

Divide & Conquer Technique

Finding Max and Min

Divide and Conquer

General Info

- We solve a problem recursively, applying three steps at each level of the recursion.
 - Divide the problem into a number of subproblems that are smaller instances of the same problem.
 - Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
 - Combine the solutions to the subproblems into the solution for the original problem.
- When the subproblems are large enough to solve recursively, we call that the ***recursive case***.
- Once the subproblems become small enough that we no longer recurse, we say that the recursion “bottoms out” and that we have gotten down to the ***base case***.

Recurrence

A natural way to characterize the running time

- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.
- Recurrences can take many forms.
- Three methods for solving recurrences—that is, for obtaining asymptotic “ θ ” or “ O ” bounds on the solution:
 - In the ***substitution method***, we guess a bound and then use mathematical induction to prove our guess correct.
 - The ***recursion-tree method*** converts the recurrence into a tree whose nodes represent the costs incurred at various levels of the recursion. We use techniques for bounding summations to solve the recurrence.
 - The ***master method*** provides bounds for recurrences of the form
 - $T(n) = aT(n/b) + f(n)$, where $a \geq 1, b > 1$, and $f(n)$ is a given function.
 - Creates a subproblems, each of which is $1/b$ the size of the original problem, and in which the divide and combine steps together take $f(n)$ time.

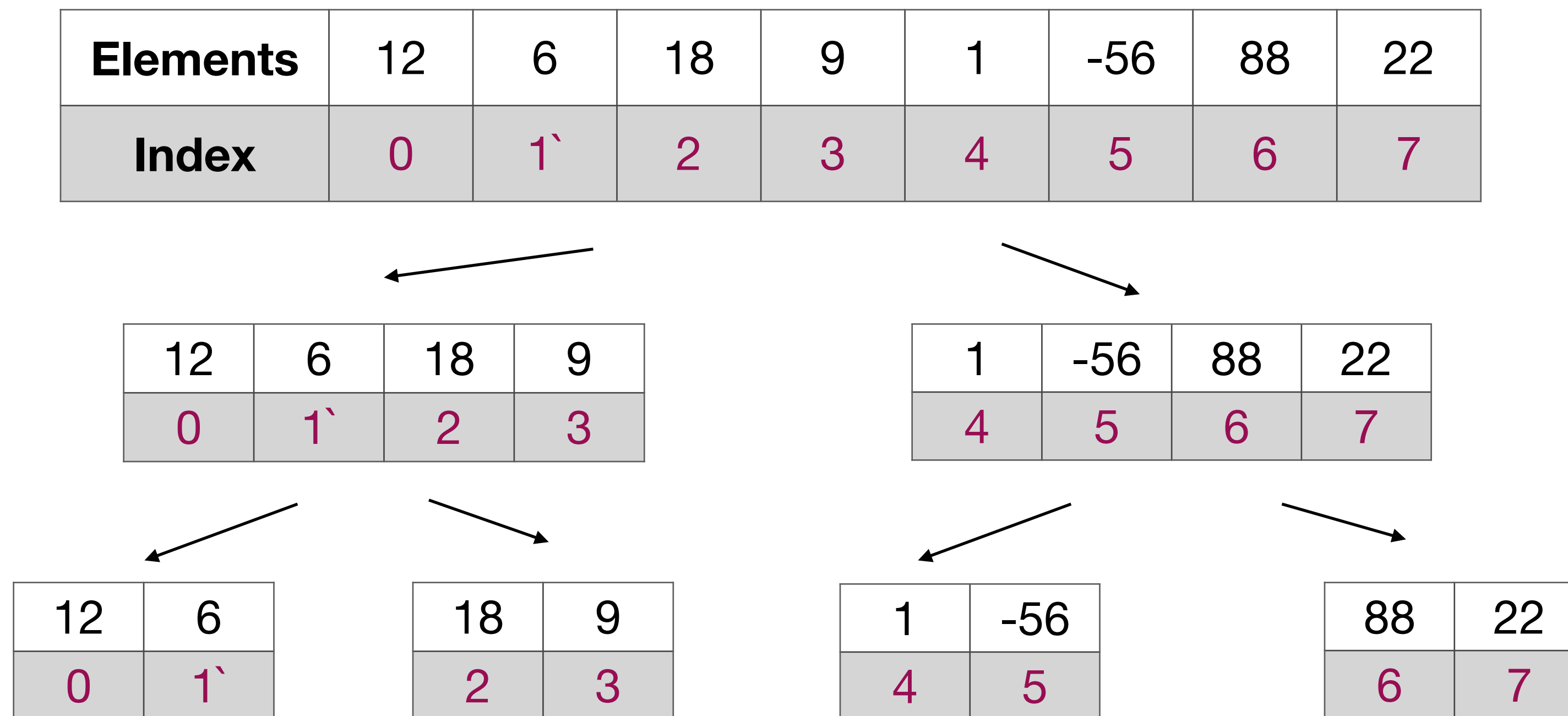
Divide and Conquer - Example

Finding the Max & Min

- ***Problem Statement:*** An array of elements is provided and you are supposed to find the Maximum element and Minimum element from the array in an efficient way.
- Solution:
 - Oh my god! It's so easy, just compare ***all*** the elements and easily we can find.
 - Can we call the above approach as efficient? Or Is there any way other than this?

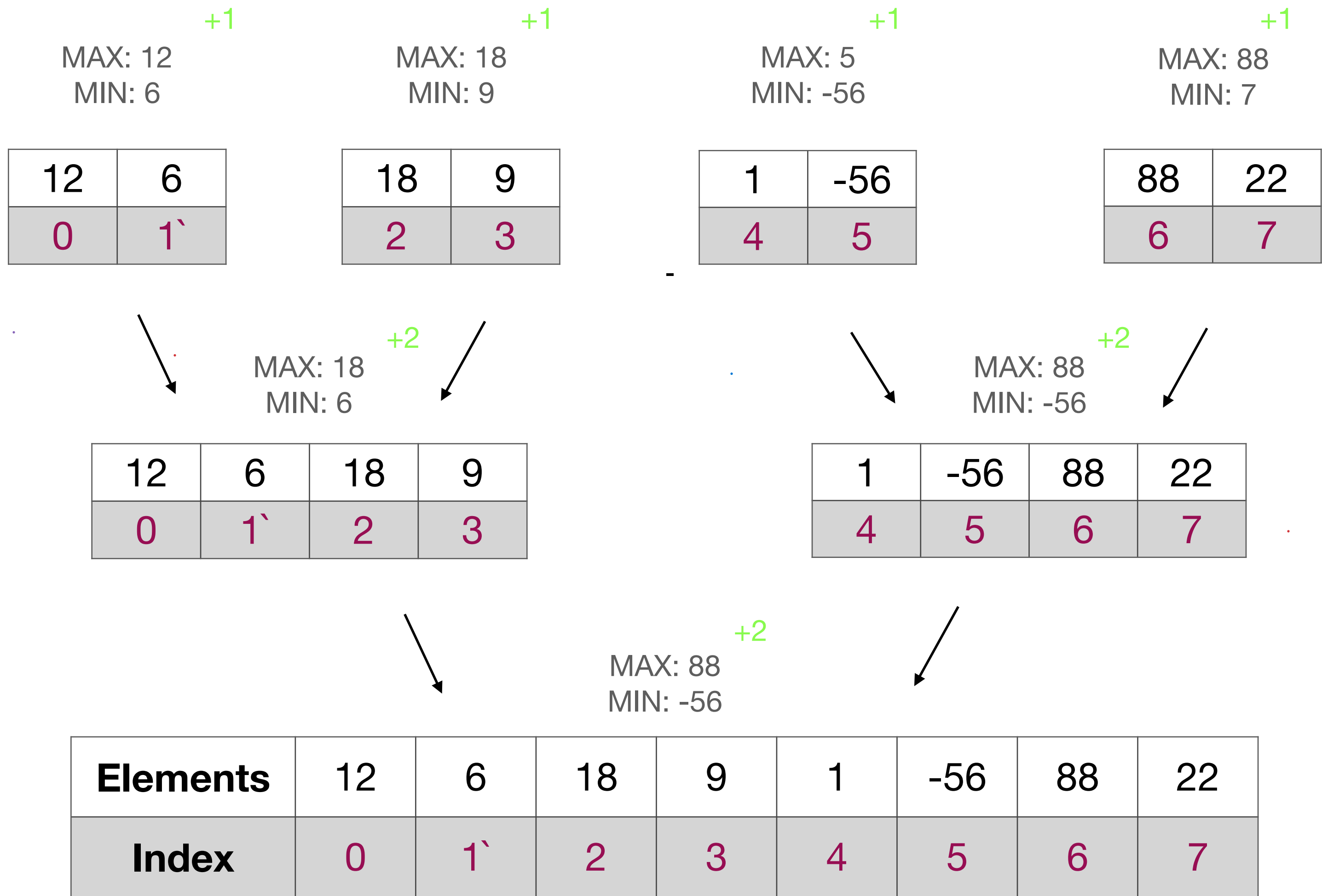
Finding Max & Min

Divide & Conquer



Finding Max & Min


Divide & Conquer




Total Comparison = 10

If not?


Linear Way



Elements	12	6	18	9	1	-56	88	22
Index	0	1	2	3	4	5	6	7



Elements	12	6	18	9	1	-56	88	22
Index	0	1	2	3	4	5	6	7



Elements	12	6	18	9	1	-56	88	22
Index	0	1	2	3	4	5	6	7

For Max ?
Also for Min?

Complexity Analysis

Master method

$$T(n) = \begin{cases} 0, & \text{if } n = 1 \\ 1, & \text{if } n = 2 \\ 2T(n/2) + 2, & \text{if } n \geq 2 \end{cases}$$

$$= T(n/2) + T(n/2) + 1 + 1$$

$$= 2T(n/2) + 2$$

To expand the Equation, substitute the values of n ,

$$T(n/2), T(n/4), T(n/8) \dots$$

Finally, we get

$$\begin{aligned} T(n) &= \frac{3n}{2} - 2 \\ &= \mathcal{O}(n) \end{aligned}$$

Extra Reading

Do it at your convenience

- Using divide and Conquer technique, solve the Maximum Sub Array Problem
- Is it possible to find the non duplicate elements in a sorted array using the divide and conquer strategy? Check out [here](#)
- If you would like to see more solved examples, Click [here](#)

Give a Try

Friend Finder - Must Try once

Jean-Locutus Πcard is searching for his incapacitated friend, Datum, on Gravity Island. The island is a narrow strip running north–south for n kilometers, and Πcard needs to pinpoint Datum's location to the nearest integer kilometer so that he is within visual range. Fortunately, Πcard has a tracking device, which will always tell him whether Datum is north or south of his current position (but sadly, not how far away he is), as well as a teleportation device, which allows him to jump to specified coordinates on the island in constant time.

Unfortunately, Gravity Island is rapidly sinking. The topography of the island is such that the north and south ends will submerge into the water first, with the center of the island submerging last. Therefore, it is more important that Πcard find Datum quickly if he is close to either end of the island, lest he short-circuit. Describe an algorithm so that, if Datum is k kilometers from the nearest end of the island (i.e., he is either at the k th or the $(n - k)$ th kilometer, measured from north to south), then Πcard can find him after visiting $O(\log k)$ locations with his teleportation and tracking devices.

Source: [MIT Open Courseware](#), Instructors: Erik Demaine, Jason Ku, and Justin Solomon

Friend Finder

Solution

Solution: We can generalize the idea of binary search to search quickly from both ends. The idea will be to alternately search inward from either end of the island exponentially until Π card just passes Datum, and then use normal binary search to pinpoint Datum's precise location. Specifically, tell Π card to alternately teleport to 2^i and $n - 2^i$ for increasing i starting at 0 until either Datum is found, or until Datum has been passed, i.e., Datum is observed north of 2^{j-1} but south of 2^j for some j , or south of $n - 2^{j-1}$ but north of $n - 2^j$ for some j . Reaching this state will take $O(j)$ time, since at most $2j$ locations will be visited, and then binary searching within the remaining 2^{j-1} kilometer stretch of the island will also take at most $O(j)$ time. But since either $2^{j-1} < k < 2^j$ or $n - 2^j < n - k < n - 2^{j-1}$, then $j - 1 < \lg k < j$ and $j = O(\log k)$, as desired. This algorithm is correct because it identifies a bounded range where Datum is known to exist, and reduces to binary search which will correctly return Datum's location.

Source: [MIT Open Courseware](#), Instructors: Erik Demaine, Jason Ku, and Justin Solomon