Data Structures : Lab 12

Edumi (Tony) Sparks
11-10-21

1) a)

best Case : Comparisons 0 Scenario Hash lands on one with no keys attached to it

Worst Case : Comparisons 3 Scenario Longest list is 3 keys. Any other key whose hash = 2 for Unsuccessful.

$\alpha = 9/10 = 4/5$

b)

Average Case for Successful Searches:

| 10 | 70 | 32 | 52 | 12 | 23 | 26 | 99 |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 3 | 1 | 1 | 1 |

$= 12/9 = 3/2$

c)

Average Case for UnSuccessful Searches:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 3 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

$= 8/10 = 4/5$

d) What is the overall average Case:

$1.5 \times 0.3 = 1.01$
$0.8 \times 0.7$

---

2)

$\alpha = 3.25$ $C_{ave} = 3.15625$
$u = 3.25$
$S = 1 + 3.25/2 = 2.625$

---

3)

$\alpha = 0.816$ $C_{ave} = 8.03779$
$u = 1/2(1 + 1/(1-\alpha)^2) = 15.2684$
$S = 1/2(1 + 1/1-\alpha) = 3.21739$

No because the load factor is greater than the keys.

---

4) Screen Shot @ end

5) Screenshot @ end

- Tony Sparks

codingground | Compile and Execute C Online (GNU GCC v7.1.1)

Execute | Share | main.c | STDIN

```c
#include <stdio.h>
#include <stdlib.h>

// Structure that represents a node of the 2-3 tree
struct Node {
    struct Node* pLeft;
    int leftVal;
    struct Node* pMiddle;
    int leftVal;
    struct Node* pRight;
};

// Utility function that creates a new node for the 2-3 tree
struct Node* createNewNode(int left, int right) {
    struct Node* pNewNode = (struct Node*) malloc(sizeof(struct Node));
            sizeof(struct Node));
    pNewNode->leftVal  = left;
    pNewNode->rightVal = right;
    pNewNode->pLeft    = NULL;      //initially unconnected
    pNewNode->pMiddle  = NULL;      //initially unconnected
    pNewNode->pRight   = NULL;      //initially unconnected
    return pNewNode;
}


// Utility function to traverse and print all the values in a 23-tree
void print23Tree(struct Node* pNode) {
    //recursive traversal!
}

int main() {
    struct Node *pRoot = createNewNode(0, 0);
    pRoot->leftVal=8;

    struct Node *pTemp = createNewNode(3, 5);
    pRoot->pLeft=pTemp;

    pTemp = createNewNode(9, 0);
    pRoot->pMiddle=pTemp;

    print23Tree(pRoot);
}
```

```
    pNewNode->leftVal  = left;
    pNewNode->rightVal = right;
    pNewNode->pLeft    = NULL;      //initially unconnected
    pNewNode->pMiddle  = NULL;      //initially unconnected
    pNewNode->pRight   = NULL;      //initially unconnected
    return pNewNode;
}

// Utility function to traverse and print all the values in a 23-tree
void print23Tree(struct Node* pNode) {
    if (pNode!=NULL){
        print23Tree(pNode->pLeft);
        printf("%d ", pNode->leftVal);
        print23Tree(pNode->pMiddle);
        printf("%d ", pNode->rightVal);
        print23Tree(pNode->pRight);
    }
}

int main() {
    struct Node *pRoot = createNewNode(0, 0);
    pRoot->leftVal=8;

    struct Node *pTemp = createNewNode(3, 5);
    pRoot->pLeft=pTemp;

    pTemp = createNewNode(9, 0);
    pRoot->pMiddle=pTemp;

    print23Tree(pRoot);
}
```

```
/tmp/qxpOvdddb6.o
3 5 8 9 0 0
```