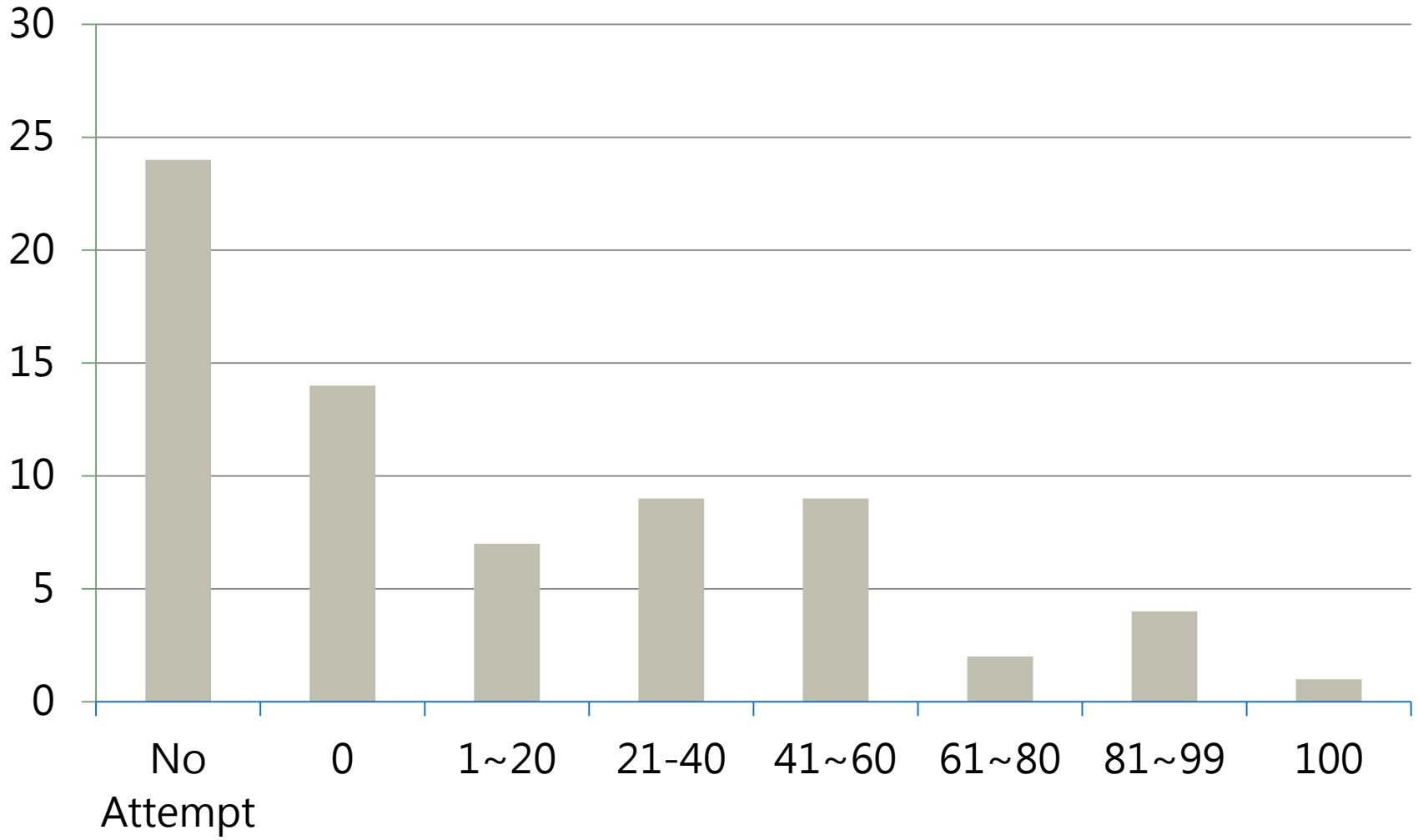
Power Socket

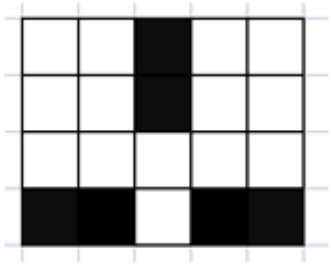
Tony Wong
January 11, 2014

Statistics



Description

- There is a wall with some holes.
- A socket = 6 holes are in a particular arrangement
- How many sockets are there in the wall?



- Size of wall = W x H
- Number of holes = N
- Socket (Figure 3) can be rotated

Sample

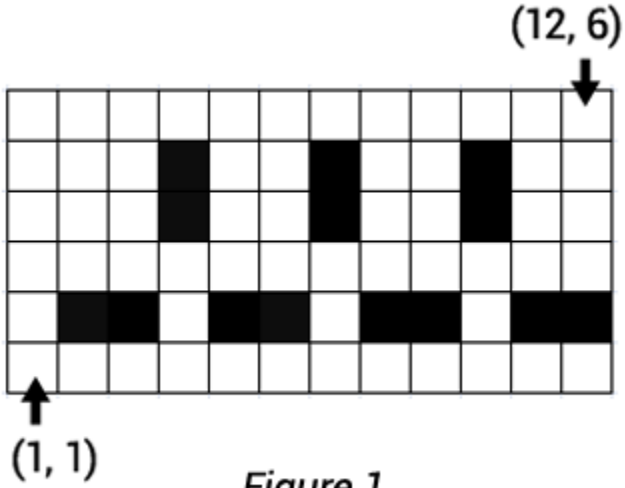


Figure 1

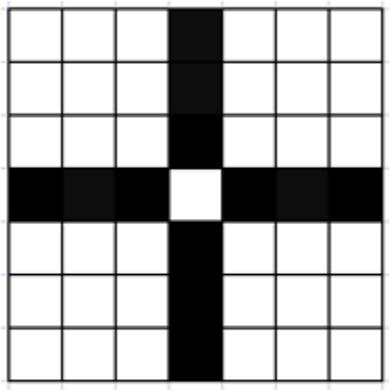


Figure 2

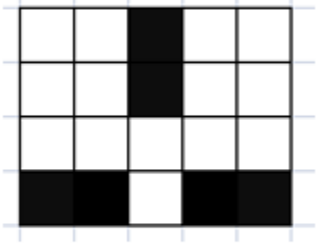
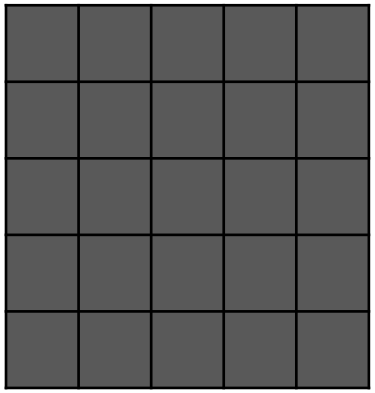


Figure 3

Question

- What is the output of the following wall?



- What about a 40000 x 40000 wall with all four 100x100 corners filled with holes?
  - i.e. hole if and only if x=1..100, 39901..40000 and y=1..100, 39901..40000
  - Total: 40000 holes

Constraints

- In test cases worth 30% of the total points
  - $1 \le W, H \le 1000$, $6 \le N \le 1000$, only upright
- In test cases worth 50% of the total points
  - $1 \le W, H \le 1000$, $6 \le N \le 1000$
- In test cases worth 70% of the total points
  - $1 \le W, H \le 40000$, $6 \le N \le 2000$
- In all test cases
  - $1 \le W, H \le 40000$, $6 \le N \le 50000$

- It is worth noting that $N << WH$

Solution

- 50%
  - Runtime: O(WH)
  - Memory: O(WH)

50% Solution

- Create a W x H array
- Fill array with 0
- For each hole, mark as 1, and the check if this hole can form a socket
  - Each hole can be checked in constant time
  - More explained later

Solution

- 50%
  - Runtime: O(WH)
  - Memory: O(WH)
- 70%
  - Runtime: O(N*N)
  - Memory: O(N)

70%

- Just store the input as-is in an array
- For each hole, try 4 different directions by linear searching the array.
- Best case: 4 searches (1 search / direction)
- Worst case: 20 searches (5 searches / direction)

- Quite easy to implement

Solution

- 50%
  - Runtime: O(WH)
  - Memory: O(WH)
- 70%
  - Runtime: O(N*N)
  - Memory: O(N)
- 100%
  - Runtime: O(N lg N)
  - Memory: O(N)

100%: Binary search

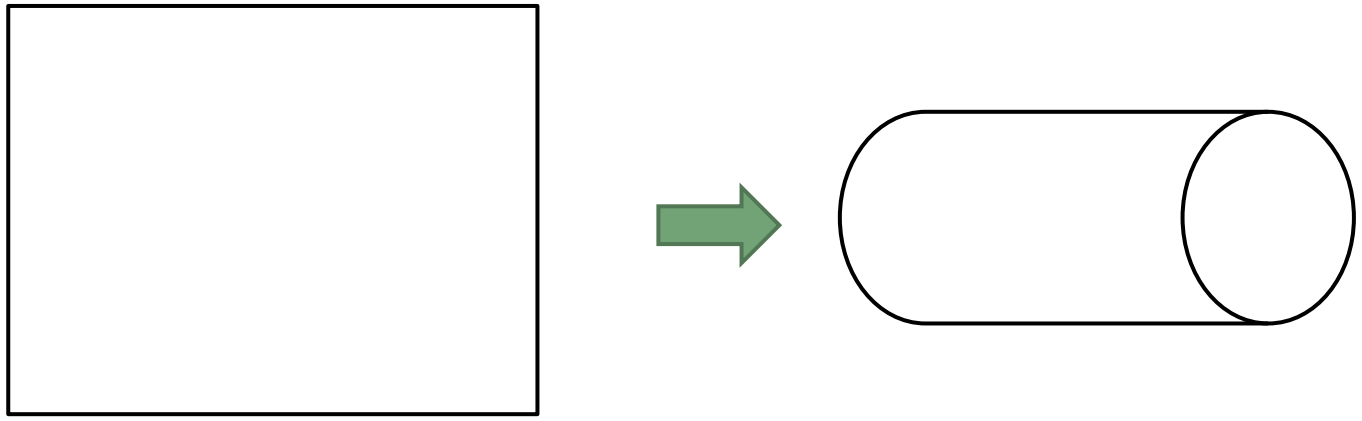
- Change 70% linear search into binary search
- You can either:
  - Change the if statement into

```
if (x == xx && y == yy) {
...
} else if (x < xx || (y == yy && y < yy)) {
...
} else {
...
}
```

  - Use a numeric search key
    - Key = 41000x + y
    - 41000 x 40000 + 40000 = 1640040000 < 2147483647

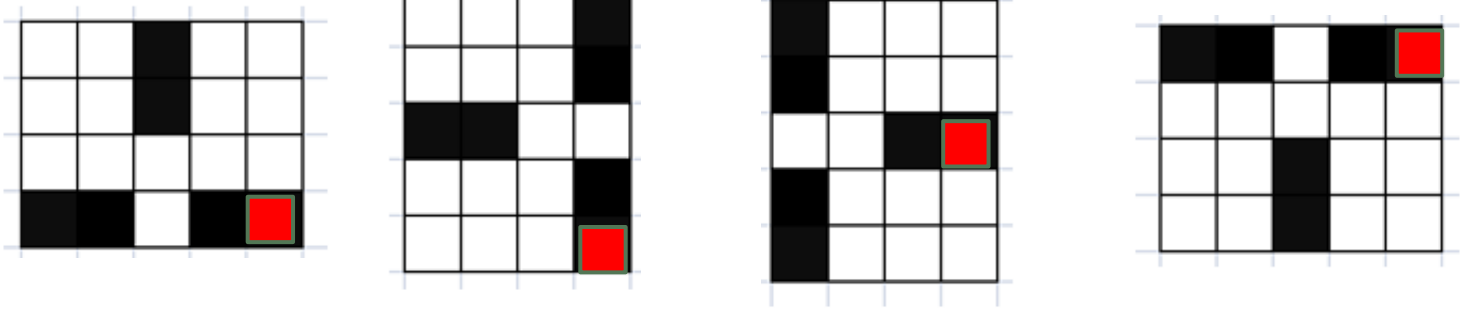
Note

- If you choose 41000, range checking can be skipped
- You should NOT use 40000 if you don't perform range checking
  - Otherwise you will be wrapping the wall into a cone
  - Example: (100, -4) == (99, 39996)



How to check on-the-fly?

- Of course you can store all the holes into an array and process one by one.
- We can actually check immediately after every input if we search the smaller keys every time.
- The max search key of the 4 directions are:



Wait….

- Since HKOI has no memory limit
- Can't I just use a very large array?
  - e.g. char a[40000][40000];

- Questions

1. Does the system allow you to allocate 1.6GB memory?
2. If yes, how much time will it take?
3. Also, how much time required to initialize the array?
4. How much time required to check the holes?

Want to know the details?

- Attend "Miscellaneous CS Topics"
  - Computer architecture
  - Operating system
  - Programming languages
  - Software development