CS 91 USACO

Bronze Division

Unit 3: Problem Solving Using Algorithms

LECTURE 16: EXAMPLE WINNING SOLUTIONS

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Practice Problem: Combination Lock
- Overview of the Warmholes problem

Practice: Combination Locks (combo)

SECTION 1

Problem Statement

- Farmer John's cows keep escaping from his farm and causing mischief. To try and prevent them from leaving, he purchases a fancy combination lock to keep his cows from opening the pasture gate.
- Knowing that his cows are quite clever, Farmer John wants to make sure they cannot easily open the lock by simply trying many different combinations. The lock has three dials, each numbered 1..N (1 <= N <= 100), where 1 and N are adjacent since the dials are circular. There are two combinations that open the lock, one set by Farmer John, and also a "master" combination set by the lock maker.
- The lock has a small tolerance for error, however, so it will open even if the numbers on the dials are each within at most 2 positions of a valid combination.

Problem Statement

- For example, if Farmer John's combination is (1,2,3) and the master combination is (4,5,6), the lock will open if its dials are set to (1,3,5) (since this is close enough to Farmer John's combination) or to (2,4,8) (since this is close enough to the master combination). Note that (1,5,6) would not open the lock, since it is not close enough to any one single combination.
- Given Farmer John's combination and the master combination, please determine the number of distinct settings for the dials that will open the lock. Order matters, so the setting (1,2,3) is distinct from (3,2,1).

INPUT FORMAT (file combo.in):

Line 1:	The integer N.
Line 2:	Three space-separated integers, specifying Farmer John's combination.
Line 3:	Three space-separated integers, specifying the master combination (possibly the same as Farmer John's combination).

Each dial is numbered 1..50. Farmer John's combination is (1,2,3), and the master combination is (5,6,7).

SAMPLE INPUT:

```
50
1 2 3
5 6 7
```

OUTPUT FORMAT (file cowsignal.out):

- Line 1: The number of distinct dial settings that will open the lock.

SAMPLE OUTPUT:

249

Nature of the Problem

- Create a non-recurring list of qualified lock settings.
- Use conditional statement (or generating function) to create the neighboring lock settings (modulo is not appropriate here)
- The size of the non-recurring list is the output value.

Deal with Lock with Modulus

4	2	3
3	1	2
2	50	1
1	49	50
50	48	49

All numbers decreased by 1
We can use easier number modulus

3	1	2
2	0	1
1	49	0
0	48	49
49	47	48

Nature of the Problem

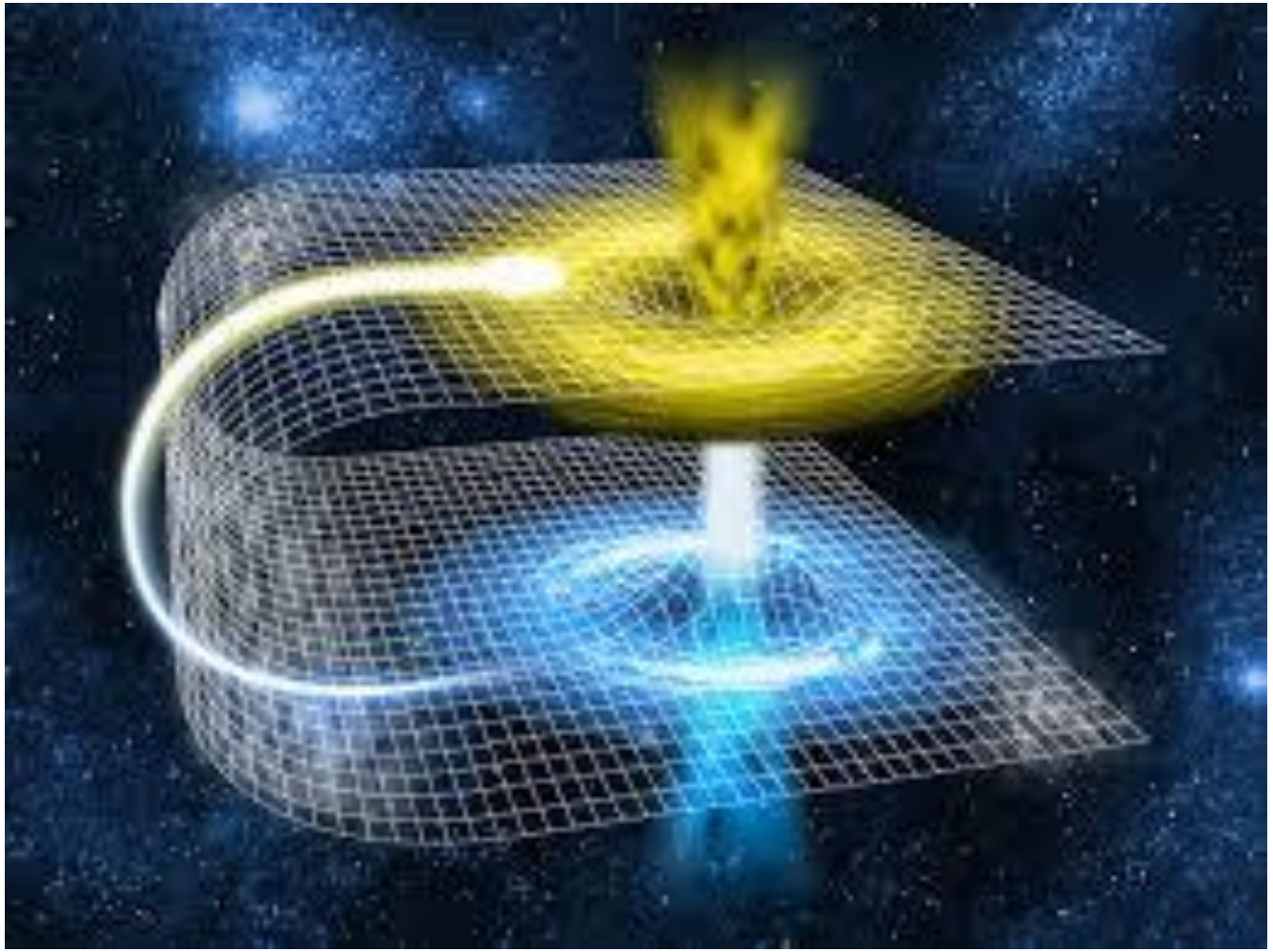
- There is another way to solve this problem.
- That is to calculate the total possibility of the FJ’s key been unlocked and the total cases of the master key been unlocked and minus the shared cases.

Modulus Distance

```
public static int distance(int a, int b, int N){
        int ax = Math.min(a, b);
        int bx = Math.max(a, b);
        int aN = ax+N;
        int d1 = Math.abs(ax-bx);
        int d2 = Math.abs(aN-bx);
        return (d1<=d2) ? d1 : d2;
    }
```

Overview of Warmwhole Problem

SECTION 2

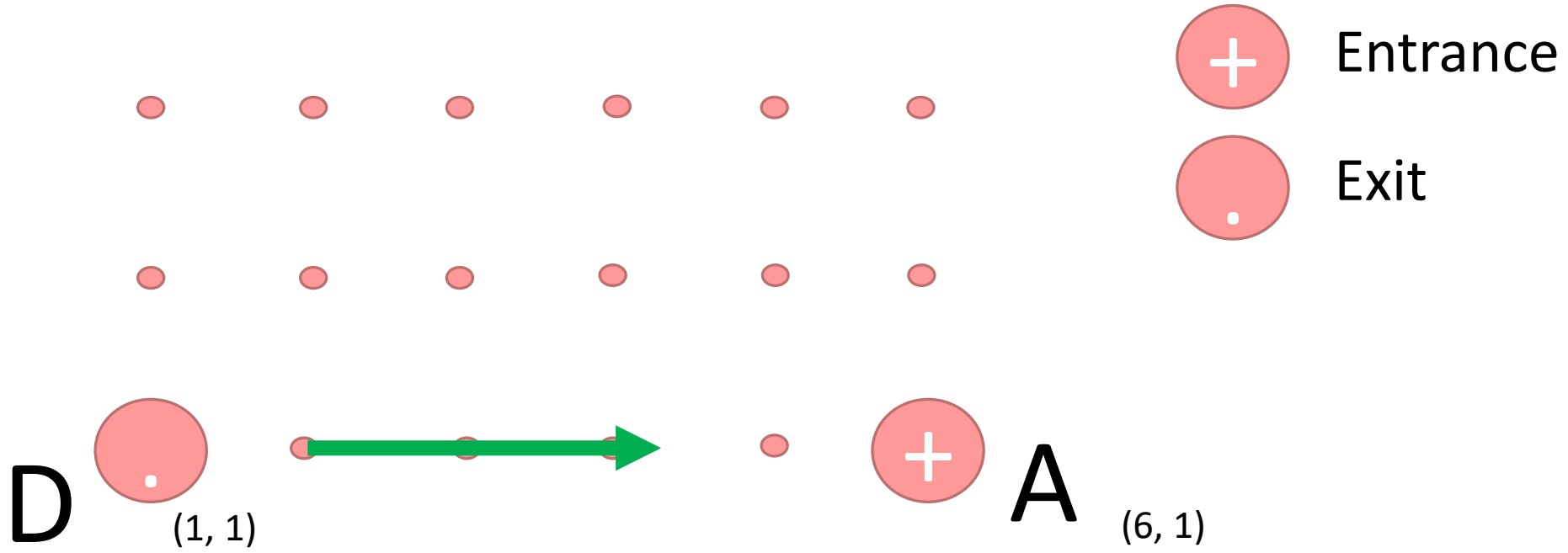


Worm Hole Problem

Worm Hole Problem

A wormhole is a theoretical **passage** through **space-time** that could create shortcuts for long journeys across the universe. **Wormholes** are predicted by the theory of general **relativity**. But be wary: wormholes bring with them the dangers of sudden collapse, high radiation and dangerous contact with exotic matter.

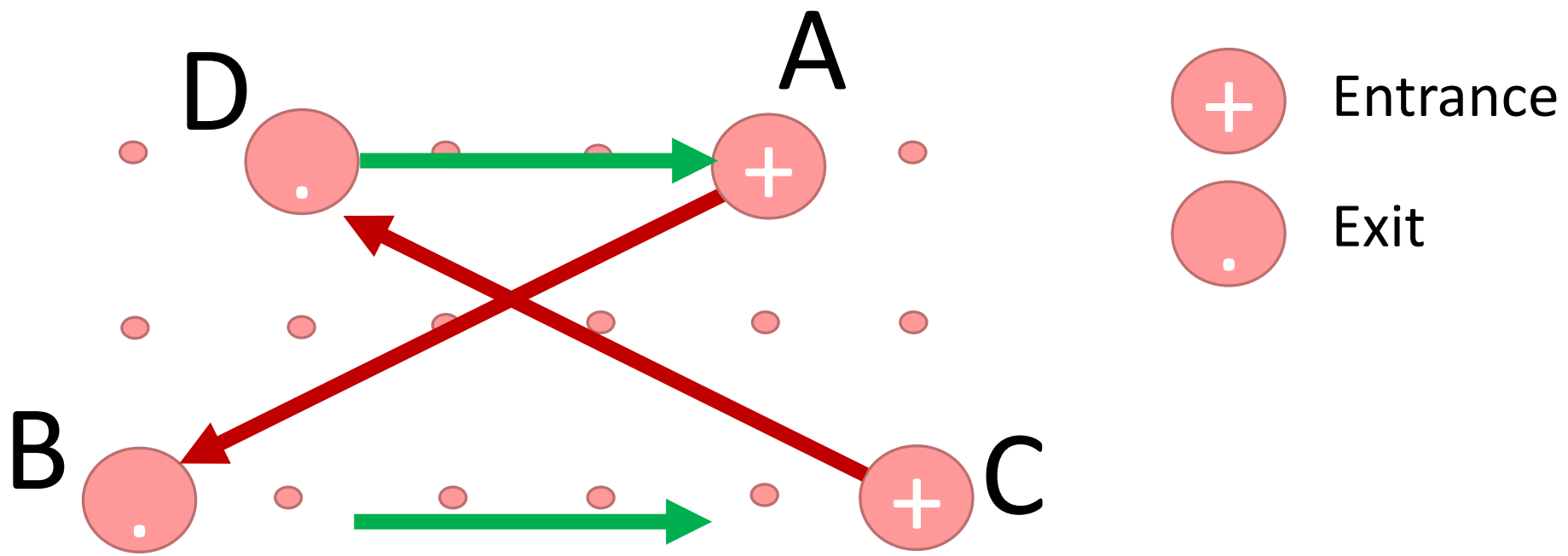
Worm Hole Analysis



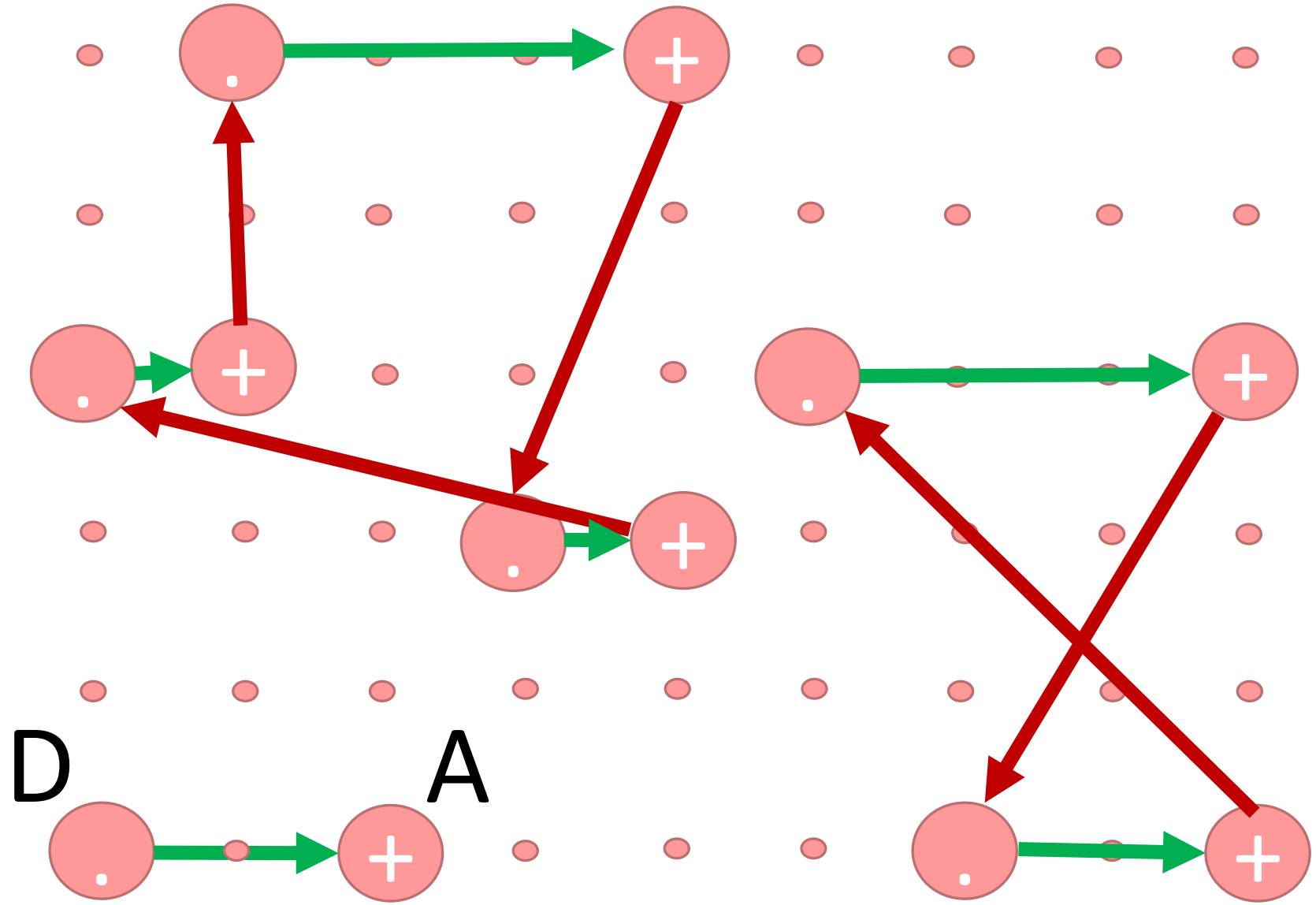
If a pairing contains pair [A, D], then there is a cycle, there is a trap.

Worm Hole Analysis

Next Neighbor on the Right



If we start from A
If a pairing contains pair [A, B] and A is the next on the right for C point, while [C, D] paired, and
[C, D] is the last pair we can find in this looping, and D has a point as next neighbor on the right. Then, there is a cycle.



- If we tried all of the wormholes as starting points, and
- there exists a certain starting point that loops around partner and next neighbor to the right and end at a point which has a next neighbor to the right, then there is a cycle. Otherwise there is no a cycle.

Nature of the Problem

- Starts from the recursive pairing problem.
- Finding a of the possible pairings for a group of wormholes.
- If a pairings has cycle, then we increase the trapped case by one.
- Then, at each printing of pairing location, we should check each pairing if there exists a cycle in the pairing. An isCycle() need to be checked.

Nature of the isCycle() Problem

- Repeating the steps on finding partner, then find the partner's next neighbor on the right as a new point to find the next partner.
- If the last neighbor on the right exists, then, there is a cycle.
- If the last next neighbor to the right does not exists, then, there is no cycle.

isCycle() Function

```
public static boolean isCycle() {
    for(int start = 0; start <N; start++) {
      int pos = start;
      for(int count = 0; count < N; count++) {
         if (pos != -1) pos = next_on_right[partners[pos]];
      }
      if(pos != -1) return true;
    }
    return false;
}
```

Techniques Needed to Solve this Problem

1. Generation of all possible pairing.
2. Check with there is a cycle by depth-first-searching

We will discuss the basic graphs theory, generation of all possible paring and, then, come back to solve this problem.