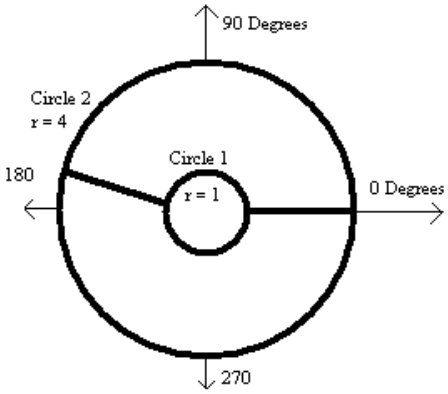
Driving in Circles

Filename: `circles`

The University of Central Florida (UCF) has a unique layout that makes it more interesting to drive through. Specifically, the roads form circles around the Student Union, with radial cross streets connecting circles. The whole setup can be disorienting for the delivery trucks that need to bring in packages to the center of UCF, since most drivers are not used to driving in circles when they already know where they are going. Luckily, no street or road on campus is one-way so drivers avoid that confusion.

As a new driver for a delivery company that includes UCF and other similarly arranged locations, Ali is finding that it can be difficult to figure out the fastest way to get to a delivery with all the circles. Determined to minimize the distance he has to travel to get to a delivery point, Ali decides that he'll make an application for his new smart phone to help him out. You just put in the layout of the roads and streets and it tells you how far you have to travel. Who knows, maybe there will be a market for just such an app!



The Problem:

Given a description of the map as well as starting and ending locations, determine the shortest trip.

If you need a value for π, use the one provided by your environment or 3.141592653589793 if one is not provided.

The Input:

The first line of the input will contain a positive integer, t ($t \leq 100$), giving the number of trips to follow. Each trip begins with a line containing the integers, c ($1 \leq c \leq 10$) and r ($c\text{-}1 \leq r \leq 20$), where c is the number of circular roads that surround a single center point and r is the number of radial streets connecting the circular roads. The next line will contain c integer circle radii in strictly increasing order, each separated by a single space.

The next r lines will each contain two integers, separated by a single space, describing a radial street, d ($1 \leq d < c$) and a ($0 \leq a < 360$). d is the one-based index of the starting circular road and represents connecting circular road d with next larger circular road d+1. a is an integer angle in degrees in relation to the center of all circular roads, with 0 degrees pointing directly to the East, and the angle increasing in the counter-clockwise direction (90 degrees would be North, etc.).

All circular roads will be connected by radial streets, and it will always be possible to reach one circular road from another. Radial streets will only connect to valid circular roads. No road or street will lie on top of another road.

After the r radial street descriptions, there will be four more integers on a single line, *sc sa fc fa* ($1 \leq sc, fc \leq c$ and $0 \leq sa, fa < 360$), denoting the starting and ending locations. *sc* is the one-based index of the starting circular road, and *sa* is the angle of the starting point compared to the center of the circular road. *fc* and *fa* are the destination circular road and the angle of the point on the destination circular road, respectively. Again, angles are in degrees and given in the same orientation as the radial street descriptions (0 degrees is East, 90 is North, etc.).

The Output:

For each trip, print a line containing only the shortest distance from the starting point to the ending point, traveling along circular roads and radial cross streets. This distance should be output rounded to two decimal points (for example, 2.435 should be rounded to 2.44 and 2.934 should be rounded to 2.93).

Sample Input:

```
2
1 0
1
1 0 1 180
2 2
1 4
1 0
1 170
2 180 1 90
```

Sample Output:

```
3.14
5.09
```