1 (https://www.inneconstructions.com/constructio

You've built an in-flight entertainment system with on-demand movie streaming.

Users on longer flights like to start a second movie right when their first one ends, but they complain that the plane usually lands before they can see the ending. So you're building a feature for choosing two movies whose total runtimes will equal the exact flight length.

Write a function that takes an integer flightLength (in minutes) and an array of integers movieLengths (in minutes) and returns a boolean indicating whether there are two numbers in movieLengths whose sum equals flightLength.

When building your function:

- Assume your users will watch exactly two movies
- · Don't make your users watch the same movie twice
- · Optimize for runtime over memory

Gotchas

We can do this in O(n) time, where n is the length of movieLengths.

Remember: your users shouldn't watch the same movie twice. **Are you sure your method won't** give a false positive if the array has one element that is half flightLength?

Breakdown

How would we solve this by hand? We know our two movie lengths need to sum to flightLength. So for a given firstMovieLength, we need a secondMovieLength that equals flightLength - firstMovieLength.

To do this by hand we might go through movieLengths from beginning to end, treating each item as firstMovieLength, and for each of those check if there's a secondMovieLength equal to flightLength - firstMovieLength.

How would we implement this in code? We could nest two loops (the outer choosing firstMovieLength, the inner choosing secondMovieLength). That'd give us a runtime of $O(n^2)$. We can do better.

To bring our runtime down we'll probably need to replace that inner loop (the one that looks for a matching secondMovieLength) with something faster.

We could sort the movieLengths first—then we could use $\underline{binary \, search}$ to find secondMovieLength in $O(\lg n)$ time instead of O(n) time. But sorting would cost O(nlg(n)), and we can do even better than that.

Could we check for the existence of our secondMovieLength in constant time?

What data structure gives us convenient constant-time lookups?

A set!

So we could throw all of our movieLengths into a set first, in O(n) time. Then we could loop through our possible firstMovieLengths and replace our inner loop with a simple check in our set. This'll give us O(n) runtime overall!

Of course, we need to add some logic to make sure we're not showing users the same movie twice...

But first, we can tighten this up a bit. Instead of two sequential loops, can we do it all in one loop? (Done carefully, this will give us protection from showing the same movie twice as well.)

Solution

We make one pass through movieLengths, treating each item as the firstMovieLength. At each iteration, we:

- 1. See if there's a matchingSecondMovieLength we've seen already (stored in our movieLengthsSeen set) that is equal to flightLength firstMovieLength. If there is, we short-circuit and return true.
- 2. Keep our movieLengthsSeen set up to date by throwing in the current firstMovieLength.

```
function canTwoMoviesFillFlight(movieLengths, flightLength) {
    // movie lengths we've seen so far
    var movieLengthsSeen = new Set();

    for (var i = 0; i < movieLengths.length; i++) {
        var firstMovieLength = movieLengths[i];

        var matchingSecondMovieLength = flightLength - firstMovieLength;
        if (movieLengthsSeen.has(matchingSecondMovieLength)) {
            return true;
        }

        movieLengthsSeen.add(firstMovieLength);
    }

    // we never found a match, so return false
    return false;
}</pre>
```

We know users won't watch the same movie twice because we check movieLengthsSeen for matchingSecondMovieLength before we've put firstMovieLength in it!

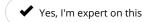
Complexity

O(n) time, and O(n) space. Note while optimizing runtime we added a bit of space cost.

Bonus

- 1. What if we wanted the movie lengths to sum to something close to the flight length (say, within 20 minutes)?
- 2. What if we wanted to fill the flight length as nicely as possible with *any* number of movies (not just 2)?
- 3. What if we knew that movieLengths was sorted? Could we save some space and/or time?

Did you get it right?





Like this problem? Pass it on!

f Share



u=https://www.intexxi>i6សlvak9620tt/iq%250icord/jag9620jpt/ein/ligk/%20question%21&via=interviewcake&relate entertainment) entertainment)

Yo, follow along!



(http:////www.wictebroccolorcolonteinvienvienvienke)ke)

Subscribe to our weekly question email list » (/free-weekly-coding-interview-problem-newsletter)

Programming interview questions by company:

- Google interview questions (/google-interview-questions)
- Facebook interview questions (/facebook-interview-questions)
- Amazon interview questions (/amazon-interview-questions)

Programming interview questions by language:

- Java interview questions (/java-interview-questions)
- Python interview questions (/python-interview-questions)
- Ruby interview questions (/ruby-interview-questions)
- JavaScript interview questions (/javascript-interview-questions)
- **NEW:** Testing and QA interview questions (/testing-and-qa-interview-questions)
- **NEW:** SQL interview questions (/sql-interview-questions)

Copyright © 2016 Cake Labs, Inc. All rights reserved.

228 Park Ave S #82632, New York, NY US 10003 (804) 876-2253

Privacy (/privacy-policy) | Terms (/terms-and-conditions)