DSA PROJECT

Digital Address Book

Complexity

Shah Raza

Note

Before starting to calculate the complexity of this project, it is to be noted that although the complexity of a few built-in functions such as **strcmp** and **strcpy** used in this Project is **O(n)** but in the following slides I will assume their complexities to be **O(1)** just to focus on the complexities of Linked List and Hash Table Functions.

AddressBook class functions

1. Add
2. Delete
3. Search
4. Display
5. TextFile
6. IsEmpty

1. Add

```
void Add(Contact *c)                                    for n>0
  {
     Contact *temp;
     temp= new Contact;                                     1
     strcpy(temp->name,c->name);                            1
     strcpy(temp->email,c->email);                          1
     strcpy(temp->address,c->address);                      1
     temp->ncode=c->ncode;                                  1
     temp->number= c->number;                               1
     if(head==NULL)                                         1
     {
        head=temp;
        head->next=NULL;
     }
```

As n>0, this portion will not be executed

```
else
   {
      Contact *i;
      i=head;                                  1
      while(i->next!=NULL)                     n
         i=i->next;                            n-1
      i->next=temp;                            1
      i=i->next;                               1
      i->next=NULL;                            1
   }
}
```

T(n) = 2n + 10
Time Complexity= O(n)

2. Delete

```
void Delete(char *name)                                  for n>1
  {
    if(head==NULL)                                            1
    {
        cout<<"Contact does not exist.\n";
        return ;
    }
    Contact *temp,*i;
    i=head;                                                   1
    int Count=0;                                              1
    if(!strcmp(i->name,name))                                 1
    {
        head=i->next;
        delete i;
    }
```

This will not execute in worst case

```
 else
 {
 while(i!=NULL)                                          n+1
 {
    if(!strcmp(i->next->name,name))                      n
    {
       temp=i->next;                                     n
       i->next=i->next->next;                            n
       delete temp;                                      n
       i=i->next;                                        n
       Count++;                                          n
       Continue;                                         n
    }
    i=i->next;
 }
 if(!Count)                                              1
     cout<<"Contact does not exist.\n";
 else
     cout<<"Contact deleted.\n";                         1
 }
}
```

T(n) = 8n + 7
Time Complexity = O(n)

3. Search

```
void Search(char *name)                                  for n>0
  {
    if(head==NULL)                                          1
    {
      cout<<"Contact does not exist.\n";
      return ;
    }
    Contact *i;
    i=head;                                                 1
    int Count=0;                                            1
```

```
while(i!=NULL)                                                    n+1
   {
      if(!strcmp(i->name,name))                                   n
      {
         cout<<"\t\t\tName: "<<i->name<<endl;                     n
         cout<<"\t\t\tNumber: 0"<<i->ncode<<"-"<<i->number<<endl;  n
         cout<<"\t\t\tEmail: "<<i->email<<endl;                  n
         cout<<"\t\t\tAddress: "<<i->address<<endl;              n
         Count++;                                                 n
      }
      i=i->next;                                                  n
   }
   if(!Count)                                                     1
       cout<<"Contact does not exist.\n";
 }
```

T(n) = 8n + 5
Time Complexity = O(n)

4. Display

```
void Display()                                                          for n>0
  {
    if(head==NULL)                                                        1
    {
      cout<<"No Contacts Available.\n";
      return ;
    }
    Contact *temp;
    temp=head;                                                            1
    while(temp!=NULL)                                                     n+1
    {
      cout<<"\t\t\tName: "<<temp->name<<endl;                             n
      cout<<"\t\t\tNumber: 0"<<temp->ncode<<"-"<<temp->number<<endl;      n
      cout<<"\t\t\tEmail: "<<temp->email<<endl;                           n
      cout<<"\t\t\tAddress: "<<temp->address<<endl;                       n
      temp=temp->next;                                                    n
    }
  }
```

T(n) = 6n +3
Time Complexity = O(n)

5. TextFile

```
void TextFile(ofstream &myfile)                                          for n>0
  {
     if(head==NULL)                                                        1
     {
        myfile<<"No Contacts Available.\n";
        return ;
     }
     Contact *temp;
     temp=head;                                                            1
     while(temp!=NULL)                                                     n+1
     {
        myfile<<"\t\t\tName: "<<temp->name<<endl;                          n
        myfile<<"\t\t\tNumber: 0"<<temp->ncode<<"-"<<temp->number<<endl;   n
        myfile<<"\t\t\tEmail: "<<temp->email<<endl;                       n
        myfile<<"\t\t\tAddress: "<<temp->address<<endl;                    n
        temp=temp->next;                                                   n
     }
  }
```

T(n) = 6n + 3
Time Complexity = O(n)

6. IsEmpty

```
bool IsEmpty()
{
    if(head==NULL)                    1
        return 1;
    return 0;                         1
}
```

T(n) = 2
Time Complexity = O(1)

Hash Table Functions

1. HashFunction
2. InsertContact
3. DeleteContact
4. SearchContact
5. DisplayContacts
6. GenerateTextFile

1. HashFunction

```
int HashFunction(char *name)
{
    int len= strlen(name);                          1
    int Prime=3,sum=0;                              2
    for(int i=0;i<len;i++)                          1+n+1+n
        sum=(sum*Prime)+name[i];                    n
    return sum%Max;                                 1
}
```

T(n) = 3n + 6
Time Complexity = O(n)

2. InsertContact

```
void InsertContact()
{
    Contact c1;
    cout<<"\t\t\tEnter Name: ";                                  1
    scanf(" %[^\n]s",c1.name);                                   1
    cout<<"\t\t\tEnter Network Code: (0)";                        1
    cin>>c1.ncode;                                               1
    cout<<"\t\t\tEnter Number: ";                                1
    cin>>c1.number;                                              1
    cout<<"\t\t\tEnter Email: ";                                 1
    scanf(" %[^\n]s",c1.email);                                  1
    cout<<"\t\t\tEnter Address: ";                               1
    scanf(" %[^\n]s",c1.address);                                1
    int index=HashFunction(c1.name);                             n
    table[index].Add(&c1);                                       n
}
```

T(n) = 2n + 10
Time Complexity = O(n)

3. DeleteContact

```
void DeleteContact()
{
    Contact c;
    cout<<"\t\t\tEnter the name of the Contact you want to Delete: ";        1
    scanf(" %[^\n]s",c.name);                                                 1
    int index=HashFunction(c.name);                                           n
    table[index].Delete(c.name);                                              n
}
```

T(n) = 2n + 2
Time Complexity = O(n)

4. SearchContact

```
void SearchContact()
{
    Contact c;
    cout<<"\t\t\tEnter the name of the Contact you want to Search: ";        1
    scanf(" %[^\n]s",c.name);                                                 1
    int index=HashFunction(c.name);                                           n
    table[index].Search(c.name);                                              n
}
```

T(n) = 2n + 2
Time Complexity = O(n)

5. DisplayContacts

```
void DisplayContacts()                                  for Max=10
{
    for(int i=0;i<Max;i++)                              1+11+10
    {
        if(table[i].IsEmpty())                          10
            continue;
        table[i].Display();                             10n
    }
}
```

T(n) = 10n + 32
Time Complexity = O(n)

6. GenerateTextFile

```
void GenerateTextFile()                                  for Max=10
  {
    ofstream myfile("Contacts.txt");                          1
    for(int i=0;i<Max;i++)                               1+11+10
    {
      if(table[i].IsEmpty())                                  10
        continue;
      table[i].TextFile(myfile);                              10n
    }
    myfile.close();                                           1
    cout<<"\t\t\tFile Generated.";                            1
  }
```

T(n) = 10n + 35
Time Complexity = O(n)