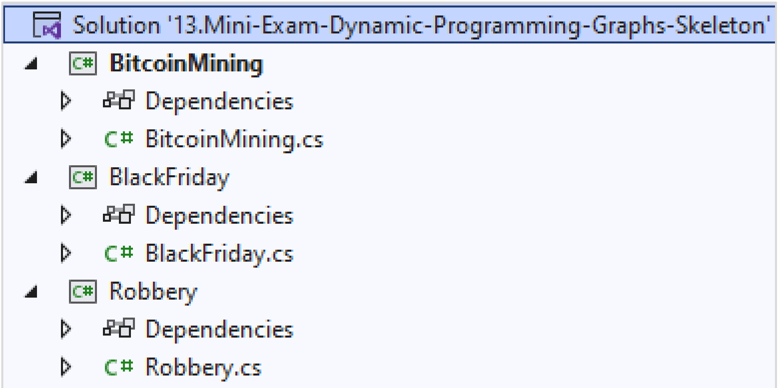
Mini Exam: Dynamic Programming & Graphs

Problems for exercises and homework for the "Data Structures and Algorithms Basics" course from the official "Applied Programmer" curriculum.

You can check your solutions here: https://judge.softuni.org/Contests/3685/Mini-Exam-Dynamic-Programming-Graphs

Use the provided skeleton:



1. Robbery

You are robber who **just** stole a TV. Now you must **escape** the cops **without** being **caught**.

You are given a **map** of the city **streets**. There are few **rules**:

- Going from **one point to other** costs you some **energy** (displayed as a value on each arrow) and takes one turn.
- Each point is being watched by a **video camera**. A point can be **black** (a camera is **not** watching it) or **white** (a camera is watching it).
- You can only travel to points where the **camera is off**.

Find the path that requires the **least energy** to go to the final point. **Print the required energy**.

Input

- On the first line you will receive a number - **n** specifying the **number of points**.
- On the second line you will receive a number - **c** specifying the **number of point connections**.
- On the next **c** lines you will receive the **connections** in the format "`<firstPoint> <secondPoint> <distance>`"
- On the next line you will receive all points in the format "`<point1><color1> <point2><color2> …<pointK><colorK>`".
- On the next line, you will receive the **starting point**.
- On the next line, you will receive the **ending point**.

Output

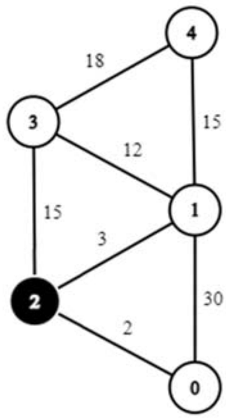
- Print the energy of the path that requires the **least energy** to go to the final point.

Constraints

- The number of points in the city will be between **[2…20000]**.

- The distance of a connection will be a valid integer between **[0…10000].**
- The points will always be numbers starting from **[0… number of points – 1]**.
- The color will be either "**b**" or "**w**" – **b** means the camera is currently **not** watching, **w** means the camera is currently watching.
- There will always be a valid path from **start** to **end**.

Examples

Input	Output	Comments
5 7 0 1 30 0 2 2 1 3 12 1 4 15 2 1 3 2 3 15 3 4 18 0b 1b 2w 3b 4b 0 4	45	 • Start – 0. • Destination – 4. • The path that requires the least energy to go to the final point is 0->2->1->4 (20 energy). • However, camera 2 is watching, so we can't go through this point. • Therefore, the solution is 0->1->4 (45 energy).
6 9 0 1 30 0 2 2 1 3 12 1 4 15 2 1 3 2 3 15 3 4 18 4 5 13 3 5 11 0b 1b 2w 3b 4w 5b 0	53	

5		

2. Bitcoin Mining

Bitcoin users can control how quickly their transactions are processed by setting the fee rate. The **higher the fee rate, the faster the transaction will be processed**. Each block in the blockchain can only contain up to **1MB (1 000 000 bytes) of information.**

Since space is limited, a **limited number of transactions can be included in each block**. This means **Bitcoin miners are incentivized to prioritize the transaction with the highest fees**.

Write a program that based on the **block capacity**, you need to **decide which transactions** to put in it to **maximize the fees** of the pending transactions.

Input

- On the first line, you will receive an integer – **n** – number of the **pending transactions**.
- On the next **n** lines, you will receive a transaction in the following format: **`"{hash} {size} {fees} {from} {to}"`**.
    - Size will be in **bytes**.

Output

- Print the **used size of the block** in the following format: **`"Total Size: {totalSize}"`**.
- Print the **total fees of all transactions in the block** in the following format: **`"Total Fees: {totalFees}"`**.
- Print **transaction hashses** that will be **included in the block**.
    - Order doesn't matter.

Constraints

- **`n will be in the range [1… 25].`**
- **`size`** will be in the range [0… 1 000 000].
- **`fees`** will be in the range [0… 1 000 000].

Examples

Input	Output
5 25d8dd2f 342000 23213 coinbase.btc atanasov.btc 16d27e46 542000 523213 coinbase.btc shopov.btc 6247072a 242000 13213 coinbase.btc ani.btc fc951abc 600000 113213 procoinbase.btc atanasov.btc 0a4a5f32 450000 153213 procoinbase.btc peter.btc	Total Size: 992000 Total Fees: 676426 0a4a5f32 16d27e46
5 25d8dd2f 100000 2321 coinbase.btc atanasov.btc	Total Size: 850000 Total Fees: 82603

`16d27e46 200000 52323 coinbase.btc shopov.btc`
`6247072a 300000 1323 coinbase.btc ani.btc`
`fc951abc 150000 11323 procoinbase.btc atanasov.btc`
`0a4a5f32 100000 15313 procoinbase.btc peter.btc`
	`0a4a5f32`
	`fc951abc`
	`6247072a`
	`16d27e46`
	`25d8dd2f`

3. Black Friday

The year is 1955 and online shopping doesn't exist. However, **"Black Friday"** is approaching, and Roi wants to be prepared.

Roi wants to **visit every shop** in the town for **as little time as possible**. So, you are appointed to solve this problem.

You will be given the **number** of **shops** on the first line, then the number of **roads** (**n**), and on the next **n** lines you will receive which shops the road connects and the travel time.

Assume you can **start from any** shop and your target is to **visit every one** of them with the **minimum travel time**.

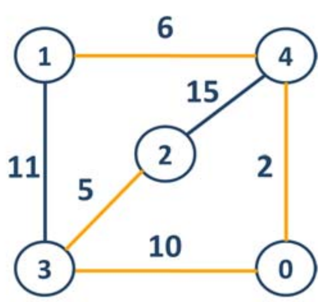
Input

- On the **first line** you will be given the **number of** the **shops.**
- On the **second** line you will be given the **number of streets** (**n**).
- On the **next n lines,** you will be given a connection in the format: **"`{firstShop} {secondShop} {time}`"**.

Output

- Print the **total time** of the trip you have chosen.

Examples

Input	Output	Comment
5 6 0 3 10 0 4 2 4 1 6 1 3 11 2 3 5 2 4 15	23	The minimum travel time to visit all shops is: • 2 -> 3 (5) • 3 -> 0 (10) • 0 -> 4 (2) • 4 -> 1 (6)

4 4 1 2 5 2 3 6 0 2 10 0 1 5	16	