-Contact)

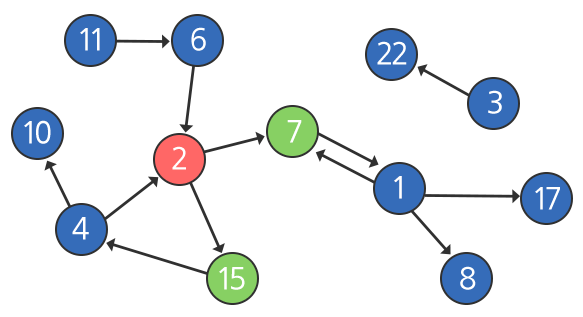
2020. 3. 6. 10:53

### 문제

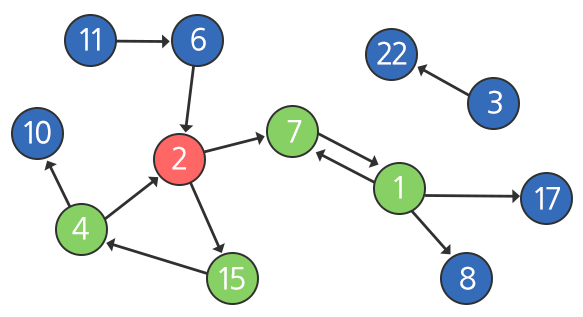
비상연락망과 연락을 시작하는 당번에 대한 정보가 주어질 때, 가장 나중에 연락을 받게 되는 사람 중 번호가 가장 큰 사람을 구하는 함수를 작성하시오.  
   
**[예시]**  
  
아래는 비상연락망을 나타낸 그림이다.



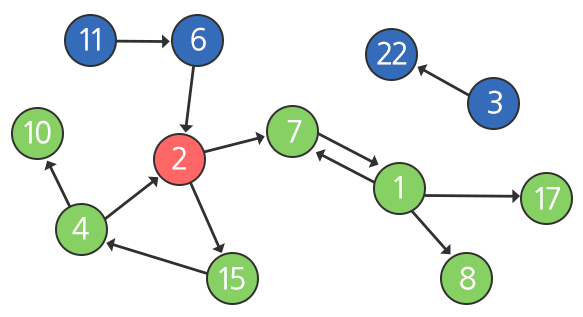
각 원은 개개인을 의미하며, 원 안의 숫자는 그사람의 번호를 나타내고 빨간원은 연락을 시작하는 당번을 의미한다.  
  
화살표는 연락이 가능한 방향을 의미한다.  
  
위의 예시에서는 7번과 1번은 서로 연락이 가능하다,  
  
하지만 2번과 7번의 경우 2번은 7번에게 연락할 수 있지만 7번은 2번에게 연락할 수 없다.  
   
비상연락망이 가동되면 아래 그림과 같이 연락을 시작하는 당번인 2번은 연락 가능한 7번과 15번에 동시에 연락을 취한다 (다자 간 통화를 사용한다고 가정).



그 다음 아래와 같이 7번은 1번에게, 15번은 4번에게 연락을 취한다 (이 과정은 동시에 일어난다고 가정한다).



그 다음 아래와 같이 1번은 8번과 17번에게 동시에 연락하며, 이와 동시에 4번은 10번에게 연락한다.  
  
7번과 2번의 경우는 이미 연락을 받은 상태이기 때문에 다시 연락하지 않는다.



위의 모습이 연락이 끝난 마지막 모습이 되며, 마지막에 동시에 연락 받은 사람은 8번, 10번, 17번의 세 명이다.  
  
이 중에서 가장 숫자가 큰 사람은 17번이므로 17을 반환하면 된다.  
   
※ 3, 6, 11, 22번은 시간이 지나도 연락을 받지 못한다.

### 풀이방법

BFS(너비 우선 탐색)을 사용해서 문제를 해결한다.

최대 깊이의 단말 노드 중에서 가장 큰 값을 찾는 것이다. 문제를 예시로 들면,

각 깊이에서 노드는 아래와 같다.

깊이 1 : 2

깊이 2 : 7, 15

깊이 3: 4, 1

깊이 4 : 10, 8, 17

최대 깊이는 4이고, 깊이 4에서 단말 노드 중 가장 큰 노드의 값은 17이다.

이 문제를 푸는데 두 가지 신경써야할 부분이 있었다.

첫 번째는 입력을 받는 방법이다. 문제에서 노드 간의 간선의 개수에 대한 정보는 주어지지 않는다.

한 줄에 모두 입력되는데, 이 한줄을 모두 입력받고 공백을 단위로 from과 to를 저장해야된다는 말이다.

보통 다른 문제를 해결할 때는 입력을 받을 때 Scanner 클래스를 사용해서 입력을 받았지만,

공백을 포함한 한 줄을 입력받기 위해 BufferedReader 클래스를 사용해 입력받았다.

혹시, Scanner 클래스를 사용해서 공백을 포함하는 라인을 입력받는 방법을 아시는 분은 알려주시기 바랍니다.

두 번째는 단말 노드에 대한 정보이다.  처음에는 모든 단말 노드를 다 찾고 이 중에서 최대값을 구했다.

단말 노드마다 깊이가 다르다는 점을 간과한 것이다.

문제에서 요구하는 것은 최대 깊이에 있는 단말노드 중 가장 큰 값을 원한다.

따라서 노드에 대한 깊이도 같이 저장할 필요가 있었다.

### 소스코드

package samsung;

import java.util.\*;

import java.io.\*;

public class s\_1238 {

public static class Node{

int depth;

int v;

Node(int depth, int v){

this.depth = depth;

this.v = v;

}

}

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

for(int t = 1; t <= 10; t++) {

StringTokenizer tk = new StringTokenizer(br.readLine());

int n = Integer.parseInt(tk.nextToken());

int first = Integer.parseInt(tk.nextToken());

int[][] a = new int[n+1][n+1];

int[] visit = new int[n+1];

Queue <Node> q = new LinkedList<>();

ArrayList <Node> result = new ArrayList<>();

int max\_depth = 1;

tk = new StringTokenizer(br.readLine());

while(tk.hasMoreTokens()) {

int from = Integer.parseInt(tk.nextToken());

int to = Integer.parseInt(tk.nextToken());

a[from][to] = 1;

}

q.add(new Node(1, first));

result.add(new Node(1, first));

visit[first] = 1;

while(!q.isEmpty()) {

Node node = q.poll();

int v = node.v;

int depth = node.depth;

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

if(a[v][i] == 1 && visit[i] == 0) {

visit[i] = 1;

result.add(new Node(depth + 1, i));

q.add(new Node(depth+1, i));

}

}

max\_depth = Math.max(max\_depth, depth);

}

int max = 1;

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

if(result.get(i).depth == max\_depth) {

max = Math.max(max, result.get(i).v);

}

}

System.out.println("#" + t + " " + max);

}

}

}

## [[SWEA] 6719. 성수의 프로그래밍 강좌 시청](https://dheldh77.tistory.com/entry/SWEA-6719-%EC%84%B1%EC%88%98%EC%9D%98-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%EA%B0%95%EC%A2%8C-%EC%8B%9C%EC%B2%AD)

2020. 3. 6. 10:40

### 문제

성수는 이제 프로그래밍을 시작하기로 마음 먹은 초보다.  
  
그렇기에 프로그래밍 강좌를 통해 자신의 프로그래밍 실력을 끌어 올리려고 한다.  
  
성수의 실력이 A라고 할 때, 수준이 M인 강좌를 시청하고 나면 성수의 실력은 (A+M)/2가 된다.  
  
즉, 성수는 자신이 보는 강좌가 좋은 지 아닌지 판단하지 않고 그대로 강좌를 받아들이기 때문에,  
  
실력보다 낮은 수준의 강좌를 보면 실력이 낮아질 수 있다.  
  
현재 성수는 아직 아무런 실력이 없다. 즉 실력이 0이다.  
  
성수는 볼 수 있는 강좌 총 N개 찾았고 시간 문제상 이 중에서 K개를 적절한 순서로 선택해 한 번씩 시청하려고 한다.  
  
성수가 같은 강좌를 두 번 이상 보는 일은 없다고 할 때, 성수가 가질 수 있는 실력의 수치는 최대 몇인지 구하는 프로그램을 작성하라.

### 풀이방법

그리디 알고리즘으로 문제를 해결한다.

성수의 실력을 가장 많이 늘릴 수 있는 방법은 ,

N개의 강좌 중 가장 높은 K개의 강좌를 고르고, 이 중에서 낮은 순서대로 강좌를 듣는 것이다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_6719 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

int n = sc.nextInt();

int k = sc.nextInt();

double r = 0;

ArrayList <Integer> a = new ArrayList<>();

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

int tmp = sc.nextInt();

a.add(tmp);

}

Collections.sort(a);

for(int i = a.size() - k; i < a.size(); i++) {

r = (r + a.get(i)) / 2;

}

System.out.printf("#%d %.6f\n", t, r);

}

}

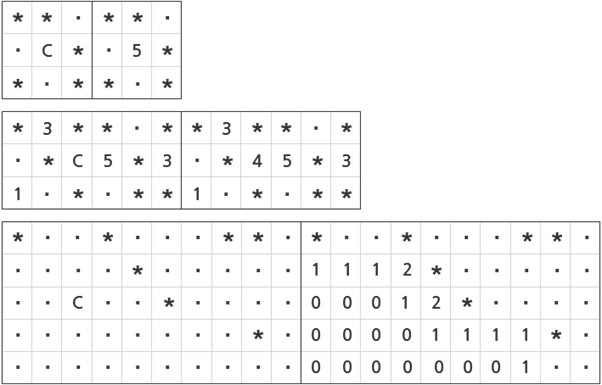
}

## [[SWEA] 1868. 파핑파핑 지뢰찾기](https://dheldh77.tistory.com/entry/SWEA-1486-%EC%9E%A5%ED%9B%88%EC%9D%B4%EC%9D%98-%EB%86%92%EC%9D%80-%EC%84%A0%EB%B0%98-1)

2020. 3. 6. 10:34

### 문제

‘파핑 파핑 지뢰 찾기’라는 유명한 게임이 있다. 이 게임은 RXC 크기의 표를 이용하는 게임인데,  
  
표의 각 칸에는 지뢰가 있을 수도 있고 없을 수도 있다.  
  
표의 각 칸을 클릭했을 때, 그 칸이 지뢰가 있는 칸이라면 ‘파핑 파핑!’이라는 소리와 함께 게임은 끝난다.  
  
지뢰가 없는 칸이라면 변이 맞닿아 있거나 꼭지점이 맞닿아 있는 최대 8칸에 대해 몇 개의 지뢰가 있는지가 0에서 8사이의 숫자로 클릭한 칸에 표시된다.  
  
만약 이 숫자가 0이라면 근처의 8방향에 지뢰가 없다는 것이 확정된 것이기 때문에 그 8방향의 칸도 자동으로 숫자를 표시해 준다.  
  
실제 게임에서는 어떤 위치에 지뢰가 있는지 알 수 없지만, 이 문제에서는 특별히 알 수 있다고 하자.  
  
지뢰를 ‘\*’로, 지뢰가 없는 칸을 ‘.’로, 클릭한 지뢰가 없는 칸을 ‘c’로 나타냈을 때 표가 어떻게 변화되는지 나타낸다.



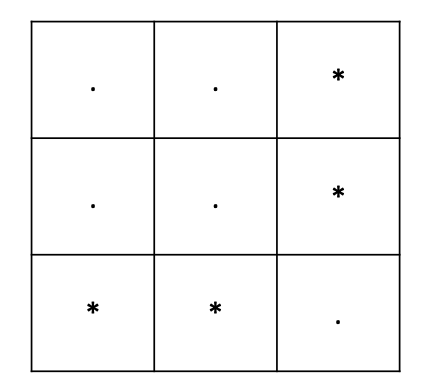
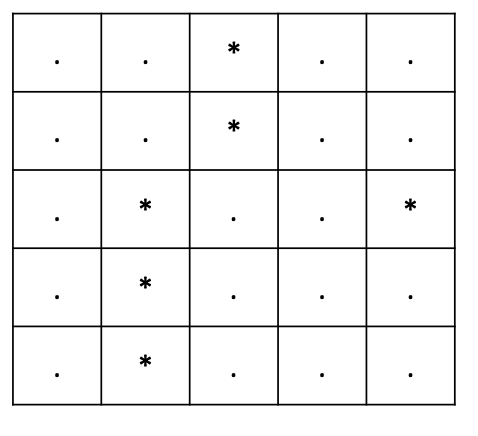
세 번째 예에서는 0으로 표시 될 칸들과 이와 인접한 칸들이 한 번의 클릭에 연쇄적으로 숫자가 표시된 것을 볼 수 있다.  
  
파핑 파핑 지뢰 찾기를 할 때 표의 크기와 표가 주어질 때, 지뢰가 있는 칸을 제외한 다른 모든 칸의 숫자들이 표시되려면 최소 몇 번의 클릭을 해야 하는지 구하는 프로그램을 작성하라.

### 풀이방법

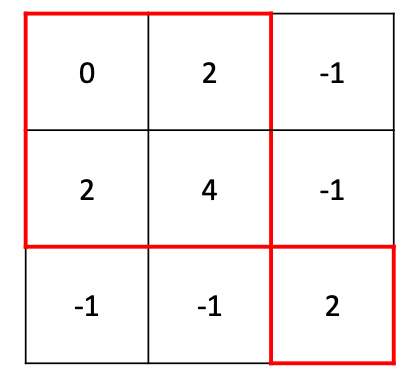
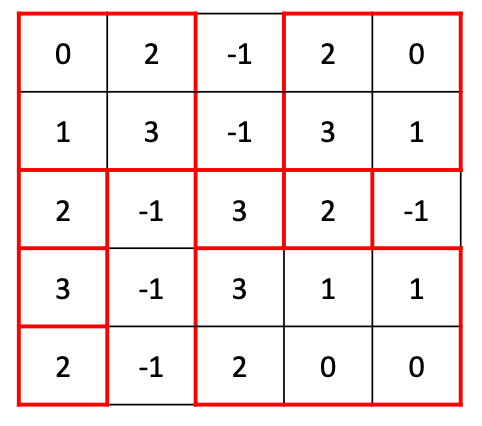
높이 우선 탐색(DFS)로 영역을 모두 찾는 문제이다.

예시를 들어서 풀이 방법을 보면,

입력이 아래와 같은 때,

case 1case 2

지뢰에 대한 정보와 이에 따른 최소 클릭 횟수는 아래와 같다.

case 1case 2

0에 인접한 8방향의 노드가 한 영역이 되고 0이 없을 때는 지뢰가 아닌 노드 하나하나가 영역이 된다.

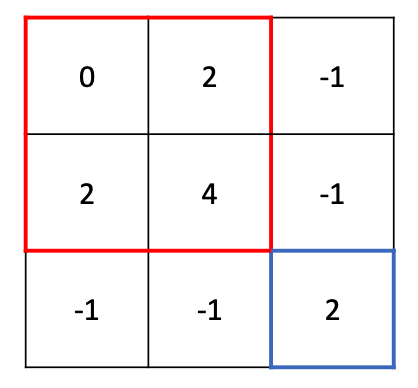
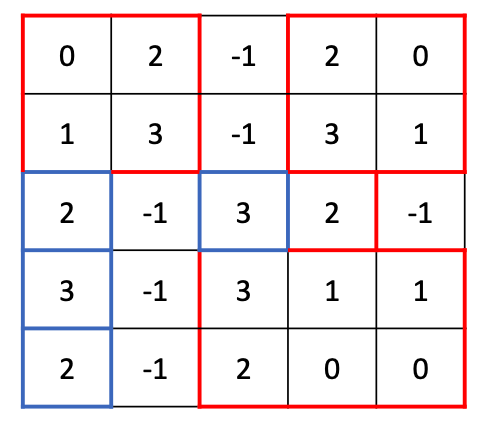
또한, 영역의 개수는 최소 클릭 횟수이다. 즉, case1 은 최소 클릭 횟수가 2가 되고, case2는 8이 된다.(빨간 사각형의 개수)

따라서 깊이 우선 탐색을 할 때, 원소값이 0인 정점을 시작으로 인접한 8방향의 정점을 탐색한다.

인접한 정점 중 원소값이 0인 정점이 있다면 계속해서 탐색을 한다.

원소값이 0인 정점 시작으로 깊이 우선 탐색을 모두 탐색했다면, 아래와 같이 사각형을 찾은 것이다.

빨간색 사각형은 깊이 우선 탐색으로 찾은 사각형, 파란색 사각형은 앞으로 찾아야할 사각형이다.

case 1case 2

이후, 원소 값이 양수이면서, 방문하지 않은 정점들을 모두 찾아 깊이 우선 탐색을 한 횟수에 더해주면 최소 클릭 횟수가 된다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_1868 {

static int[] dx = {-1, 0, 1, 0, -1, -1, 1, 1};

static int[] dy = {0, 1, 0, -1, 1, -1, -1, 1};

static int[][] a;

static int[][] v;

static int n;

public static void dfs(int x, int y) {

for(int i = 0; i < 8; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < n && ny < n) {

if(v[nx][ny] == 0 && a[nx][ny] > 0)

v[nx][ny] = 1;

else if(v[nx][ny] == 0 && a[nx][ny] == 0) {

v[nx][ny] = 1;

dfs(nx,ny);

}

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

n = sc.nextInt();

char[][] map = new char[n][n];

a = new int[n][n];

v = new int[n][n];

int area = 0;

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

String s = sc.next();

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

map[i][j] = s.charAt(j);

}

}

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

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

if(map[i][j] == '\*') {

a[i][j] = -1;

v[i][j] = 1;

continue;

}

int cnt = 0;

for(int k = 0; k < 8; k++) {

int nx = i + dx[k];

int ny = j + dy[k];

if(nx >= 0 && ny >= 0 && nx < n && ny < n) {

if(map[nx][ny] == '\*') cnt++;

}

}

a[i][j] = cnt;

}

}

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

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

if(a[i][j] == 0 && v[i][j] == 0) {

area++;

v[i][j] = 1;

dfs(i, j);

}

}

}

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

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

if(v[i][j] == 0) area++;

}

}

System.out.println("#" + t + " " + area);

}

}

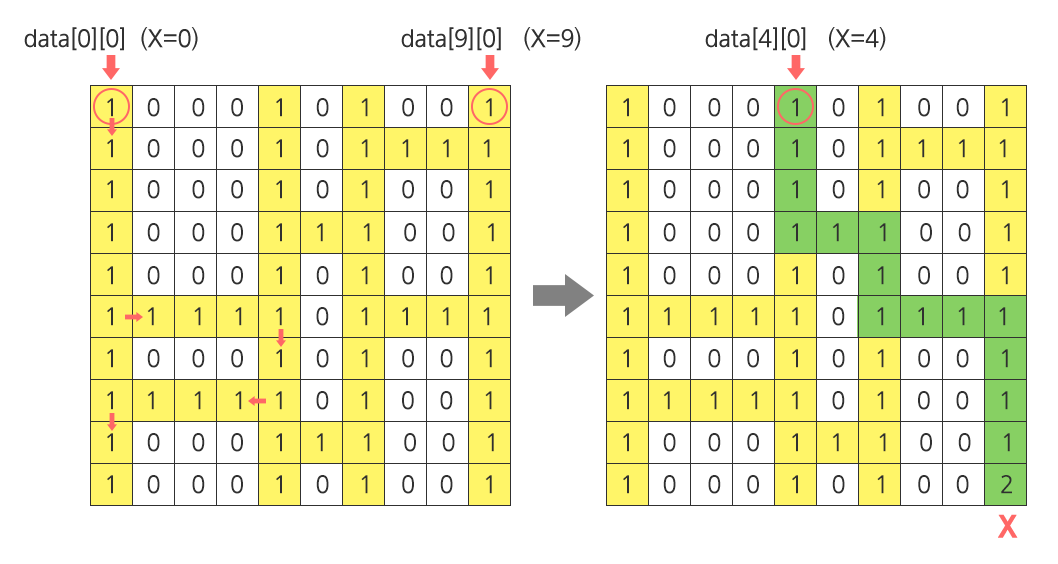
}

## [[SWEA] 1210. [S/W 문제해결 기본] 2일차 - Ladder1](https://dheldh77.tistory.com/entry/SWEA-1210-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-2%EC%9D%BC%EC%B0%A8-Ladder1)

2020. 3. 6. 09:53

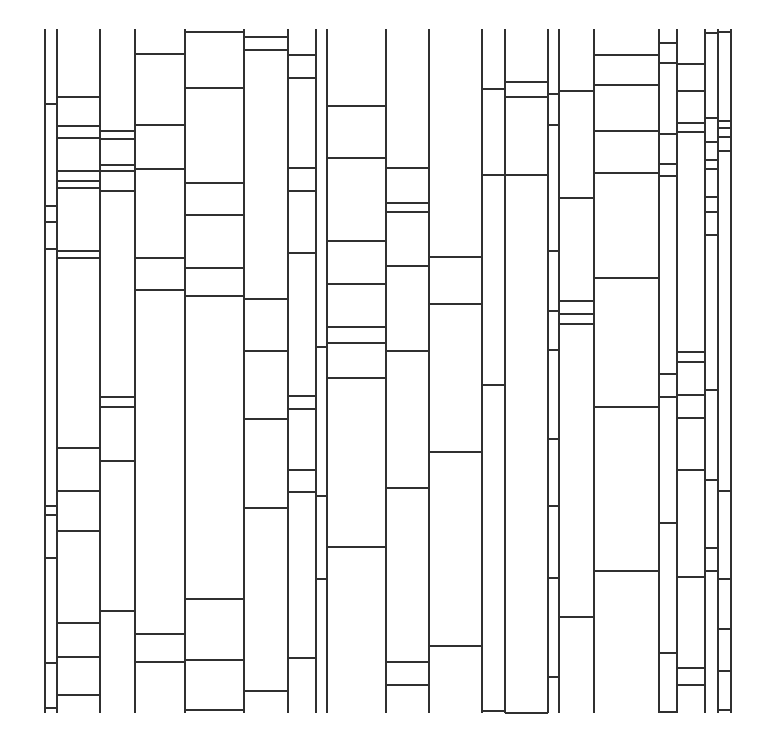
### 문제

점심 시간에 산책을 다니는 사원들은 최근 날씨가 더워져, 사다리 게임을 통하여 누가 아이스크림을 구입할지 결정하기로 한다.  
  
김 대리는 사다리타기에 참여하지 않는 대신 사다리를 그리기로 하였다.  
  
사다리를 다 그리고 보니 김 대리는 어느 사다리를 고르면 X표시에 도착하게 되는지 궁금해졌다. 이를 구해보자.  
  
아래 <그림 1>의 예를 살펴보면, 출발점 x=0 및 x=9인 세로 방향의 두 막대 사이에 임의의 개수의 막대들이 랜덤 간격으로 추가되고(이 예에서는 2개가 추가됨) 이 막대들 사이에 가로 방향의 선들이 또한 랜덤하게 연결된다.  
  
X=0인 출발점에서 출발하는 사례에 대해서 화살표로 표시한 바와 같이, 아래 방향으로 진행하면서 좌우 방향으로 이동 가능한 통로가 나타나면 방향 전환을 하게 된다.  
  
방향 전환 이후엔 다시 아래 방향으로만 이동하게 되며, 바닥에 도착하면 멈추게 된다.  
  
문제의 X표시에 도착하려면 X=4인 출발점에서 출발해야 하므로 답은 4가 된다. 해당 경로는 별도로 표시하였다.



<그림 1> 사다리 게임에 대한 설명 (미니맵)

아래 <그림 2>와 같은 **100 x 100 크기의 2차원 배열로 주어진 사다리에 대해서, 지정된 도착점에 대응되는 출발점 X를 반환하는 코드를 작성하라**(‘0’으로 채워진 평면상에 사다리는 연속된 ‘1’로 표현된다. 도착 지점은 '2'로 표현된다).



<그림 2> 테스트 케이스에 의해 생성되는 실제 사다리의 모습

### 풀이방법

입력된 사다리의 모양을 2차원 배열에 저장한다.

행을 i, 열을 j, 2차원 배열을 a[i][j]라고 했을 때(0 <= i <= 99 , 0 <= j <= 99),

i == 0 일때, a[i][j] == 1이 되는 원소를 찾는다.

i == 0 일때, a[i][j] == 1이되는 원소를 찾았다면, 이 때, j값을 따로 저장한다.

아래의 3가지 형태로 진행하고, i 가 99가 되었을 때, a[i][j] == 2라면 위에서 따로 저장한 j값을 출력한다.

1) 사다리에서 좌측으로 갈 수 있는 경우( j + 1 == 1 && j + 1 < 100)

   j++

2) 사다리에서 우측으로 갈 수 있는 경우( j - 1 == 1 && j -1 >= 0)

   j--

3) 사다리에서 좌측도 우측도 갈 수 없는 경우. 즉, 아래로 내려가야할 경우

  i++

### 소스코드

package samsung;

import java.util.\*;

public class s\_1210 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int tc = sc.nextInt();

int find = 0;

int[][] a = new int[100][100];

for(int i = 0; i < 100; i++) {

for(int j = 0; j < 100; j++) {

a[i][j] = sc.nextInt();

}

}

for(int i = 0; i < 100; i++) {

if(a[0][i] == 1) {

int x = 0;

int y = i;

int[][] v = new int[100][100];

v[x][y] = 1;

while(true) {

if(y-1 >= 0 && a[x][y-1] == 1 && v[x][y-1] == 0) y--;

else if(y+1 < 100 && a[x][y+1] == 1 && v[x][y+1] == 0) y++;

else x++;

v[x][y] = 1;

if(x == 99) {

if(a[x][y] == 2) {

find = 1;

System.out.println("#" + t + " " + i);

}

break;

}

}

}

if(find == 1) break;

}

}

}

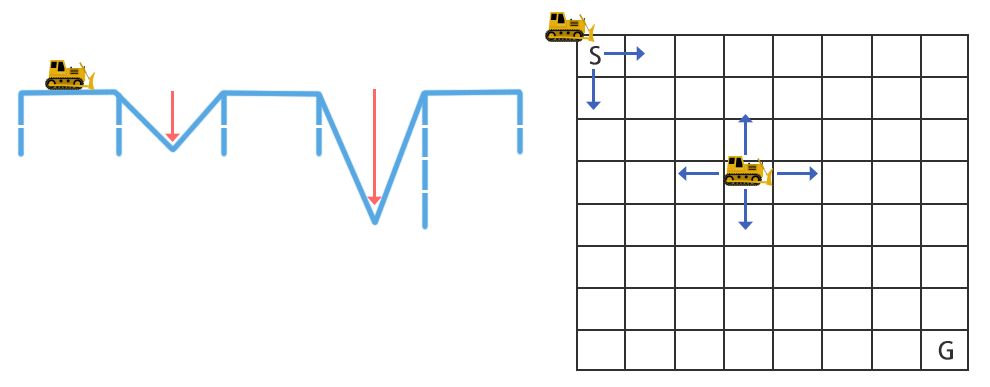
}

## [[SWEA] 1249. [S/W 문제해결 응용] 4일차 - 보급로](https://dheldh77.tistory.com/entry/SWEA-1249-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EC%9D%91%EC%9A%A9-4%EC%9D%BC%EC%B0%A8-%EB%B3%B4%EA%B8%89%EB%A1%9C)

2020. 3. 5. 10:56

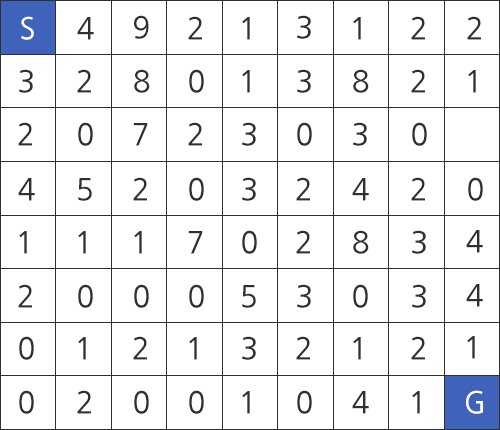
### 문제

2차 세계 대전에서 연합군과 독일군의 전투가 점점 치열해지고 있다.  
  
전투가 진행중인 지역은 대규모 폭격과 시가전 등으로 인해 도로 곳곳이 파손된 상태이다.  
  
그림 1(a)에서와 같이 도로들은 전투로 인해 트럭이나 탱크와 같은 차량들이 지날 갈 수 없다.  
  
전투에서 승리하기 위해서는 기갑사단과 보급부대가 신속하게 이동하기 위한 도로가 있어야 한다.  
  
공병대는 출발지(S) 에서 도착지(G)까지 가기 위한 도로 복구 작업을 빠른 시간 내에 수행하려고 한다.  
  
도로가 파여진 깊이에 비례해서 복구 시간은 증가한다.  
  
출발지에서 도착지까지 가는 경로 중에 복구 시간이 가장 짧은 경로에 대한 총 복구 시간을 구하시오.  
  
깊이가 1이라면 복구에 드는 시간이 1이라고 가정한다.



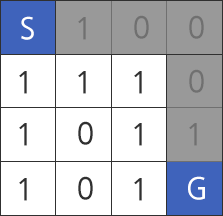
**그림 1 (a) 파손된 도로                                                             (b) 지도 형태와 이동 방향**

지도 정보는 그림1(b)와 같이 2차원 배열 형태로 표시된다.  
  
출발지는 좌상단의 칸(S)이고 도착지는 우하단의 칸(G)가 된다.  
  
이동 경로는 상하좌우 방향으로 진행할 수 있으며, 한 칸씩 움직일 수 있다.  
  
지도 정보에는 각 칸마다 파여진 도로의 깊이가 주어진다. 현재 위치한 칸의 도로를 복구해야만 다른 곳으로 이동할 수 있다.



**그림 2 지도 정보**

이동하는 시간에 비해 복구하는데 필요한 시간은 매우 크다고 가정한다.  
  
따라서, 출발지에서 도착지까지 거리에 대해서는 고려할 필요가 없다.  
  
지도 정보는 그림2에서 보듯이 2차원 배열의 형태이다.  
  
출발지(S)와 도착지(G)는 좌상단과 우하단이 되고 입력 데이터에서는 0으로 표시된다.  
  
출발지와 도착지를 제외한 곳이 0인 것은 복구 작업이 불필요한 곳이다.  
  
다음과 같은 지도에서 복구 작업 시간이 최소인 시간은 2이고 회색으로 칠해진 경로가 된다.



### 풀이방법

너비우선탐색(BFS, Breadth First Search)를 이용해 해결한다.

너비우선탐색으로 최단 경로를 찾는데 한 가지 경우를 고려해주어여한다.

**방문했던 노드라도 새로운 최단경로가 있다면 재탐색해야되는 점이다.**

예를 들어 아래와 같은 입력이 주어졌을 때,

|  |  |  |
| --- | --- | --- |
| 0 | 3 | 0 |
| 5 | 4 | 2 |
| 4 | 1 | 0 |

(0,0)을 시작으로 같은 레벨의 노드로 너비 우선탐색을 시작한다.

|  |  |  |
| --- | --- | --- |
| 0 | 3 |  |
| 5 |  |  |
|  |  |  |

(0,0)에 인접한 노드((1,0), (0,1))를 방문하고 큐에 담는다.

|  |  |  |
| --- | --- | --- |
| 0 | 3 |  |
| 5 | 9 |  |
| 9 |  |  |

(1,0)에 인접한 노드((2,0), (1,1))를 방문하고 큐에 담는다.

|  |  |  |
| --- | --- | --- |
| 0 | 3 | 3 |
| 5 | 7 |  |
| 9 |  |  |

이 때, (1,1)의 값이 변하게 된다.

위의 사이클에서 이미  (1,1)를 방문했지만, (1,1)은 (1,0)와도 인접하기 때문이다.

이 경우 (1,1)에는 인접한 두 노드를 경유하는 길이 중 더 짧은 값이 들어가게 된다. 즉, (1,1)은 9에서 7로 바뀌게 된다.

값이 바뀌게된 노드는 전차가 상, 하, 좌, 우로 이동할 수 있기 때문에 경로의 길이가 바뀔 수 있으므로 재탐색을 해주어야한다.

따라서 큐에 다시 넣어 탐색한다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_1249 {

public static class Node{

int x;

int y;

Node(int x, int y){

this.x = x;

this.y = y;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

int[] dx = {-1, 0, 1, 0};

int[] dy = {0, 1, 0, -1};

for(int t = 1; t <= tc; t++) {

int n = sc.nextInt();

int[][] a = new int[n][n];

int[][] v = new int[n][n];

int[][] r = new int[n][n];

Queue <Node> q = new LinkedList<>();

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

String s = sc.next();

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

a[i][j] = s.charAt(j) - '0';

r[i][j] = -1;

}

}

r[0][0] = 0;

v[0][0] = 1;

q.add(new Node(0, 0));

while(!q.isEmpty()) {

Node node = q.poll();

int x = node.x;

int y = node.y;

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < n && ny < n) {

if(v[nx][ny] == 0) {

q.add(new Node(nx, ny));

v[nx][ny] = 1;

r[nx][ny] = r[x][y] + a[nx][ny];

}

else {

if(r[x][y] + a[nx][ny] < r[nx][ny]) {

r[nx][ny] = r[x][y] + a[nx][ny];

q.add(new Node(nx, ny));

}

}

}

}

}

System.out.println("#" + t + " " + r[n-1][n-1]);

}

}

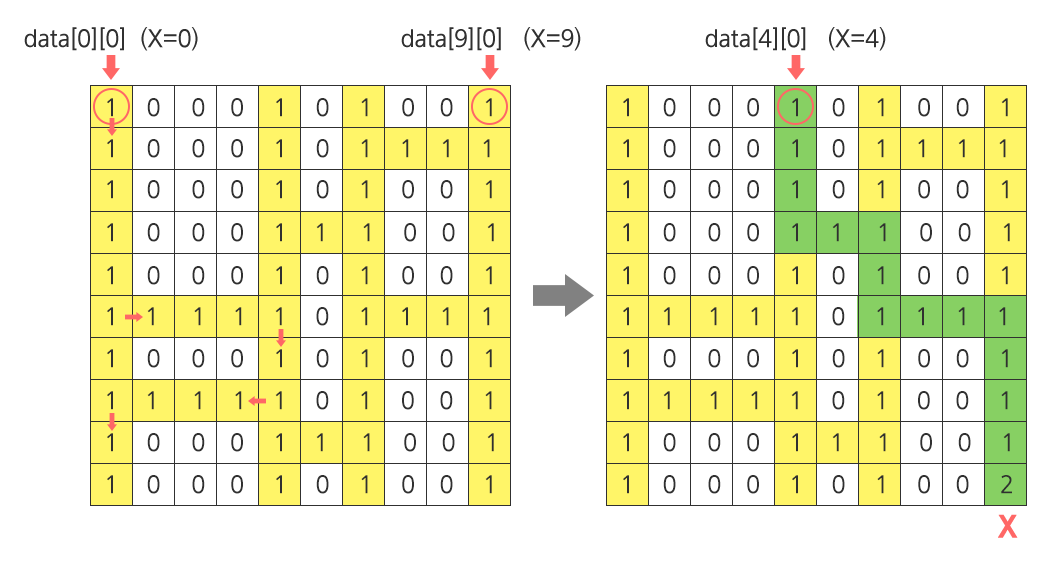
}

## [[SWEA] 1211. [S/W 문제해결 기본] 2일차 - Ladder2](https://dheldh77.tistory.com/entry/SWEA-1211-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-2%EC%9D%BC%EC%B0%A8-Ladder2)

2020. 3. 5. 10:35

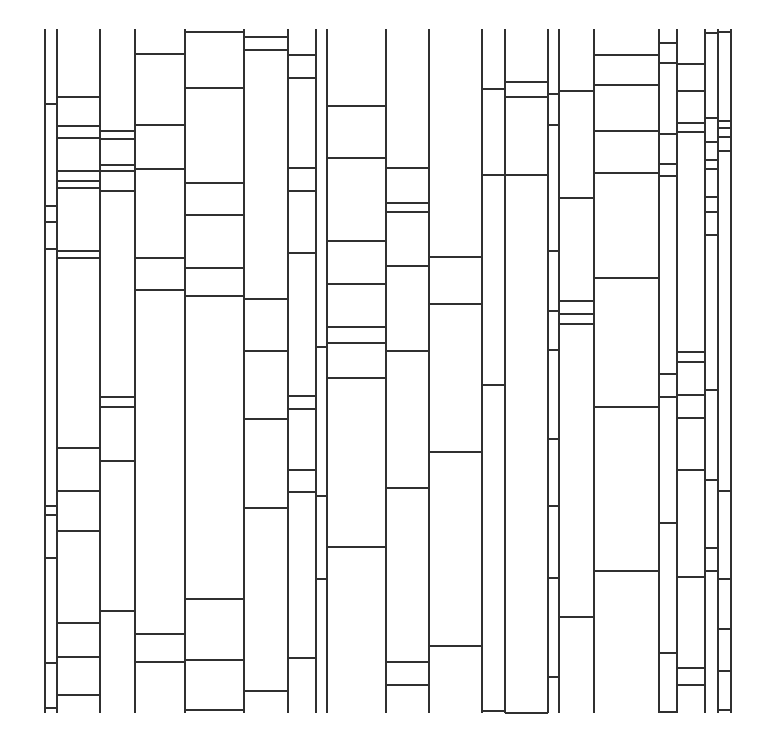
### 문제

점심 시간에 산책을 다니는 사원들은 최근 날씨가 더워져, 사다리 게임을 통하여 누가 아이스크림을 구입할지 결정하기로 한다.  
  
김 대리는 사다리타기에 참여하지 않는 대신 사다리를 그리기로 하였다.  
  
사다리를 다 그리고 보니 김 대리는 어느 사다리를 고르면 X표시에 도착하게 되는지 궁금해졌다. 이를 구해보자.  
  
아래 <그림 1>의 예를 살펴보면, 출발점 x=0 및 x=9인 세로 방향의 두 막대 사이에 임의의 개수의 막대들이 랜덤 간격으로 추가되고(이 예에서는 2개가 추가됨) 이 막대들 사이에 가로 방향의 선들이 또한 랜덤하게 연결된다.  
  
X=0인 출발점에서 출발하는 사례에 대해서 화살표로 표시한 바와 같이, 아래 방향으로 진행하면서 좌우 방향으로 이동 가능한 통로가 나타나면 방향 전환을 하게 된다.  
  
방향 전환 이후엔 다시 아래 방향으로만 이동하게 되며, 바닥에 도착하면 멈추게 된다.  
  
문제의 X표시에 도착하려면 X=4인 출발점에서 출발해야 하므로 답은 4가 된다. 해당 경로는 별도로 표시하였다.



<그림 1> 사다리 게임에 대한 설명(미니맵)

아래 <그림 2>와 같은 **100 x 100 크기의 2차원 배열로 주어진 사다리에 대해서, 모든 출발점을 검사하여 가장 짧은 이동 거리를 갖는 시작점 x(복수 개인 경우 가장 큰 x좌표)를 반환하는 코드를 작성하라.**(‘0’으로 채워진 평면상에 사다리는 연속된 ‘1’로 표현된다. 도착 지점은 '2'로 표현된다.)



<그림 2> 테스트케이스에 의해 생성되는 실제 사다리의 모습

### 풀이방법

1) 사다리의 입력을 2차원 배열 a[n][n]에 저장

2) 0행 i열이 1일 때, 즉 a[0][i] == 1일 때 탐색시작

3) 좌, 우의 원소 중 1이 있을 때, 1인 원소를 탐색

3-1) 좌, 우의 원소가 모두 0이라면 아래에있는 원소를 탐색

4) 행이 99일 때 탐색을 종료

### 소스코드

package samsung;

import java.util.\*;

public class s\_1211 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int tc = sc.nextInt();

int[][] a = new int[100][100];

int min = 10001;

int idx = 0;

for(int i = 0; i < 100; i++) {

for(int j = 0; j < 100; j++) {

a[i][j] = sc.nextInt();

}

}

for(int i = 0; i < 100; i++) {

if(a[0][i] == 1) {

int[][] v = new int[100][100];

int x = 0;

int y = i;

int cnt = 0;

while(x < 99) {

v[x][y] = 1;

if(y - 1 >= 0 && v[x][y - 1] == 0 && a[x][y - 1] > 0) {

y--;

}

else if(y + 1 < 100 && v[x][y + 1] == 0 && a[x][y + 1] > 0) {

y++;

}

else if(x + 1 < 100 && v[x + 1][y] == 0 && a[x + 1][y] > 0){

x++;

}

cnt++;

}

if(cnt < min) {

min = cnt;

idx = i;

}

}

}

System.out.println("#" + t + " " + idx);

}

}

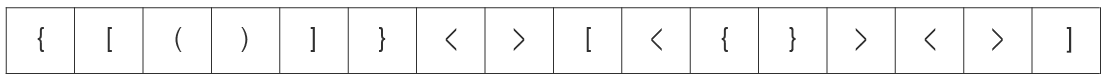
}

## [[SWEA] 1218. [S/W 문제해결 기본] 4일차 - 괄호 짝짓기](https://dheldh77.tistory.com/entry/SWEA-1218-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-4%EC%9D%BC%EC%B0%A8-%EA%B4%84%ED%98%B8-%EC%A7%9D%EC%A7%93%EA%B8%B0)

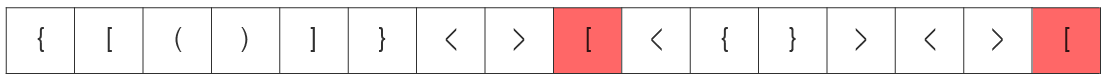
2020. 3. 5. 10:28

### 문제

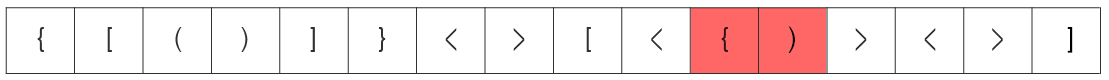
4 종류의 괄호문자들 '()', '[]', '{}', '<>' 로 이루어진 문자열이 주어진다.  
  
이 문자열에 사용된 괄호들의 짝이 모두 맞는지 판별하는 프로그램을 작성한다.  
  
예를 들어 아래와 같은 문자열은 유효하다고 판단할 수 있다.



아래와 같은 문자열은 유효하지 않은 문자열이다. 붉은색으로 표시된 괄호의 짝을 찾을 수 없기 때문이다.



아래 문자열은 열고 닫는 괄호의 개수는 유효하나 짝이 맞지 않는 괄호가 사용 되었기 때문에 유효하지 않다.



### 풀이방법

스택을 이용해 해결한다.

1) '{' , '[' , '<', '(' 기호가 나오면 스택에 담는다.

2) '}', ']', '>', ')' 닫는 기호가 나올 경우 스택에 담겨진 top의 기호와 쌍을 이루는지 확인한다.

3) 2의 조건을 만족하면 pop으로 스택의 top을 삭제한다.

4) 모든 입력을 다 받아들인 후 스택이 비었다면 유효하고 하나라도 있다면 유효하지 않는다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_1218 {

public static void main(String[] ags) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int tc = sc.nextInt();

String s = sc.next();

Stack <Character> st = new Stack();

int r = 0;

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

char c = s.charAt(i);

if(c == ')' && st.peek() == '(') st.pop();

else if(c == ']' && st.peek() == '[') st.pop();

else if(c == '}' && st.peek() == '{') st.pop();

else if(c == '>' && st.peek() == '<') st.pop();

else {

st.push(c);

}

}

if(st.isEmpty()) r = 1;

else r = 0;

System.out.println("#" + t + " " + r);

}

}

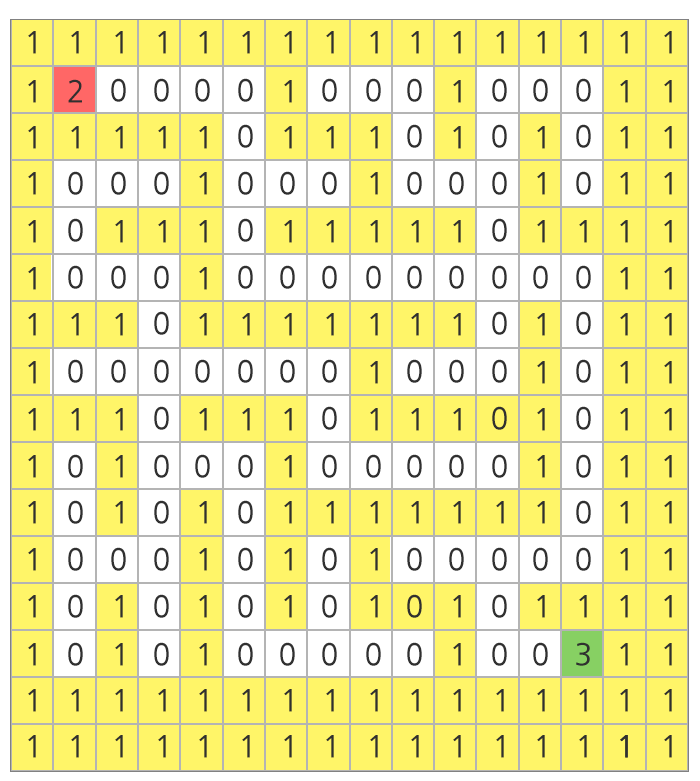
}

## [[SWEA] 1227. [S/W 문제해결 기본] 7일차 - 미로2](https://dheldh77.tistory.com/entry/1227-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-7%EC%9D%BC%EC%B0%A8-%EB%AF%B8%EB%A1%9C2)

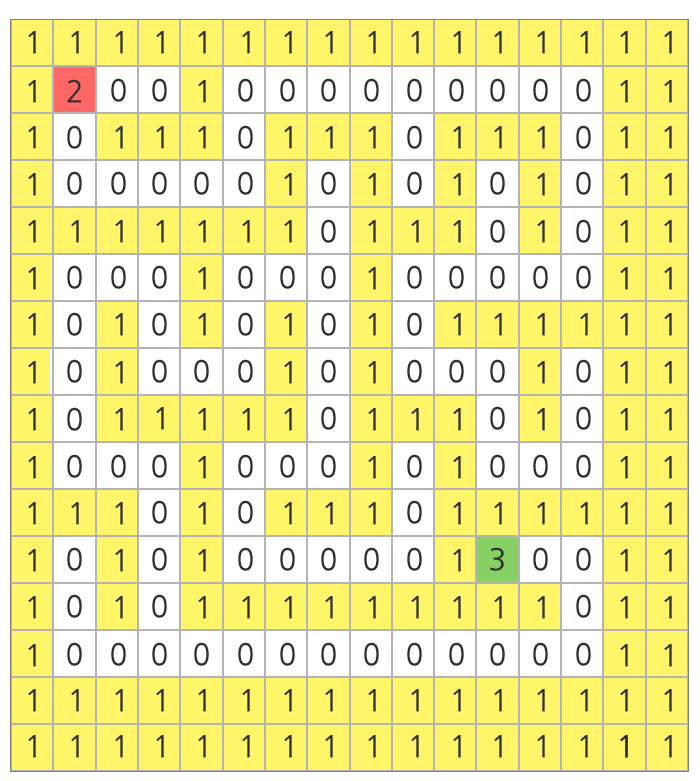
2020. 3. 4. 08:12

### 문제

아래 그림과 같은 미로가 있다. 100\*100 행렬의 형태로 만들어진 미로에서 흰색 바탕은 길, 노란색 바탕은 벽을 나타낸다.  
  
가장 좌상단에 있는 칸을 (0, 0)의 기준으로 하여, 가로방향을 x 방향, 세로방향을 y 방향이라고 할 때, 미로의 시작점은 (1, 1)이고 도착점은 (13, 13)이다.  
  
주어진 미로의 출발점으로부터 도착지점까지 갈 수 있는 길이 있는지 판단하는 프로그램을 작성하라.  
  
아래의 예시에서는 도달 가능하다.



아래의 예시에서는 출발점이 (1, 1)이고, 도착점이 (11, 11)이며 도달이 불가능하다.



위의 예시는 공간상의 이유로 100x100이 아닌 16x16으로 주어졌음에 유의한다.

### 풀이방법

DFS(깊이 우선 탐색)을 이용해 문제 해결.

2의 값이 저장된 노드를 중심으로 상, 하, 좌, 우 갈 수 있는 노드를 탐색하며 모든 경로를 찾는다.

모든 경로를 찾았을 때, 3이 저장된 노드를 방문하는 경로를 찾지 못했다면 0을 출력, 찾았다면 1을 출력한다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_1227 {

static int[][] map;

static int[][] v;

static int[] dx = {-1, 0, 1, 0};

static int[] dy = {0, 1, 0, -1};

static int find;

public static void dfs(int x, int y) {

if(find == 1)

return;

if(map[x][y] == 3) {

find = 1;

return;

}

v[x][y] = 1;

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < 100 && ny < 100) {

if(v[nx][ny] == 0 && map[nx][ny] != 1) {

dfs(nx,ny);

}

}

}

v[x][y] = 0;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int tc = sc.nextInt();

map = new int[100][100];

v = new int[100][100];

int x = 0;

int y = 0;

find = 0;

for(int i = 0; i < 100; i++) {

String s = sc.next();

for(int j = 0; j < 100; j++) {

map[i][j] = s.charAt(j) - '0';

if(map[i][j] == 2) {

x = i;

y = j;

}

}

}

dfs(x, y);

System.out.println("#" + t + " " + find);

}

}

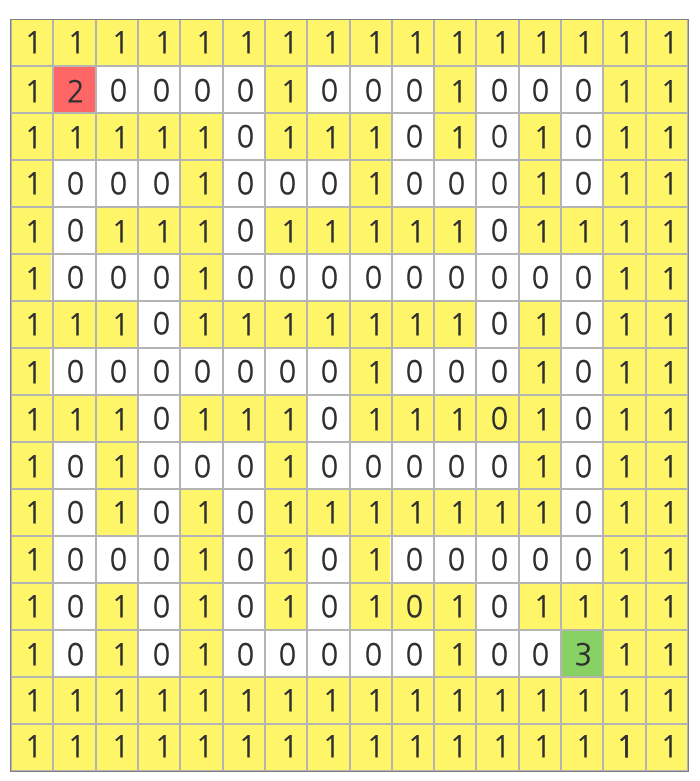
}

## [[SWEA] 1226. [S/W 문제해결 기본] 7일차 - 미로1](https://dheldh77.tistory.com/entry/1226-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-7%EC%9D%BC%EC%B0%A8-%EB%AF%B8%EB%A1%9C1)

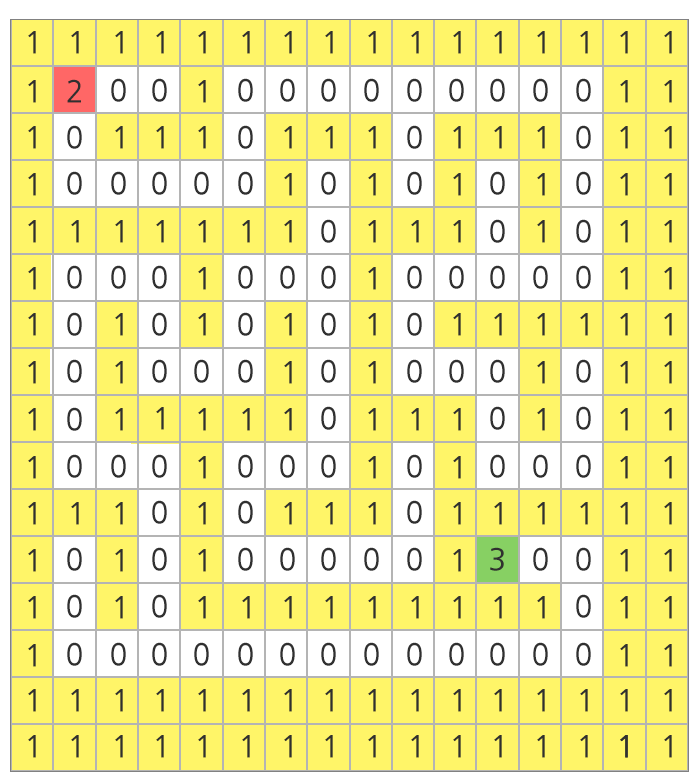
2020. 3. 4. 08:11

### 문제

아래 그림과 같은 미로가 있다. 16\*16 행렬의 형태로 만들어진 미로에서 흰색 바탕은 길, 노란색 바탕은 벽을 나타낸다.  
  
가장 좌상단에 있는 칸을 (0, 0)의 기준으로 하여, 가로방향을 x 방향, 세로방향을 y 방향이라고 할 때, 미로의 시작점은 (1, 1)이고 도착점은 (13, 13)이다.  
  
주어진 미로의 출발점으로부터 도착지점까지 갈 수 있는 길이 있는지 판단하는 프로그램을 작성하라.  
  
아래의 예시에서는 도달 가능하다.



아래의 예시에서는 출발점이 (1, 1)이고, 도착점이 (11, 11)이며 도달이 불가능하다.



### 풀이방법

DFS(깊이 우선 탐색)을 이용해 문제 해결.

2의 값이 저장된 노드를 중심으로 상, 하, 좌, 우 갈 수 있는 노드를 탐색하며 모든 경로를 찾는다.

모든 경로를 찾았을 때, 3이 저장된 노드를 방문하는 경로를 찾지 못했다면 0을 출력, 찾았다면 1을 출력한다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_1226 {

static int[][] a;

static int[][] v;

static int[] dx = {-1, 0, 1, 0};

static int[] dy = {0, 1, 0, -1};

static int find;

public static void dfs(int x, int y) {

if(find == 1)

return;

if(a[x][y] == 3) {

find = 1;

return;

}

v[x][y] = 1;

for(int i = 0; i < 4; i++) {

int nx = x + dx[i];

int ny = y + dy[i];

if(nx >= 0 && ny >= 0 && nx < 16 && ny < 16) {

if(v[nx][ny] == 0 && a[nx][ny] != 1) {

dfs(nx, ny);

}

}

}

v[x][y] = 0;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int tc = sc.nextInt();

a = new int[16][16];

v = new int[16][16];

int x = 0;

int y = 0;

find = 0;

for(int i = 0; i < 16; i++) {

String s = sc.next();

for(int j = 0; j < 16; j++) {

a[i][j] = s.charAt(j) - '0';

if(a[i][j] == 2) {

x = i;

y = j;

}

}

}

dfs(x, y);

System.out.println("#" + t + " " + find);

}

}

}

좋아요공감

공유하기

글 요소

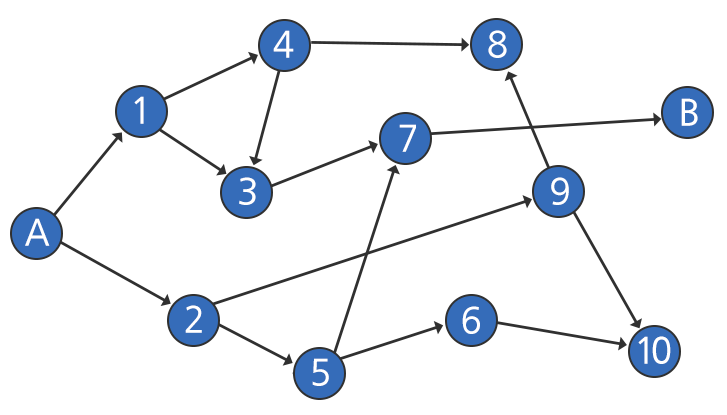
구독하기

## [[SWEA] 1219. [S/W 문제해결 기본] 4일차 - 길찾기](https://dheldh77.tistory.com/entry/1219-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-4%EC%9D%BC%EC%B0%A8-%EA%B8%B8%EC%B0%BE%EA%B8%B0)

2020. 3. 4. 08:06

### 문제

그림과 같이 도식화한 지도에서 A도시에서 출발하여 B도시로 가는 길이 존재하는지 조사하려고 한다.  
  
길 중간 중간에는 최대 2개의 갈림길이 존재하고, 모든 길은 일방 통행으로 되돌아오는 것이 불가능하다.  
  
다음과 같이 길이 주어질 때, A도시에서 B도시로 가는 길이 존재하는지 알아내는 프로그램을 작성하여라.  
  
 - A와 B는 숫자 0과 99으로 고정된다.  
  
 - 모든 길은 순서쌍으로 나타내어진다. 위 예시에서 2번에서 출발 할 수 있는 길의 표현은 (2, 5), (2, 9)로 나타낼 수 있다.  
  
 - 가는 길의 개수와 상관없이 한가지 길이라도 존재한다면 길이 존재하는 것이다.  
  
 - 단 화살표 방향을 거슬러 돌아갈 수는 없다.



### 풀이방법

DFS(깊이 우선 탐색)을 이용해 문제 해결.

입력에 따라 각 노드와 간선을 나타내는 인접행렬을 만들고, 0노드에서 출발해 99번째 노드로 갈 수 있는 경로가 존재하는지 찾는다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_1219 {

static int[][] a;

static int[] v;

static int find;

public static void dfs(int x) {

if(find == 1)

return;

if(x == 99) {

find = 1;

return;

}

v[x] = 1;

for(int i = 0; i < 100; i++) {

if(v[i] == 0 && a[x][i] == 1)

dfs(i);

}

v[x] = 0;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int tc = sc.nextInt();

int n = sc.nextInt();

a = new int[100][100];

v = new int[100];

find = 0;

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

int x = sc.nextInt();

int y = sc.nextInt();

a[x][y] = 1;

}

dfs(0);

System.out.println("#" + t + " " + find);

}

}

}

## [[SWEA] 1224. [S/W 문제해결 기본] 6일차 - 계산기3](https://dheldh77.tistory.com/entry/SWEA-1224-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-6%EC%9D%BC%EC%B0%A8-%EA%B3%84%EC%82%B0%EA%B8%B03)

2020. 3. 4. 08:01

### 문제

문자열로 이루어진 계산식이 주어질 때, 이 계산식을 후위 표기식으로 바꾸어 계산하는 프로그램을 작성하시오.  
  
예를 들어  
  
“3+(4+5)\*6+7”  
  
라는 문자열로 된 계산식을 후위 표기식으로 바꾸면 다음과 같다.  
  
"345+6\*+7+"  
  
변환된 식을 계산하면 64를 얻을 수 있다.  
  
문자열 계산식을 구성하는 연산자는 +, \* 두 종류이며 문자열 중간에 괄호가 들어갈 수 있다.  
  
이 때 괄호의 유효성 여부는 항상 옳은 경우만 주어진다.  
  
피연산자인 숫자는 0 ~ 9의 정수만 주어진다.

### 풀이방법

문제에서는 2가지를 요구하고 있다. 주어진 계산식을 후위표기식으로 바꾸는 것과 후위표기식으로 바꾼 식을 계산하는 것이다.

**[1. 주어진 계산식을 후위표기식으로 바꾸는 것]**

후위 표기식으로 바꾸기 위해서는 스택을 이용한다.

주어진 계산식을 한 글자씩 읽어들일 때, 숫자일 경우 그대로 출력하고, '+' 또는 '\*' 기호일 경우 우선 순위에 따라 스택에 담거나 출력한다.

즉, 아래와 같은 조건과 과정을 따른다.

입력을 한 문자씩 읽을 때,

**1) 숫자일 경우는 바로 출력**

**2) '(' 기호일 경우, push**

**3) ')' 기호일 경우, 스택에 '(' 연산자가 나올때까지 pop**

**4) '+', '\*' 기호일 경우, 스택이 비었는지 확인**

**2-1) 스택이 비었을 경우, push**

**2-2) 스택이 차 있을 경우 (우선순위 : '\*' > '+' > '(' )**

**2-2-a) 읽어들인 입력이 스택의 top에 있는 값보다 우선순위가 높을 경우 push**

**2-2-b) 읽어들인 입력의 스택의 top에 있는 값보다 우선순위가 낮거나 같은 경우 스택의 값을 pop한 뒤 입력을 push**

**5) 입력을 끝까지 읽었을 때, 스택에 들어있는 내용을 모두 출력**

**[2. 후위표기식으로 표현된 식을 계산하는 것]**

후위표기식을 계산하는 방법 또한 스택을 이용한다. 1의 과정에서 출력내용을 문자열에 담았을 때, 한 문자씩 읽어들인다.

읽어들인 문자가 숫자일 경우 스택에 담고, '+' 또는 '\*" 기호일 경우 스택에 담긴 두 숫자를 뽑아 더하거나 곱해 다시 스택에 담는다.

즉, 아래와 같은 조건과 과정을 따른다.

입력을 한 문자씩 읽을 때,

**1) 숫자일 경우 스택에 push**

**2) '+'기호일 경우 pop으로 스택에 담겨진 두 개의 숫자를 뽑아 덧셈하고 이 결과를 다시 스택에 push**

**3) '\*'기호일 경우 pop으로 스택에 담겨진 두 개의 숫자를 뽑아 곱셈하고 이 결과를 다시 스택에 push**

**4) 입력을 끝까지 읽었을 때, 스택에 들어있는 결과를 출력**

### 소스코드

package samsung;

import java.util.\*;

public class s\_1224 {

public static int get\_prior(char c) {

if(c == '\*')

return 1;

if(c == '+')

return 3;

if(c == '(')

return 5;

else

return -1;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int n = sc.nextInt();

String s = sc.next();

Stack <Character> op = new Stack();

String r = "";

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

char c = s.charAt(i);

if(c - '0' >= 0 && c - '0' <= 9) {

r += String.valueOf(c);

}

else if( c == ')') {

char cur = op.pop();

while(!op.isEmpty() && cur != '(') {

r += String.valueOf(cur);

cur = op.pop();

}

}

else {

if(op.isEmpty()) {

op.push(c);

}

else {

if(c == '(')

op.push(c);

else if(get\_prior(op.peek()) > get\_prior(c))

op.push(c);

else {

r += String.valueOf(op.pop());

op.push(c);

}

}

}

}

while(!op.isEmpty()) {

r += op.pop();

}

Stack <Integer> nums = new Stack();

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

char c = r.charAt(i);

if(c - '0' >= 0 && c -'0' <= 9) {

nums.push(c - '0');

}

else {

int tmp1 = nums.pop();

int tmp2 = nums.pop();

if(c == '\*') nums.push(tmp1 \* tmp2);

else if(c == '+') nums.push(tmp1 + tmp2);

}

}

System.out.println("#" + t + " " + nums.pop());

}

}

}

## [[SWEA] 1223. [S/W 문제해결 기본] 6일차 - 계산기2](https://dheldh77.tistory.com/entry/SWEA-1223-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-6%EC%9D%BC%EC%B0%A8-%EA%B3%84%EC%82%B0%EA%B8%B02)

2020. 3. 4. 07:53

### 문제

문자열로 이루어진 계산식이 주어질 때, 이 계산식을 후위 표기식으로 바꾸어 계산하는 프로그램을 작성하시오.  
  
예를 들어  
  
“3+4+5\*6+7”  
  
라는 문자열로 된 계산식을 후위 표기식으로 바꾸면 다음과 같다.  
  
"34+56\*+7+"  
  
변환된 식을 계산하면 44를 얻을 수 있다.  
  
문자열 계산식을 구성하는 연산자는 +, \* 두 종류이며 피연산자인 숫자는 0 ~ 9의 정수만 주어진다.

### 풀이방법

문제에서는 2가지를 요구하고 있다. 주어진 계산식을 후위표기식으로 바꾸는 것과 후위표기식으로 바꾼 식을 계산하는 것이다.

**[1. 주어진 계산식을 후위표기식으로 바꾸는 것]**

후위 표기식으로 바꾸기 위해서는 스택을 이용한다.

주어진 계산식을 한 글자씩 읽어들일 때, 숫자일 경우 그대로 출력하고, '+' 또는 '\*' 기호일 경우 우선 순위에 따라 스택에 담거나 출력한다.

즉, 아래와 같은 조건과 과정을 따른다.

입력을 한 문자씩 읽을 때,

**1) 숫자일 경우는 바로 출력**

**2) '+', '\*'기호일 경우, 스택이 비었는지 확인**

**2-1) 스택이 비었을 경우, push**

**2-2) 스택이 차 있을 경우 ('\*'기호가 '+'기호보다 우선순위가 높다)**

**2-2-a) 읽어들인 입력이 스택의 top에 있는 값보다 우선순위가 높을 경우 push**

**2-2-b) 읽어들인 입력의 스택의 top에 있는 값보다 우선순위가 낮거나 같은 경우 스택의 값을 pop한 뒤 입력을 push**

**3) 입력을 끝까지 읽었을 때, 스택에 들어있는 내용을 모두 출력**

**[2. 후위표기식으로 표현된 식을 계산하는 것]**

후위표기식을 계산하는 방법 또한 스택을 이용한다. 1의 과정에서 출력내용을 문자열에 담았을 때, 한 문자씩 읽어들인다.

읽어들인 문자가 숫자일 경우 스택에 담고, '+' 또는 '\*" 기호일 경우 스택에 담긴 두 숫자를 뽑아 더하거나 곱해 다시 스택에 담는다.

즉, 아래와 같은 조건과 과정을 따른다.

입력을 한 문자씩 읽을 때,

**1) 숫자일 경우 스택에 push**

**2) '+'기호일 경우 pop으로 스택에 담겨진 두 개의 숫자를 뽑아 덧셈하고 이 결과를 다시 스택에 push**

**3) '\*'기호일 경우 pop으로 스택에 담겨진 두 개의 숫자를 뽑아 곱셈하고 이 결과를 다시 스택에 push**

**4) 입력을 끝까지 읽었을 때, 스택에 들어있는 결과를 출력**

### 소스코드

package samsung;

import java.util.\*;

public class s\_1223 {

public static int get\_prior(char c) {

if(c == '\*')

return 1;

else if(c == '+')

return 3;

else

return -1;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int n = sc.nextInt();

String s = sc.next();

Stack <Character> op = new Stack();

String r = "";

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

char c = s.charAt(i);

if(c - '0' >= 0 && c - '0' <= 9) {

r += String.valueOf(c);

}

else {

if(op.isEmpty()) {

op.push(c);

}

else {

if(get\_prior(op.peek()) <= get\_prior(c)) {

r += String.valueOf(op.pop());

op.push(c);

}

else {

op.push(c);

}

}

}

}

while(!op.isEmpty()) {

r += String.valueOf(op.pop());

}

Stack <Integer> nums = new Stack();

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

char c = r.charAt(i);

if(c - '0' >= 0 && c -'9' <= 0) nums.push(c - '0');

else {

int tmp1 = nums.pop();

int tmp2 = nums.pop();

if(c == '\*') nums.push(tmp1 \* tmp2);

else if(c == '+') nums.push(tmp1 + tmp2);

}

}

System.out.println("#" + t + " " + nums.pop());

}

}

}

## [[SWEA] 1222. [S/W 문제해결 기본] 6일차 - 계산기1](https://dheldh77.tistory.com/entry/SWEA-1222-SW-%EB%AC%B8%EC%A0%9C%ED%95%B4%EA%B2%B0-%EA%B8%B0%EB%B3%B8-6%EC%9D%BC%EC%B0%A8-%EA%B3%84%EC%82%B0%EA%B8%B01)

2020. 3. 4. 07:46

### 문제

문자열로 이루어진 계산식이 주어질 때, 이 계산식을 후위 표기식으로 바꾸어 계산하는 프로그램을 작성하시오.  
  
예를 들어  
  
“3+4+5+6+7”  
  
라는 문자열로 된 계산식을 후위 표기식으로 바꾸면 다음과 같다.  
  
"34+5+6+7+"  
  
변환된 식을 계산하면 25를 얻을 수 있다.  
  
문자열 계산식을 구성하는 연산자는 + 하나뿐이며 피연산자인 숫자는 0 ~ 9의 정수만 주어진다.

### 풀이방법

문제에서는 2가지를 요구하고 있다. 주어진 계산식을 후위표기식으로 바꾸는 것과 후위표기식으로 바꾼 식을 계산하는 것이다.

**[1. 주어진 계산식을 후위표기식으로 바꾸는 것]**

후위 표기식으로 바꾸기 위해서는 스택을 이용한다.

주어진 계산식을 한 글자씩 읽어들일 때, 숫자일 경우 그대로 출력하고, '+'기호일 경우 스택에 담거나 출력한다.

즉, 아래와 같은 조건과 과정을 따른다.

입력을 한 문자씩 읽을 때,

**1) 숫자일 경우는 바로 출력**

**2) '+'기호일 경우, 스택이 비었는지 확인**

**2-1) 스택이 비었을 경우, push**

**2-2) 스택이 차 있을 경우, pop하고 출력**

**3) 입력을 끝까지 읽었을 때, 스택에 들어있는 내용을 모두 출력**

**[2. 후위표기식으로 표현된 식을 계산하는 것]**

후위표기식을 계산하는 방법 또한 스택을 이용한다. 1의 과정에서 출력내용을 문자열에 담았을 때, 한 문자씩 읽어들인다.

읽어들인 문자가 숫자일 경우 스택에 담고, '+'기호일 경우 스택에 담긴 두 숫자를 뽑아 더하고 다시 스택에 담는다.

즉, 아래와 같은 조건과 과정을 따른다.

입력을 한 문자씩 읽을 때,

**1) 숫자일 경우 스택에 push**

**2) '+'기호일 경우 pop으로 스택에 담겨진 두 개의 숫자를 뽑아 덧셈하고 이 결과를 다시 스택에 push**

**3) 입력을 끝까지 읽었을 때, 스택에 들어있는 결과를 출력**

### 소스코드

package samsung;

import java.util.\*;

public class s\_1222 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

for(int t = 1; t <= 10; t++) {

int n = sc.nextInt();

String s = sc.next();

Stack <Character> st = new Stack();

String result = "";

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

char c = s.charAt(i);

if(c - '0' >= 0 && c - '0' <= 9) result += String.valueOf(c);

else {

if(st.isEmpty()) {

st.push(c);

}else {

result += String.valueOf(st.pop());

st.push(c);

}

}

}

while(!st.isEmpty()) {

result += String.valueOf(st.pop());

}

Stack <Integer> st2 = new Stack();

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

char c = result.charAt(i);

if(c - '0' >= 0 && c - '0' <= 9) {

st2.push(c - '0');

}

else {

int op1 = st2.pop();

int op2 = st2.pop();

st2.push(op1 + op2);

}

}

System.out.println("#" + t + " " + st2.pop());

}

}

}

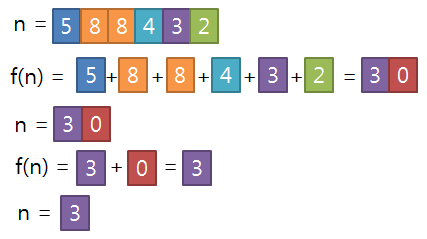
## [[SWEA] 3750. Digit sum](https://dheldh77.tistory.com/entry/SWEA-3750-Digit-sum)

2020. 3. 4. 07:11

[3750](https://dheldh77.tistory.com/tag/3750), [SWEA](https://dheldh77.tistory.com/tag/SWEA)

### 문제

자연수 n에 대해 함수 f(n)은 n의 각 자릿수를 더한 값이다.  
  
예를 들어 n = 588432라면, f(n) = 5 + 8 + 8 + 4 + 3 + 2 = 30인 것이다.  
  
어떤 자연수 n이 주어질 때, n이 한 자리수가 될 때까지 n에 f(n)을 대입하는 것을 반복하면, 최종적으로 n이 어떤 값이 되는지 구하는 프로그램을 작성하라.  
  
예를 들어 n = 588432라면 f(n) = 30이므로 n = 30이 되고, 이 때 f(n) = 3으로 최종적으로 n = 3이 되는 것이다.



### 풀이방법

1) 숫자를 문자열로 입력받고 각 자리수를 더해서 새로운 문자열로 만든다.

2) 문자열의 길이가 1이 될 때까지 1)의 과정을 반복한다.

3) 문자열의 길이가 1일 때 값을 출력한다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_3750 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

String s = sc.next();

int n = s.length();

while(true) {

if(n == 1) {

System.out.println("#" + t + " " + s);

break;

}

long num = 0;

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

num += s.charAt(i) - '0';

}

s = String.valueOf(num);

n = s.length();

}

}

}

}

## [[SWEA] 5986. 새샘이와 세 소수](https://dheldh77.tistory.com/entry/SWEA-5986-%EC%83%88%EC%83%98%EC%9D%B4%EC%99%80-%EC%84%B8-%EC%86%8C%EC%88%98)

2020. 2. 26. 11:02

### 문제

정수론에서, 세 소수 문제란 다음과 같다.  
  
“5보다 큰 모든 홀수는 정확히 세 소수의 합으로 표현될 수 있다. (같은 소수를 합에 사용해도 된다.)”  
  
예를 들어, 7 = 2 + 2 + 3, 11 = 2 + 2 + 7, 25 = 7 + 7 + 11로 나타낼 수 있다.  
  
1939년 러시아 수학자 I. M. Vinogradov는 충분히 큰 홀수는 세 소수의 합으로 표현할 수 있다는 것을 증명했다.  
  
여기서 충분히 크다는 것은 3315 ≈ 107000000 이상의 수라는 의미이다.  
  
현재 가장 발전된 하한은 약 e3100 ≈ 101346 이상의 수이다.  
  
러시아 수학자 I. M. Vinogradov 를 존경하는 새샘이는 직접 세 소수 문제를 풀어보기로 했다.  
  
하지만 이 수는 너무 크기 때문에 컴퓨터로도 이 범위까지의 모든 수를 증명할 수는 없었다.  
  
대신 어떤 크지 않은 홀수에 대해, 세 소수의 합으로 나타낼 수 있는 경우의 수를 구하기로 했다.  
  
5보다 큰 홀수 N을 입력 받아 N = x + y + z (단, x ≤ y ≤ z 이고, x, y, z는 소수) 로 나타나는 경우의 수를 구하는 프로그램을 작성

### 풀이방법

1) 입력으로 받은 n보다 작거나 같은 소수를 모두 찾는다.

2) 이 소수들 중 3개의 수를 뽑아 n을 만들 수 있는 경우의 수를 찾는다.

3개의 수만 뽑으면 되는 문제이므로 3중 반복문으로 해결한다. 단, 순열이 아닌 조합이므로 i, j, k 순으로 반복문을 돌릴 때,

i 는 0부터 j는 i부터 k는 j부터 반복한다.

### 소스코드

package samsung;

import java.util.\*;

public class s\_5986 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int tc = sc.nextInt();

for(int t = 1; t <= tc; t++) {

int n = sc.nextInt();

int cnt = 0;

ArrayList <Integer> a = new ArrayList<>();

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

int flag = 0;

for(int j = 2; j <= Math.sqrt(i); j++) {

if(i % j == 0) {

flag = 1;

break;

}

}

if(flag == 0)

a.add(i);

}

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

for(int j = i; j < a.size(); j++) {

if(a.get(i) + a.get(j) >= n)

break;

for(int k = j; k < a.size(); k++) {

int sum = a.get(i) + a.get(j) + a.get(k);

if(sum == n) {

cnt++;

}

}

}

}

System.out.println("#" + t + " " + cnt);

}

}

}

[[SWEA] 9280. 진용이네 주차타워](https://dheldh77.tistory.com/entry/SWEA-9280-%EC%A7%84%EC%9A%A9%EC%9D%B4%EB%84%A4-%EC%A3%BC%EC%B0%A8%ED%83%80%EC%9B%8C)

2020. 2. 25. 10:31

**문제**

부지런한 진용이는 정문 앞에서 유료 주차장 운영하고 있다. 이 주차장은 *1* 부터 *n* 까지 번호가 매겨진 *n* 개의 주차 공간을 가지고 있다.  
매일 아침 모든 주차 공간이 비어 있는 상태에서 영업을 시작하며, 다음과 같은 방식으로 운영된다.

* 차가 주차장에 도착하면, 진용이는 비어있는 주차 공간이 있는지 검사한다.
* 비어있는 공간이 없다면, 빈 공간이 생길 때까지 차량을 입구에서 기다리게 한다.
* 빈 주차 공간이 있으면 진용이는 곧바로 주차를 시키며, 주차 가능한 공간 중 번호가 가장 작은 주차 공간에 주차하도록 한다.
* 만약 주차를 기다리는 차량이 여러 대라면, 입구의 대기장소에서 자기 차례를 기다려야 한다. 운전자들은 예의가 바르기 때문에 새치기를 하지 않는다.

주차요금은 차량의 무게와 주차 공간마다 따로 책정된 단위 무게당 금액을 곱한 가격이다. 진용이네 주차장에서는 종일 이용권만을 판매하기 때문에 이용시간은 고려하지 않는다.

진용이는 오늘 주차장을 이용할 *m* 대의 차량들이 들어오고 나가는 순서를 알고 있다.  
진용이의 주차장이 오늘 하루 벌어들일 총 수입을 계산하는 프로그램을 작성하라.

* 첫 번째 줄에 자연수 *n* 과 *m* 이 주어진다. (1 ≤ *n*  ≤ 100, 1 ≤ *m*  ≤ 2000)
* *n* 개의 줄에 *i* 번째 주차 공간의 단위 무게당 요금 R*i* 가 정수로 주어진다. (1 ≤ R*i* ≤ 100)
* *m* 개의 줄에 차량 *i* 의 무게 W*i* 가 정수로 주어진다. 차량번호 *i* 와 차량의 도착 순서는 아무런 관계가 없다. (1 ≤ W*i*  ≤ 10000)
* 이후 *2m* 개의 줄에 차량들의 주차장 출입 순서가 하나의 정수 *x* 로 주어진다.  
  주어진 정수 *x* 가 양수면, *x* 번 차가 주차장에 들어옴을 뜻한다.  
  *x* 가 음수면, *-x* 번 차가 주차장을 나감을 뜻한다.

**풀이방법**

큐(Stack) 자료구조의 선입선출(FIFO, First In First Out)을 이용한다.

1) 세 개의 자료구조가 필요하다. m개의 입력을 받아들일 큐, 주차장으로 사용될 배열, 주차장이 가득 찼을 경우를 대비한 큐

2) m개의 입력을 큐에 넣는다. 배열을 0으로 초기화한다.

3) 큐에서 값을 하나씩 뽑아 배열에 낮은 인덱스 순으로 원소에 큐의 값을 넣어준다.

4) 큐에서 음수가 나올 경우 -1을 곱해 해당하는 원소의 값을 0으로 바꿔준다.

4-1) 큐에서 양수가 나왔는데 배열이 모두 0이 아닐 때, 1)에서 만들어준 대비 큐에 넣는다.

5) 4)의 단계가 나올때까지 4-1)을 반복한다.

6) 4)의 단계가 나왔다면 대비 큐에서 값을 뽑아 배열에 넣는다.

7) 4) - 6)의 과정을 m을 담은 큐가 빌 때까지 반복한다.

**소스코드**

package samsung;

import java.util.\*;

public class s\_9280 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int test = sc.nextInt();

for(int t = 1; t <= test; t++) {

int n = sc.nextInt();

int m = sc.nextInt();

int sum = 0;

int[] r = new int[n];

int[] w = new int[m];

int[] p = new int[n];

Queue <Integer> q = new LinkedList();

Queue <Integer> rq = new LinkedList();

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

r[i] = sc.nextInt();

}

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

w[i] = sc.nextInt();

}

for(int i = 0; i < 2 \* m; i++) {

int tmp = sc.nextInt();

q.add(tmp);

}

int cnt = 0;

while(!q.isEmpty()) {

int num;

if(cnt < n && !rq.isEmpty()) {

num = rq.poll();

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

if(p[i] == 0) {

p[i] = num;

sum += w[num - 1] \* r[i];

cnt++;

break;

}

}

}else {

num = q.poll();

if(num > 0) {

if(cnt < n) {

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

if(p[i] == 0) {

p[i] = num;

sum += w[num - 1] \* r[i];

cnt++;

break;

}

}

}

else {

rq.add(num);

}

}

else {

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

if(p[i] == -1 \* num) {

p[i] = 0;

cnt--;

}

}

}

}

}

System.out.println("#" + t + " " + sum);

}

}

}

## [[SWEA] 8931. 제로](https://dheldh77.tistory.com/entry/SWEA-8931-%EC%A0%9C%EB%A1%9C)

### 문제

재현이는 재민이를 도와서 동아리 장부를 관리하고 있다.

재현이는 영수증을 모아서 동아리의 지출 금액을 세고 있고, 재민이는 재현이가 부르는 액수를 순서대로 적고 있다.

재현이는 가끔 잘못된 수를 부르는 실수를 하는데, 이 때마다 0을 외쳐서, 가장 최근에 재민이가 쓰고 지우지 않았던 수를 지우게 시킨다.

재현이가 모든 수를 부른 후 재민이가 받아 적은 수의 합은 무엇일까?

### 풀이방법

스택(Stack) 자료구조의 후입선출(LIFO, Last In First Out)을 이용한다.

 재현이가 말하는 숫자를 스택에 집어넣되, 재현이가 0을 말하면 스택에 가장 최근에 입력된 값을 삭제한다.

 소스코드

package samsung;

import java.util.\*;

public class s\_8931 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int test = sc.nextInt();

for(int t = 1; t <= test; t++) {

int n = sc.nextInt();

Stack <Integer> s = new Stack<>();

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

int tmp = sc.nextInt();

if(tmp == 0)

s.pop();

else

s.push(tmp);

}

int sum = 0;

while(!s.isEmpty()) {

int tmp = s.pop();

sum += tmp;

}

System.out.println("#" + t + " " + sum);

}

}

}