Atividade da semana 07

Antonio Erick Freitas Ferreira - 542631

Para esta atividade, eu implementei uma classe chamada Node, onde cada objeto tem propriedades como: pai, pi, tempo de descoberta, tempo de encerramento, lista de adj, cor, contador de caminhos, contador de ciclos e outros. Cada Node representa um vertice do grafo.

Questão 03 - Para encontrar a quantidade de caminhos que existem entre um vértice e todos os outros vertices fazendo uma alteração no algoritmo padrão da busca em largura, basta comparar na lista de adjacência do vertice, se algum desses vértices é cinza ou preto e se esse vértice não é a origem, caso essa condição seja satisfeita, basta somar +1 no contador de caminhos.

```cpp
void BFS(vector<Node*> grafo,Node* s){
    //Busca em largura usa fila

    //Reinicia o grafo.
    for(int i = 1; i < grafo.size();i++){
        grafo[i]->color = 'b';
        grafo[i]->start = 0;
        grafo[i]->end = 0;
        grafo[i]->pi = nullptr;
        grafo[i]->numCaminho = 0;

    }

    s->color = 'c';
    s->start = 0;
    s->pi = nullptr;

    queue<Node*> fil;
    fil.push(s);
    while(!fil.empty()){
        Node* aux = fil.front();
        fil.pop();
        for(Node* k : aux->adj){
            if(k->color == 'b'){
                k->pi = aux;
                k->color = 'c';
                k->start = aux->start+1;
                k->numCaminho++; //Pois se eu chegu
                fil.push(k);
            }
            else if(k->color != 'b' && k != s){
                k->numCaminho++;
            }
        }
        aux->color = 'p';
    }
}
```

Questão 07- Para saber se existe ciclos em um grafo em tempo O(|V| + |E|) basta fazer uma simples alteração na busca em profundidade. A busca em profundidade (DFS) faz o caminho mais longo de um vertice origem até o vértice mais distante possível. Caso durante esse algoritmo seja encontrado algum vertice cinza, quer dizer que há um ciclo, pois houve como sair de um vértice A, navegar pelo grafo sem repetir arestas e chegar no mesmo vértice A.

```
28  void DFS(vector<Node*> grafo){
29      tempo = 0;
30      //Busca em profundidade usa a pilha de recursao
31          for(int i = 1; i < grafo.size();i++){
32              if(grafo[i]->color == 'b'){ //Vertice ainda não descoberto.
33                  Visit(grafo,grafo[i]);
34              }
35          }
36  }
```

```
 9  void Visit(vector<Node*> grafo,Node* u){
10      tempo++;
11      u->start = tempo;
12      u->color = 'c';
13      for(auto k : u->adj){
14          if(k->color == 'b'){
15              k->pi = u;
16              Visit(grafo,k);
17          }else{
18              k->countCicle++;
19          }
20      }
21      u->color = 'p';
22      tempo++;
23      u->end = tempo;
24
25  }
```

Observe que esse algoritmo contará quantos ciclos tem para cada vértice.