Jimmy's Perfect Vacation

Filename: vacation

Jimmy and his family are planning a vacation. They all love amusement parks, and they always try to go on every ride that the park has. Unfortunately, they only have a certain amount of time in the day. Not only does timing make things complicated, some of the paths between two rides have been blocked off for parades, construction, and mothers tending to crying babies. To make matters even worse, Jimmy is very sensitive to rides and throws up every time he gets off, so he never wants to go near a ride he has already ridden. They could write out by hand to figure out the fastest way to visit each ride, but Jimmy has a good friend (you)! Help Jimmy and his family find the fastest way to visit all of the rides. Of course, they will be going on a lot more vacations, so your program should be able to handle multiple vacations.

The Problem:

Given each ride the amusement park has, as well as a list of blocked paths between two rides, find the fastest time it will take for Jimmy and his family to visit each one. Assume that he and his family walk at 1 meter per second, and they can always travel to their destination in a straight line assuming that the path is not blocked. Rides always take a constant time of 120 seconds per ride (they always visit the parks during the off-season so the waiting time is negligible).

The Input:

Input will begin with a single integer, *n*, which is the number of different parks Jimmy and his family have planned to visit. For each park, there will be two non-negative integers, *r* and *b,* which are the number of rides and blocked paths respectively ($r < 11$, $b < r^2$). On the following *r* lines are two integers, *x* and *y*, denoting the Cartesian coordinates of the ride (in meters). The rides are numbered from 1 to *r* for simplicity, and no two rides will have the same coordinates. After these *r* lines are *b* lines, each giving two integers *i* and *j* stating that the path between ride *i* and ride *j* is blocked, and therefore, the path from *j* to *i* is blocked as well. All coordinates will be non-negative integers less than 1,000, and Jimmy and his family always start their visit at the entrance, which is at the origin, (0,0). There will never be a blocked path that includes the entrance, and you may never revisit the entrance (they consider that exiting the park and make you buy another ticket!). Assume there is a path between every pair of rides unless it was blocked.

The Output:

For each vacation, first output a header, on a line by itself, stating "`Vacation #`*`k`*`:`" where *k* is the number of the vacation, starting with 1. If Jimmy can and his family can ride every ride, output a new line "`Jimmy can finish all of the rides in `*`s`*` seconds.`" where *s* is the number of seconds rounded to the nearest thousandth of a second it takes to ride every ride (as an example of rounding, a time of 100.5455 would be rounded up to 100.546, whereas a time of 100.5454 would be rounded down to 100.545). If Jimmy and his family cannot ride every ride, output one line saying "`Jimmy should plan this vacation a different day.` " instead. Output a single blank line after the output for each vacation.

Sample Input:

```
3
4 0
0 2
2 2
2 4
4 4
5 4
10 10
12 35
64 60
3 7
100 857
1 2
1 3
1 4
1 5
5 2
0 5
0 10
0 20
0 50
0 25
3 4
1 2
```

Sample Output:

```
Vacation #1:
Jimmy can finish all of the rides in 488.000 seconds.

Vacation #2:
Jimmy should plan this vacation a different day.

Vacation #3:
Jimmy can finish all of the rides in 670.000 seconds.
```