

Problem 1

Here is the **Important** part of the problem.

1. Entry 1
2. Entry 2
3. $O(\log_2(n))$

Problem 2

Algorithm:

```
i ← 1
while houses[i] == "keep going":
    i* = 2
return bsearch(i / 2, i)
```

Runtime:

The bounds of i will always be $2^k \leq h \leq 2^{k+1}$ where $\lfloor \log(h) \rfloor = k$.
 $\log(2^{k+1} - 2^k) \leq \log(k) + C \rightarrow O(\log(h))$.

This means the algorithm runs in $O(\log(h))$ time.

Correctness:

Hypothesis: Returns the index to where houses will return "Party's here!" without making changes to houses.

Base Case: $h = 1$, the alg. returns 1. ✓

Induction Step:

Assume:

$\forall i < n$, it returns i when $i = h$.

The function considers some range $[i, 2i]$ where $i = \lfloor \log_2(h) \rfloor$. We also know this will return the correct result as, we know $h \geq i$ and $h \leq 2i$. Meaning we have not throw out the correct h and do not have to consider any previous numbers.

This proves that our algorithm will always find the correct h according to our hypothesis.

Problem 3

Hypothesis: Given any board, $2^k \times 2^k$, with one tile missing/already filled. Every tile can be filled by placing some rotated version of an L piece.

Base Case:

$k = 1$ is a 2×2 board, with one spot missing. It can be solved trivially with one L piece.

Assume:

$\forall p < k$, the $2^p \times 2^p$, is possible to cover the entire board with L pieces.

Consider:

A $2^k \times 2^k$ board. We divide and conquer by going into all 4 sub-quadrant, splitting up the board into 4 equally sized sub-quadrants. We continue this until we reach all 2×2 sub-quadrants, from here, we fill the trivial sub-quadrant with a spot missing. Now we can consider this sub-quadrant to be itself a missing spot, and all other recursive calls can back out a step. From here every 2×2 grid can be treated as a 1×1 grid. This means we now have a $2^{k-1} \times 2^{k-1}$ grid with a single spot missing.

This proves our hypothesis, since we assumed that the $k - 1$ case must be true.