SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

K. J. Somaiya College of Engineering, Mumbai – 400 077
(A Constituent College of Somaiya Vidyavihar University)

Somaiya TRUST

Data Structures IA-2

Cash Flow Minimiser

Project By:
1. Riya Hemani- 16010121065
2. Devesh Jain- 16010121070
3. Ritvik Jindal- 16010121073

Problem Definition

Using graph algorithm to reduce the amount involved in cash flow and number of transactions to the minimum possible extent. This also helps us to settle debts quickly.

Appropriateness of Data Structure Chosen- GRAPHS

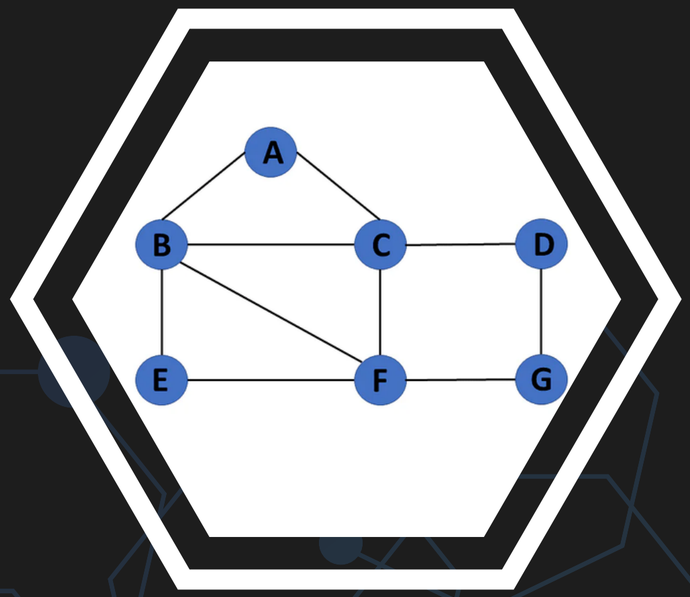
Appropriateness of Data Structure Chosen

Graphs in data structures are non-linear data structures made up of a finite number of nodes or vertices and the edges that connect them. They are used to address real-world problems in which it represents the problem area as a network like telephone networks. For example, it can represent a single user as nodes or vertices in a telephone network, while the link between them via telephone represents edges.

Appropriateness of Data Structure Chosen

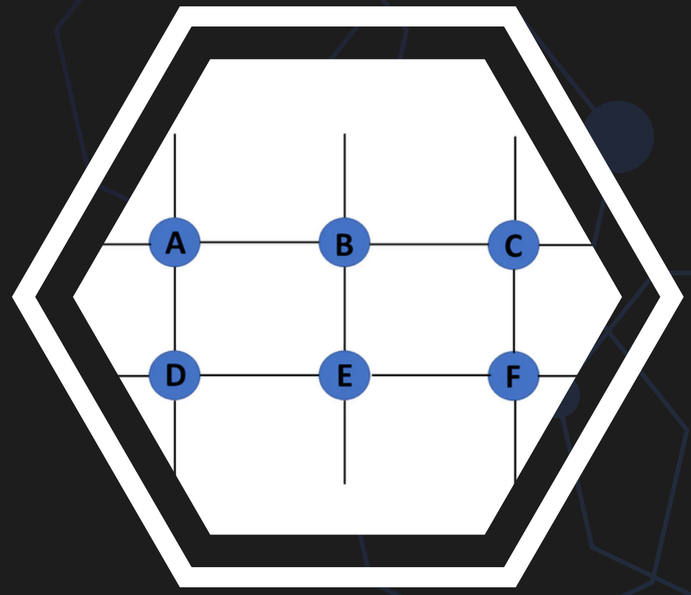
In the case of our problem statement, it represents the network of a group settling debts between each other. Since each member of the group owes a certain amount to other members, overall it leads to hassle and a lot of transactions, but using graphs we can reduce and eliminate a maximum of those transactions down to a few transactions that minimizes the amount needed to be transferred as well the people involved.

Types of Graphs

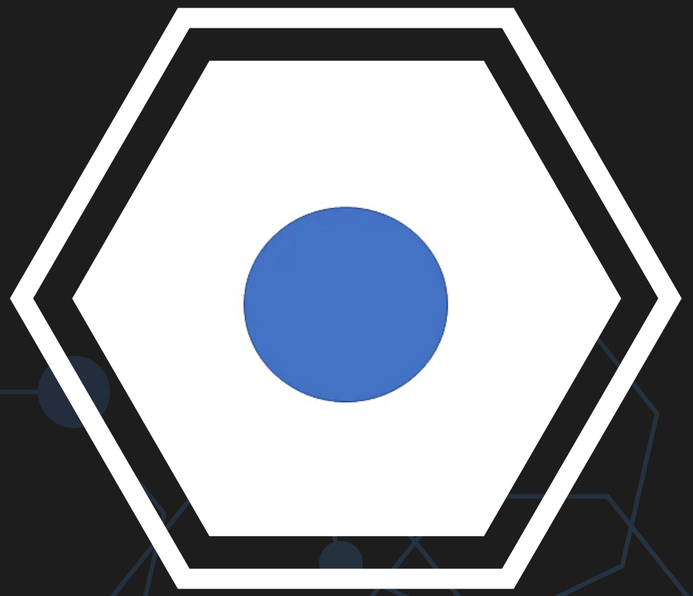


- Finite: The graph G=(V, E) is called a finite graph if the number of vertices and edges in the graph is limited in number.

- Infinite: The graph G=(V, E) is called a finite graph if the number of vertices and edges in the graph is interminable.

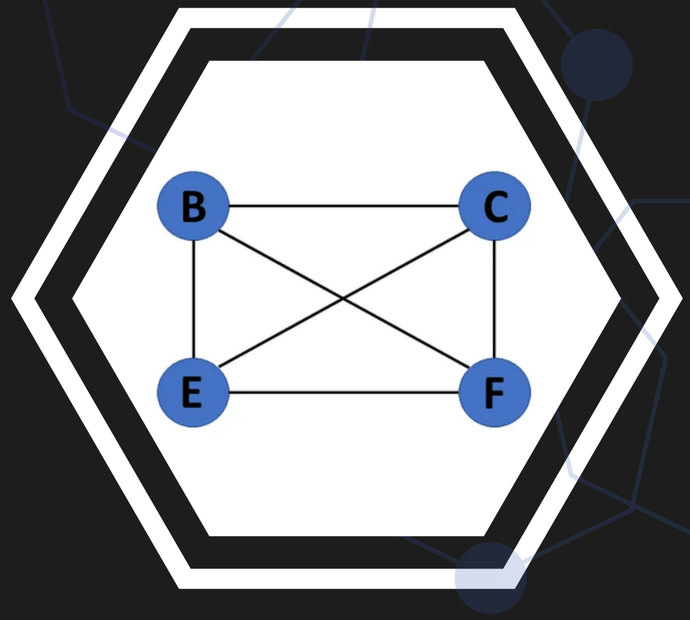


Types of Graphs



- Trivial: A graph G= (V, E) is trivial if it contains only a single vertex and no edges.

- Simple: A simple graph is a graph that does not contain more than one edge between the pair of vertices.



Implementation Details

```c
/* Importing the libraries that are needed for the program to run*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
/* Defining the variables that will be used in the program and the maximum number of people*/
#define N 100
#define min(i, j) (((i) < (j)) ? (i) : (j))
int graph[N][N];
char people[N][N];
int n;

/*returns index of minimum value in a[]*/
int get_min(int a[])
{
    int min_ind = 0;
    for(int i=1; i<n; i++)
    {
        if (a[i] < a[min_ind])
        {
            min_ind = i;
        }
    }
    return min_ind;
}
```

```
25
26  /*returns index of maximum value in a[]*/
27  int get_max(int a[])
28  {
29      int max_ind = 0;
30      for(int i=1; i<n; i++)
31      {
32          if (a[i] > a[max_ind])
33          {
34              max_ind = i;
35          }
36      }
37      return max_ind;
38  }
39
```

```c
int main()
{
    printf("      ***** Welcome to our Data Structures Project on Cash Flow Minimiser. *****\n\n");
    printf("                                                                    -made by devesh,riya,ritvik ");
    printf("\nEnter no. of people: ");
    scanf("%d",&n);
    printf("Enter names of people\n");
    //accept names of people who owe each other
    for(int i=0; i<n; i++)
    {
        printf("Enter name %d : ",i+1);
        scanf("%s",&people[i]);
    }

    for(int i=0; i<n; i++)
    {
        for(int j=0; j<n; j++)
        {
            if(i == j)
            {
                graph[i][j]=0;
                continue;
            }
            printf("How much %s has to pay %s?  Rs.",people[i],people[j]);
            scanf("%d",&graph[i][j]);
        }
    }
    min_cashflow(graph);
}
```

```
/* amt[p] indicates the net amount to be credited/debited to from person 'p'
   If amt[p] is positive, then i'th person will get amt[i]
   If amt[p] is negative, then i'th person will give  -amt[i]*/
void min_cashflow_rec(int amt[])
{
    int mcr = get_max(amt), mdb = get_min(amt);
    if(amt[mcr] == 0 && amt[mdb] == 0)
    {
        return;
    }
    int minval = min(-amt[mdb], amt[mcr]);
    amt[mcr] -= minval;
    amt[mdb] += minval;
    printf("\n%s will pay ₹ %d to %s",people[mdb],minval,people[mcr]);
    min_cashflow_rec(amt);
}

```

```
/*Given a set of persons as graph[] where graph[i][j] indicatesthe amount that person i needs
  to pay person j, this function finds and prints the minimum cash flow to settle all debts.*/
void min_cashflow(int graph[][N])
{
    int amt[n];
    for(int i=0; i<n; i++)
    {
        amt[i] = 0;
    }
    for(int p=0; p<n; p++)
    {
      for(int i=0; i<n; i++)
      {
        amt[p] += (graph[i][p] -  graph[p][i]);
      }
    }
    min_cashflow_rec(amt);
}

```

Test Cases and Result Analysis

Test Case 1 :

```
      ** Welcome to our Data Structures Project on Cash Flow Minimiser. **

                                                                  -made by devesh,riya,ritvik
Enter no. of people: 3
Enter names of people
Enter name 1 : Ritvik
Enter name 2 : Devesh
Enter name 3 : Riya
How much Ritvik has to pay Devesh?   Rs.4500
How much Ritvik has to pay Riya?   Rs.5800
How much Devesh has to pay Ritvik?   Rs.3600
How much Devesh has to pay Riya?   Rs.2100
How much Riya has to pay Ritvik?   Rs.7900
How much Riya has to pay Devesh?   Rs.8200

Riya will pay Rs.7000 to Devesh
Riya will pay Rs.1200 to Ritvik
Process returned 0 (0x0)   execution time : 62.697 s
Press any key to continue.
```

Result Analysis

For Test Case 1:

- The first line contains the input for number of people i.e 3 involved in the transaction.
- Then it takes the input for the 3 names (Ritvik,Riya,Devesh) and how much each person owes to the other.
- Lastly, after calculating the amount,it shows the min. payment, i.e., the amt. Riya has to pay to Ritvik & Devesh

Test Case 2 :

```
      ** Welcome to our Data Structures Project on Cash Flow Minimiser. **

                                                                    -made by devesh,riya,ritvik
Enter no. of people: 5
Enter names of people
Enter name 1 : Isha
Enter name 2 : Devesh
Enter name 3 : Anusha
Enter name 4 : Riya
Enter name 5 : Ritvik
How much Isha has to pay Devesh?   Rs.67000
How much Isha has to pay Anusha?   Rs.45000
How much Isha has to pay Riya?   Rs.31000
How much Isha has to pay Ritvik?   Rs.78000
How much Devesh has to pay Isha?   Rs.23000
How much Devesh has to pay Anusha?   Rs.79000
How much Devesh has to pay Riya?   Rs.84000
How much Devesh has to pay Ritvik?   Rs.35000
How much Anusha has to pay Isha?   Rs.94000
How much Anusha has to pay Devesh?   Rs.68000
How much Anusha has to pay Riya?   Rs.32000
How much Anusha has to pay Ritvik?   Rs.93000
How much Riya has to pay Isha?   Rs.44000
How much Riya has to pay Devesh?   Rs.33000
How much Riya has to pay Anusha?   Rs.26000
How much Riya has to pay Ritvik?   Rs.18000
How much Ritvik has to pay Isha?   Rs.57000
How much Ritvik has to pay Devesh?   Rs.61000
How much Ritvik has to pay Anusha?   Rs.58000
How much Ritvik has to pay Riya?   Rs.42000

Anusha will pay Rs.68000 to Riya
Anusha will pay Rs.8000 to Devesh
Isha will pay Rs.3000 to Ritvik
Anusha will pay Rs.3000 to Ritvik
Process returned 0 (0x0)   execution time : 319.893 s
Press any key to continue.
```

Result Analysis

For Test Case 2:

- The first line contains the input for number of people i.e 5 involved in the transaction.
- Then it takes the input for the 5 names (Ritvik,Riya,Devesh,Anusha,Isha) and how much each person owes to the other.
- Lastly, after calculation, it reduced the 20 payments down to 4 between Anusha to Ritvik, RIya & Devesh and Isha to Ritvik

Learning outcomes of application

Our application is very useful in the management of the finances in a group. Many existing application, like Splitwise, use a similar concept which allows people to deal with the money they owe or are owed within the same group in the easiest and most minimalistic way possible.

THANK YOU