

Lab: Searching

- Exercises
 - Exercise 0: Preparation
 - Exercise 1: Testing Our Procedures
 - Exercise 2: Extending linear-search-vector
 - Exercise 3: Searching Files
 - Exercise 4: Binary Searching the Class
 - Exercise 5: Binary Searching the Class, Revisited
 - Exercise 6: Observing Binary Search
 - Exercise 7: A Guessing Game

Exercises

Exercise 0: Preparation

- a. If you have not done so already, please scan the reading on searching. In particular, you should look at the sample procedures.
- b. Make a copy of the library file that contains the code from that reading.
- c. Start DrScheme.

Exercise 1: Testing Our Procedures

- a. Using `linear-search-list`, search for the letter `a` in various lists of characters. (It's probably easiest to create a list of characters with `string->list`.)
- b. Using `linear-search-vector`, search for the letter `#\a` in various vectors of characters.
- c. Develop some tests for `search-list-for-keyed-value`.

Exercise 2: Extending linear-search-vector

Write a procedure that takes a predicate and vector as parameters and, using `linear-search-vector` as a helper, finds a value in the vector that matches the predicate or returns `#f` if no such value exists. (Like `linear-search-vectors`, this searches vectors; unlike `linear-search-vector` (and like `linear-search-list`) this returns a matching value, rather than an index.

Exercise 3: Searching Files

Define and test a Scheme procedure, (`search-file pred? port`) that reads in Scheme values from a given input port, applying a specified test to each one. When it finds a value that passes the test, it should return that value; if it gets the end-of-file object before finding a value that passes the test, it should call the `error` procedure to print an appropriate diagnostic.

Exercise 4: Binary Searching the Class

In the accompanying library file, you'll find `lastnames`, a vector of the last names of people in this class.

Call the `binary-search` procedure with appropriate arguments to determine the position of your surname in this vector.

You probably want the `get-key` procedure to resemble

```
(lambda (name) name)
```

Exercise 5: Binary Searching the Class, Revisited

If you look at the library file, you'll find a list of student records called `students`.

Write a procedure, (`lookup-student lastname`) that calls the `binary-search` procedure with appropriate arguments and returns the information for the appropriate student.

Exercise 6: Observing Binary Search

Add calls to `display` and `newline` to the definition of `binary-search`, so that it prints out the values of `lower-bound` and `upper-bound` each time the kernel procedure is called. How many recursive calls are made as binary search finds your surname in the list? How many are made in the course of an unsuccessful search for the surname "Stone"?

Exercise 7: A Guessing Game

The *divide-and-conquer* principle can be applied in other situations. For example, we can apply it to a guessing game in which one player, A, selects a number in the range from 1 to some value and the other player, B, tries to guess it by asking yes-or-no questions of the form "Is your number less than n ?" (putting in specific values for n). The most efficient strategy for B to use is repeated bisection of the range within which A's number is known to lie.

Write a Scheme procedure that takes the part of B in this game. Your procedure should take the maximum possible value as a parameter. When invoked, it should print out a question of the specified form and read in the user's response (presumably, the symbol `yes` or the symbol `no`), then repeat the process until the range of possible values has been narrowed to contain only one number. The procedure should then display and identify that number. A sample run might look like this:

> (**guessing-game** 100)

Is your number less than 51? **yes**

Is your number less than 26? **no**

Is your number less than 38? **no**

Is your number less than 44? **no**

Is your number less than 47? **yes**

Is your number less than 45? **no**

Is your number less than 46? **no**

Since your number is less than 47 but not less than 46, it must be 46.