Universidade Federal de Pernambuco
Centro de Informática

Final da Seletiva 2015

Maratona de Programação

28 de fevereiro de 2015

Caderno de Problemas

Este caderno contém 10 problemas; as páginas estão numeradas de 1 a 14, contando esta página de rosto. Verifique se o caderno está completo.

Informações Gerais

A) Sobre a entrada

1) A entrada do seu programa deve ser lida da *entrada padrão* (*stdin*).
2) A entrada é composta por vários casos de teste, cada um descrito em um número de linhas que depende do problema. A primeira linha da entrada é sempre o número de casos de teste.
3) Quando uma linha da entrada contém vários valores, estes serão separados por um único espaço em branco. Não haverá nenhum outro espaço em branco.
4) Cada linha, incluindo a última, contém o caractere final-de-linha.

B) Sobre a saída

1) A saída do seu programa deve ser escrita na *saída padrão* (*stdout*).
2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco. Não deverá haver nenhum outro espaço em branco.
3) Cada linha, incluindo a última, deve conter o caractere final-de-linha.

Problema A

André e os Mentos

André é um maratonista do CIn-UFPE. Todo sábado, durante os treinos, ele come de tudo: salgadinho, refrigerante, biscoito, água e mentos. Principalmente mentos. Mas o problema, porém, é que toda vez que André vai tirar alguns mentos do tubo, ele tem que parar de codar por alguns instantes, o que atrapalha sua concentração.

O mentos vem em um tubo com duas pontas. Cada vez que André quer chupar alguns, ele escolhe um certo sabor, e olha pra cada ponta do mentos. Em cada uma, se houver um mentos do sabor escolhido, ele pega. Se não houver nenhum daquele sabor nas pontas, ele não pega nenhum, e só parou de codar à toa. Para diminuir a perda de tempo durante o contest, André decidiu minimizar suas paradas para pegar mentos. Ele fez um corte fino ao longo do tubo, para poder ver com antecedência quais sabores tem dentro dele. Mas ele não vai pegar do meio, e fez isso apenas para poder decidir melhor quais sabores irá escolher tirar das pontas em cada uma de suas paradas.

Agora, André precisa calcular o número mínimo de vezes que ele deve parar para pegar seus mentos, seguindo o método descrito, até eles acabarem. Ele calcularia isso facilmente usando Transformada de Fourier, mas ele está ocupado codando uma questão. Por isso cabe a você, um companheiro de time dele, fazer isso para ajudá-lo.

Entrada

A primeira linha contém um inteiro **T** ($1 \leq$ **T** ≤ 200), o número de casos de teste. Cada caso de teste começa com uma linha com um inteiro **N**, o número de mentos do tubo ($1 \leq$ **N** ≤ 1000). Na linha seguinte, há **N** inteiros, o **i**-ésimo deles é o número do sabor do **i**-ésimo mentos no tubo. Cada um desses números está entre 1 e 10^9.

Saída

Para cada caso imprima uma linha contendo "Caso #**X**: **Y**", onde **X** é o número do caso atual, iniciando em 1, e **Y** é a quantidade mínima de vezes que André precisa parar para pegar mentos.

Exemplo de Entrada	**Exemplo de Saída**
3 5 1 3 1 3 2 5 1 2 3 2 1 7 1 1 2 3 3 4 2	Caso #1: 4 Caso #2: 3 Caso #3: 5

Explicação do primeiro exemplo: Uma sequência ótima de sabores para André pegar é:
{1 3 1 3 2}: pega o sabor 1
{3 1 3 2}: pega o sabor 2
{3 1 3}: pega o sabor 3
{1}: por último, pega o sabor 1 de novo

Autor: *Mário Henrique*

Problema B

Baile de Formatura

A turma de Ciência da Computação do CIn-UFPE de 2025.1 está se formando! É uma formatura muito especial, não só porque todos os projetões dos alunos desta turma viraram multinacionais, mas também porque o número 2025 é um quadrado perfeito! Por isso, os alunos decidiram tornar todos os números da cerimônia quadrados perfeitos: datas, quantidade de convidados, hash do nome da turma, até a quantidade de formandos (roleta russa FTW!).

Os organizadores da festa estavam conseguindo atender a essa exigência, até chegar a hora de comprar os salgados. Eles vinham em caixas com **N** salgados de uma vez. Se **N** não for um quadrado perfeito, terão que comprar mais de uma caixa. Calcule o número mínimo de salgados que eles devem comprar para atender à demanda excêntrica dos formandos.

Entrada

A primeira linha contém um inteiro **T** ($1 \leq \mathbf{T} \leq 1000$), o número de casos de teste. Cada uma das próximas T linhas contém um número **N** ($1 \leq \mathbf{N} \leq 10^9$), o número de salgados que vem numa caixa só.

Saída

Para cada caso imprima uma linha contendo "Caso #**X**: **Y**", onde **X** é o número do caso atual, iniciando em 1, e **Y** é o número mínimo de salgados que eles devem comprar.

Exemplo de Entrada	**Exemplo de Saída**
5 5 9 10 12 13	Caso #1: 25 Caso #2: 9 Caso #3: 100 Caso #4: 36 Caso #5: 169

Autor: Mário Henrique

Problema C

Calçada da Fama

Na calçada da fama, há várias estrelas no chão com os nomes dos artistas. Alguns vândalos, talvez por inveja ou talvez pelo simples fato de querer vandalizar, estavam pichando várias dessas estrelas e colocando outros nomes no lugar:

O prefeito, afim de tentar minimizar esse problema, instalou várias câmeras nessa calçada. A calçada pode ser vista como um segmento [1..**N**], onde cada posição possivelmente se encontra uma estrela.

Cada câmera protege um segmento [**a**..**b**], inclusivos. Deseja-se saber, para cada estrela, se elas estão cobertas por uma câmera ou não. Você foi contratado para fazer esse trabalho.

Entrada

A primeira linha da entrada contém **T** ($1 \leq \mathbf{T} \leq 100$), o número de casos de teste. Cada caso de teste começa com dois inteiros **N** e **C** ($1 \leq \mathbf{N} \leq 10^9$, $1 \leq \mathbf{C} \leq 10^4$), o tamanho da calçada e o número de câmeras, respectivamente. A seguir há **C** linhas, cada uma descrevendo uma câmera **i** com dois inteiros $\mathbf{a_i}$ e $\mathbf{b_i}$ ($1 \leq \mathbf{a_i} \leq \mathbf{b_i} \leq \mathbf{N}$), representando o intervalo coberto pela câmera. A seguir, há um número **E** ($1 \leq \mathbf{E} \leq 10^4$), o número de estrelas. A seguir há uma linha com **E** inteiros $\mathbf{x_i}$, indicando a posição da estrela **i** na calçada ($1 \leq \mathbf{x_i} \leq N$).

Saída

Para cada caso imprima “Caso **#X**: **Y**”, onde **X** é o número do caso atual, começando em 1, e **Y** é o número de estrelas que estão cobertas por alguma câmera.

Exemplo de Entrada	**Exemplo de Saída**
2 10 3 1 3 1 4 7 8 3 2 5 8 2 1 1 2 3 1 1 2	Caso #1: 2 Caso #2: 3

Autor: Gustavo Stor

Problema D
Defesa ao Grafo

Tower Defense é um famoso jogo de estratégia onde o jogador deve posicionar torres de defesa para proteger algo - seja um castelo, um tesouro ou até você mesmo - contra uma horda de monstros. Há várias variações do jogo: em alguns tipos, o mapa se assemelha a um tabuleiro, e os monstros tem um caminho especifico a seguir; em outros tipos, o mapa é aberto e os monstros podem chegar ao destino final por vários meios diferentes.

Graph Defense é uma variação do Tower Defense comum. Aqui, o mapa é representado como um grafo de **N** vértices e **M** arestas. Cada vértice é uma posição em que um monstro ou uma torre (ou ambos) podem estar, em um dado momento, e as arestas representam conexões bidirecionais entre esses vértices (i.e. se há uma aresta de **u** para **v**, um monstro que está no vértice **u** em um dado momento pode ir para o vértice **v** no momento seguinte e vice-versa). O castelo, que você deseja proteger, se encontra no vértice **F**.

Cada torre **i** possui um alcance $\mathbf{C_i}$, um ataque $\mathbf{A_i}$ e está no vértice $\mathbf{V_i}$. Todos os vértices que estão a no máximo $\mathbf{C_i}$ arestas de distância de $\mathbf{V_i}$ receberão $\mathbf{A_i}$ de dano a cada unidade de tempo. As torres não se movem, e existem desde o início do jogo. O castelo possui um escudo mágico protetor que faz com que nenhuma torre consiga atacar o vértice **F** onde ele se encontra, tampouco propagar o ataque, ou seja, o vértice **F** é uma barreira e nada passa por ele, a não ser os monstros, possivelmente.

Cada monstro **i** surge durante o decorrer do jogo em um vértice $\mathbf{K_i}$ e possui $\mathbf{H_i}$ pontos de vida. Os monstros nunca ficam parados e, a cada unidade de tempo, se movem para um vértice adjacente. Eles sempre vão seguir para o destino final, o castelo, pelo caminho que causará o menor dano possível. Os monstros morrem quando alcançam 0 ou menos pontos de vida. Um monstro só consegue invadir o castelo quando chega ao destino **F** vivo. Se houver uma torre que alcança a posição inicial $\mathbf{K_i}$ do monstro, ela irá inflingir dano já no primeiro instante em que o monstro surge. Um monstro pode surgir já no castelo.
Você foi contratado para fazer uma simulação do jogo. Depois de todas as aparições de monstros, quantos conseguiram invadir o castelo ainda com vida?

Entrada

A primeira linha da entrada contém **T** ($1 \leq \mathbf{T} \leq 100$), o número de casos de teste. Cada caso de teste começa com três inteiros **N** ($1 \leq \mathbf{N} \leq 1000$), **M** ($0 \leq \mathbf{M} \leq (\mathbf{N}*(\mathbf{N}-1))/2$) e **F** ($1 \leq \mathbf{F} \leq \mathbf{N}$), o número de vértices, arestas e o vértice em que se encontra o castelo, respectivamente. A seguir há **M** linhas, cada uma com dois inteiros **u** ($1 \leq \mathbf{u} \leq \mathbf{N}$) e **v** ($1 \leq \mathbf{v} \leq \mathbf{N}$ e $\mathbf{v} \mathrel{!}= \mathbf{u}$), indicando a existência de uma aresta que liga os vértices **u** e **v**. Não haverá mais de uma aresta entre um mesmo par de vértices. A seguir há um número **P** ($0 \leq \mathbf{P} \leq 100$), indicando o número de torres. Cada uma das próximas **P** linhas conterá três inteiros $\mathbf{V_i}$ ($1 \leq \mathbf{V_i} \leq \mathbf{N}$ e $\mathbf{V_i} \mathrel{!}= \mathbf{F}$), $\mathbf{A_i}$ ($1 \leq \mathbf{A_i} \leq 10^5$), e $\mathbf{C_i}$ ($1 \leq \mathbf{C_i} \leq 1000$), indicando que a **i**-ésima torre se encontra no vértice $\mathbf{V_i}$ com $\mathbf{A_i}$ de ataque e $\mathbf{C_i}$ de alcance, conforme explicado na descrição do problema. Pode haver mais de uma torre no mesmo vértice, e não haverá nenhuma torre no vértice **F**. Por fim, haverá um inteiro **Q** ($1 \leq \mathbf{Q} \leq 10^4$), indicando o número de monstros. Cada uma das próximas **Q** linhas contém dois inteiros $\mathbf{K_i}$ ($1 \leq \mathbf{K_i} \leq \mathbf{N}$) e $\mathbf{H_i}$ ($1 \leq \mathbf{H_i} \leq 10^8$), indicando o vértice onde o **i**-ésimo monstro nasce e a quantidade de pontos de vida que ele tem no começo, respectivamente. É garantido que existe pelo menos um caminho que, não fosse pelos ataques das torres, o monstro conseguiria chegar ao castelo.

Saída

Para cada caso imprima “Caso **#X**: **Y**”, onde **X** é o número do caso atual, começando em 1, e **Y** é o número de monstros que conseguiram chegar ao castelo com vida.

Exemplo de Entrada	**Exemplo de Saída**
2 1 0 1 0 3 1 3 1 2 1 1 9 8 1 1 2 2 3 3 4 3 5 4 7 5 6 8 4 9 5 2 6 2 3 7 4 2 9 1 15 2 2 3 9 4 14 5 11 6 50 7 20 8 15 9 15	Caso #1: 3 Caso #2: 6

No segundo caso, os monstros que conseguem chegar vivos ao castelo são os monstros que surgem nos nós: 1, 3, 5, 6, 7 e 9.

Autor: Gustavo Stor

Problema E

Estimando a Média

Guga fez **N** provas em toda sua vida acadêmica. Agora, perto de se formar, ele quer saber qual foi o maior período de tempo contíguo em que ele possuiu a maior média aritmética.

Entrada

A primeira linha contém um inteiro **T** ($1 \leq \mathbf{T} \leq 100$), o número de casos de teste. Cada caso começa com uma linha com um número **N** ($1 \leq \mathbf{N} \leq 10^5$), o número de provas que Guga realizou em toda sua vida acadêmica. Em seguida, há uma linha com **N** inteiros $\mathbf{P_i}$ ($0 \leq \mathbf{P_i} \leq 10000$), o **i**-ésimo inteiro representa a nota da **i**-ésima prova.

Saída

Para cada caso imprima uma linha contendo "Caso #**X**: **Y**", onde **X** é o número do caso atual, iniciando em 1, e **Y** é o tamanho da maior sequência de provas que contém a maior média obtida por Guga.

Exemplo de Entrada	**Exemplo de Saída**
2 2 8 8 1 0	Caso #1: 2 Caso #2: 1

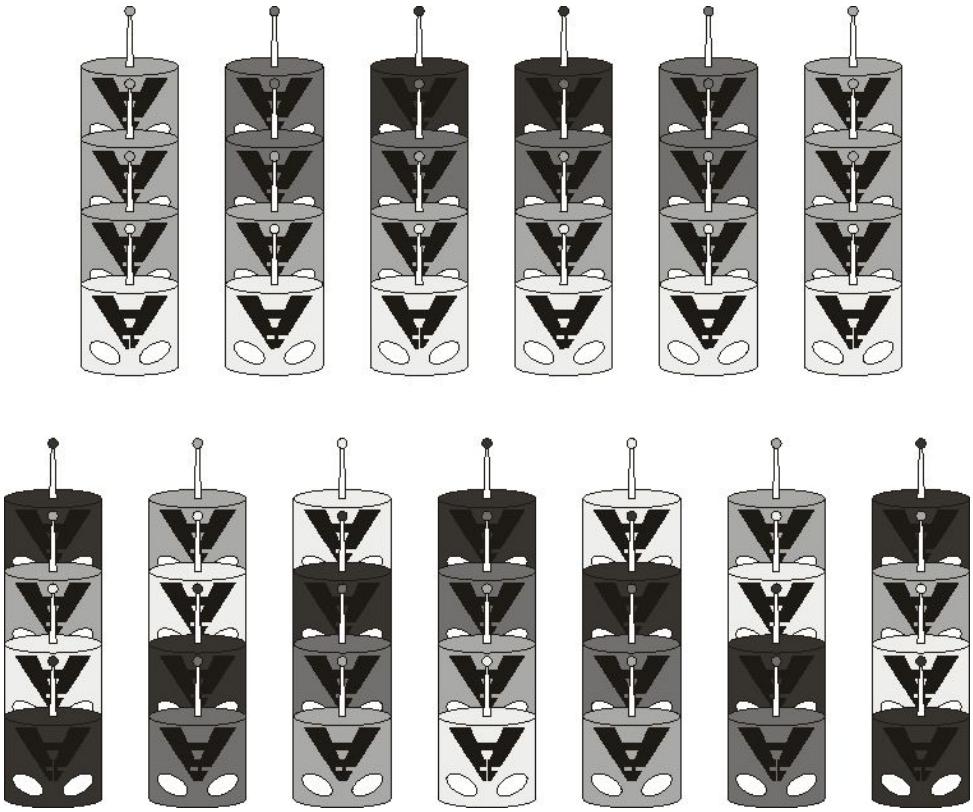
Autor: Duhan Caraciolo

Problema F

Formação de Robôs

A inventora de Heitor Ado, a doutora Ruína Balística, terminou a construção de um novo exército de robôs, e ele está pronto para ser enviado para conquistar o mundo. Os robôs possuem **N** cores diferentes, cada uma demonstrando o tipo de armamento que ele usa. Heitor mandou você, um de seus lacaios, organizá-los em formação, ou seja, em várias fileiras lado a lado, formando uma matriz. Heitor lhe deu certas regras para isso:

- Deve haver no mínimo duas fileiras
- Todas as fileiras devem ter a mesma quantidade de robôs
- Considerando as cores dos robôs, toda a formação deve ser simétrica em relação a um eixo central paralelo às fileiras



As imagens acima mostram duas possíveis organizações dos robôs para diferentes quantidades de cores. As fileiras são dispostas verticalmente, e na primeira imagem o eixo simétrico paralelo às fileiras passa entre as duas fileiras do centro; já na segunda imagem, o eixo simétrico passa pelo centro da fileira central.

Se você não for capaz de organizar os robôs da forma como Heitor pediu, a doutora Ruína irá dissecá-lo e transformá-lo em um deles. Dadas as quantidades de robôs de cada cor, decida se você pode cumprir a ordem dele ou se deve fugir enquanto ainda há tempo.

Entrada

A primeira linha contém um inteiro **T** ($1 \leq$ **T** ≤ 1000), o número de casos de teste. Cada caso começa com uma linha com um número **N** ($1 \leq$ **N** ≤ 100), o número de cores diferentes. Em seguida, há uma linha com **N** inteiros $\mathbf{A_i}$ ($1 \leq \mathbf{A_i} \leq 1000$), o número de robôs com a **i**-ésima cor.

Saída

Para cada caso imprima uma linha contendo "Caso #**X**: **Y**", onde **X** é o número do caso atual, iniciando em 1, e **Y** é a string “Challenge Accepted!”, se for possível organizar os robôs do jeito que Heitor quer, ou “Run for your life!”, caso contrário.

Exemplo de Entrada	**Exemplo de Saída**
4 1 50 2 10 10 3 2 2 5 3 2 3 3	Caso #1: Challenge Accepted! Caso #2: Challenge Accepted! Caso #3: Challenge Accepted! Caso #4: Run for your life!

Autor: Mário Henrique

Problema G

Guga e a String

Guga tem uma string **S** contendo apenas letras minúsculas e quer fazer operações nela. Cada operação pode ser de um dos seguintes tipos:

- 0 x, deslocar cada vogal de **S** x posições da esquerda pra direita (voltando para o começo, caso necessário)
- 1 x, deslocar cada consoante de **S** x posições da esquerda pra direita (voltando para o começo, caso necessário)
- 2, imprimir como **S** se encontra atualmente

*Guga considera que as vogais são **a**, **e**, **i**, **o** e **u**.*
Uma operação do tipo 0 só desloca vogais por posições de **S** que possuem vogais.
Uma operação do tipo 1 só desloca consoantes por posições de **S** que possuem consoantes.

Por exemplo,
A string “computador” após a operação 1 2 fica “dorcumapot”, ou seja, cada consoante vai para a posição em S da segunda próxima consoante.
A string “abe” após a operação 0 1, fica “eba”.

Entrada
A primeira linha da entrada contém **T** ($1 \leq \mathbf{T} \leq 100$), o número de casos de teste. A primeira linha de cada caso de teste possui **S** ($1 \leq |\mathbf{S}| \leq 10^4$), a string que Guga possui. A segunda linha de cada caso possui **Q** ($1 \leq \mathbf{Q} \leq 10^5$) , o número de operações que Guga irá executar em **S**. Cada uma das próximas **Q** linhas possuem uma operação como explicado acima. Para cada operação, $0 \leq \mathbf{x} \leq |\mathbf{S}|$.

Saída
Para cada caso imprima “Caso #**X**:”, onde **X** é o número do caso atual, começando em 1. Para cada operação 2, imprima em uma nova linha como a string **S** se encontra depois de todas as operações anteriores terem sido executadas. A saída possui aproximadamente $3*10^6$ caracteres.

Exemplo de Entrada	**Exemplo de Saída**
2 computador 2 0 2 2 abecidfugh 4 2 0 2 1 3 2	Caso #1: campotodur Caso #2: abecidfugh ifugahbecd

Autor: Duhan Caraciolo

Problema H
Honorável Presente

Guga ganhou um grafo conexo de aniversário, com **N** nós e **N**-1 arestas bidirecionais. Cada aresta conecta dois nós e possui um peso. Quando André descobriu a existência do presente de Guga pensou na seguinte brincadeira: Dado um número inteiro **X**, quantos pares (**A**,**B**) (**A** $\leq$ **B**) existem tal que o menor caminho do nó **A** para o nó **B** possui apenas arestas com peso menor ou igual a **X**?
Agora Guga e André estão precisando de um programa que responda essa pergunta, para que assim eles possam brincar infinitamente e saber se acertaram a resposta ou não.

Entrada
A primeira linha da entrada contém um inteiro **T** ($1 \leq$ **T** ≤ 50), o número de casos de teste. A primeira linha de cada caso de teste contém **N** ($1 \leq$ **N** $\leq 10^5$), o número de nós que o grafo de Guga possui. Cada uma das **N**-1 linhas possui três inteiros **A**, **B** e **C** ($1 \leq$ **A**, **B** $\leq$ **N**, $1 \leq$ **C** $\leq 10^6$), indicando que existe uma aresta do nó **A** para o nó **B** com peso **C**. A próxima linha contém um inteiro **X** ($1 \leq$ **X** $\leq 10^6$), o maior peso permitido no caminho, como explicado acima.

Saída
Para cada caso imprima "Caso #**C**: **Y**", onde **C** é o número do caso atual, iniciando em 1, e **Y** é a resposta da questão.

Exemplo de Entrada	**Exemplo de Saída**
3 3 1 2 2 1 3 2 2 4 1 2 3 2 3 5 3 4 7 6 1 10	Caso #1: 6 Caso #2: 7 Caso #3: 1

Autor: Duhan Caraciolo

Problema I
Interessante Rotina de Krapekar

O inteiro 6174 é conhecido como a constante de Krapekar em homenagem ao matemático indiano Dattathreya Ramachandra Kaprekar. Esse número é interessante graças ao fato que se **X** é um número de 4 dígitos (zeros iniciais são permitidos para completar os 4 dígitos) em que todos os dígitos não são iguais entre si, a rotina de Krapekar iniciando no número **X** sempre converge para 6174. Ou seja, a rotina de Krapekar converge para 6174 se, e somente se, **X** possui 4 dígitos com pelo menos dois deles diferentes entre si. A rotina de Krapekar é executada da seguinte forma:

```
int krapekar(int X) {
    int cnt = 0;
    while (X != 6174) {
        int maior = maior_numero_com_digitos_de(X);
        int menor = menor_numero_com_digitos_de(X);
        X = maior - menor;
        cnt = cnt + 1;
    }
    return cnt;
}
```

`maior_numero_com_digitos_de(X)` é o maior número que pode ser formado usando-se os dígitos de X.
`menor_numero_com_digitos_de(X)` é o menor número que pode ser formado usando-se os dígitos de X.

Por exemplo:
`maior_numero_com_digitos_de`(3524) = 5432
`menor_numero_com_digitos_de`(3524) = 2345
`maior_numero_com_digitos_de`(10) = 1000 //pois 10 = 0010 com quatro dígitos
`menor_numero_com_digitos_de`(10) = 1

Entrada

A primeira linha da entrada contém **T** ($1 \leq \mathbf{T} \leq 10^4$), o número de casos de teste. Cada caso de teste consiste de uma linha contendo um inteiro **X** ($0 \leq \mathbf{X} \leq 9999$).

Saída

Para cada caso de teste imprima "Caso **#X**: **Y**", onde **X** é o número do caso atual, iniciando em 1, e **Y** é o retorno da rotina de krapekar ou -1 caso a rotina entre em loop infinito.

Exemplo de Entrada	**Exemplo de Saída**
3 3524 0 10	Caso #1: 3 Caso #2: -1 Caso #3: 5

Autor: Duhan Caraciolo

Problema J
Jacutingas vs Jaburus

Há **N** jacutingas em uma floresta, cada um em sua respectiva árvore. Há **N** jaburus cansados voando nesta floresta, e eles desejam pousar em árvores diferentes o mais cedo possível (jaburus são muito briguentos e não conseguem dividir uma mesma árvore). A cada $\mathbf{P_i}$ minutos, a jacutinga **i** sai da árvore para voar um pouco, e pode-se considerar que ela volta instantaneamente. A cada $\mathbf{C_i}$ minutos, o jaburu **i** pode tentar pousar em uma árvore em que a jacutinga não se encontre, e caso não consiga, volta instantaneamente a voar. Pode-se considerar que jaburus voam mais rápido que jacutingas e conseguem ocupar as árvores mais rápido do que elas. Dado uma estratégia ótima entre os jaburus, qual o menor tempo em que todos os jaburus estarão relaxando, cada um em uma árvore diferente?

Entrada
A primeira linha da entrada contém **T** ($1 \leq \mathbf{T} \leq 100$), o número de casos de teste. Cada caso de teste começa com um inteiro **N** ($1 \leq \mathbf{N} \leq 9$), o número de jacutingas e de jaburus. A segunda linha do caso de teste contém **N** inteiros $\mathbf{P_i}$ ($1 \leq \mathbf{P_i} \leq 10^4$), como descrito na questão. A terceira e última linha do caso de teste contém mais **N** inteiros $\mathbf{C_i}$ ($1 \leq \mathbf{C_i} \leq 10^4$), como também descrito na questão.

Saída
Para cada caso imprima "Caso **#X**: **Y**", onde **X** é o número do caso atual, começando em 1, e **Y** é a resposta da questão.

Exemplo de Entrada	**Exemplo de Saída**
2 2 1 3 2 2 1 1 1	Caso #1: 6 Caso #2: 1

Autor: Gustavo Stor