Algorithms

Binary Search Homework 4

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: LeetCode 69 - Sqrt(x)

- Given a **non-negative** integer x, compute and return the **square root** of x.
- Truncate the number and return the integer part only
  - sqrt(20) = 4.47213595 ⇒ 4
- Use binary search
- int mySqrt(int x)
  - $0 <= x <= 2^{31} - 1$
- Input ⇒ output
  - 4 ⇒ 2
  - 20 ⇒ 4
  - 8 ⇒ 2

Problem #2: Spoj PIE

- double largest_area(vector<double> &pie_radius, int N)
- There are **N people** and **M pies**.
- Each pie is a **circle** with a different **radius**.
- Each person should get an **equally sized** (*but not necessarily equally shaped*) piece from a **single** pie. It is ok to have leftovers.
- What is the **largest** possible piece size each person can get?
- Input ⇒ Output
  - [4, 3, 3], people = 4 ⇒ 25.1327 (the largest area of an equally sized piece)
    - circle_area(4) = 50.2655. Split on 2 persons, each take 25.1327.
  - [5], people = 25 ⇒ 3.1416 circle_area(5) = 78.5398 / 25 ⇒ 3.1416
  - [1 4 2 3 4 5 6 5 4 2], people = 6 ⇒ 50.2655

Problem #2: Spoj PIE

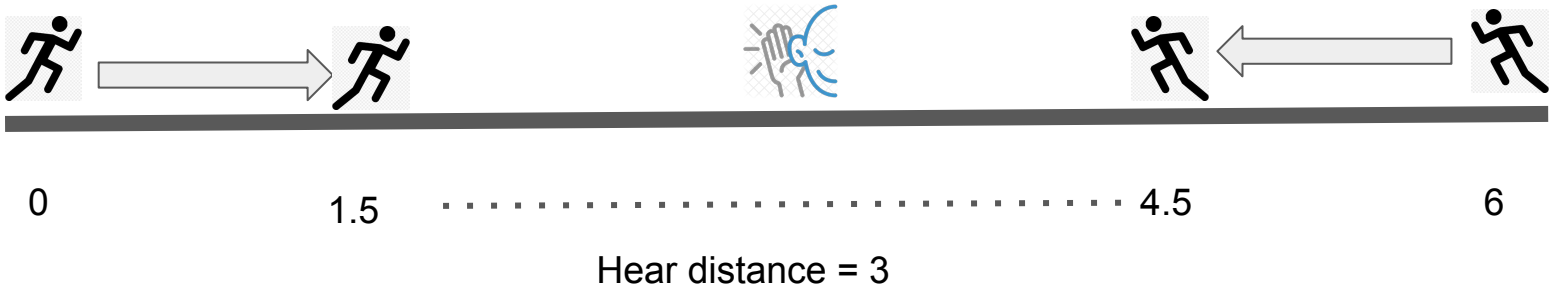
- To test this problem on SPOJ, you can do the following
- Read the SPOJ problem
  - The description is lengthy. Mine is shorter
    - Tricky text. There is +1 to add to the people (the host). E.g. case 2: 24 persons
  - You need to write main(). The function read T test cases
  - Properly format the output
- To make your life easier, Find code: **02_spoj_pie_EMPTY.cpp**
  - The code is ready for you to only code the **largest_area** function
    - I prepared the reading and writing for you
  - Test it directly, or copy paste the test case from the Spoj website
- Tip: For the binary search use: for (int i = 0; i < 100; i++)
  - while(fabs(start-end) < 1e-9) fails

Problem #4: Spoj GLASNICI

- double min_time(vector<double> &positions, double msg_dist)
- There is a road with N people positioned on it. As a team, a message from the **first** person should be **propagated** to the **last** person *as follows*:
  - Each person may run at a speed of **one unit** per second.
    - Move to a specific location: Either on the left or the right [my added constraint]
  - If a person **shouts** the message, anyone that is **at most msg_dist units** far can hear the message.
  - The team **arranges** their **move locations** to propagate the message as **fast** as possible
- Return the **minimum time** to propagate the message, given each person's position (positive values **sorted** in ascending order) and **msg_dist** (the max distance where a shout between two people can be heard)
  - $0 \le msg\_dist \le 1{,}000{,}000$
  - $1 \le positions.length \le 100{,}000$ and $0 \le positions[i] \le 1{,}000{,}000{,}000$

Problem #4: Spoj GLASNICI

- Input: positions = [0, 6] and msg_dist = 3 ⇒ min_time = 1.5
- We have 2 people: one at position 0 and another at position 6
- Person(1): moves from 0 to 1.5 then shouts the msg
  - Once shout, the shout can heard at position 1.5+3 = 4.5
- Person(2): moves back **in parallel** from 6 to 4.5
- As each person moves one unit per second, *the distance equals time*
- So we need 1.5 seconds so the team to propagate the message



Problem #4: Spoj GLASNICI

- Input: positions = [0, 4, 4, 8] and msg_dist = 2 ⇒ min_time = 1
- We have 4 people
- Person(1): moves from 0 to 1 then shouts the msg ⇒ heard at position (3)
- Person(2): moves from 4 to 3 then shouts the msg ⇒ heard at position (5)
- Person(3): moves from 4 to 5 then shouts the msg ⇒ heard at position (7)
- Person(4): moves from 8 to 7 to get the message
- So 1 unit of time is enough for the team to efficiently send the message to the last person

Problem #4: Spoj GLASNICI

- Input: positions = [5, 7, 9, 20] and msg_dist = 70 ⇒ min_time = 0
  - Once person 1 shouts, the last can hear. No need for a team effort
- Input: positions = [0, 9, 18] and msg_dist = 5 ⇒ min_time = 4
  - Person 1 moves to 4, Person 2 doesn't move. Person 3 moves to 14
- Input: positions = [0, 2.473, 3.707, 6, 7.400, 8.800, 8.928, 11.200, 11.600, 14.224, 15.400, 15.800, 18.600, 20, 20.200]
  and msg_dist = 1.4 ⇒ min_time = 0.9
- Feel free to read the SPOJ text. I simplified it and added some extra logic
- Find **03_spoj_GLASNICI_EMPTY.cpp**

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."