

# Data Structures and Algorithms

## Solutions 7

Markus Innerebner, Romans Kasperovics  
dsa(a)inf.unibz.it

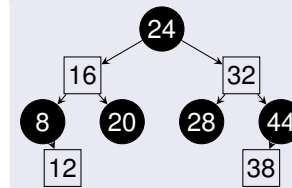
Free University of Bozen - Bolzano, Italy

Thu, April 16, 2009

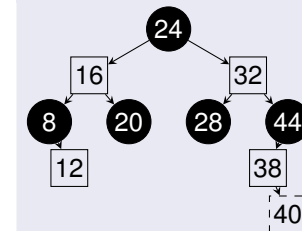
## Solution 07

### Task 1: Insert 40

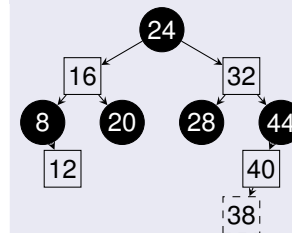
#### 1) Initial



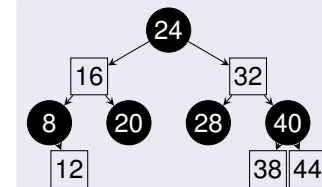
#### 2) Insertion Position



#### 3) Case 2



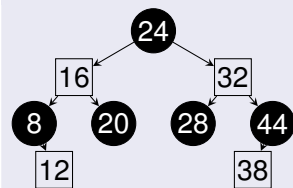
#### 4) Case 3



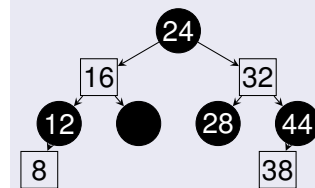
## Solution 07

### Task 1: Delete 20

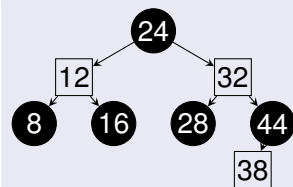
#### 1) Initial



#### 2) Case 3m



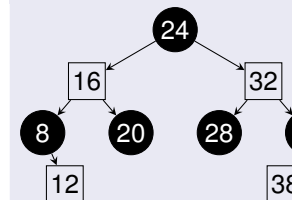
#### 3) Case 4m



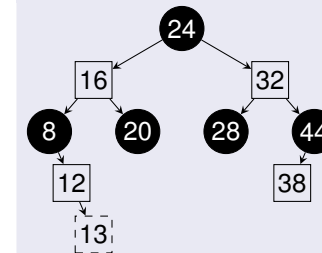
## Solution 07

### Task 1: Insert 13

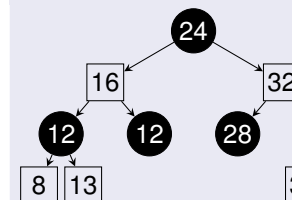
#### 1) Initial



#### 2) Insertion Position



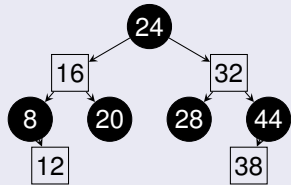
#### 3) Case 3m



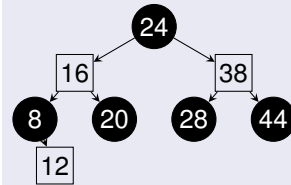
## Solution 07

### Task 1: Delete 32

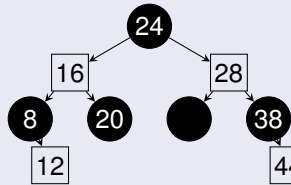
#### 1) Initial



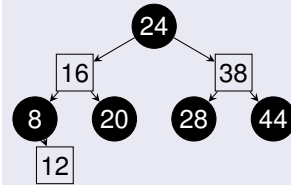
#### 2) Option I Case 1



#### 2) Option II Case 3



#### 4) Option II Case 4



## Solution 07

### Task 2: Init

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>
#define DELETEDVAL -2
#define EMPTYVAL -1
#define EMPTYINDEX -1
typedef enum { false, true } bool;
```

```
int* init(int m) {
    int i;
    int* result = (int*) malloc (m * sizeof(int));
    if (result != NULL)
        for(i=0; i<m; i++)
            result[i] = EMPTYVAL;
    return result;
}
```

## Solution 07

### Task 2: Hashval, Search

```
int hashval(int key, int i, int m) {
    return (key % 5 + i * ((2 * key + 1) % m)) % m;
}
int search(int H[], int m, int key, bool *found) {
    int p, i;
    *found = false;
    for (i=0; i<m; i++) {
        p = hashval(key, i, m);
        if (H[p] == key) {
            *found = true;
            return p;
        }
        if (H[p] == EMPTYVAL)
            break;
    }
    return EMPTYINDEX;
}
```

## Solution 07

### Task 2: Insert

```
int insert(int H[], int m, int key, bool* inserted) {
    int p, i = 0, j = EMPTYINDEX;
    do {
        p = hashval(key, i++, m);
        if (j == EMPTYINDEX && (H[p] == EMPTYVAL || H[p] == DELETEDVAL))
            j = p;
    } while (H[p] != -1 && H[p] != key && i < m);
    if (j == EMPTYINDEX || H[j] == key) {
        *inserted = false;
        return EMPTYINDEX;
    } else {
        *inserted = true;
        H[j] = key;
        return j;
    }
}
```

## Solution 07

### Task 2: Delete

```
int delete(int H[], int m, int key, bool* deleted) {  
    int p = search(H, m, key, deleted);  
    if (*deleted) {  
        H[p] = DELETEDVAL;  
        return p;  
    }  
    return EMPTYINDEX;  
}
```