Fundamental Algorithms, Assignment 5
Solutions

1. Consider hashing with chaining using as hash function the sum of the numerical values of the letters (A=1,B=2,...,Z=26) mod 7. For example, \( h(\text{JOE}) = 10+15+5 \mod 7 = 2 \). Starting with an empty table apply the following operations. Show the state of the hash table after each one. (In the case of Search tell what places were examined and in what order.)

- Insert COBB
- Insert RUTH
- Insert ROSE
- Search BUZ
- Insert DOC
- Delete COBB

Solution: Let \( T[0\cdots6] \) be the hash table which is \{NIL, NIL, NIL, NIL, NIL, NIL, NIL\} initially. Let \( \text{num}(\cdot) : \{A, B, \cdots, Z\} \rightarrow [1\cdots, 26] \) be the specified bijection which maps a letter to its numerical value. We have

- Insert COBB:
  \[
  \text{num}(C) + \text{num}(O) + \text{num}(B) + \text{num}(B) \mod 7 = (3 + 15 + 2 + 2) \mod 7 = 22 \mod 7 = 1
  \]
  \( T[1] \) is empty, so “COBB” is placed in \( T[1] \).
  \( T[0\cdots6] = \{\text{NIL}, \text{“COBB”}, \text{NIL}, \text{NIL}, \text{NIL}, \text{NIL}, \text{NIL}\} \).

- Insert RUTH:
  \[
  \text{num}(R) + \text{num}(U) + \text{num}(T) + \text{num}(H) \mod 7 = (18 + 21 + 20 + 8) \mod 7 = 67 \mod 7 = 4
  \]
  \( T[4] \) is empty, so “RUTH” is placed in \( T[4] \).
  \( T[0\cdots6] = \{\text{NIL}, \text{“COBB”}, \text{NIL}, \text{NIL}, \text{“RUTH”}, \text{NIL}, \text{NIL}\} \).

- Insert ROSE:
  \[
  \text{num}(R) + \text{num}(O) + \text{num}(S) + \text{num}(E) \mod 7 = (18 + 15 + 19 + 5) \mod 7 = 57 \mod 7 = 1
  \]
  So “ROSE” is placed as the head of the linked list in \( T[1] \).
  \( T = \{\text{NIL}, \text{“ROSE”} \rightarrow \text{“COBB”}, \text{NIL}, \text{NIL}, \text{“RUTH”}, \text{NIL}, \text{NIL}\} \).

- Search BUZ:
  \[
  \text{num}(B) + \text{num}(U) + \text{num}(Z) \mod 7 = (2 + 21 + 26) \mod 7 = 49 \mod 7 = 0
  \]
  \( T[0] \) is empty, it would not contain “BUZ”
“NIL” (representing “not found”) is returned. Hash table $T$ remains unchanged.

- **Insert DOC:**
  
  
  
  \[
  \text{num}(D) + \text{num}(O) + \text{num}(C) \pmod{7} \\
  = (4 + 15 + 3) \pmod{7} = 22 \pmod{7} = 1
  \]
  
  So “DOC” is placed as the head of the linked list in $T[1]$. 
  
  $T = \{\text{NIL}, \text{“DOC”} \rightarrow \text{“ROSE”} \rightarrow \text{“COBB”}, \text{NIL}, \text{NIL}, \text{“RUTH”}, \text{NIL}, \text{NIL}\}$.

- **Delete COBB:**
  
  As calculated before, the key for COBB is 1.
  
  So “COBB” is fetched in $T[1]$. After “DOC” and “ROSE” are examined, “COBB” is found and then deleted.
  
  $T = \{\text{NIL}, \text{“DOC”} \rightarrow \text{“ROSE”}, \text{NIL}, \text{NIL}, \text{“RUTH”}, \text{NIL}, \text{NIL}\}$.

2. Consider a Binary Search Tree $T$ with vertices $a, b, c, d, e, f, g, h$ and $ROOT[T] = a$ and with the following values ($N$ means NIL)

<table>
<thead>
<tr>
<th>vertex</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>N</td>
<td>e</td>
<td>e</td>
<td>a</td>
<td>d</td>
<td>g</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>left</td>
<td>h</td>
<td>N</td>
<td>N</td>
<td>e</td>
<td>c</td>
<td>N</td>
<td>f</td>
<td>N</td>
</tr>
<tr>
<td>right</td>
<td>d</td>
<td>N</td>
<td>g</td>
<td>N</td>
<td>b</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>key</td>
<td>80</td>
<td>170</td>
<td>140</td>
<td>200</td>
<td>150</td>
<td>143</td>
<td>148</td>
<td>70</td>
</tr>
</tbody>
</table>

Draw a nice picture of the tree. Illustrate INSERT[i] where key[i]=100.

**Solution:** Here is the picture, without the key values.

```
               a
              /  \
             d    e
            /\    /\ 
           c  b  g  f
          / \   /
         h   N  N
```

For INSERT[i]:

We start at root $a$ with $key[a] = 80$. As $80 < 100$ we replace $a$ by its right child $d$ with $key[d] = 200$. As $100 < 200$ we replace $d$ by its left child $e$ with $key[e] = 150$. As $100 < 150$ we replace $e$ by its left child $c$ with $key[c] = 140$. As $100 < 140$ we replace $c$ by its left child. But its left child is NIL so we make the new vertex $i$ its left child by setting $p[i] = c$ and $left[c] = i$. 
3. Continuing with the Binary Search Tree of the previous problem:

(a) Which is the successor of $c$. Illustrate how the program `SUCCESSOR` will find it.

Solution: The successor of $c$ is $f$. As $c$ has a right child $g$, `SUCCESSOR` will call `MIN[g]` which will go to the left as long as possible, ending (in one step) at $f$.

(b) Which is the minimal element? Illustrate how the program `MIN` will find it.

Solution: $h$. Start at root $a$. Go to left: $h$. Go to left: NIL. Return $h$.

(c) Illustrate the program `DELETE[e]`

Solution: There are two approaches (equally correct) to `DELETE[x]` when $x$ has two children. One can effectively replace $x$ by the maximum of its left tree or the minimum of its right tree.

Solution 1: $e$ has a left child $c$. Applying `MAX[c]` gives $g$. $g$ has a left child $f$. So we splice $f$ into $g$’s place by resetting $right[c] = f$ and $p[f] = c$ and we put $g$ in $e$’s place, setting $left[d] = g$, $left[g] = c$, $right[g] = b$, and $p[g] = d$.

Solution 2: $e$ has right child $b$. Applying `MIN[b]` gives $b$ itself. We splice $b$ into $e$’s place by resetting $p[c] = b$ and $left[b] = c$ and $p[b] = d$ and $left[d] = e$

4. Set $N = 2^K$. We’ll represent integers $0 \leq x < N$ by $A[0 \cdots (K - 1)]$ with $x = \sum_{i=0}^{k-1} A[i]2^i$. (This is the standard binary representation of $x$, read right to left.) Consider the following algorithms:

Procedure `JACK[A]`

$I \leftarrow 0$

$A[0] + +$

WHILE (A[I] = 2 AND $I < K - 1$)

$A[I] \leftarrow 0$

$I + +$

$A[I] + +$

END WHILE

and:

ANYA[A]

FOR $J = 1$ TO $N - 1$

DO `JACK[A]`

END FOR
(a) If the input to $JACK[A]$ is the binary representation of $x$ with $0 \leq x \leq N - 2$ describe what the output will be.

**Solution:** $JACK$ increments by one, the final value of $A$ will be the binary representation of $x+1$. For example, if $A$ (reading right to left) is 1100111 then it becomes 1100112, 1100120, 110020, 1100100 and then stops.

(b) For “time” we will mean here the number of times the line: “WHILE ($A[I] = 2$ AND $I < K - 1$)” is reached. We want here the “time” as a function of $N$. What is the worst-case time for $JACK$? What is the best-case time for $JACK$?

**Solution:** The worst case is $K = \Theta(\lg N)$, starting, e.g., at 01111111111. The best case is $1 = \Theta(1)$, when you start with $A[0] = 0$, so when $x$ is an odd number.

(c) Assume the array $A$ is initially all zeroes. Describe what $ANYA$ is doing.

**Solution:** $ANYA$ is going through all the numbers (in their binary representation) from 0 to $N - 1$.

(d) (*) Again assume the array $A$ is initially all zeroes and “time” as above. What is the time for $ANYA$ in $\Theta$-land?

**Solution:** The FOR loops goes $N$ times and each time is $O(\lg N)$ which would give $O(N \lg N)$. But actually it is linear! Note that $N/2$ values (the odd ones) have “time” 1, $N/4$ (ending in 01) have time 2, in general $N2^{-i}$ have time $i$. So the total time is

$$\sum i = 1^k N2^{-i} \cdot i = N \sum i2^{-i}$$

As we discussed with BUILDMAXHEAP, $\sum_{i=1}^{\infty} i2^{-i} = 2$ (a constant) so the total time is $O(N)$. 