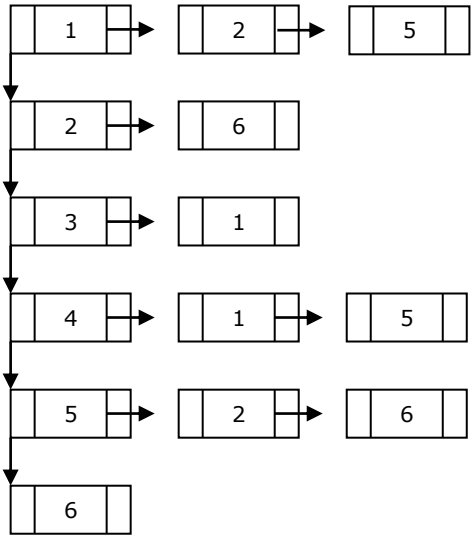
Aula 14 – Grafos Implementação

Lista de Adjacência: essa representação é uma lista de lista, onde a primeira lista indica os vértices e, para cada vértice, uma segunda lista indica seus adjacentes.
Exemplo: Dados V = {1,2,3,4,5,6} e A = {(1,2),(1,5),(2,6),(3,1),(4,1),(4,5),(5,2),(5,6)}



Declaração do Grafo Representado por Lista de Adjacência

```
//Lista de Adjacentes
struct adjacente{
       struct adjacente *prox; //armazena o endereço do próximo adjacente
       int a;
       };

//Lista de Vértices
struct vertice{
       struct vertice *proxv; //armazena o endereço do próximo vértice
       int v;
       struct adjacente *proxa; //armazena o endereço do próximo adjacente
                                //da lista de adjacentes
       };
```

Etapa I – (Entrega no dia de hoje – 29/11/2017 – até as 22:00)
Implementação dos Procedimentos de Conecta e Imprime utilizando Representação de Grafos por Lista de Adjacência

1 - (4.0 Pontos) Conecta Vértice e Adjacente. Lembrando que os vértices e os adjacentes devem estar em ordem crescente. O usuário irá digitar o arco (vértice, adjacente).

- Para conectar dois vértices quaisquer v1 e v2 (lembre-se que um vértice torna-se adjacente), é necessário primeiro percorrer a lista de vértices procurando por v1:

- Se v1 for encontrado na lista de vértices, deve-se percorrer a lista de adjacentes a v1 procurando por v2 (os vértices devem ser inseridos em ordem crescente).

- Se v2 não existir na lista de vértices adjacentes, ele deve ser inserido (em ordem crescente).

- Caso v1 não exista na lista de vértices (em ordem crescente), ele deve ser inserido e depois deve-se inserir v2 na lista de adjacentes a v1.

2 – (1.0 Ponto) Imprime grafo. A impressão deverá ser apresentada da seguinte maneira:

vértice -> adjacente – adjacente – adjacente - ...

vértice -> adjacente – adjacente - ...

```
/*Grafos por Lista de Adjacência*/

#include<stdio.h>
#include<malloc.h>

//Lista de Adjacentes
struct adjacente{
       struct adjacente *prox;
       int a;
       };

//Lista de Vértices
struct vertice{
       struct vertice *proxv;
       int v;
       struct adjacente *proxa;
       };

//Aloca um nodo do tipo vértice
struct vertice *aloca_vertice(int v){
  struct vertice *novo;
  novo=(struct vertice *)malloc(sizeof(struct vertice));
  novo->v=v;
  novo->proxv=NULL;
  novo->proxa=NULL;
  return (novo); //retorna um endereço de memória
 }

//Aloca um nodo do tipo adjacente
struct adjacente *aloca_adjacente(int a){
  struct adjacente *novo;
  novo=(struct adjacente *)malloc(sizeof(struct adjacente));
  novo->a=a;
  novo->prox=NULL;
  return (novo); //retorna um endereço de memória
  }
```

```
//Chamadas para alocar
*grafo=aloca_vertice(v1); //quando o grafo for NULL
novo_v=aloca_vertice(v1); //para os vértices
novo_a=aloca_adjacente(v2); //para os adjacentes


//Função Principal
int main(){
 struct vertice *grafo=NULL;
 int v,a;
 do{
  printf("\nPara encerrar digite 0 para o vertice e o adjacente\n");
  printf("Digite o Vertice: ");
  scanf("%i",&v);
  printf("Digite o Adjacente: ");
  scanf("%i",&a);
  if((v!=0)&&(a!=0)){
   conecta(&grafo,v,a);
   }
  }while((v!=0)&&(a!=0)); //a saída será quando o usuário digitar 0 0
 }
```

Etapa II - (Entrega no dia 06/12/2017 – até as 22:00)

Implementação dos Procedimentos de Desconecta Adjacente e Imprime utilizando Representação de Grafos por Lista de Adjacência

3 - (4.0 Pontos) Função Desconecta Adjacente. A função recebe um arco (vértice, adjacente) e a mesma deverá desconectar o adjacente relativo a esse vértice, permanecendo os demais. Ao desconectar o único adjacente de um vértice, este (vértice) deverá permanecer com o campo `proxa=NULL`. Não é permitido desconectar vértices, apenas seus adjacentes.

4 – (1.0 Ponto) Função Pesquisa Vértice e Adjacente. Deverá ser inserido pelo usuário um arco (vértice e adjacente) e a função retorna se existe ou não o vértice e o adjacente (arco).

Observações:

- Data de Entrega: dia 06 de dezembro de 2017, pelo Moodle, até as 22 horas, sendo que essa aula será destinada ao término da implementação e plantão de dúvidas.