Magic Containment

Due: 11:59pm on October 26th, 2024

Objectives

- Give practice with sorting in C.
- Give practice with 2 pointer logic in C.

Story

Your program printed an incantation, and seemingly the wrong one, because one of your apprentices performed the spell and the magic is still lingering. Luckily, the spell did not destroy the forest and instead has miraculously made the magical motes slow down to the point where they are measurable and seemingly sphere shaped. The state of the magic gave you an idea.

You had some students work on producing various magic containment devices. Each device has a limited capacity, and can contain only one mote of magic. A magic containment device acts as a box with 3 dimensions, a height, length, and width. A magical mote can fit in the box as long as the mote's volume does not exceed the device's volume.

Based on your devices and magical motes, you want to know the least amount of magic that cannot be contained by your devices.

Problem

Given a list of magical devices as their dimensions and a list of magical motes as their radii, determine the least volume of magic that cannot be contained by the given magical devices.

Input Specification

Input begins with a line containing 2 integers, M and D, $(1 \leq M, D \leq 500,000)$, representing the number of magical motes and the number of magic containment devices respectively.

The following line contains M integers representing the radius of a magical mote. Each radius will be at most 1,000,000

The following D lines each contain 3 integers representing the length, width, and height of a magical device. Each dimension will be at most 1,000,000.

(To make the problem easier) It's guaranteed that changing the radius by at most 10^{-6} relatively will not change the answer.

Output Specification

Your program should print the least volume of the uncontained motes. (To make the problem easier) Your answer will be accepted if is off by at most 10^{-6} relative to the correct answer.

Example Input Output

Below is are example of cases that your program will be tested against. The below cases are not meant to be extensive. It is your responsibility. You should work on developing your own cases.

Input	**Output**
`5 4`	`46034.804351`
`6 21 1 9 10`	
`5 5 5`	
`10 6 18`	
`7 17 11`	
`11 7 17`	
`2 2`	`0.000000`
`10 10`	
`20 20 20`	
`20 20 20`	

Example Explanation

Sample Case 1

In the first case there are 5 motes. The volumes for the motes are approximately the following

- 904.778684
- 38792.386087
- 4.188790
- 3053.628059
- 4188.790205

In the first case there are 4 magic containment devices. The volumes for the devices are

- 125
- 1080
- 1309
- 1309

We can put the 3rd mote in the 1st device and the 1st mote in the 3rd device. None of the remaining motes can fix in any of the devices. The sum of the volumes of the uncontained motes is 38792.386087+3053.628059+4188.790205 =

Sample Case 2

Both motes can be contained.

Hints

- Use types with floating points (preferably `double`)
- Compute the volume of each device and mote
- Store the volume in 2 arrays (one for devices and one for motes).
- Geometry
  - Volume of a box (device) is $h \times l \times w$
  - Volume of a sphere (magical mote) is $\frac{4}{3} \times \pi \times r^3$
- Sort the arrays by volume.
- Always try to contain the largest mote not contained currently. If there is no device that contain it move to the next one.
- Track an index in the array of motes AND an index in the array of devices. Start with the largest device first.
- When a mote can be contained, move both indices.
- When a mote cannot be contained, move only one of the indices.
- Track sum of uncontained mote volume in some variable.

Grading Criteria

Programs that do not compile using gcc will receive a grade of 0. Programs that exit with a non-zero return code will be considered to have crashed on the case. Programs on a particular case that take longer than the maximum of 10 seconds and 5 times the length of the instructor's solutions will be deemed too inefficient for that case. Programs that do not produce the exact output of the instructors will be considered incorrect. Outputting input prompts will result in a program being considered incorrect.

Any case on which a program crashes, takes too much time, or produces wrong output will be awarded 0 points, even if the program produces partially correct output.

Any solutions that does not implement a Sorting algorithm will receive at most 50 points. YOU CANNOT USE A BUILT IN SORT IN C.

- Whitespace is acceptable (5 points)
- Comments are reasonable (5 points)
- Variable names are not vague (5 points)
- Use standard input/output (5 points)
- No extra output; do not prompt for input (e.g. "please enter the number of incantations") (5 points)
- Read using a loop (5 points)
- Make volumes for motes (5 points)
- Make volumes for devices (5 points)
- Try to contain the largest possible mote first (5 points)
- Tracks "unused" devices (5 points)
- 10 test cases (5 points each)