Fundamental Algorithms, Assignment 5
Solutions

1. Some exercises in which \( n \) is NOT the data size but we want the answer in terms of \( n \). (Answers in \( \Theta \)-land.)

   (a) How long does \textsc{merge-sort} on \( n^2 \) items take?
   \textbf{Solution:} On \( n \) items it would be our mantra \( \Theta(n \log n) \) so on \( n^2 \) it would be \( \Theta(n^2 \log(n^2)) \). But \( \log(n^2) = 2 \log(n) \) and the 2 gets absorbed in the \( \Theta \) so the answer is \( \Theta(n^2 \log n) \).

   (b) Suppose that when \( n = 2^m \), \textsc{anna} takes time \( \Theta(m^2 2^m) \). How long does it take as a function of \( n \)?
   \textbf{Solution:} As \( m = \log n \) this is \( \Theta(n \log^2 n) \). (Note that \( \log^2 n \) (the square of the \( \log \)) and \( \log(n^2) \) (the \( \log \) of the square) are very different!)

   (c) Suppose that when \( n = 2^m \), \textsc{bob} takes time \( \Theta(5^m) \). How long does it take as a function of \( n \)?
   \textbf{Solution:} \( 5^m = (2^c)^m = (2^m)^c = n^c \) where \( c = \log 5 \).

   (d) How long does \textsc{counting-sort} take to sort \( n^2 \) items with each item in the range 0 to \( n^3 - 1 \).
   \textbf{Solution:} \( \Theta(n^3) \) as the main time is going through the mostly empty slots.

   (e) How long does \textsc{radix-sort} take to sort \( n^2 \) items with each item in the range 0 to \( n^3 - 1 \) and base \( n \) is used.
   \textbf{Solution:} The numbers have three digits in base \( n \) (for example 0 to 999 in decimal or 0 to 7 in binary) so there are three applications of \textsc{counting-sort}. Three is a constant so lets just look at \textsc{counting-sort}. Here the time is \( \Theta(n^2) \) as the main time is to put the \( n^2 \) items into the \( n \) slots. So the total time is \( \Theta(n^2) \).

2. 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 $T[0 \cdots 6]$ be the hash table which is $\{\text{NIL, NIL, NIL, NIL, NIL, NIL, NIL}\}$ initially.  
Let $\text{num}(\cdot) : \{A, B, \cdots, Z\} \to \{1 \cdots, 26\}$ 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 $T[1]$.  
  $T[0 \cdots 6] = \{\text{NIL, “COBB”, NIL, NIL, NIL, NIL, NIL}\}$.

- Insert RUTH:  
  \[
  \text{num}(R) + \text{num}(U) + \text{num}(T) + \text{num}(H) \mod 7 = (18 + 21 + 20 + 8) \mod 7 = 67 \mod 7 = 4
  \]
  $T[0 \cdots 6] = \{\text{NIL, “COBB”, NIL, NIL, “RUTH”, NIL, NIL}\}$.

- 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 $T[1]$.  

- 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:  
  \[
  \text{num}(D) + \text{num}(O) + \text{num}(C) \mod 7 = (4 + 15 + 3) \mod 7 = 22 \mod 7 = 1
  \]
  So “DOC” is placed as the head of the linked list in $T[1]$.  

- 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.

\[ T = \{ \text{NIL}, \text{“DOC”} \rightarrow \text{“ROSE”}, \text{NIL}, \text{NIL}, \text{“RUTH”}, \text{NIL}, \text{NIL} \} \]

3. Wally Wonkle, NYU drone, is asked to create a hash table (by chaining) of the 20000 NYU student records. Each record takes about 10K bytes of memory. He decides to use a hash table of size one million and The Boss goes ballistic, accusing him of negligent use of precious computer space. Argue coherently that this method is not using very much more space than a hash table of size 20000. What is the advantage in using size one million instead of size 20000?

**Solution:** The 20000 records are themselves going to take up 200 megabytes of memory. The only extra space with the hash table is the places that are empty. True, any empty space in an array still takes up some space but if it, say, took up five bytes that would only add up to 5 megabytes of memory which doesn’t change the total memory allotment that much. The big advantage is that you would almost never have collisions and so SEARCH and DELETE would be faster as your chain would generally have only one element (if the item being searched for did exist) or zero elements (if it did not).

4. Consider a Binary Search Tree \( T \) with vertices \( a, b, c, d, e, f, g, h \) and \( \text{ROOT}[T] = a \) 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 \( \text{INSERT}[i] \) where \( \text{key}[i] = 100 \).

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

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

For \( \text{INSERT}[i] \):

We start at root \( a \) with \( \text{key}[a] = 80 \). 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 $left[c] = i$. 