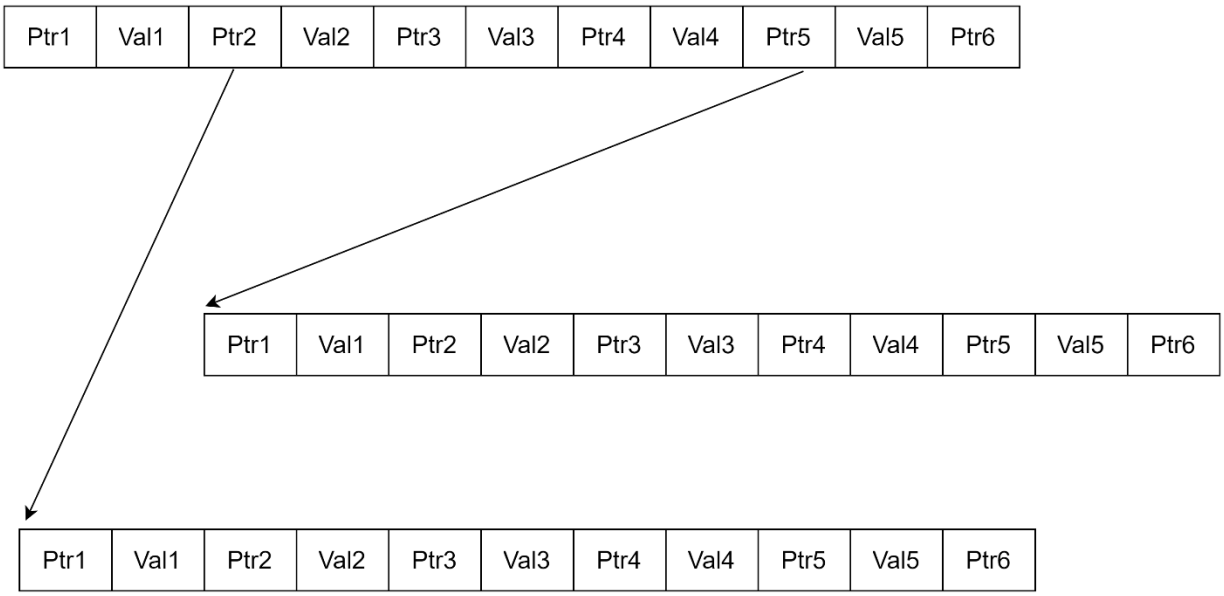
Scheme Assignment – B+ Trees

Introduction

There are many different types of trees. You may know tries and/or binary trees. One type of tree that is used in many different situations is the B+ tree. B+ trees are used when we are dealing with blocks of data, often loaded from disk. Filesystems and databases use B+ trees.

The fundamental idea is that we will have a block of both pointers and values:



The size of the block is based on the disk's block size.

To search for S, we load the head node block from disk, then we search the data items for the value we are looking for. If we find it, great. If not, one of three conditions must be true:

1) S < Smallest value (Val1). If so, follow the first pointer (1).
2) S > Largest value (Val 5). If so, follow the last pointer (6)
3) S between two values. If so, follow pointer between those two values.

This approach minimizes the number of disk blocks you must load. Binary tree allows you to find an item in O(log n) time. B+ tree allows you to find an item in O(log SIZE) time, where SIZE = number of items in the block. Here memory access time is far outweighed by disk block load time.

More information and an illustration available: https://en.wikipedia.org/wiki/B%2B_tree

Details

We won't be using pointers in Scheme. Instead, you will be constructing a tree (sample provided below) with data items and lists interspersed. You will then create a function "walk" that takes the tree and a data item to search for. Your function will return true if the item is in the tree and false if the item is not in the tree.

Rules:

1) You must use Scheme R5RS in Dr. Racket. No other language/IDE is acceptable.
2) You must not hard code anything. Your solution should work for any tree of any depth or nodes of any width > 1.
3) You must not use any functions not included in the slides unless you write them.
4) You must construct your own tree(s) for testing. You must test your code.
5) Your solution must be recursive.

This is not a long assignment. My solution is 5 lines. Yours may be longer (or shorter!) but it should not be dramatically longer or shorter.

Example Tree:

```
(define ones (list '() 2 '() 4 '() 6 '() 7 '() 8 '()))

(define sixties (list '() 52 '() 54 '() 66 '() 67 '() 68 '()))

(define tens (list ones 10 '() 30 '() 50 sixties 70 '() 90 '()))

(define ninehundreds (list '() 930 '() 940 '() 960 '() 970 '() 988 '()))

(define tree (list tens 100 '() 300 '() 500 '() 700 '() 900 ninehundreds ))
```

Test:

```
(walk tree 0) ; returns false

(walk tree 6) ; returns true

(walk tree 67) ; returns true

(walk tree 750) ; returns false

(walk tree 970) ; returns true

(walk tree 900) ; returns true

(walk tree 1100) ; returns false
```

Dr. Racket will save your definitions (File/Save As) as a ".rkt" file. You can submit that way or as a basic text file (the file extension doesn't matter).

Because this is such a short assignment, there is no meaningful rubric. It will be graded:

0 – missing or no significant effort shown

50 – works for at least a few non-trivial cases

100 - correct