Problem B: Missile Command

As Chief Bureaucrat at Missile Command, it has recently come to your attention that the existing performance guidelines do not sufficiently penalize frivolous use of expensive ammunition. Therefore, you must write a new battle summary analysis tool which takes into account excess ammunition consumption during battle.

A battle consists of the following elements:

- Shots. A shot is a circularly explosive countermeasure. A shot has a fixed position and is active for 2 seconds, during which its radius varies from 0 to 1km and then back to 0 according to the formula:
  $r = (1 - (t - 1)^2)^{1/2}$
- The ground, at y = 0.
- Missiles. A missile is a point particle that moves at a constant velocity. If a missile collides with an active shot, the missile is neutralized (the shot persists). If a missile hits the ground before being neutralized, it is considered to have hit its target.

Performance is evaluated on a simple point scale. The performance criteria are as follows:

- Every neutralized missile adds 1 point.
- Every missile allowed to hit its target subtracts 5 points.
- Every unnecessary shot subtracts 20 points. The number of unnecessary shots in a battle is the difference between the actual number of shots fired and size of the minimum subset of those shots that would have neutralized the same number of missiles.

Input (from file b.in)

Input will be given in the following format (legend follows):

```
nb
nm
mx my mdx mdy mt
...
ns
sx sy st
...
...
```

In the following legend, indentation denotes repetition of the indented block a number of times equal to the value of the preceding input item:

nb (0 < *nb*) – number of battles
    nm (0 <= *nm* <= 20) – number of missiles
        mx/*my* (0.0 < *my*) – initial missile position (in km)
        mdx/*mdy* – missile velocity (in km/s)
        mt (0.0 <= *mt*) – time since battle start of the missile's entrance (in seconds)
    ns (0 <= *ns* <= 20) – number of shots
        sx/*sy* (1.0 <= *sy*) – shot position at time of detonation (in km)
        st (0.0 <= *st*) – time since battle start of the shot's detonation (in seconds)

Output (to stdout)

For each battle, output a line containing the score for that battle.

Sample Input

```
2
2
4.0 8.0 0.0 -1.0 0.0
4.0 8.0 1.0 -1.0 0.0
1
4.0 4.0 3.0
3
4.0 10.0 0.0 -1.0 0.0
5.0 10.0 3.0 -6.0 4.0
13.0 10.0 -3.0 -5.0 4.0
3
4.0 5.0 3.0
7.0 8.0 4.0
9.0 4.0 4.0
```

Sample Output

```
-4
-17
```