

## Automatic Feedback using Large Language Models

\* 1. This survey requires an understanding of Python code. Are you familiar with Python as a programming language?

☐ Yes

☐ No

## Automatic Feedback using Large Language Models

Before proceeding with the signing to start the experiment, please consider the following information.

You have been invited to participate in a research study on grades and automatic feedback.

The group is led by Prof. Luis de la Fuente Valentin, and the researcher involved is Marcelo Guerra Hahn.

This research has the approval of the Ethics Committee of the International University of La Rioja (Ref. PI050/2024).

If you agree to participate in this research, you will be asked to complete 1 questionnaire:

Questionnaire on the evaluation of automatic feedback in programming problems

Implications for the participant:

- You understand that your participation is completely voluntary.
- You can withdraw from the study at any time without explanation and without any repercussions on your current situation.
- The information obtained will be used exclusively for the specific purposes of this study.
- You will not receive any financial incentive or goods for your participation in the study, but the school will receive a general report on the research results.
- You also have the right to not answer any specific questions. Additionally, you have the right to receive a copy of this document as well as the signed consent form.

Risks of the research for the participant:

There are no risks of any kind from participating in this study.

### \* 2. Statement of Consent

I agree to participate in the study and use my answers described above. Further, I agree to print a copy of this consent form for my records if I deem it appropriate.

☐ Yes

☐ No

## Automatic Feedback using Large Language Models

\* 3. **Please select the FIRST number in the list and click NEXT.**

(This question is used as a randomization mechanism)



## Automatic Feedback using Large Language Models

You will be provided with a computing problem (the same for all parts) and five different students solutions. For each the solutions a hint was generated using a Large Language Model (LLM). For each solution and hint please answer the three questions.

### Problem

Alphabet Soup

=====

Two words formed by capital letters are given. It is assumed that they are ordered lexicographically, being the first smaller or equal to the second. It is intended to find the pair of letters that distinguishes them and that determines this order, if this pair exists. The pair exists as long as the first word is neither a proper prefix nor equal to the second.

Input

-----

Two lines, each with a word. Each word has 20 letters at most and may have repetitions.

Output

-----

A line with the pair of letters requested, if it exists. Otherwise, you will have the word `Nenhum` (i.e., None).

Examples

-----

### Input 1

```
```sh
MPMCAC
MPAMJ
```
```

### Output 1

```
```sh
MA
```
```

### Input 2

```
```sh
CCM
CCMP
```
```

### Output 2

```
```sh
Nenhum
```
```

### Input 3

```
```sh
MPAMJ
MPMCAC
```
```

### Output 3

```
```sh
AM
```
```

Credits

-----

DCC/FCUP 2010 - Ana Paula Tomás

## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 1

```
import string

def dados():
    lista = []
    a = input()
    a = string.split(a)
    for x in a:
        lista = lista + [x]
    return lista

def inicio():
    a = []
    b = []

    a = dados()
    b = dados()

    for i in range(min(len(a),len(b))):
        if a[i] != b[i]:
            print(a[i],b[i])
            return
    print("Nenhum")
    return

return

inicio()
```

## Feedback 1

It looks like you're on the right track by comparing the characters of the two words. However, make sure to move the print statement for "Nenhum" outside the loop so it only gets executed if no differing characters are found. Also, consider the case where one word is a proper prefix of the other and handle it accordingly. This will ensure that you correctly identify when to print the pair of differing characters or "Nenhum".

\* 4. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 5. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 6. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Alphabet Soup

=====

Two words formed by capital letters are given. It is assumed that they are ordered lexicographically, being the first smaller or equal to the second. It is intended to find the pair of letters that distinguishes them and that determines this order, if this pair exists. The pair exists as long as the first word is neither a proper prefix nor equal to the second.

## Input

-----

Two lines, each with a word. Each word has 20 letters at most and may have repetitions.

## Output

-----

A line with the pair of letters requested, if it exists. Otherwise, you will have the word `Nenhum` (i.e., None).

## Examples

-----

### ### Input 1

```
```sh
MPMCAC
MPAMJ
```
```

### ### Output 1

```
```sh
MA
```
```

### ### Input 2

```
```sh
CCM
CCMP
```
```

### ### Output 2

```
```sh
Nenhum
```
```

### ### Input 3

```
```sh
MPAMJ
MPMCAC
```
```

### ### Output 3

```
```sh
AM
```
```

## Credits

-----

## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 2

```
import string

def dados():
    lista = []
    a = input()
    a = string.split(a)
    for x in a:
        lista = lista + [x]
    return lista

def inicio():
    a = []
    b = []

    a = input()
    b = input()
    print(a,b)
    for i in range(min(len(a),len(b))):
        if a[i] == b[i]:
            pass
        else:
            print(a[i],)
            print(b[i])
            return
    return
inicio()
```

### Feedback 1

Think about how you can compare the two words character by character to find the first position where they differ. If you find such a position, those two characters form the pair you need to output. If you reach the end of one word without finding a difference, you should check if the first word is a proper prefix of the second or if they are equal.

\* 7. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 8. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 9. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Messy Cicadas

=====

Fed up to hear that they began the day singing, some cicadas decided to imitate their rivals. Not realizing the logic of travel in narrow lines, we found them flustered: having left a certain place in Indian line, they began to walk around, systematically passing through places where they previously had been, as if they were lost in a maze. Finally, they found the place where they intended to arrive. Anyway, we would like to know what they could have done if they had had a better sense of guidance and had not "visited" the same places repeatedly.

Task

----

Write a program that analyzes the sequence of places that they were "visiting", which includes the starting point and, in the end, the arrival point, and find a path that respects this sequence but removes all unnecessary turns. Only the last passage in each of these places can be considered useful. All others constitute deviations. In case of mistake, all the return that the cicadas have given, since they first visited a place until they returned, will be considered unnecessary. The place of arrival was not visited twice, but any of the remaining points, including the origin, may have been.

Input

-----

Only the sequence of places where the cicadas went through, one location for each line of data is given. The last line has the value 0. The number of distinct places does not exceed 30. Each location is identified by a positive integer lower than 10000.

## Output

-----

The sequence of places they would visit if they were not confused, each place in a separate line.

## Examples

-----

### ### Input 1

```
```sh
1600
15
2315
15
1315
0
```
```

### ### Output 1

```
```sh
1600
15
1315
```
```

### ### Input 2

```
```sh
1513
3171
178
1316
1600
1513
1774
178
3171
672
1315
0
```
```

### ### Output 2

```
```sh
1513
1774
178
3171
672
1315
```
```

## Credits

-----

ToPAS'2006 - Ana Paula Tomás



## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 1

```
def cigarras():
    res=[]
    i=0
    x=input()
    while x!=0:
        if x not in res:
            res.append(x)
        else:
            while res[i-1]!=x:
                res.remove(res[i-1])
                i=i-1
            i=i+1
            x=input()
    print(res)
```

### Feedback 1

Think about how you can use a dictionary to keep track of the last position of each place visited. This will help you identify which places are revisited and need to be removed from the path.

Consider this: Instead of trying to manipulate the list as you go, maintain a record of the latest index for each place. After collecting all the input, you can then determine the correct sequence by only considering the most recent visit to each place.

\* 10. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 11. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 12. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Messy Cicadas

=====

Fed up to hear that they began the day singing, some cicadas decided to imitate their rivals. Not realizing the logic of travel in narrow lines, we found them flustered: having left a certain place in Indian line, they began to walk around, systematically passing through places where they previously had been, as if they were lost in a maze. Finally, they found the place where they intended to arrive. Anyway, we would like to know what they could have done if they had had a better sense of guidance and had not "visited" the same places repeatedly.

Task

----

Write a program that analyzes the sequence of places that they were "visiting", which includes the starting point and, in the end, the arrival point, and find a path that respects this sequence but removes all unnecessary turns. Only the last passage in each of these places can be considered useful. All others constitute deviations. In case of mistake, all the return that the cicadas have given, since they first visited a place until they returned, will be considered unnecessary. The place of arrival was not visited twice, but any of the remaining points, including the origin, may have been.

Input

----

Only the sequence of places where the cicadas went through, one location for each line of data is given. The last line has the value 0. The number of distinct places does not exceed 30. Each location is identified by a positive integer lower than 10000.

Output

-----

The sequence of places they would visit if they were not confused, each place in a separate line.

Examples

-----

### Input 1

```
```sh
1600
15
```

```
2315
15
1315
0
...
```

### Output 1

```
```sh
1600
15
1315
...
```

### Input 2

```
```sh
1513
3171
178
1316
1600
1513
1774
178
3171
672
1315
0
...
```

### Output 2

```
```sh
1513
1774
178
3171
672
1315
...
```

Credits

-----

ToPAS'2006 - Ana Paula Tomás

Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 2

```
def cigarras():
    res=[]
    i=0
    x=input()
    while x!=0:
        if x not in res:
            res.append(x)
        else:
            while res[i-1]!=x:
                res.remove(res[i-1])
                i=i-1
            i=i+1
            x=input()
    return res
```

## Feedback 1

Consider maintaining a dictionary that stores the most recent index of each place as you read the input. This will help you identify which places need to be in the final path based on their last occurrence. After processing the input, you can build the final path by including only these last occurrences in the correct order.

\* 13. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 14. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 15. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

#### Minimalist Optics

=====

It is intended to define the links between nodes of a network taking into account the expected gross income for each direct connection between two nodes. If possible, it should be preserve the network connectivity and minimize the costs of its global maintenance and, having already guaranteed these two conditions, maximize global net income. Maintenance costs are identical for all calls.

#### Input

-----

In the first line three integers are given: the number of nodes in the network, the number of possible calls and the cost of maintaining each connection. This is followed by a line with three integers, for each connection: the first two identify their extremes and the third is their gross performance. Nodes are identified by consecutive numbers, starting from 1.

#### Output

-----

`rendimento optimo` (i.e., optimal income) followed by the net income value of any optimal network or the message `impossivel` (i.e., impossible) if it is not possible to guarantee connectivity.

#### Examples

-----

### Input 1

```
```sh
8 9 2
1 2 6
1 5 6
4 8 6
3 6 3
7 4 2
2 5 4
6 2 4
5 6 1
2 3 3
```
```

### Output 1

```
```sh
impossivel
```
```

### Input 2

```
```sh
8 13 2
1 2 6
1 5 6
8 7 6
4 8 6
3 4 4
3 6 3
7 4 2
2 5 4
6 2 4
5 6 1
4 6 9
7 6 5
2 3 3
```
```

### Output 2

```
```sh
rendimento optimo: 27
```
```

Credits  
-----

DCC/FCUP 2009 - Ana Paula Tomás

Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 1

```
# -*- coding: cp1252 -*-

#OPTICA MINIMALISTA com o prim

grafo1={1:{2:6,5:6},4:{8:6},3:{6:3},7:{4:2},2:{5:4,3:3},6:{2:4},5:{6:1},8:{}}

def PRIM(grafo,custo): #grafo pesado
    d={}
    Q=[]
    total=1
    for v in grafo:
        d[v]=-1
        Q.append(v)
    inicial=Q[0]
    d[inicial]=0
    while Q!=[]:
        (u,Q)=EXTRACT_MAX(Q,d)
        if d[u]==-1:
            print('impossivel')
            return
        total=total+d[u]
        for v in grafo[u]:
            if v in Q and grafo[u][v]>d[v]:
                d[v]=grafo[u][v]
    print('rendimento optimo:',total-custo*(len(grafo)-1))
    return

def EXTRACT_MAX(lista,d):
    (ma,x)=max((d[n],n) for n in lista)
    lista.remove(x)
    return (x,lista)
```

```
(vertices,arestas,custo)=map(int,input().split())
grafo={}
lista=[]
for i in range(1,vertices+1):
    grafo[i]={}
for i in range(1,arestas+1):
    (ext1,ext2,rendimento)=map(int,input().split())
    lista.append((ext1,ext2,rendimento))
for (a,b,c) in lista:
    grafo[a][b]=c

PRIM(grafo,custo)
```

## Feedback 1

To solve the problem of finding the optimal network connectivity with maximal net income using Prim's algorithm, you need to ensure that the graph remains connected and you minimize the maintenance costs while maximizing the gross income. Consider how Prim's algorithm can help you build a Minimum Spanning Tree (MST), which ensures connectivity and minimal cost. Once you have the MST, think about how you can calculate the net income by subtracting the total maintenance cost from the gross income of the MST.

\* 16. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 17. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 18. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Minimalist Optics

=====

It is intended to define the links between nodes of a network taking into account the expected gross income for each direct connection between two nodes. If possible, it should be preserve the network connectivity and minimize the costs of its global maintenance and, having already guaranteed these two conditions, maximize global net income. Maintenance costs are identical for all calls.



## Input

-----

In the first line three integers are given: the number of nodes in the network, the number of possible calls and the cost of maintaining each connection. This is followed by a line with three integers, for each connection: the first two identify their extremes and the third is their gross performance. Nodes are identified by consecutive numbers, starting from 1.

## Output

-----

`rendimento optimo` (i.e., optimal income) followed by the net income value of any optimal network or the message `impossivel` (i.e., impossible) if it is not possible to guarantee connectivity.

## Examples

-----

### ### Input 1

```
```sh
8 9 2
1 2 6
1 5 6
4 8 6
3 6 3
7 4 2
2 5 4
6 2 4
5 6 1
2 3 3
```
```

### ### Output 1

```
```sh
impossivel
```
```

### ### Input 2

```
```sh
8 13 2
1 2 6
1 5 6
8 7 6
4 8 6
3 4 4
3 6 3
7 4 2
2 5 4
6 2 4
5 6 1
4 6 9
7 6 5
2 3 3
```
```

### ### Output 2

```
```sh
rendimento optimo: 27
```
```

...

Credits

-----

DCC/FCUP 2009 - Ana Paula Tomás

Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 2

```
def optica_minimalista():
    #obter input
    x = input().split()
    nNos = int(x[0])
    nLig = int(x[1])
    cManu = int(x[2])

    adj = []

    for i in range(nLig):
        x = input().split()
        inic = int(x[0])
        fim = int(x[1])
        rBruto = int(x[2])
        adj.append([inic,fim,rBruto])

    #ordenar a lista de adj
    adj = maxRendimento(adj)
    #obter a mst e o rendimento bruto max
    mst = getMST(nNos,adj)

    #imprimir o valor
    if len(mst[0]) == 1:
        print(("rendimento optimo: %d" % (mst[1]-(cManu*(nNos-1)))))
    else:
        print("impossivel")

def getMST(n,adj):
    # criar um conjunto para cada vertice
    setList = []
    for i in range(1,n+1):
        setList.append(set())
        setList[i-1].add(i)
```

```

def getMST(n,adj):

    #coloca o rendimento bruto a zero
    rBruto = 0

    #percorre toda a lista de adjacencias
    for i in adj:
        #pega nos dois vertices de cada ligacao
        x= i[0]
        y= i[1]
        #e testa se estao no mesmo set, caso nao estejam:
        if find_set(setList,x) != find_set(setList,y):
            setList = unionSet(setList,x,y)    #une os sets a que cada um pertence
            rBruto += i[2]    #e aumenta-se o rendimento bruto

    return setList,rBruto

def find_set(setList,x):    #encontrar o set do vertice x na lista de sets setList
    for i in range(len(setList)):
        if x in setList[i]:
            return i

#une o set do elemento x com o set do elemento y
def unionSet(setList,x,y):
    doublebreak = False
    for i in range(len(setList)):
        for z in range(len(setList)):
            if x in setList[i] and y in setList[z]: #junta o set de x com o set de y e apaga o antigo set de y
                setList[i] = setList[i].union(setList[z])
                del(setList[z])
                doublebreak = True    #ta feito o que queriamos, doublebreak para parar os dois ciclos for
                break
        if doublebreak == True: break
    return setList

```

```

#coloca por ordem decrescente de peso todas as arestas
def maxRendimento(adj):
    for i in range(len(adj)):
        for j in range(len(adj)):
            if adj[i][2] > adj[j][2]:
                (adj[i],adj[j]) = (adj[j],adj[i])
    return adj

optica_minimalista()

```

### Feedback 1

In your current implementation, you're using a custom method to sort the edges and manage sets. Consider using a standard algorithm like Kruskal's algorithm, which is specifically designed for this kind of problem (finding a Minimum Spanning Tree). It efficiently handles edge sorting and union-find operations. Additionally, make sure your algorithm correctly handles all nodes to ensure full connectivity.

\* 19. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 20. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 21. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Sociology

=====

The students of the Sociology course decided to make an experience between them. They agreed that each one would invent a story about himself, tell it to their colleagues who they relate better, and would tell them also all the stories that were told them. Based on this experience, they agreed to consider that a person's group would be the set of people, including herself, whose history became known and who made known their own. They knew that under these conditions, a person's group is necessarily the group of each of the people in their group.

They intended to determine the number of groups with four or more people and the number of people outside these groups. When they were about to start the experiment, some students suggested their replacement for a simulation on a computer to various hypothetical scenarios.

Input

-----

The first line is given the number of scenarios to consider. Following are the descriptions of the scenarios. The first line of each scenario contains the number of students ( $\geq 4$ ). Next appears one line per student: the first number in this line identifies the student and the following is the number of students with whom they consider to relate better. The remaining numbers in this line identify these students, and there may be none. Consecutive integers are used, starting from 1, to number people.

Output

-----

For each scenario, you will have a line with identification of this case and, in the next line, two integers separated by a space. The first is the number of groups with four people or more. The second is the number of people outside these groups. Either numbers can be 0.

Example

-----

### Input

```
```txt
4
4
1 3 2 4 3
4 0
2 2 1 3
3 2 2 1
6
1 2 3 5
2 2 3 4
4 1 2
3 2 2 1
6 1 5
5 2 6 1
8
1 4 6 2 4 5
3 1 2
2 2 3 4
4 1 5
6 0
5 3 4 8 7
7 1 5
8 2 5 3
10
1 4 6 2 4 5
3 2 2 1
9 0
2 2 3 4
4 2 5 9
6 1 1
5 3 4 8 7
7 1 5
8 1 5
10 1 9
```
```

### Output

```
```txt
Caso #1
0 4
Caso #2
1 0
```

Caso #3

1 2

Caso #4

2 2

...

Credits

-----

DCC/FCUP 2008 - António Porto

Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 1

```
#!/usr/bin/env python

#Implementacao do algoritmo de Tarjan
def fconexos(grafo):

    grupos = [ ]
    pilha = [ ]
    low = { }

    def visit(no):
        if no in low: return

        n = len(low)
        low[no] = n
        pos = len(pilha)
        pilha.append(no)

        for successor in grafo[no]:
            visit(successor)
            low[no] = min(low[no], low[successor])

        if n == low[no]:
            x = pilha[pos:]
            del pilha[pos:]
            grupos.append(x)
            for y in x:
                low[y] = len(grafo)

    for no in grafo:
        visit(no)

    return grupos
```

```

def escrever(result,x):
    g = 0
    s = 0
    for i in range(len(result)):
        if len(result[i]) < 4:
            s+=len(result[i])
        else:
            g+=1

    print("Caso #"+str(x+1))
    print(g,s)

def sociologia():
    cenario = {}
    saida = []
    num_cenarios = input()
    c = 0
    for i in range(num_cenarios):
        num_alunos = input()
        for k in range(num_alunos):
            linha = input()
            s = linha.split()
            lista = []
            if (int(s[1]) != 0):
                for n in range(2,len(s)):
                    lista.append(s[n])
            cenario[s[0]] = lista

        grupos = fconexos(cenario)
        escrever(grupos,i)
sociologia()

```

#### Feedback 1

Your implementation of Tarjan's algorithm to find strongly connected components is a good approach to identifying the groups. However, ensure that your graph representation correctly reflects the input format, and remember to initialize the graph structure properly for each scenario. Additionally, after finding the strongly connected components, you need to count the number of components with four or more students and those with fewer than four students accurately. Pay attention to the way you reset or reuse variables between scenarios.

\* 22. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 23. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 24. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Sociology  
=====

The students of the Sociology course decided to make an experience between them. They agreed that each one would invent a story about himself, tell it to their colleagues who they relate better, and would tell them also all the stories that were told them. Based on this experience, they agreed to consider that a person's group would be the set of people, including herself, whose history became known and who made known their own. They knew that under these conditions, a person's group is necessarily the group of each of the people in their group.

They intended to determine the number of groups with four or more people and the number of people outside these groups. When they were about to start the experiment, some students suggested their replacement for a simulation on a computer to various hypothetical scenarios.

Input  
-----

The first line is given the number of scenarios to consider. Following are the descriptions of the scenarios. The first line of each scenario contains the number of students ( $\geq 4$ ). Next appears one line per student: the first number in this line identifies the student and the following is the number of students with whom they consider to relate better. The remaining numbers in this line identify these students, and there may be none. Consecutive integers are used, starting from 1, to number people.

Output  
-----

For each scenario, you will have a line with identification of this case and, in the next line, two integers separated by a space. The first is the number of groups with four people or more. The second is the number of people outside



these groups. Either numbers can be 0.

Example

-----

### Input

```
```txt
4
4
1 3 2 4 3
4 0
2 2 1 3
3 2 2 1
6
1 2 3 5
2 2 3 4
4 1 2
3 2 2 1
6 1 5
5 2 6 1
8
1 4 6 2 4 5
3 1 2
2 2 3 4
4 1 5
6 0
5 3 4 8 7
7 1 5
8 2 5 3
10
1 4 6 2 4 5
3 2 2 1
9 0
2 2 3 4
4 2 5 9
6 1 1
5 3 4 8 7
7 1 5
8 1 5
10 1 9
```
```

### Output

```
```txt
Caso #1
0 4
Caso #2
1 0
Caso #3
1 2
Caso #4
2 2
```
```

Credits

-----

DCC/FCUP 2008 - António Porto

## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 2

```
import string

#####ler dados#####
def ler_dados():
    lista = []
    a = input()
    a = string.split(a)
    for x in a:
        lista = lista + [int(x)]
    return lista

def Trabalho_a():
    a = []
    b = []
    temp = []
    n_cenarios = []
    x = 0
    temp1 = []
    graf_key = []

    cenarios = int(input())
    for i in range(cenarios):
        grafo = {}
        descr_cenarios = int(input())
        n_cenarios.append(descr_cenarios)
        for j in range(descr_cenarios):
            a = ler_dados()
            temp = []
            if a[1] != 0:
                for z in range(2, len(a)):
                    temp.append(a[z])
            grafo[a[0]] = temp
```

```

        else:
            grafo[a[0]] = []
        caminhos(grafo,descr_cenarios,i)

def caminhos(grafo,temporaria,temporaria2):
    graf_key = grafo.keys()
    grupo1_temp=[]
    grupo2_temp=[]
    grupos=[]
#1PASSAGEM!!!(caminho numa direccao)
    for toma in graf_key:
        grup_temp = []
        for ab in range(len(graf_key)):
            a = bfs(grafo,toma,ab+1, [])
            if a == True:
                if ab not in grup_temp:
                    grup_temp.append(ab)
        inc (grup_temp)
        grupo1_temp.append(grup_temp)

        final = []
        final2= []

    final=elimina_rep(grupo1_temp)

#2 PASSAGEM!!!(Caminho em direccao contraria)
    final2=teste(grafo,final,temporaria,temporaria2)

```

```
def teste(grafo, grupo1_temp, temp, temporaria2):
```

```
    tamanho = 0
    contador=0
    final=[]
    b=[]
    c=[]
    c=elimina_rep(grupo1_temp)
    for i in range(len(c)):
        b=[]
        for j in range(len(c[i])):
            a = bfs(grafo,c[i][j],c[i][0], [])
            if a==True:
                b.append(c[i][j])
        final.append(b)
    for z in range(len(final)):
        if len(final[z])>=4:
            contador+=1
            tamanho=tamanho+len(final[z])
    print"Caso #%d"%temporaria2
    if (tamanho-temp)<0:
        print(contador, " ",(tamanho-temp)*-1)
    else:
        print(contador,tamanho-temp)
```

```
def bfs(g,s,t,vis):
```

```
    if s==t:
        return True
    vis.append(s)
    adj=g[s]
    for x in adj:
        if not (x in vis):
            if bfs(g,x,t,vis):
                return True
    vis.pop()
    return False
```

```

def inc(lista):
    for i in range(len(lista)):
        lista[i]=lista[i]+1
    return lista

def elimina_rep(lista):
    a=[]
    for i in range(len(lista)):
        if lista[i] not in a:
            a.append(lista[i])
    return a

def elimina_azia(lista):
    final=[]
    for i in range(len(lista)):
        if len(lista[i])==1:
            valor_eliminar=lista[i][0]
            for a in range(len(lista)):
                if valor_eliminar in lista[a]:
                    final.append(elimina_valor(lista[a],valor_eliminar))
    print(final)

def elimina_valor(lista,valor):
    for i in range(len(lista)):
        if valor==lista[i]:
            temp=i
    del lista[temp]
    return lista

Trabalho_a()

```

### Feedback 1

It seems that your approach is on the right track, but there's a small issue with the way you're handling input and the graph traversal. Make sure to properly manage and initialize variables for each scenario to avoid carrying over data from previous scenarios. Also, ensure that your graph traversal correctly identifies all strongly connected components (SCCs). It might help to use a dedicated library function for finding SCCs to simplify your code. Consider checking your input parsing and how you build the graph to ensure accuracy.

\* 25. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 26. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 27. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

Automatic Feedback using Large Language Models

## Problem

Order

=====

An animal will be transported by land of a zoo to another under a rare species exchange protocol. The trip will be made in a rectangular cage, whose dimensions (length, width and height) will have to be within certain intervals. These dimensions are still conditioned both by the infrastructure of the transport network and by the type of vehicles that may ensure transportation in each section of the path. The problem was analyzed by a technical team that provided the capacity of each section of the network, taking into account the type of vehicles available. To reduce the animal's stress, it is strongly recommended that the cage has a maximum length. What will be this value?

Input

-----

The first line has five integers separated by one space: minimum and maximum width, minimum and maximum length, and minimum height of the cage. The second line has two integers that identify the origin and the destination. Each of the remaining lines, except for the latter, has five integers separated by spaces: the first two identify the extremes of a section, and the following the maximum values for the width, length and height of the cage if the route includes this section. The last line has the value -1. The dimensions are in decimeters. Any connection is bidirectional. Locations (extremes of sections) are identified by not necessarily consecutive integers.

Output

-----

The length of the cage that will carry the animal (expressed in decimeters) or 0 if the transport is not possible.

Example

-----

### Input

```
```txt
20 30 25 40 15
4 3
1 6 25 30 50
4 6 30 27 20
6 3 15 35 25
50 7 29 20 35
1 50 30 25 20
3 7 23 25 15
4 50 22 30 15
1 3 20 30 23
7 4 30 40 16
-1
```
```

### Output

```
```txt
27
```
```

Credits

-----

DCC/FCUP 2008 - Ana Paula Tomás

## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 1

```
Q = []
arcos = {}
adj = {}
dist = {}

def encomenda():
    x = input().split()
    lmin = int(x[0])
    lmax = int(x[1])
    cmin = int(x[2])
    cmax = int(x[3])
    amin = int(x[4])
    x = input().split()
    origem = int(x[0])
    destino = int(x[1])

    while True:
        x = input().split()
        ori = int(x[0])
        if ori == -1: break
        dest = int(x[1])
        larg = int(x[2])
        comp = int(x[3])
        alt = int(x[4])

        if larg >= lmin and alt >= amin and comp >= cmin:
            arcos[(ori,dest)] = comp
            arcos[(dest,ori)] = comp
            if ori not in Q: Q.append(ori)
            if dest not in Q: Q.append(dest)

            try:
                adj[ori].append(dest)
            except:
                adj[ori] = [dest]
```



```

        try:
            adj[dest].append(ori)
        except:
            adj[dest] = [ori]
    dijkstra(origem,destino)

def dijkstra(origem,destino):
    for i in Q:
        dist[i] = 0
    dist[origem] = 999

    while Q != []:
        a = getMin()
        u = Q[a]
        del(Q[a])
        if u == destino:
            print(dist[u])
            break
        for v in adj[u]:
            alt = min(dist[u],arcos[(u,v)])
            if alt > dist[v]:
                dist[v] = alt

def getMin():
    maxValue = 0
    indice = 0
    for i in range(len(Q)):
        if dist[Q[i]] > maxValue:
            maxValue = dist[Q[i]]
            indice = i
    return indice

encomenda()

```

### Feedback 1

Your approach is quite close, but there's a subtle issue in the Dijkstra implementation. Make sure that the initial distances are set correctly to handle the maximum constraint effectively. Also, ensure that you initialize your priority queue properly. Instead of using a list for Q, consider using a priority queue (like `heapq` in Python) to manage the nodes more efficiently. This will help in retrieving the node with the maximum distance more efficiently.

\* 28. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 29. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 30. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

Automatic Feedback using Large Language Models

## Problem

Order

=====

An animal will be transported by land of a zoo to another under a rare species exchange protocol. The trip will be made in a rectangular cage, whose dimensions (length, width and height) will have to be within certain intervals. These dimensions are still conditioned both by the infrastructure of the transport network and by the type of vehicles that may ensure transportation in each section of the path. The problem was analyzed by a technical team that provided the capacity of each section of the network, taking into account the type of vehicles available. To reduce the animal's stress, it is strongly recommended that the cage has a maximum length. What will be this value?

Input

-----

The first line has five integers separated by one space: minimum and maximum width, minimum and maximum length, and minimum height of the cage. The second line has two integers that identify the origin and the destination. Each of the remaining lines, except for the latter, has five integers separated by spaces: the first two identify the extremes of a section, and the following the maximum values for the width, length and height of the cage if the route includes this section. The last line has the value -1. The dimensions are in decimeters. Any connection is bidirectional. Locations (extremes of sections) are identified by not necessarily consecutive integers.

Output

-----

The length of the cage that will carry the animal (expressed in decimeters) or 0 if the transport is not possible.

Example

-----

### Input

```
```txt
20 30 25 40 15
4 3
1 6 25 30 50
4 6 30 27 20
6 3 15 35 25
50 7 29 20 35
1 50 30 25 20
3 7 23 25 15
4 50 22 30 15
1 3 20 30 23
7 4 30 40 16
-1
```
```

### Output

```
```txt
27
```
```

Credits

-----

DCC/FCUP 2008 - Ana Paula Tomás

## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 2

```
Q = []
arcos = {}
adj = {}
dist = {}

def encomenda():
    x = input().split()
    lmin = int(x[0])
    lmax = int(x[1])
    cmin = int(x[2])
    cmax = int(x[3])
    amin = int(x[4])
    x = input().split()
    origem = int(x[0])
    destino = int(x[1])

    while True:
        x = input().split()
        ori = int(x[0])
        if ori == -1: break
        dest = int(x[1])
        larg = int(x[2])
        comp = int(x[3])
        alt = int(x[4])

        if larg >= lmin and alt >= amin and comp >= cmin:
            arcos[(ori,dest)] = comp
            arcos[(dest,ori)] = comp
            if ori not in Q: Q.append(ori)
            if dest not in Q: Q.append(dest)

            try:
                adj[ori].append(dest)
            except:
                adj[ori] = [dest]
```

```

        try:
            adj[dest].append(ori)
        except:
            adj[dest] = [ori]
    dijkstra(cmax,origem,destino)

def dijkstra(cmax,origem,destino):
    for i in Q:
        dist[i] = 0
    dist[origem] = cmax

    while Q != []:
        a = getMin()
        u = Q[a]
        del(Q[a])
        if u == destino:
            print(dist[u])
            break
        for v in adj[u]:
            alt = min(dist[u],arcos[(u,v)])
            if alt > dist[v]:
                dist[v] = alt

def getMin():
    maxValue = 0
    indice = 0
    for i in range(len(Q)):
        if dist[Q[i]] > maxValue:
            maxValue = dist[Q[i]]
            indice = i
    return indice

encomenda()

```

### Feedback 1

You are on the right track with using Dijkstra's algorithm. However, consider switching your priority queue implementation to heapq for efficiency. This will help you get the node with the maximum distance more effectively than the current list-based approach. Also, make sure that the initial distances are correctly initialized to handle the maximum constraint properly.

\* 31. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 32. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 33. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Infrastructure

=====

An animal will be transported in a rectangular cage whose dimensions (length, width and height) will have to be within certain intervals. These dimensions are still conditioned both by the infrastructure of the transport network and by the type of vehicles that may ensure transportation in each section of the path. The problem was analyzed by a technical team that provided the capacity of each section of the network, taking into account the type of vehicles available. Considering the restrictions of the cage and the network, it is intended to determine the **number of sections** that could pass.

Input

-----

The first line has the number of places (extremes of sections).

In the second line, five integers separated by a space: minimum and maximum width, minimum and maximum length, and minimum cage height. The third line has two integers that identify origin and destination. Each of the remaining lines, except for the latter, has five integers separated by spaces: the first two identify the extremes of a section, and the following the maximum values for the width, length and height of the cage, if the route includes this section. The last line has the value -1. The dimensions are in decimeters. Any connection is bidirectional. The locations are identified by **consecutive** integers.

Output

-----

The number of sections in which it could pass, taking into account the dimension restrictions, but **not** the origin and destination.

## Examples

-----

### ### Input 1

```
```txt
6
20 25 15 40 5
4 1
1 4 10 30 23
3 5 22 30 15
1 2 25 30 50
5 2 20 30 50
3 2 30 27 20
2 4 15 35 25
5 6 29 20 35
1 5 30 25 20
4 6 23 25 15
6 3 30 40 16
-1
```
```

### ### Output 1

```
```txt
8
```
```

### ### Input 2

```
```txt
6
20 30 25 40 15
3 4
1 2 25 30 50
3 2 19 27 20
2 4 15 35 25
5 6 29 20 35
1 5 30 25 20
4 6 23 25 15
3 5 12 30 15
1 4 20 30 23
6 3 30 20 16
-1
```
```

### ### Output 2

```
```txt
4
```
```

## Credits

-----

## Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

### Solution 1

```
def infra():
    g = {}
    firstline = readinput()
    [origem,destino] = readinput()
    g[origem] = []
    g[destino] = []
    while True:
        lista = readinput()
        if lista[0] == -1 : break
        [u,v,l,c,a] = lista
        if verifica(firstline,l,c,a) == True:
            c = min(c,firstline[3])
            if u not in g: g[u] = []
            if v not in g: g[v] = []
            g[u].append((v,c))
            g[v].append((u,c))
    Q = PQ()
    comp = {}
    for x in g:
        comp[x] = 0
        Q.push(x,comp[x])
    comp[origem] = 2**30
    Q.change_pri(origem,2**30)
    visited = set()
    while not Q.empty():
        u = Q.front()
        Q.pop()
        visited.add(u)
        for (v,c) in g[u]:
            if v not in visited:
                k = min(comp[u],c)
                if comp[v]< k:
                    comp[v] = k
                    Q.change_pri(v,k)
    return comp[destino]
```

### Feedback 1

To tackle the problem, focus on constructing a graph where each node represents a location and each edge represents a transport section with specific cage dimension constraints. Use BFS starting from the origin to count valid sections that satisfy these constraints, ensuring to maintain a visited set to avoid redundant processing and to check each section's dimensions against the given criteria. This approach will help efficiently determine the number of valid transport sections between locations, excluding the origin and destination pair.



\* 34. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 35. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 36. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Infrastructure

=====

An animal will be transported in a rectangular cage whose dimensions (length, width and height) will have to be within certain intervals. These dimensions are still conditioned both by the infrastructure of the transport network and by the type of vehicles that may ensure transportation in each section of the path. The problem was analyzed by a technical team that provided the capacity of each section of the network, taking into account the type of vehicles available. Considering the restrictions of the cage and the network, it is intended to determine the **\*\*number of sections\*\*** that could pass.

Input

-----

The first line has the number of places (extremes of sections).

In the second line, five integers separated by a space: minimum and maximum width, minimum and maximum length, and minimum cage height. The third line has two integers that identify origin and destination. Each of the remaining lines, except for the latter, has five integers separated by spaces: the first two identify the extremes of a section, and the following the maximum values for the width, length and height of the cage, if the route includes this section. The last line has the value -1. The dimensions are in decimeters. Any connection is bidirectional. The locations are identified by **consecutive** integers.

Output

-----

The number of sections in which it could pass, taking into account the dimension restrictions, but **not** the origin and destination.

Examples

-----

### Input 1

```
```txt
6
20 25 15 40 5
4 1
1 4 10 30 23
3 5 22 30 15
1 2 25 30 50
5 2 20 30 50
3 2 30 27 20
2 4 15 35 25
5 6 29 20 35
1 5 30 25 20
4 6 23 25 15
6 3 30 40 16
-1
```
```

### Output 1

```
```txt
8
```
```

### Input 2

```
```txt
6
20 30 25 40 15
3 4
1 2 25 30 50
3 2 19 27 20
2 4 15 35 25
5 6 29 20 35
1 5 30 25 20
4 6 23 25 15
3 5 12 30 15
1 4 20 30 23
6 3 30 20 16
-1
```
```

### Output 2

```
```txt
```

4  
...

Credits

-----

DCC/FCUP 2009 (Prova Prática - A) - Ana Paula Tomás

Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 2

```
def a():
    ext=read_input()
    lmin,lmax,cmin,cmax,amin= read_input()
    ori,dest=read_input()
    first=read_input()
    score=0
    while first != [-1]:
        if lmin <= first[2] <= lmax and cmin <= first[3] <= cmax and amin <= first[4]:
            score+=1
        first = read_input()
    print(score)

def read_input():
    x=map(int,input().split())
    return x

a()
```

## Feedback 1

Focus on simplifying the input reading process and ensure you correctly handle the termination condition when reading sections. Use straightforward conditions to check if each section's dimensions meet the specified constraints, incrementing a counter accordingly. This approach will help streamline your solution while ensuring clarity and correctness in counting valid sections based on cage dimension criteria.

\* 37. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 38. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 39. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Electronic Business

=====

A flower \_online\_ store forwards its customers' orders to associated non-virtual stores. These stores should give a quick response, being responsible for delivering the bouquet to the customer. The map of the distribution region is divided into non-overlapping areas. In each area, there is only one store. This store serves the zone if it has availability. If, for any reason, it cannot do so, the online store contacts the remaining stores in increasing order of \_distance\_ to the destination address, until it finds one that accepts the service. For distance calculation, it considers only paths formed by segments that connect stores in neighboring regions. It is intended to implement a program to support this operation, indicating how stores must be contacted to respond to an order.

Input

-----

The first line has the total number of regions (or stores) and the identifier of the destination region. The regions are

defined by consecutive numbers, starting from 1. In each of the following lines, there are the identifiers of two adjacent regions and the distance between them (approximate to the units). Ends with -1.

#### Output

-----

The order in which stores must be contacted (ties are resolved by indicating stores in ascending order of identifier).

#### Example

-----

### Input

```txt

```
9 7
1 2 32
1 3 14
1 4 15
2 5 35
3 2 20
5 3 22
3 4 25
5 6 24
2 6 40
7 2 40
7 8 33
6 7 15
6 8 25
9 6 22
9 8 21
5 9 17
5 4 33
9 4 45
-1
```
```

### Output

```txt

```
7 6 8 9 5 2 3 1 4
```
```

#### Credits

-----

DCC/FCUP 2009 - Ana Paula Tomás

#### Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 1

```
from heapq import *

inf = 9999
dist = []
previous = {}
arcos = {}
adj = {}

def negocio():
    x = input().split()
    nLojas = x[0]
    destino = x[1]
    while True:
        x = input().split()
        x[0] = int(x[0])
        if x[0] == -1: break
        x[0] -= 1
        x[1] = int(x[1])-1
        x[2] = int(x[2])

        arcos[(x[0],x[1])] = x[2]
        arcos[(x[1],x[0])] = x[2]

        try:adj[x[0]].add(x[1])
        except:
            adj[x[0]] = set()
            adj[x[0]].add(x[1])

        try:adj[x[1]].add(x[0])
        except:
            adj[x[1]] = set()
            adj[x[1]].add(x[0])

    z = dijkstra(int(nLojas),int(destino)-1)
```

```

def dijkstra(nLojas,destino):
    for i in range(nLojas):
        dist.append(inf)
    dist[destino] = 0
    Q = []
    final = []
    for i in range(nLojas):
        heappush(Q,(dist[i],i))
    while Q != []:
        u = heappop(Q)[1]
        final.append(u+1)
        for v in adj[u]:
            alt = dist[u] + arcos[(u,v)]
            if alt < dist[v]:
                dist[v] = alt
                previous[v] = u
                #try:
                #    heappop(Q)
                heappush(Q,(alt,v))

            #except:
            #    continue
    for i in range(len(final)):
        try:
            if final[i] == final[i+1]:
                del(final[i])
        except:
            break
    for i in range(nLojas): print final[i],

negocio()

```

### Feedback 1

Focus on implementing Dijkstra's algorithm to find the shortest path from the destination region (provided in the input) to all other regions. This algorithm is suitable because it efficiently handles graphs with non-negative weights, which fits the problem's requirement of calculating distances between adjacent regions. Ensure you correctly initialize data structures for storing distances (dist), previous nodes (previous), and adjacency lists (adj). Pay attention to correctly parsing and storing input data, especially handling adjacency relationships and distances between regions. Remember that ties in distance should be resolved by indicating stores in ascending order of identifier. This approach will guide you towards building a solution that outputs the correct order of stores to contact based on their proximity to the destination.

\* 40. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 41. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 42. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

## Automatic Feedback using Large Language Models

### Problem

Electronic Business

=====

A flower \_online\_ store forwards its customers' orders to associated non-virtual stores. These stores should give a quick response, being responsible for delivering the bouquet to the customer. The map of the distribution region is divided into non-overlapping areas. In each area, there is only one store. This store serves the zone if it has availability. If, for any reason, it cannot do so, the online store contacts the remaining stores in increasing order of \_distance\_ to the destination address, until it finds one that accepts the service. For distance calculation, it considers only paths formed by segments that connect stores in neighboring regions. It is intended to implement a program to support this operation, indicating how stores must be contacted to respond to an order.

Input

-----

The first line has the total number of regions (or stores) and the identifier of the destination region. The regions are



defined by consecutive numbers, starting from 1. In each of the following lines, there are the identifiers of two adjacent regions and the distance between them (approximate to the units). Ends with -1.

#### Output

-----

The order in which stores must be contacted (ties are resolved by indicating stores in ascending order of identifier).

#### Example

-----

### Input

```txt

```
9 7
1 2 32
1 3 14
1 4 15
2 5 35
3 2 20
5 3 22
3 4 25
5 6 24
2 6 40
7 2 40
7 8 33
6 7 15
6 8 25
9 6 22
9 8 21
5 9 17
5 4 33
9 4 45
-1
```
```

### Output

```txt

```
7 6 8 9 5 2 3 1 4
```
```

#### Credits

-----

DCC/FCUP 2009 - Ana Paula Tomás

#### Solutions

In the next section, we'll provide you with a student solution and feedback on the solution. Please review the solution and the feedback and answer the questions below.

## Solution 2

```
def negocioelectronico():
    x = input().split()
    nLojas = int(x[0])
    iDestino = int(x[1])

    V = []
    E = {}

    #le todas as ligacoes entre as regioes
    x = input().split()
    while x[0] != '-1':
        inic = int(x[0])
        fim = int(x[1])
        dist = int(x[2])
        #Cria uma lista com todos os vertices
        if inic not in V:
            V.append(inic)
        if fim not in V:
            V.append(fim)
        #Cria um dicionario com todas as ligacoes no sentido lato
        E[inic,fim]=dist
        E[fim,inic]=dist
        x = input().split()
    #enviamos todas as ligacoes existentes e respectivos pesos, o ponto a partir do qual pretendemos que seja ligado a
    # todas as regioes e uma lista com os vertices
    print(dijkstra(E,iDestino,V))

#retorna o indice da regioao mais proxima
def Extract_Min(Q,dist):
    minimo = 999999
    for i in Q:
        if dist[i-1] < minimo:
            minimo = dist[i-1]
            ind = i-1
    return ind
```

```
#calcula as distancias minimas entre todas as regioes e a regioao de entrega
def dijkstra(G,s,Q):
    dist = []
    parent = []
    for i in range(len(Q)):
        dist.append(999999)
        parent.append(None)
    #As regioes sao identificadas por numeros consecutivos a partir de 1 mas os indices da lista comecam no zero.
    #Por isso temos de reduzir uma unidade. p.e -> Regiao 1 encontra-se no indice 0
    dist[s-1]=0
    S = []
    while Q != []:
        #procura o indice da regioao mais proxima
        ind = Extract_Min(Q,dist)
        #Guarda os vertices conforme encontra-mos os minimos
        S.append(ind+1)
        #remove a entrada da regioao da lista das regioes por visitar
        del(Q[Q.index(ind+1)])
        #No dicionario com todas as ligacoes procura aquelas que ligam a regioao pretendida a todas as outras
        for i in G:
            if i[0] == ind+1:
                if dist[i[1]-1] > dist[ind] + G[i]:
                    dist[i[1]-1] = dist[ind] + G[i]
                    parent[i[1]-1] = ind
    return S
```

### Feedback 1

Instead of using a custom Extract\_Min function to find the minimum distance vertex each time, consider using Python's built-in min function with a key to efficiently find the minimum element in a list or set. This will simplify your code and improve its readability and performance. Remember to adjust your data structures (dist, parent, Q) accordingly to utilize this approach effectively within your dijkstra function. This change will help streamline your implementation of Dijkstra's algorithm for finding the shortest paths from the destination region to all other regions in the map of distribution regions.

\* 43. The feedback provided contains the solution to the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 44. The feedback provided contains relevant information to advance towards the solution.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess

\* 45. The feedback provided contains irrelevant information for solving the problem.

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neither Agree nor Disagree
- ☐ Disagree
- ☐ Strongly Disagree
- ☐ I don't know enough about the topic to assess