John Brehm

John Dye

Comp 482

6 May, 2017

Instant Insanity Programming Project Report

A. Using a simulation it takes approximately ~130 assignments.

B. Using a simulation ~4.7 numbers will occur less than 4 times in the puzzle.

C. My algorithm takes a binary search like approach to finding the obstacle.

Before starting any calculations I sorted the cubes by the average of the sum of all their sides to ttry and group cubes with similar numbers together.

The algorithm then begins and checks if the entire set of cubes has an obstacle, if it does it continues the algorithm, else there is no obstacle.

It then begins checking the set sizes similar to a binary search. If an obstacle is found then the smallest set is <= that set. If no obstacle is found then the min obstacle is >= that set.

After determining what size the obstacle is the algorithm runs through every possible combinations of the found size and returns the first set which has an obstacle.

As of right now I don't actually know if the code works nor what the actual obstacles are. The thread library checker function checkThreads() is horribly inefficient with 4 nested loops (O(n4)?) but I couldn't think of another way to compare the thread lists. Also just making the lists of threads takes a significant amount of time to complete as is. I also have not implemented the DFS which finds 2 threads at the same time but this might not even be necessary if the size of the obstacle is greater than 15 (this is what you suggested the cutoff point be in class). If I could improve the runtime of the thread library functions and implement the 2 thread DFS then this algorithm may be able to find the obstacle in a reasonable amount of time.

π input

4 7 10 13 16 19

22 26 29 32 3 6

9 12 16 19 22 25

28 31 2 6 9 12

15 18 21 24 28 31

2 5 8 11 14 18

21 24 27 30 1 4

8 11 14 17 20 23

26 30 1 4 7 10

13 16 20 23 26 29

32 3 6 10 13 16

19 22 25 28 32 3

6 9 12 15 18 22

25 28 31 2 5 8

12 15 18 21 24 27

30 2 5 8 11 14

17 20 24 27 30 1

4 7 10 14 17 20

23 26 29 32 3 7

10 13 16 19 22 25

29 32 3 6 9 12

15 19 22 25 28 31

2 5 9 12 15 18

21 24 27 31 2 5

8 11 14 17 21 24

27 30 1 4 7 11

14 17 20 23 26 29

1 4 7 10 13 16

19 23 26 29 32 3

6 9 13 16 19 22

25 28 31 3 6 9

12 15 18 21 25 28

e input

3 6 9 11 14 17

20 22 25 28 30 1

4 7 9 12 15 17

20 23 26 28 31 2

4 7 10 13 15 18

21 23 26 29 32 2

5 8 11 13 16 19

21 24 27 30 32 3

6 8 11 14 17 19

22 25 27 30 1 4

6 9 12 14 17 20

23 25 28 31 1 4

7 10 12 15 18 21

23 26 29 31 2 5

8 10 13 16 18 21

24 27 29 32 3 5

8 11 14 16 19 22

24 27 30 1 3 6

9 12 14 17 20 22

25 28 31 1 4 7

9 12 15 18 20 23

26 28 31 2 5 7

10 13 15 18 21 24

26 29 32 2 5 8

11 13 16 19 22 24

27 30 32 3 6 9

11 14 17 19 22 25

28 30 1 4 6 9

12 15 17 20 23 25

28 31 2 4 7 10

13 15 18 21 23 26

29 32 2 5 8 10