Solutions for Program Assignment 3

CSCI2100A

March 30, 2015

1 Exercise 3.24

1.1 Analysis

This is a very basic problem of binary tree. The simplest way is to use a recursive function. In this function, the steps are as follows: (1) determine the root of the current subtree in preorder sequence (1st position); (2) locate its position in inorder sequence; (3) recursively deal with the left subtree and the right subtree; (4) print out the root in postorder sequence.

1.2 Sample code

```
#include <stdio.h>
#include <string.h>

char inorder[100], preorder[100];
char line[200];

void trace(int ihead, int itail, int phead, int ptail)
{
        char split = preorder[phead];
        int i;
    int len;

        if (ihead > itail) return;

        for (i=ihead; inorder[i] != split; ++i)
                ;

        len = i - ihead;
        trace(ihead, i-1, phead+1, phead+len);

        len = itail - i - 1;
        trace(i+1, itail, ptail - len, ptail);

        printf("%c", split);
}

int main() {
    int n;

        for (;;) {
```

```
                if (gets(line) == NULL) break;
                sscanf(line, "%s %s", preorder, inorder);
                n = strlen(inorder);
                trace(0, n-1, 0, n-1);
                printf("\n");
        }

        return 0;
}
```

2 Exercise 3.35

2.1 Analysis

Due to the data size, you can actually use a brutal-force algorithm to compute the solution, i.e., enumerate starting point and ending point and search for the path between them. The time complexity should be $O(n^3)$. However, a dynamic programming based method can be use to reduce the complexity. For this method, you need to keep some variable in the node and do a top-down searching but a bottom-up updating. The code below will show the variables and updating rule in more details.

2.2 Sample Code

```
/*Modified from the version coded by Ianmaoac in GitHub*/
#include <stdio.h>
#include <string.h>

#define MAXN (100010)
#define MOD (1000000007)
struct node
{
    int to,next;
    long long w;
} edges[MAXN*2];

int box[MAXN],size,n;
long long ans,sum[MAXN];

void add(int from,int to,long long w)
{
    edges[size].to=to;
    edges[size].w=w;
    edges[size].next=box[from];
    box[from]=size++;
}
void init()
{
    int a,b,c,i;
    scanf("%d",&n);
    for (i=0; i<MAXN; i++)
        box[i] = -1;
    for (i=0; i<MAXN; i++)
```

```
        sum[i] = 0;
    size=ans=0;
    for(i=0;i<n-1;++i)
    {
        scanf("%d%d%d",&a,&b,&c);
        add(a,b,c);
        add(b,a,c);
    }
}
void dfs(int x,int father)
{
    int i, ne;
    for(i=box[x];i!=-1;i=edges[i].next)
    {
        ne=edges[i].to;
        if(ne==father)continue;
        dfs(ne,x);
        ans=(ans+sum[x]*edges[i].w%MOD*(sum[ne]+1))%MOD;
        sum[x]=(sum[x]+edges[i].w*(sum[ne]+1))%MOD;
    }
    ans=(ans+sum[x])%MOD;
}
int main()
{
    init();
    dfs(1,-1);
    printf("%lld\n", ans);
    return 0;
}
```