CSCD320 Homework1, Winter 2012, Eastern Washington University. Cheney, Washington.

Name: Eric Fode **EWU ID:005301214**

Solution for Problem 1

Proof. Prove that $T(n) = 2T(n/3) + n^2 = \theta(n^2)$ where $n_0 = 0$ using induction
Claim: if $T(n) = 2T(n/3) + n^2$ then $T(n) \leq c(n^2)$ for some constant c
Assume:

$$T(1) = 1$$

$$c = 4$$

Base Case of 1:

$$\begin{aligned} T(3) &= 2T(3/3) + 1^2 \\ &= 2 * 1 + 1 \\ &= 3 \\ 4 * 1^2 &> 3 \\ 4 &> 3 \end{aligned}$$

Hypothesis: assume $T(m) \leq c * m^2$ is true for all $m = 3, ..., n$
Inductive step: when $m = n + 1$

$$\begin{aligned} 4(n+1)^2 &\geq 2T(n/3) + n^2 \\ &\geq 2T(\frac{n+1}{3}) + (n+1)^2 \\ &\geq 8(\frac{n+1}{3})^2 + (n+1)^2 \\ &\geq \frac{8}{9}(n+1)^2 + (n+1)^2 \\ 4(n+1)^2 &\geq (\frac{17}{9})(n+1)^2 \end{aligned}$$

$\square$

CSCD320 Homework1, Winter 2012, Eastern Washington University. Cheney, Washington.

Name: Eric Fode **EWU ID:005301214**

Solution for Problem 2

1. $T(n) = 4T(n/2) + 3n^2 - 9n$
   $a = 4, b = 2, f(n) = 3n^2 - 9n$ case 2
   $$3n^2 - 9n = \theta(n^2)$$
   so
   $$T(n) = \theta(n^2)$$

2. $T(n) = 4T(n2) + 2n^3 - 100n^2$
   $a = 4, b = 2, f(n) = 2n^3 - 100n^2$ case 3
   $$2n^3 - 100n^2 = \Omega(n^{2+1})$$
   and
   $$4 * ((n)^3 - 50n^2) \leq 1/2(2n^3 - 100n^2)$$
   so
   $$T(n) = \theta(n^3)$$

3. $T(n) = 4T(n/2) + n + 5logn$
   $a = 4, b = 2, f(n) = n + 5logn$ case 1
   $$n + 5\log n = O(n^{2-1})$$
   so
   $$T(n) = \theta(n^2)$$

4. $T(n) = 8T(n/2) + n^2 + nlogn$
   $a = 8, b = 2, f(n) = n^2 + nlogn$ case 1
   $$n^2 + n\log n = O(n^{3-1})$$
   so
   $$T(n) = \theta(n^3)$$

5. $T(n) = 8T(n/2) + 4n^3 + 5n^2$
   $a = 8, b = 2, f(n) = 4n^3 + 5n^2$ case 2
   $$4n^3 + 5n^2 = \theta(n^3)$$
   so
   $$T(n) = \theta(n^3 \log n)$$

6. $T(n) = 8T(n/2) + 2^{-10}n^4 + 6n^3$
   $a = 8, b = 2, f(n) = 2^{-10}n^4 + 6n^3$ case 3
   $$2^{-10}n^4 + 6n^3 = \Omega(n^{3+1})$$
   and
   $$\frac{2^{-10}n^4}{2} + 3n^3 \leq c(2^{-10}n^4 + 6n^3)$$
   when n is large so
   $$T(n) = \theta(n^4)$$

CSCD320 Homework1, Winter 2012, Eastern Washington University. Cheney, Washington.

Name: Eric Fode **EWU ID:005301214**

Solution for Problem 3 Two Instances in which the master therom does not work because they both fall inbetween the cases described by the master therom:

$$T(n) = 2^n T(\frac{n}{2}) + n^n$$
$$T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n}$$

Solution for Problem 4

1. Idea: navigate to bottom of tree and step back one level compair subtree left max, subtree right max and this tree total return max of these three repeate until top node is reached

2. Pseudocode:

```
maxtree(tree)
  Input: A tree
  Result: The root node of the largest subtree
  begin
    if tree == null then
      return 0;
    if tree.leaf == true then
      tree.head.value = tree.head.weight;
      return tree.head;
    leftReturn ⟵ maxtree(tree.left);
    rightReturn ⟵ maxtree(tree.right);
    tree.head.value ⟵ tree.left.value + tree.right.value + tree.head.weight;
    if leftReturn.value > rightReturn.value and leftReturn.value > tree.head.value then
      return leftReturn;
    if rightReturn.value > leftReturn.value and rightReturn.value > tree.head.value then
      return rightReturn;
    if tree.head.value > leftReturn.value and tree.head.value > rightReturn.value then
      return tree.head;
    if leftReturn.value == rightReturn.value and
    rightReturn.value > tree.head.value then
      return rightReturn;
```

3. Analyze: This algorithm can be represented by the following recurance

$$T(n) = 2T(n/2) + c$$

which using the master therom can be converted to

$$T(n) = O(n)$$

Solution for Problem 5
BST
Pros:

1. Extermly space effcent, they only take up $O(n * c)$ space.
2. Fast lookup, access time for a givin index is $O(\log_2 n)$ when tree is balenced.
3. Fast find, find time is $O(\log_2 n)$ when tree is balenced
4. Items are sorted as they are inserted.
5. If converted to anouther type of data structure it is trivial to retain order of items.

Cons:

1. Insert can be and cause tree to have to restructure. If balence is maintained.
2. Remove can be expensive and cause tree to restructure. If balence is maintained.
3. Insert and remove are not constant time operations. If balence is maintained.

Best Use Cases:

1. When data will not be added or deleted frequently.
2. When searching is required.
3. When only one ordering will be used.

CSCD320 Homework1, Winter 2012, Eastern Washington University. Cheney, Washington.

Name: Eric Fode **EWU ID:005301214**

Hash Table
Pros:

1. Extermly fast lookup $\Omega(1)$.
2. Extermly fast insert (assuming that hash table is large enugh to hold new item) $\Omega(c)$.
3. Extermly fast delete $\Omega(1)$.
4. Extermly fast modify $\Omega(c)$.

Cons:

1. Slow search $O(n)$.
2. No ordering.
3. Underlying List can be very large to accomidate room for all possible hashs.

Best Use Cases:

1. When data is constantly being added and removed.
2. When data is ordered on hash.
3. When space is unlimited and speed is critical