# A. 양말 짝 맞추기

(Time Limit: 10 seconds)

양말 장사를 하는 지훈이는 오랜만에 창고 정리를 하기로 했다. 창고에는 양말 짝이 맞지 않은 채 가득 섞여 있었다. 이제 지훈이를 도와 어지럽게 섞여있는 양말의 짝을 맞추어 보자.


- 양말의 종류는 알파벳으로 구분된다.
- 양말의 오른짝은 소문자, 왼짝은 대문자이다.

즉, Jj가 J 양말의 한 쌍이며, aB는 a양말의 오른짝과 B양말의 왼짝이므로 짝이 맞지 않는다.

## Input

입력의 첫 번째 줄은 테스트 케이스의 개수 T ( 0 < T <= 30)가 주어지며, 각 테스트 케이스는 한 줄에 하나씩 창고에 있는 양말이 종류와 짝에 상관없이 연속으로 입력된다. 최대 입력되는 양말의 개수는 100개이며, 테스트 케이스에는 알파벳 대소문자만이 입력된다.

## Output

하나의 테스트 케이스마다 두 줄씩 출력한다. 각 테스트케이스의 결과 첫 줄은 “Case #X:” 와 같이 “Case” 뒤에 한 칸 띄우고” #” 과 테스트케이스 번호 X 를 출력한 후 “:” 을 찍는다. 두 번째 줄은 입력 값에 대한 결과값을 출력한다. 주어진 양말의 모든 짝이 맞는다면 ‘YES’를 출력하며, 양말의 짝이 맞지 않는 경우는 ‘NO’를 출력한다.

| Sample Input | Output for Sample Input |
|---|---|
| 3<br>aaAA<br>CAdBb<br>Bab | Case #1:<br>YES<br>Case #2:<br>NO<br>Case #3:<br>NO |

# B. 다운로드

(Time Limit: 2 seconds)


정희는 한 음악 포털 사이트에서 일정 금액을 충전하여 mp3파일을 다운로드 받아서 사용해 왔다. 지금 이 사이트에서 k곡을 다운받으면 한 곡을 무료로 다운받을 수 있는 이벤트를 진행 중이고, 현재 정희가 다운받을 수 있는 mp3파일의 개수는 n개이다. 이벤트 기간 중 정희가 다운받을 수 있는 총 mp3파일의 개수는 몇 개인가?

## Input

첫 줄에는 테스트 케이스의 개수 T ( 0 < T <= 100000 ) 가 주어진다. 한 줄당 하나의 입력이 입력되며, 각 입력 케이스는 n과 k 두 개의 정수로 이루어져 있고, 하나의 스페이스로 구분된다. ( 0 < n < 1000000, 1 < k <= 21)

## Output

각 테스트 케이스의 결과 첫 줄은 “Case #X:”와 같이 “Case”뒤에 한 칸 띄우고 ”#”과 테스트 케이스 번호 X 를 출력한 후 “:” 을 찍고, 그 다음 줄은 각 케이스마다 정희가 다운받을 수 있는 총 mp3파일의 개수를 출력한다.

| Sample Input | Output for Sample Input |
|---|---|
| 3<br>4 3<br>10 3<br>100 5 | Case #1:<br>5<br>Case #2:<br>14<br>Case #3:<br>124 |

# C. Near Duplicate Detection

(Time Limit: 10 seconds)

문서의 유사중복을 판별하기에 문서를 통째로 비교하는 것은 효율성도 낮고 효과적이지 못하기 때문에, 보통 문서에서 의미 있는 시그너처(Signature)들을 추출한 후, 각 문서들의 시그너처를 비교하여 문서 중복 여부를 판별한다. 두 문서가 서로 공유하는 문서 시그너처가 많을 수록 유사도가 높다고 할 수 있는데, 두 문서 사이의 유사도가 지정된 유사도 임계값 이상일 경우 중복으로 판별된다.

이 문제에서 유사도를 판별하는 식은 다음과 같다.

$$sim(A,B) = |A \cap B| / |A \cup B|$$

A, B는 각각 다른 두 문서에서 추출된 시그너처들의 집합이며, 문서간 유사도는 시그너처 집합 A, B의 합집합의 원소 개수에 대한 교집합의 원소 개수의 비율로 계산한다.

## Input

첫 번째 줄에는 테스트 케이스의 개수 T가 주어지고, 이어서 T개의 테스트 케이스가 주어진다. 테스트 케이스의 첫 번째 줄에는 유사도 임계값 R (0 < R <= 1, R은 실수)과 고려할 문서의 개수 D (1 < D <= 200)가 하나의 스페이스로 구분되어 주어진다. 다음 D줄에는 한 줄에 하나씩 하나의 문서를 대표하는 문서 시그너처 목록이 시그너처 개수 S (1 < S <= 500)와 함께 주어진다. 시그너처들은 하나의 스페이스로 분리되어 주어지며, 하나의 시그너처는 알파벳으로 구성된 단어들을 콜론(':')으로 연결한 공백이 없는 하나의 스트링이다. 여기서 대소문자를 구분하지 않도록 한다. 한 문서 내에 같은 시그너처가 여러 번 추출될 수도 있다. 문제를 단순하게 하기 위해 문서에서 추출된 횟수는 고려하지 않고, 추출 여부만 고려하도록 한다. 시그너처 스트링 하나의 최대 길이는 50을 넘지 않는다.

## Output

각 테스트 케이스의 첫 번째 줄은 "Case #X:" 와 같이 "Case" 뒤에 한 칸 띄우고 "#"과 테스트케이스의 번호 X를 출력한 후 ":" 를 출력하고, 그 다음 줄부터 임계값 R 이상의 유사도를 보여 중복으로 판별된 각 문서의 쌍을 출력한다.
문서는 입력된 순서대로 1번부터 번호를 부여하여, 중복으로 판별된 각 문서 쌍을 한 줄에 하나씩 출력한다. 출력 방식은 "A-B" (A < B, A와 B는 문서번호)와 같이 빠른 번호가 앞에 오도록 하고 하이픈("-")으로 연결하여 출력한다, 중복 문서 쌍이 여러 개인 경우 앞의 문서 번호가 빠른 문서 쌍을 먼저 출력하고, 앞의 문서 번호가 같은 경우 뒤의 문서 번호가 빠른 문서 쌍을 먼저 출력한다.

## Notes

유사도의 최대 오차범위는 $\pm 10^{-6}$ 으로 한다.

예)

```
if (-1e-6 < R - SIM && R - SIM < 1e-6)
    printf("R and SIM are same.");
else
    printf("R and SIM are not same.");
```

| Sample Input | Output for Sample Input |
|---|---|
| 3<br>0.3 4<br>3 case:problem wrong:order duplicated:signature<br>1 order:wrong<br>1 cASe:PrObLem<br>3 duplicated:signature duplicated:signature dummy<br>1.0 2<br>2 a:weeklong:campain the:south:Carolina<br>3 a:rally:kick a:weeklong:campain the:south:Carolina<br>0.5 3<br>2 a:weeklong:campain the:south:Carolina<br>3 a:rally:kick a:weeklong:campain the:south:Carolina<br>2 a:weeklong:campain the:south:Carolina | Case #1:<br>1-3<br>Case #2:<br>Case #3:<br>1-2<br>1-3<br>2-3 |

# D. 휴대폰번호 정렬

(Time Limit: 2 seconds)

LK 텔레콤에 근무하는 신입사원 정희에게 부장님이 새로운 업무를 지시했다. 고객들의 휴대폰 전화번호를 정렬하는 작업이다.

휴대폰 번호는 국번 3 자리와 중간국번 3자리 또는 4자리, 그리고 뒷 자리 수 4 자리로 총 10 개 혹은 11 개의 숫자로 이루어진다.

그리고 휴대폰 국번은 010, 011, 016, 017, 018, 019 의 여섯 가지가 존재한다. 국번과 중간국번, 뒷자리 수는 하이픈("-")으로 구분된다.

부장님이 원하는 요구사항은 다음과 같다. 우선, 특정 국번 번호 순서로 정렬이 되어야 하며, 국번이 같은 경우에는 중간국번 숫자끼리 비교해서 오름차순 정렬이 되어야 하며, 중간국번도 같은 경우에는 뒷자리 번호를 비교해서 오름차순으로 정렬이 되야 한다.

**단, 부장님이 원하는 정렬순서는 국번이 같을 경우 중간국번 4 자리인 휴대폰 번호가 중간국번 3 자리인 휴대폰 번호보다 항상 정렬 우선순위가 높다. 즉, 국번이 같은 경우 중간국번 4 자리인 휴대폰 번호가 중간국번 3자리인 휴대폰 번호보다 정렬 시 항상 먼저 등장한다.**

예를 들어, 다음과 같은 휴대폰 번호를 정렬한다고 하자.

011-275-3587
017-1111-2600
019-222-2222
017-111-1234
018-275-9562
010-333-1111
016-1235-3333

이 번호들을 017 / 011 / 018 / 019 / 010 / 016 국번 순으로, 그리고 중간국번 4 자리가 중간국번 3자리보다 우선순위가 높도록 정렬을 한다면, 아래와 같이 정렬이 될 것이다.

017-1111-2600
017-111-1234
011-275-3587
018-275-9562
019-222-2222
010-333-1111
016-1235-3333

위의 결과에서 첫 번째 라인의 017-1111-2600 과 두 번째 라인의 017-111-1234 를 비교해 보면, 중간국번 111 은 1111 보다 작으므로 일반적으로는 정렬 시 017-111-2600 이 먼저 등장해야 하나, 부장님의 요구사항에서 **중간국번 4 자리가 중간국번 3 자리보다 항상 정렬 우선순위가 높다**는 규칙 때문에 017-1111-2600 이 017-111-1234 보다 먼저 등장한 것을 알 수 있다.

## Input

첫 줄에는 테스트 케이스의 개수 T ( 0 < T <= 20 ) 가 주어진다.
각각의 테스트케이스의 첫 번째 줄에는 정렬할 휴대폰 번호의 개수 N(0 < N <= 50) 이 주어진다.
테스트케이스의 두 번째 줄에는 0, 1, 6, 7, 8, 9 의 6개의 숫자가 임의의 순서로 표시된다.
이 숫자들은 휴대폰 국번을 의미하며, 6개의 숫자들이 나열되는 순서는 테스트 케이스마다 바뀔 수 있다. 당신은 이 순서대로 국번 별 순서로 휴대폰번호를 정렬해야 한다.
예를 들어, 숫자 7 1 8 9 0 6 이 표시됐다면 017 / 011 / 018 / 019 / 010 / 016 국번 순으로 정렬해야 한다.
세 번째 줄부터는 라인 당 하나씩의 휴대폰 번호가 N 개 만큼 입력된다. 휴대폰 번호에서 국번은 010, 011, 016, 017, 018, 019 의 여섯 가지만 입력되며, 휴대폰 번호는 0 부터 9 까지의 숫자와 "-" 로 구성되며, 빈칸을 포함하지 않으며 그 외 다른 문자는 포함되지 않는다.
참고로, 우리나라 휴대폰 번호에서 010 국번의 경우 중간국번은 항상 4 자리지만 이 문제에서는 010 국번도 중간 국번이 3 자리가 될 수 있다고 정의한다.
즉, 모든 휴대폰번호는 국번에 상관없이 중간국번이 세자리 혹은 네 자리가 될 수 있다.

## Output

각 테스트케이스의 결과 첫 줄은 "Case #X:"와 같이 "Case"뒤에 한 칸 띄우고 "#"과 테스트 케이스 번호 X 를 출력한 후 ":" 을 찍고, 그 다음 라인부터 휴대폰번호가 정렬된 결과를 출력한다. 출력 포맷은 아래의 출력 샘플을 참조한다.

| Sample Input | Output for Sample Input |
|---|---|
| 3<br>7<br>7 1 8 9 0 6<br>011-275-3587<br>017-1111-2600<br>019-222-2222<br>017-111-1234<br>018-275-9562<br>010-333-1111<br>016-1235-3333<br>2<br>9 8 7 6 1 0<br>011-1234-1234<br>011-1234-1194<br>1<br>0 1 6 7 8 9<br>010-8940-8767 | Case #1:<br>017-1111-2600<br>017-111-1234<br>011-275-3587<br>018-275-9562<br>019-222-2222<br>010-333-1111<br>016-1235-3333<br>Case #2:<br>011-1234-1194<br>011-1234-1234<br>Case #3:<br>010-8940-8767 |

# E. Random Network

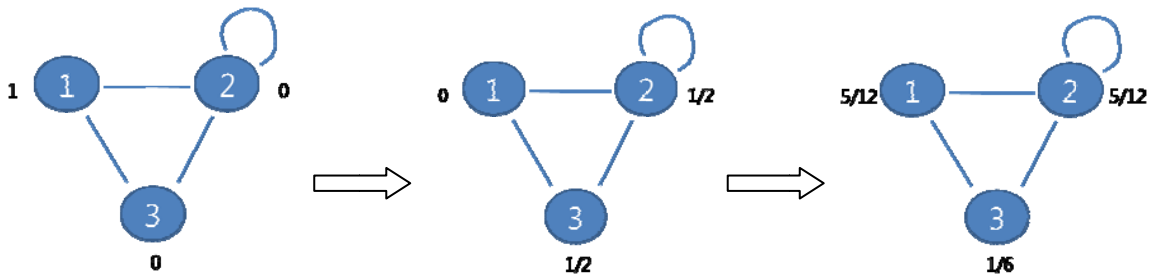
(Time Limit: 10 seconds)

Podol is a student of Soongsil University (SSU). He is currently taking care of network maintenance in SSU for his part-time job. The network consists of **N** nodes and **M** links. Each node is numbered from 1 through **N**. Several days after he took the job, he started to wonder if he sends out a packet at node 1, what would be the probability that the packet finally arrives at node **N** after taking exactly **K** steps. He knows the topology of whole network. But, unfortunately he isn't good at math. Maybe, you, one of the greatest programmers in SSU and a math lover, are going to help him, aren't you?

You can assume that the following:

- A packet must pass exactly one link for each step.
- A packet travels from its current node to a randomly selected connected node. All random selections are made with equal probability.
- All links are bidirectional. So, if there is a link between node **p** and node **q**, the packet will be able to travel from **p** to **q** and also **q** to **p**.
- More than one link can exist between two nodes.
- A node can have one or more links to itself. That means self-edges are possible. So, the packet can stay at node **u** only by passing through a self-edge of node **u**
- Maybe not all nodes are connected. That means, sometimes the packet can't travel some nodes.

Ex)




Let's assume that we have a network with 3 nodes and 4 links as above (**N** = 3, **M** = 4). The picture shows the probabilities that the packet stays at each node for each step when **K** = 2.

At step 0, the packet begins at node 1 initially. So, the probability that the packet stays at node 1 is 1 and they are all 0 at the other nodes.

At step 1, node 1 sends the packet to node 2 or node 3 with 1/2 probability each. So, the probability that the packet stays at node 2 is 1/2 and it's also same at node 3.

At step 2, node 2 sends the packet to node 1 or node 3 or node 2 itself with 1/3 probability each. And also, node 3 sends the packet to node 1 or node 2 with 1/2 probability each. So, the probability that the packet stays at node 1 is 5/12 (1/2 * 1/3 from node 2 and 1/2 * 1/2 from node 1). And the probability at node 2 is also 5/12 (0 * 1/2 from node 1 and 1/2 * 1/2 from node 3 and 1/2 * 1/3 from node 2 itself). The probability at node 3 is 1/6 (0 * 1/2 from node 1 and 1/2 * 1/3 from node 2).

As a result, the probability that the packet is at node 3 after taking exactly 2 steps is 1/6.

## Input

The first line of the input contains a single integer **T** (1 <= **T** <= 100) denoting the number of test cases. This is followed by the input data for each test case. The first line of each test case contains three integers **N**, **M**, and **K** (1 <= **N, M, K** <= 100) separated by a space. **N** and **M** are denoting the number of nodes and the number of links respectively. This is followed by **M** lines. Each of the lines contains two integers, **p** and **q** (1 <= **p**, **q** <= **N**) separated by a space which means there is a link between node **p** and node **q**.

## Output

Print exactly 2 lines for each test case. The first line should contain "Case #**X**:" where **X** is the number of test case starting from 1. The next line should contain the probability that the packet is at node **N** after taking exactly **K** steps. The packet always begins at node 1. The probability should be rounded to 4 digits after the decimal point.

| Sample Input | Output for Sample Input |
|---|---|
| 3<br>3 4 2<br>1 2<br>1 3<br>2 2<br>2 3<br>3 4 2<br>2 3<br>2 3<br>3 2<br>2 1<br>2 1 10<br>1 1 | Case #1:<br>0.1667<br>Case #2:<br>0.7500<br>Case #3:<br>0.0000 |

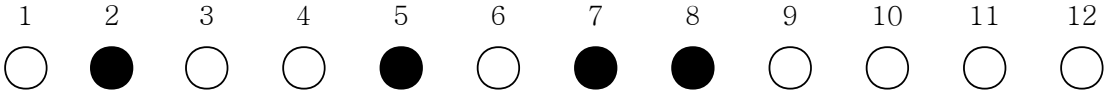
# F. 자갈게임

(Time Limit: 2 seconds)

지금부터 재미있는 게임을 하나 소개하겠다. 여러분은 조그마한 구멍 12개가 일렬로 뚫려 있는 게임판을 하나 받을 것이다. 각 구멍에 검은색 자갈이 임의로 배치됨으로써 게임은 시작된다. 이 게임의 최종 목표는 검은색 자갈의 개수를 최소로 만드는 것이고, 게임의 룰은 다음과 같다.

- 연속된 세 개의 구멍 중, 첫 번째와 두 번째 구멍에 자갈이 있고, 세 번째 구멍에 자갈이 없다면 첫 번째 구멍의 자갈을 세 번째 구멍으로 옮길 수 있으며 이 때 두 번째 구멍의 자갈은 빼낸다.
- 위와 같이 두 번째와 세 번째 구멍에 자갈이 있고, 첫 번째 구멍에 자갈이 없다면 세 번째 구멍의 자갈을 첫 번째 구멍으로 옮길 수 있으며 이 때 두 번째 구멍의 자갈은 빼낸다.
- 더 이상 빼낼 자갈이 없을 때까지 위의 두 룰을 반복해서 수행한다.

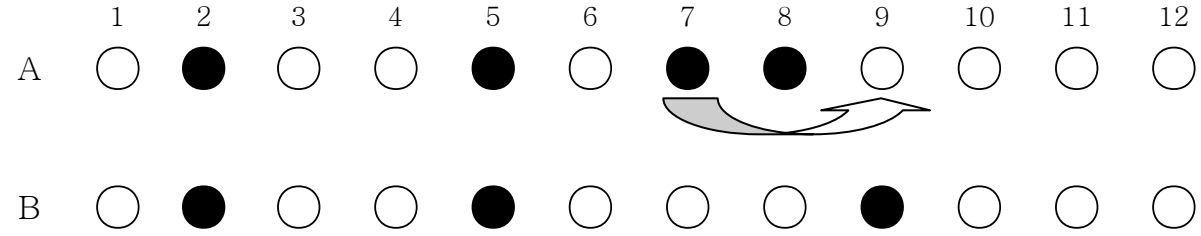
**예 – 초기상태)**




*○는 비어있는 구멍, ●는 자갈*

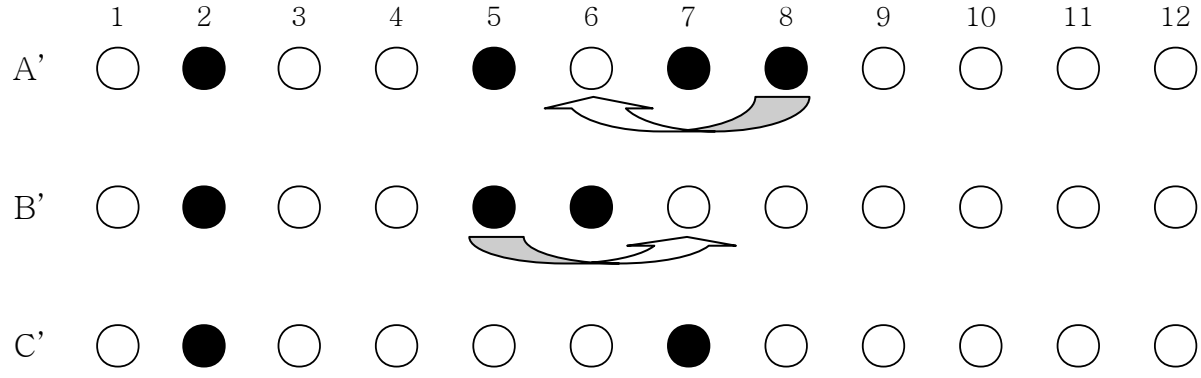
위 예는 8번 자갈을 6번 구멍으로 옮기는 방법과 7번 자갈을 9번 구멍으로 옮기는 방법이 있다.

**실행방법 1)**




실행방법 1로 옮긴 경우를 보자. A 상황에서 7번 자갈을 9번 구멍으로 옮긴 후 8번 자갈을 제거한다. 첫 번째 자갈을 제거 후 B 상황이 되어 더 이상 자갈을 빼낼 수 없기 때문에 남은 자갈의 수는 3개로 게임이 끝난다.

**실행방법 2)**



실행방법 2는 8번 자갈을 6번 구멍으로 옮긴다. 그리고 7번 자갈을 제거하면 B상황이 되고, 5번 자갈을 7번으로 옮겨서 하나의 자갈을 더 빼내어 C상황을 만들 수 있게 된다. 최종으로 남는 자갈의 수는 2개로 게임이 끝난다.

이와 같이 **자갈을 옮기는 방법과 순서**에 따라 게임의 결과가 다르게 나올 수 있다.

## Input

첫 줄에는 테스트 케이스의 개수 T ( 0 < T <= 4096 ) 가 주어진다. 하나의 케이스당 한 줄에 표시되며, 게임판의 초기상태를 나타내는 12개의 캐릭터로 이루어져 있다. 게임판의 상태는 비어있는 구멍은 'O'(알파벳 15번째 대문자),'@'는 자갈을 표시한다.

## Output

하나의 케이스 당 출력은 두 줄로 이루어져 있다. 첫 줄은 "Case #X:"와 같이 "Case"뒤에 한 칸 띄우고"#"과 테스트케이스 번호 X 를 출력한 후 ":" 을 찍고, 그 다음 라인엔 각 케이스마다 게임이 끝난 후 남을 수 있는 최소의 자갈 개수를 출력한다.

| Sample Input | Output for Sample Input |
|---|---|
| 5<br>OOO@@OOOOOOO<br>O@OO@O@@OOOO<br>O@OOOO@@@OOO<br>@@@@@@@@@@@@<br>@@@@@@@@@@O@ | Case #1:<br>1<br>Case #2:<br>2<br>Case #3:<br>3<br>Case #4:<br>12<br>Case #5:<br>1 |

# G. Helloneo's TopCoder Entry Problem

(Time Limit: 10 seconds)

Kyoowook (aka **helloneo**) is one of the students preparing for ACM-ICPC, the world's biggest programming contest. He solved many problems at UVa Online Judge and read many algorithm books. But he always had a trouble with many DP (Dynamic Programming) problems and no textbook could help him solve those problems. So, he asked one of the world finalists of ACM-ICPC, Jinho (aka **astein**) for some advices. Let's take a look at the following conversation.

**astein**: Do you know Dynamic Programming?
**helloneo**: Yes, of course, I know 0/1 Knapsack, Longest Common Subsequence, Matrix Chain Multiplication, ...
**astein**: Then, given a matrix A of N by N elements, with N <= 16, how would you find a permutation P such that sum{i=0...N-1} A[i][P[i]] is minimized where each element of P is an integer between 0 and N-1 and all distinct?
**helloneo**: **thinking**
**helloneo**: Just try each possible permutation and keep the one that returns the minimum answer.
**astein**: That gives Time Limit Exceeded, you have to use DP.
**helloneo**: How?
**astein**: Use a bitmask to memorize the previously calculated values.
**helloneo**: Wow. That makes sense.
**astein**: Go to TopCoder (http://www.topcoder.com/tc), and you will learn real DP.

Since then, **helloneo** has been practicing in TopCoder and his ability has increased fast and now, he really knows DP. ☺

For now, **helloneo** as a problem setter of Soongsil Univ. Campus Programming Contest, he would like you to solve the problem introduced by **astein**. Can you solve it?

Ex)

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 5 | 1 | 9 | 9 |
| 1 | 8 | 4 | 2 | 3 |
| 2 | 5 | 9 | 3 | 9 |
| 3 | 2 | 7 | 6 | 5 |

Figure 1

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 5 | 1 | 9 | 9 |
| 1 | 8 | 4 | 2 | 3 |
| 2 | 5 | 9 | 3 | 9 |
| 3 | 2 | 7 | 6 | 5 |

Figure 2

Figure 1 shows that given 4 by 4 matrix A, if you choose permutation P = {1, 2, 0, 3}, the sum would be 13. (A[0][1] + A[1][2] + A[2][0] + A[3][3] = 13)

But, if you choose permutation P = {1, 3, 2, 0}, the sum would be 9. Figure 2 shows this scenario. (A[0][1] + A[1][3] + A[2][2] + A[3][0] = 9)

As a result, permutation {1, 3, 2, 0} is better than {1, 2, 0, 3} and there is no other permutation better than {1, 3, 2, 0}. Therefore, the minimum sum would be 9.

## Input

The first line of the input contains a single integer T (1 <= T <= 100) denoting the number of test cases. This is followed by the input data for each test case. The first line of each test case contains a single integer N (1 <= N <= 16). N denotes the dimension of the matrix A. Next follows N lines and each contains N integers separated by a space, which represent the matrix A. Each element of the given matrix A is a non-negative integer less than or equal to 1000 (0 <= $A_{ij}$ <= 1000, 0 <= i, j <= N-1).

## Output

Print exactly 2 lines for each test case. The first line should contain "Case #X:" where X is the number of test case starting from 1. The next line should contain the minimum possible value of the following expression where $P_i$ is the i'th element of a permutation P of N integers 0,…, N-1.

$$\sum_{i=0}^{N-1} A_{iP_i}$$

## Notes

There are alternative solutions other than DP. ☺

| Sample Input | Output for Sample Input |
|---|---|
| 2<br>4<br>5 1 9 9<br>8 4 2 3<br>5 9 3 9<br>2 7 6 5<br>5<br>1 2 3 4 5<br>1 2 3 4 5<br>1 2 3 4 5<br>1 2 3 4 5<br>1 2 3 4 5 | Case #1:<br>9<br>Case #2:<br>15 |