# 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 `MERGE-SORT` on $n^2$ items take?
   `Solution:`On $n$ items it would be our mantra $\Theta(n \lg n)$ so on $n^2$ it would be $\Theta(n^2 \lg(n^2))$. But $\lg(n^2) = 2\lg(n)$ and the 2 gets absorbed in the $\Theta$ so the answer is $\Theta(n^2 \lg n)$.

   (b) Suppose that when $n = 2^m$, `ANNA` takes time $\Theta(m^2 2^m)$. How long does it take as a function of $n$.
   `Solution:`As $m = \lg n$ this is $\Theta(n \lg^2 n)$. (Note that $\lg^2 n$ (the square of the lg) and $\lg(n^2)$ (the lg of the square) are *very* different!)

   (c) Suppose that when $n = 2^m$, `BOB` takes time $\Theta(5^m)$. How long does it take as a function of $n$.
   `Solution:`$5^m = (2^c)^m = (2^m)^c = n^c$ where $c = \lg 5$.

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

   (e) How long does `RADIX-SORT` take to sort $n^2$ items with each item in the range 0 to $n^3 - 1$ and base $n$ is used.
   `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 `COUNTING-SORT`. Three is a constant so lets just look at `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(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 {NIL, NIL, NIL, NIL, NIL, NIL, NIL} initially.
Let num$(\cdot) : \{A, B, \cdots, Z\} \rightarrow [1 \cdots, 26]$ be the specified bijection which maps a letter to its numerical value. We have

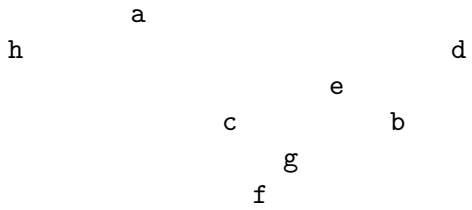
- Insert COBB:
  num(C) + num(O) + num(B) + num(B) mod 7
  = (3 + 15 + 2 + 2) mod 7 = 22 mod 7 = 1
  $T[1]$ is empty, so "COBB" is placed in $T[1]$.
  $T[0 \cdots 6]$ = {NIL, "COBB", NIL, NIL, NIL, NIL, NIL}.
- Insert RUTH:
  num(R) + num(U) + num(T) + num(H) mod 7
  = (18 + 21 + 20 + 8) mod 7 = 67 mod 7 = 4
  $T[4]$ is empty, so "RUTH" is placed in $T[4]$.
  $T[0 \cdots 6]$ = {NIL, "COBB", NIL, NIL, "RUTH", NIL, NIL}.
- Insert ROSE:
  num(R) + num(O) + num(S) + 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 $T[1]$.
  $T$ = {NIL, "ROSE"→"COBB", NIL, NIL, "RUTH", NIL, NIL}.
- Search BUZ:
  num(B) + num(U) + 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:
  num(D) + num(O) + 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 $T[1]$.
  $T$ = {NIL, "DOC"→"ROSE"→"COBB", NIL, NIL, "RUTH", NIL, 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.
$T = \{$NIL, "DOC"$\rightarrow$"ROSE", NIL, NIL, "RUTH", NIL, NIL$\}$.

3. 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)

| vertex | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| parent | N | e | e | a | d | g | c | a |
| left | h | N | N | e | c | N | f | N |
| right | d | N | g | N | b | N | N | N |
| key | 80 | 170 | 140 | 200 | 150 | 143 | 148 | 70 |

Draw a nice picture of the tree. Illustrate `INSERT[i]` where `key[i]`=100.
`Solution:`Here is the picture, without the key values.




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 $left[c] = i$.