**Problem 1:**

struct node {

string data;

node \*next;

}

current\_node = head\_node

prev\_node = next\_node = NULL

while (current\_node != NULL) {

next\_node = current\_node->next;

current\_node->next = prev\_node;

prev\_node = current\_node;

current\_node = next\_node;

}

with n loop, the time complexity is O(n)

f(n) = n > n/2 with n > 1 (C=1/2)

so it’s Ω(n)

**Problem 2:**

function F(x) {

if x→right != NULL {

print TREE-MINIMUM(x→right)→value;

F(TREE-MINIMUM(x→right))

}

p = x→parent;

while (p != NULL && x = p->right) {

x = p;

p = p->parent;

}

if ( p != NULL) {

print p→value;

F(p);

}

}

The loop to find tree-minimum and tree-successor is O(h), the recursive F(n) takes O(n) loop.

Time complexity: O(hn) with h is the height of tree, n is node number.

**Problem 3:**

Let DP[n] be the number of ways to write n as the sum of 1, 2, 4

Formula: DP(n) = DP(n-1) + DP(n-2) + DP(n-4) with n>4

The result at n index

With n <= 4, DP[1] = 1, DP[2] = 2, DP[3] = 3, DP[4] = 6

Algorithm:

int DP[n + 1];

DP[1] = 1; DP[2] = 2; DP[3] = 3; DP[4] = 6;

for (int i = 5; i <= n; i++)

DP[i] = DP[i - 1] + DP[i - 2] + DP[i – 4];

print DP[n];

Only 1 loop to n, easy to see the time complexity is O(n)

**Problem 4:**

Merge-sort divides the problem into sub-problems then conquer them, the sub-problems don’t overlap each others, the function called just once. Memoization is not useful for non-overlap functions, so it cannot improve merge-sort time complexity.

**Problem 5:**

Let DP[n] be the result of factorial at n.

Formula: DP[n] = DP[n-1] \* n

Algorithm:

DP[0] = 1;

for (int i = 1; i <= n; ++i)

DP[i] = i \* DP[i - 1];

print(DP[n]);

Only 1 loop to n, easy to see the time complexity is O(n)