Marathon Match Problem Statement

Contest: CirclesSeparation Normal view

Problem: CirclesSeparation

Problem Statement

IMPORTANT: This problem is used for two simultaneous matches: TCO'13 Marathon Round 3 and Marathon Match 81. You can compete in TCO'13 R3 only if you are eligible for TCO'13 and haven't advanced from TCO'13 R1/R2. You can compete in MM 81 if you are not eligible for TCO'13, have advanced from TCO'13 R1/R2 or just would like to skip TCO'13 R3 by some reason. Competing in both matches is not allowed. Doing so will lead to disqualification. Note that registration does not count as competing. In order to be considered a competitor, you need to make at least one submit (example or full).

There are N circles on the plane. The i-th circle is centered at (**x**[i], **y**[i]), has a radius of **r**[i] and a mass of **m**[i].

Initially it is possible that some circles overlap. Two circles overlap if their intersection has a non-zero area. In other words, circles i and j overlap if the Euclidean distance between their centers is strictly less than **r**[i] + **r**[j].

You want to move circles so that they become separated, i.e., no two circles overlap. Moving the i-th circle so that it's centered at (*fx*[i], *fy*[i]) requires the amount of work equal to **m**[i] * (Euclidean distance between (**x**[i], **y**[i]) and (*fx*[i], *fy*[i])). The work required to move several circles is equal to the sum of works needed to move each individual circle.

Your task is to choose final locations of each circle (*fx*[i], *fy*[i]) so that no two circles overlap and the work required to move the circles is as small as possible.

You will need to implement the method `minimumWork`. Its input parameters are vector <double>s **x**, **y**, **r** and **m** that represent the input set of circles. The returned vector <double> should contain the following elements: (*fx*[0], *fy*[0], *fx*[1], *fy*[1], ..., *fx*[N-1], *fy*[N-1]).

In order for your solution to be evaluated, all elements of the return value must be between -100 and 100. Technically, the circles in your solution are allowed to touch each other, but since the problem deals with floating point arithmetics, it's necessary to be careful. Due to unavoidable small errors intrinsic to floating point computations, it is possible that your solution thinks that two circles just touch and our verification program thinks that they intersect. To avoid that, it's most safe to leave a small gap between any two circles (something around 10^{-9} is definitely enough and it almost won't affect your score).

For each test case we will calculate your raw and normalized scores. The raw score is equal to the amount of work required to move the circles. The normalized score is equal to 1,000,000 * BEST / YOUR, where BEST is the lowest positive raw score currently obtained for this test case (considering the last submission from each competitor) and YOUR is your raw score. If your program failed to produce a valid solution, then the raw score is -1 and the normalized score is 0.

Your total score is equal to the arithmetic average of normalized scores on all test cases.

The test data is generated as follows. All random numbers are real numbers sampled from uniform distribution. N is equal to 50 + floor(451 * t^2) where t is a random number between 0 and 1. Each of the **x**[i], **y**[i] and **m**[i] is a random number between 0 and 1. A parameter *maxR* is generated at random between square_root(1/N) and square_root(5/N). Finally, each of the **r**[i] is generated as a random number between 0 and *maxR*.

An offline tester/visualizer is available.

Definition

Class:	CirclesSeparation
Method:	minimumWork
Parameters:	vector <double>, vector <double>, vector <double>, vector <double>
Returns:	vector <double>
Method signature:	vector <double> minimumWork(vector <double> x, vector <double> y, vector <double> r, vector <double> m)
(be sure your method is public)	

Notes

- The time limit is 10 seconds (this includes only the time spent in your code). The memory limit is 1024 megabytes.
- There is no explicit code size limit. The implicit source code size limit is around 1 MB (it is not advisable to submit codes of size close to that or larger). Once your code is compiled, the binary size should not exceed 1 MB.
- The compilation time limit is 30 seconds. You can find information about compilers that we use and compilation options here.
- There are 10 example test cases and 100 full submission (provisional) test cases. Example test cases use seeds 1 to 10.

Examples

0)

```
Seed: 1
N: 206
Total circles area: 2.1908729896456927
```

1)

```
Seed: 2
N: 137
Total circles area: 0.979493441890281
```

2)

```
Seed: 3
N: 351
Total circles area: 1.4453700132109604
```

3)

```
Seed: 4
N: 392
Total circles area: 1.6588634107567646
```

4)

```
Seed: 5
N: 134
Total circles area: 1.0796890427625727
```

5)

```
Seed: 6
N: 486
Total circles area: 3.4796511746936956
```

6)

```
Seed: 7
N: 109
Total circles area: 2.483841498043311
```

7)

```
Seed: 8
N: 125
Total circles area: 3.032110707209091
```

8)

```
Seed: 9
N: 256
Total circles area: 4.232411671726059
```

9)

```
Seed: 10
N: 83
Total circles area: 0.9591366522380305
```
