Name: Email: @umbc.edu

Arrays and Lists
January 29, 2020

In many experiments or surveys, it is common to have missing values for some observations -- variables for which no data is available. For example, many survey takers will choose not to reveal their race, income, or gender.

Suppose we are performing an experiment or survey in which each observation consists of N variables:

```
x_0, x_1, x_2, ..., x_{N-1}
```

For any observation, there are likely to be many missing values. It is possible that only a small percentage of the variables will have values.

We can use either an array or a linked list to store an observation (a set of N variables). We need to be able to access any of the N variables in an observation by index.

* How would you use an array to store an observation (set of N variables)?

* How would you use a linked list to store an observation?

* Compare the performance of the Retrieve operation for both data structures.

* Which data structure is preferable for reducing memory usage?

Solution

The most obvious way to store the data in an array is to create an N-long array, and let index i store the valuefor x_i, 0 <= i < N. Then retrieval by index is constant time: just do a normal array element access.

To store the elements in a list, we would create a Node class (or struct) with several variables:

* An index to indicate which variable is stored in the node. E.g. if x_3 is stored in the node, the index would be 3.

* A data variable to hold the value of the variable.

* A next pointer used to link the nodes into a list.

To retrive a value of a variable, x_i, we would have to iterate over the linked list until we found a node with index == i, or until we reached the end of the list, in which case the value is missing.

Although the array implementation has fast retrieval, it is not space efficient if there are a high proportion of missing values: most of th entries in the array will be unused. However, the linked list does not store missing values; we know that a variable is missing if we do *not* find it in the list.

Which implementation you would choose depends largely on what proportion of the variables you expect to be missing. If many are missing, then the array is extremely wasteful and the linked list will

```
be short, so iterating will not take so long.  On the other hand, if
few variables are missing, you might accept a small amount of wasted
memory for the sake of fast retrievals.
```