UM *FRAMEWORK* PARA ALGORITMOS BASEADOS NA TEORIA DOS GRAFOS

Acadêmico: Maicon Rafael Zatelli
Orientador: Dr. Paulo César Rodacki Gomes

Roteiro

- Introdução
- Objetivos do trabalho
- Fundamentação teórica
- Desenvolvimento do trabalho
- Conclusão
- Extensões

Introdução

Importância da teoria dos grafos

- Redes
- Transportes
- Comunicações
- Jogos

Problemas

- Muitos algoritmos
- Muitas propriedades
- Dificuldade em criar grafos com muitos vértices e arestas

Solução

- Um *framework* voltado para a teoria dos grafos

Objetivos do trabalho

- Construir um *framework* para auxiliar no desenvolvimento de softwares baseados na teoria dos grafos;
- Disponibilizar um subconjunto de algoritmos clássicos;
- Disponibilizar opção para extrair propriedades de grafos;
- Disponibilizar geradores de grafos com base em restrições;
- Persistir grafos;
- Documentar o *framework*;
- Disponibilizar uma aplicação de exemplo.

Fundamentação teórica

Conceitos básicos

- Definição de grafo
- Algoritmos de grafos
- Propriedades de grafos

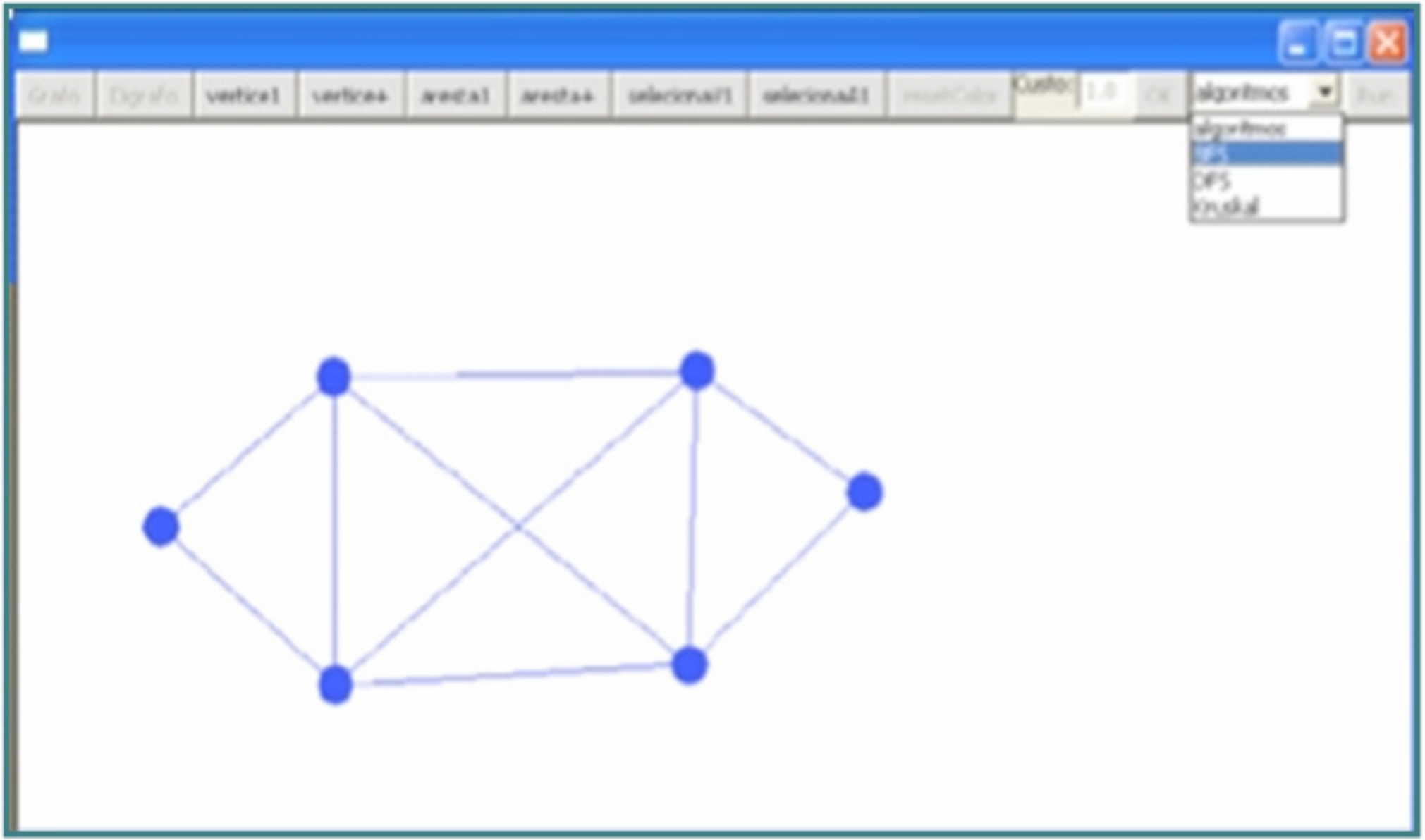
Trabalhos correlatos

- JgraphT (2005)
- TCC Hackbarth (2008)
- TCC Braun (2009)

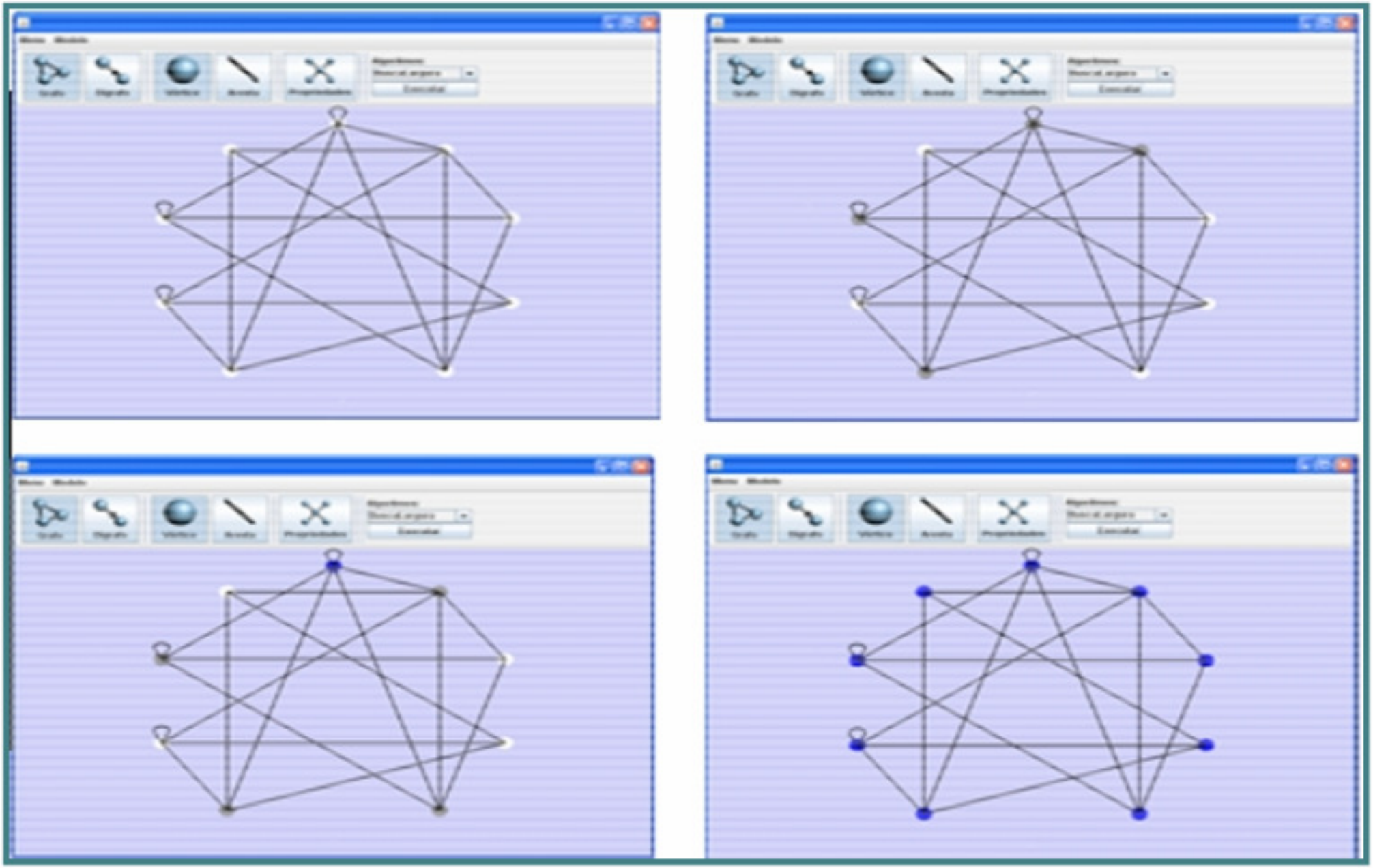
JgraphT (2005)

```
public void init(  ) {
    ListenableGraph g = new ListenableDirectedGraph(DefaultEdge.class);
    m_jgAdapter = new JGraphModelAdapter( g );
    JGraph jgraph = new JGraph( m_jgAdapter );

    adjustDisplaySettings( jgraph );
    getContentPane(  ).add( jgraph );
    resize( DEFAULT_SIZE );

    g.addVertex( "v1" );
    g.addVertex( "v2" );
    g.addVertex( "v3" );
    g.addVertex( "v4" );

    g.addEdge( "v1", "v2" );
    g.addEdge( "v2", "v3" );
    g.addEdge( "v3", "v1" );
    g.addEdge( "v4", "v3" );

    positionVertexAt( "v1", 130, 40 );
    positionVertexAt( "v2", 60, 200 );
    positionVertexAt( "v3", 310, 230 );
    positionVertexAt( "v4", 380, 70 );
}
```

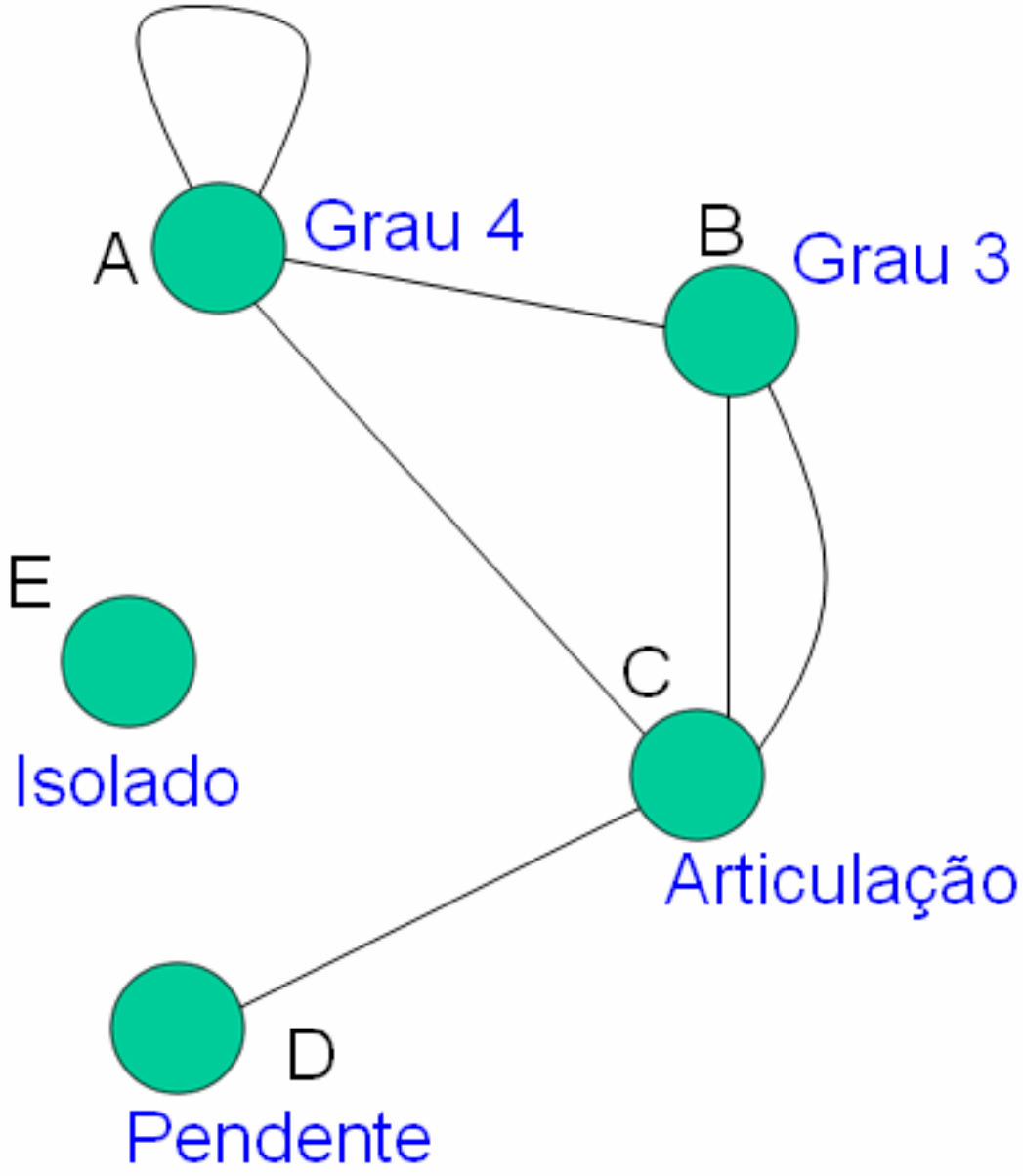
HACKBARTH (2008)

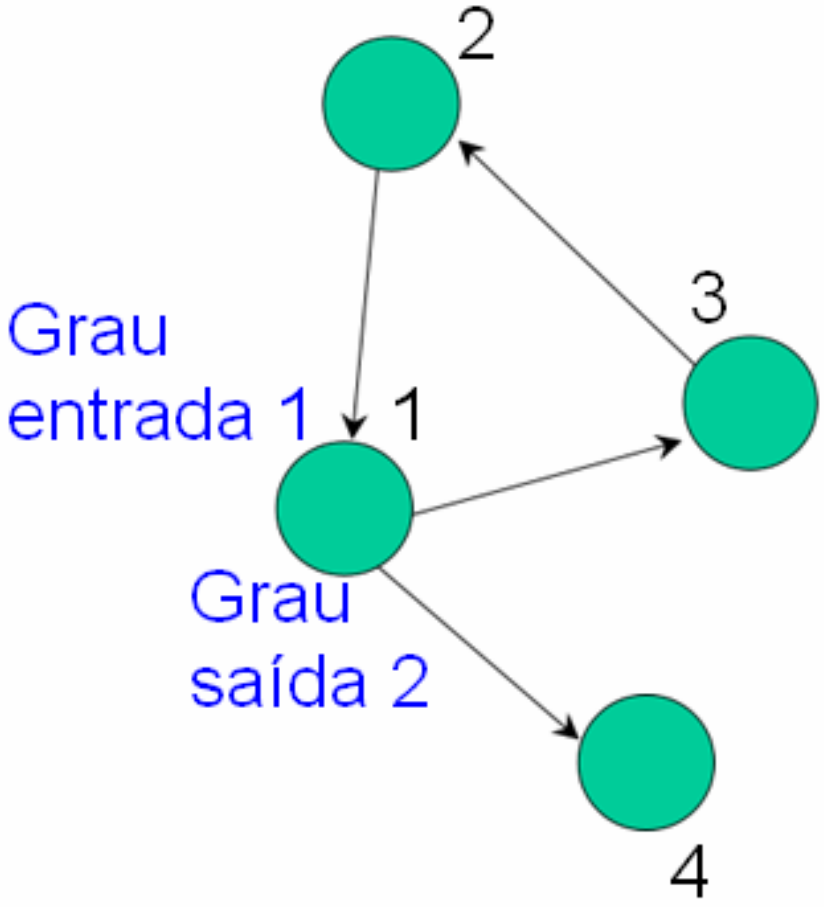


BRAUN (2009)

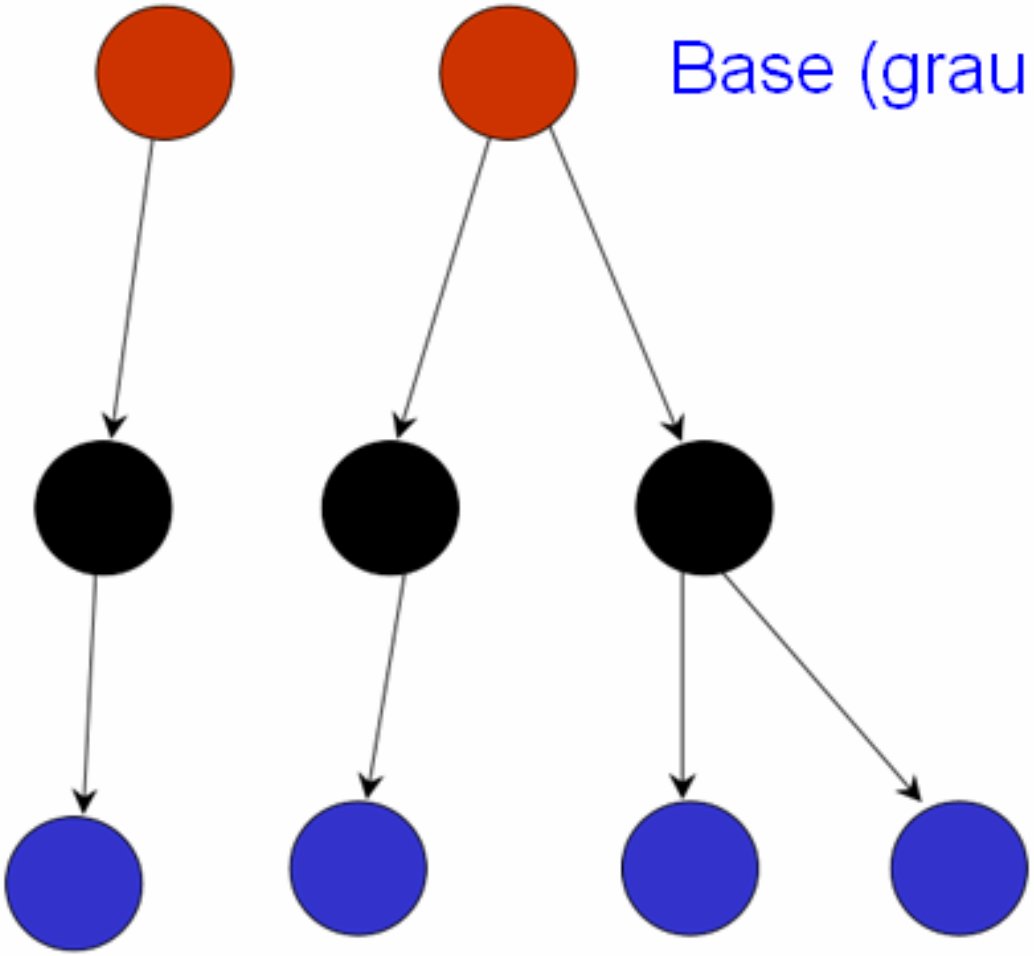


Fundamentação teórica





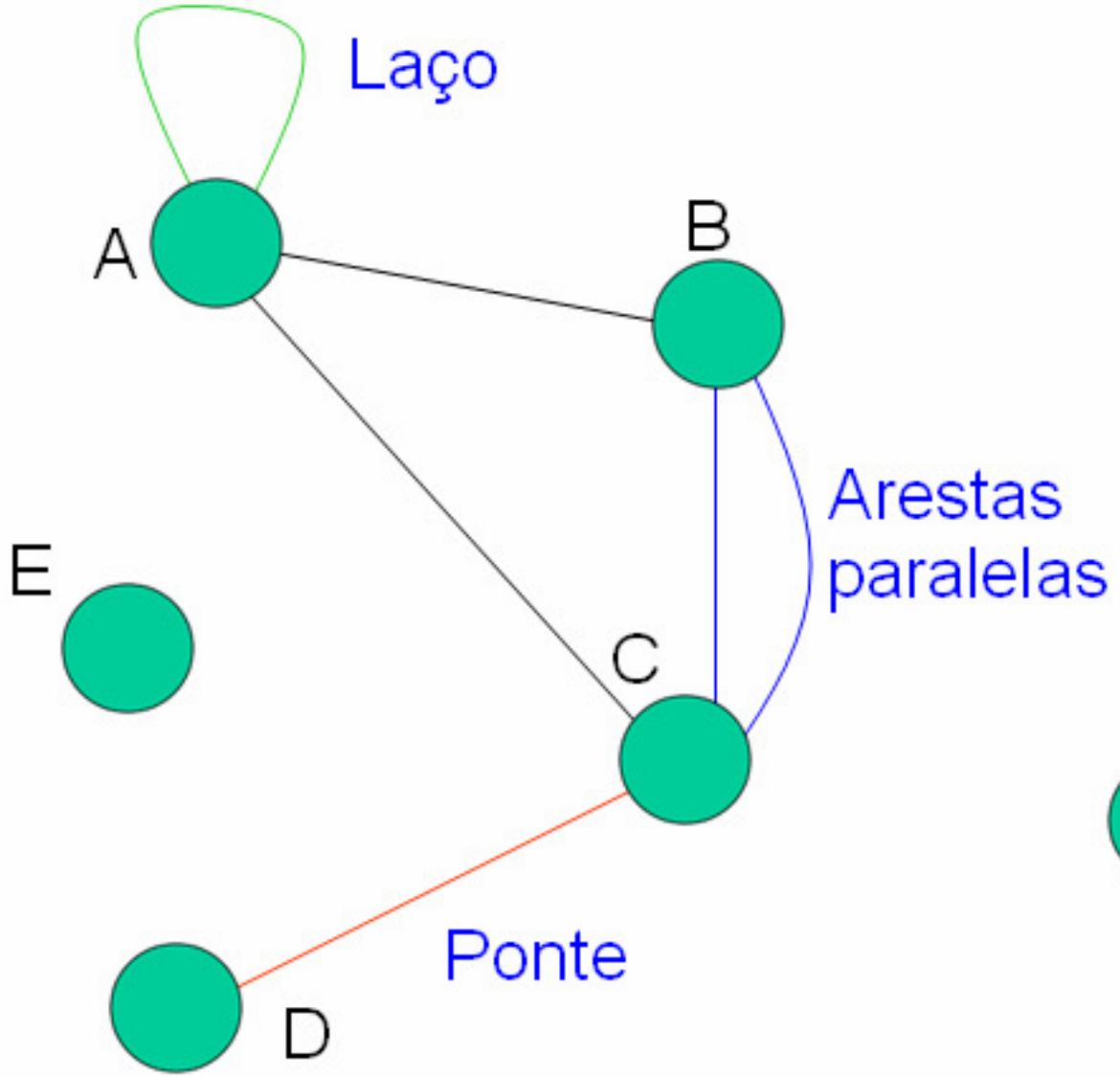
Fundamentação teórica

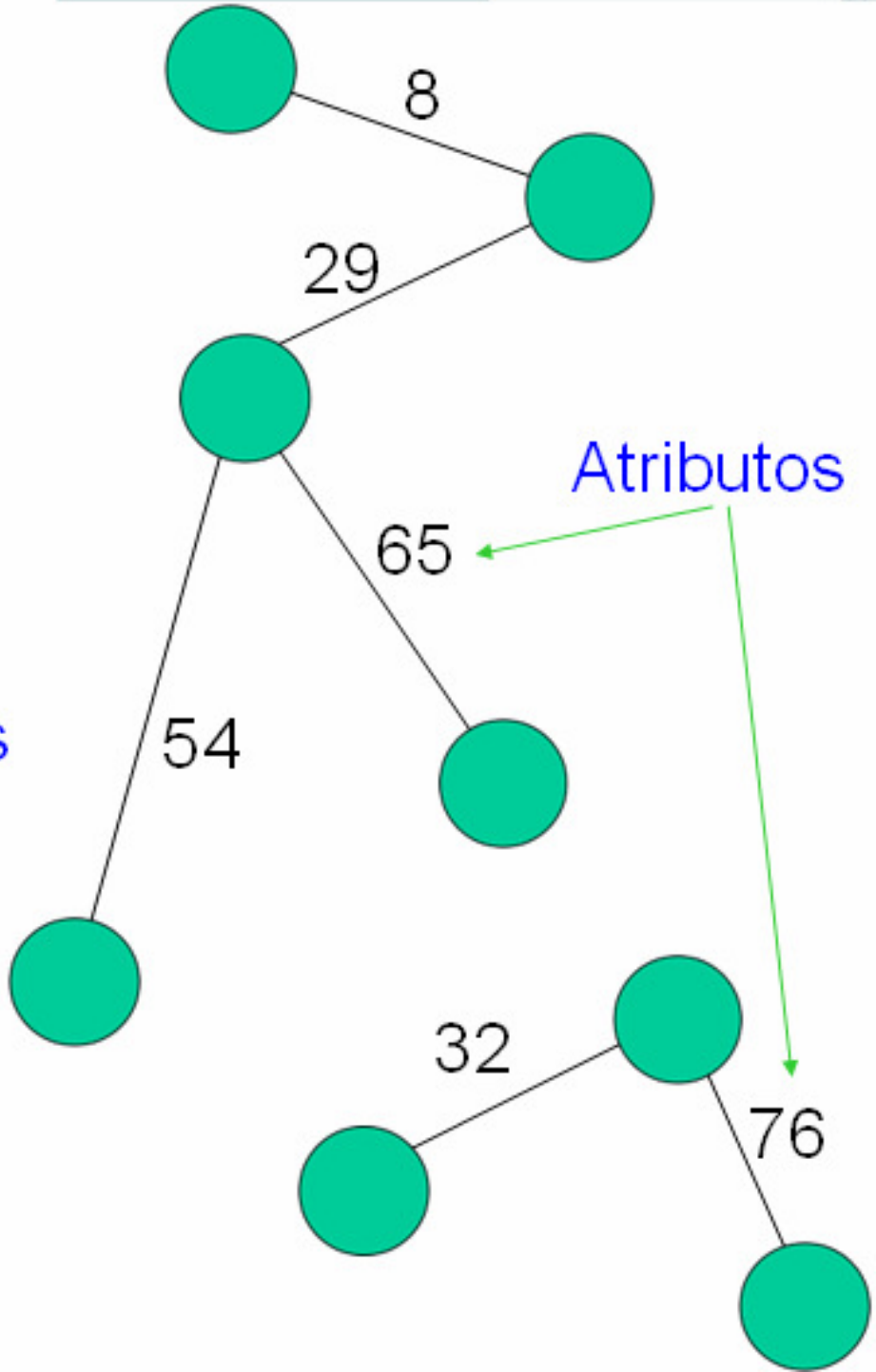


Base (grau entrada 0)

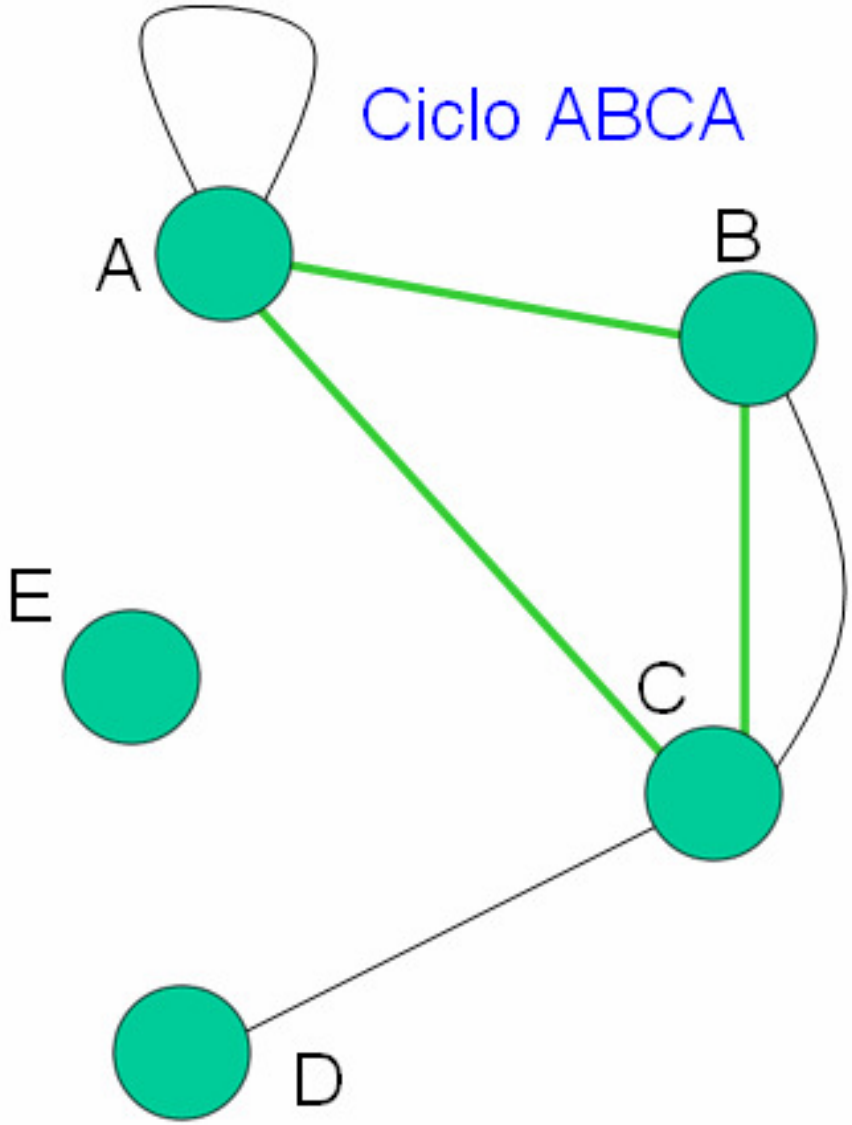
Antibase (grau saída 0)

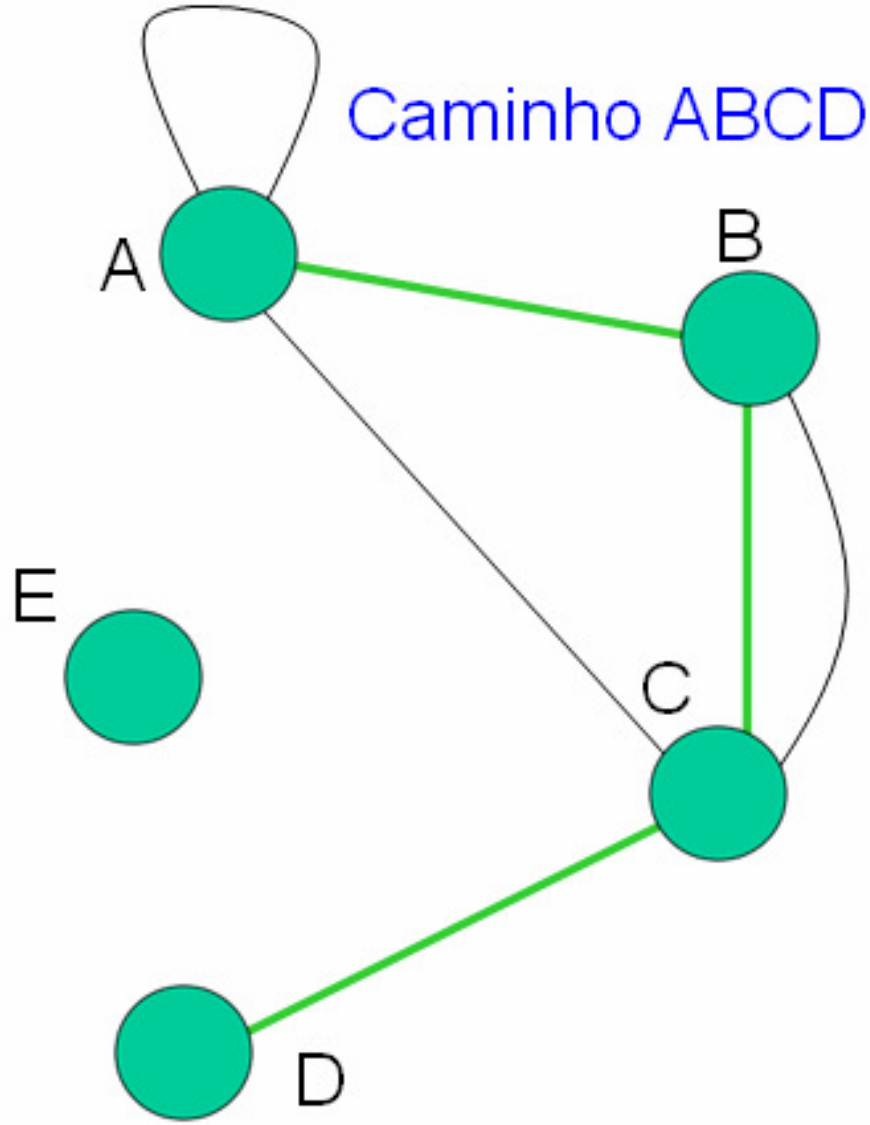
Fundamentação teórica



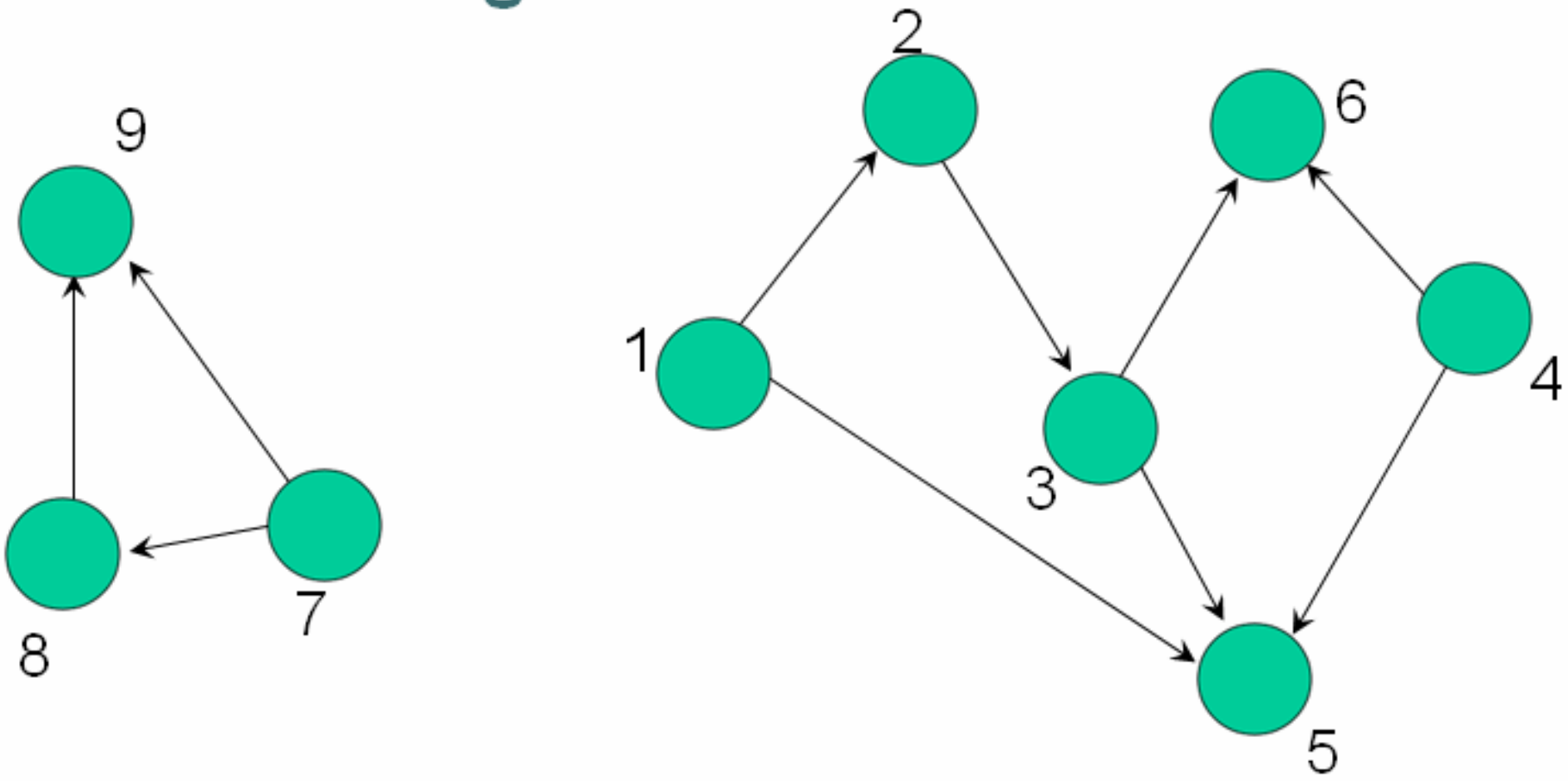


Fundamentação teórica



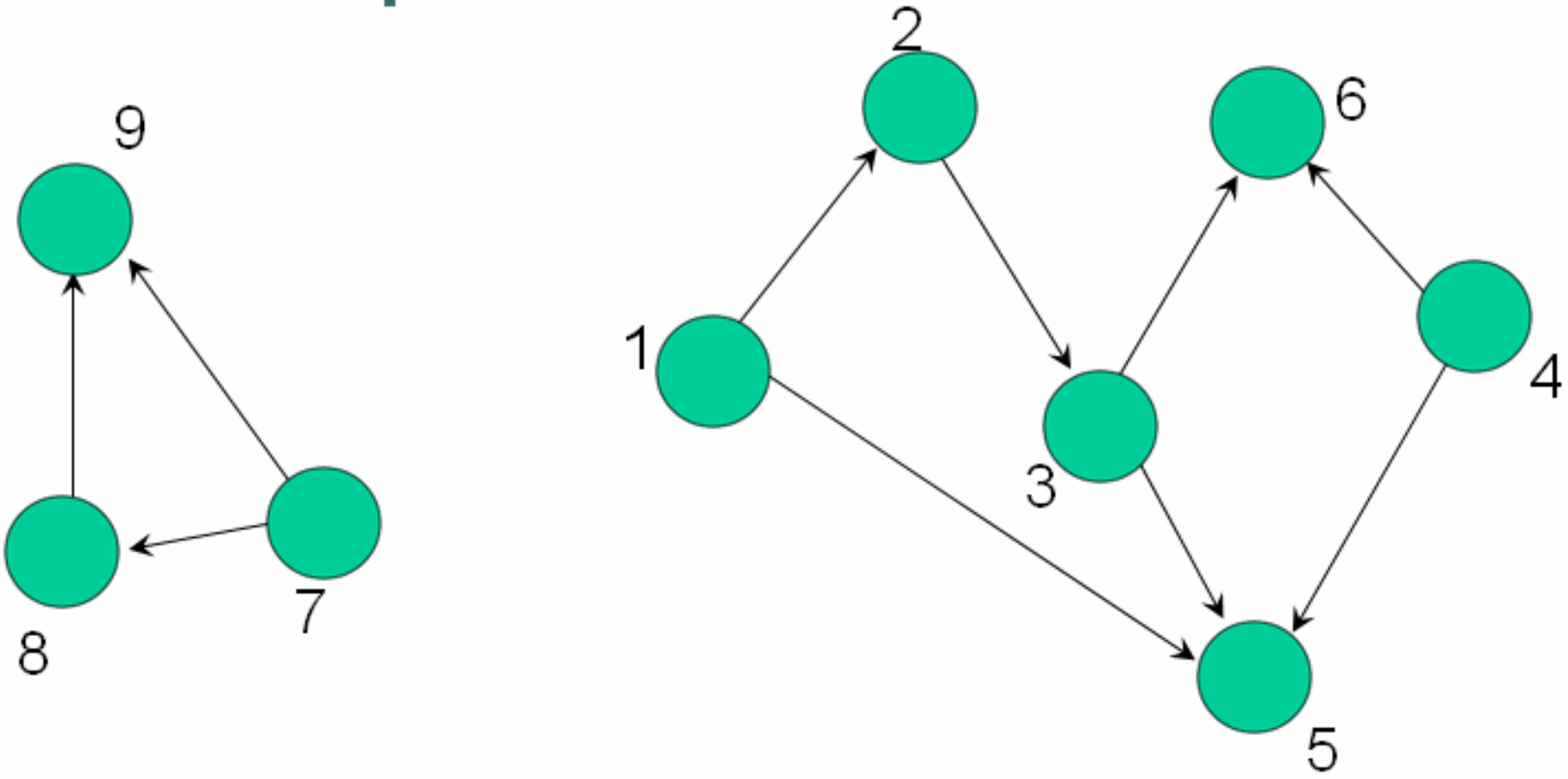


Busca em largura



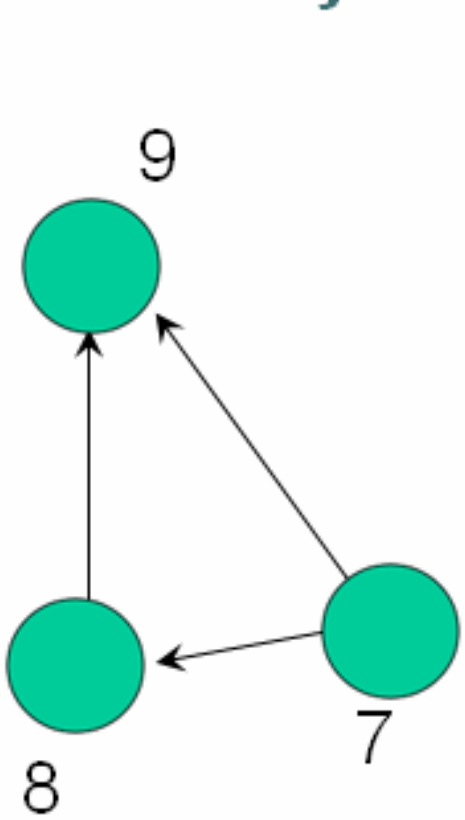
Resultado: 1 2 5 3 6 4 7 9 8

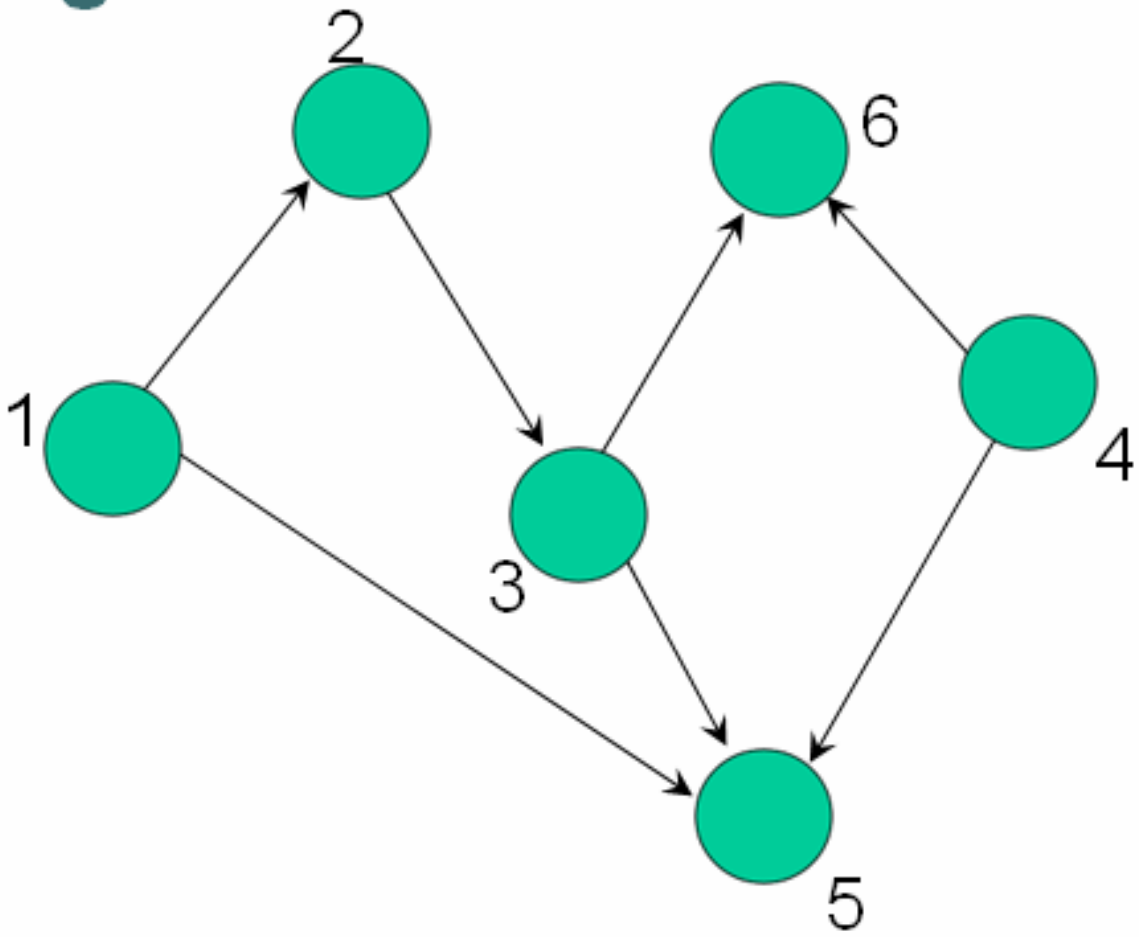
Busca em profundidade



Resultado: 1 2 3 5 6 4 7 8 9

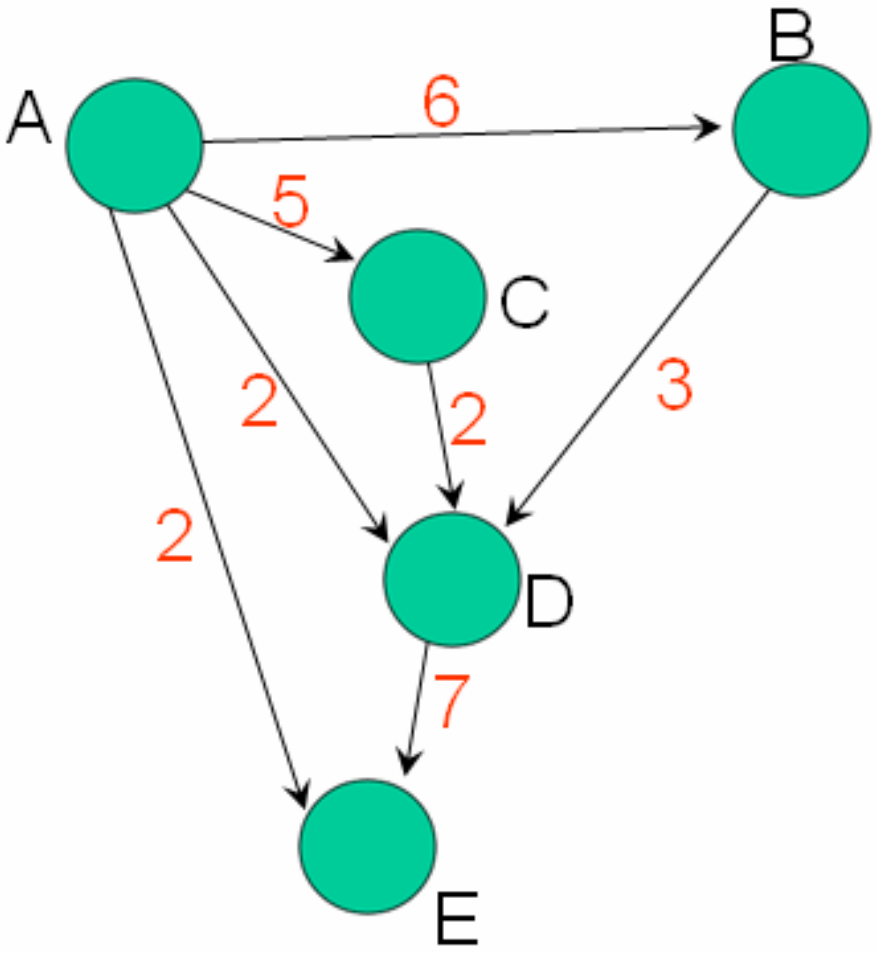
Ordenação topológica

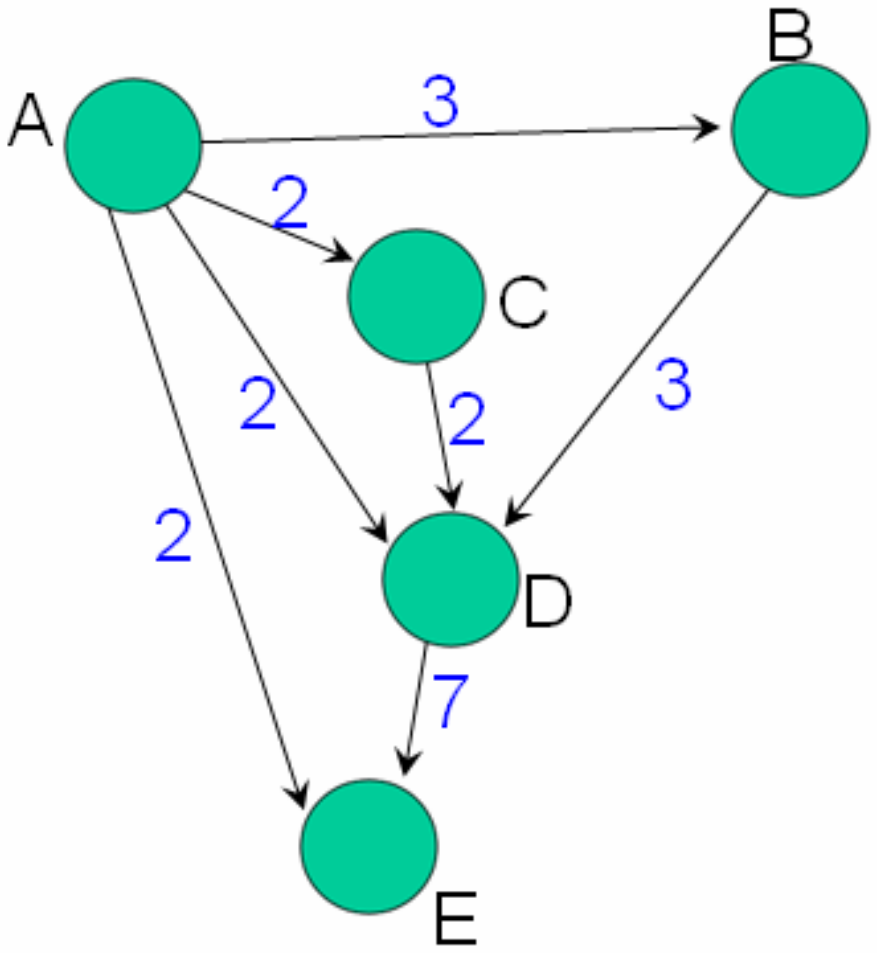




Resultado: 7 > 8 > 9 > 4 > 1 > 2 > 3 > 6 > 5

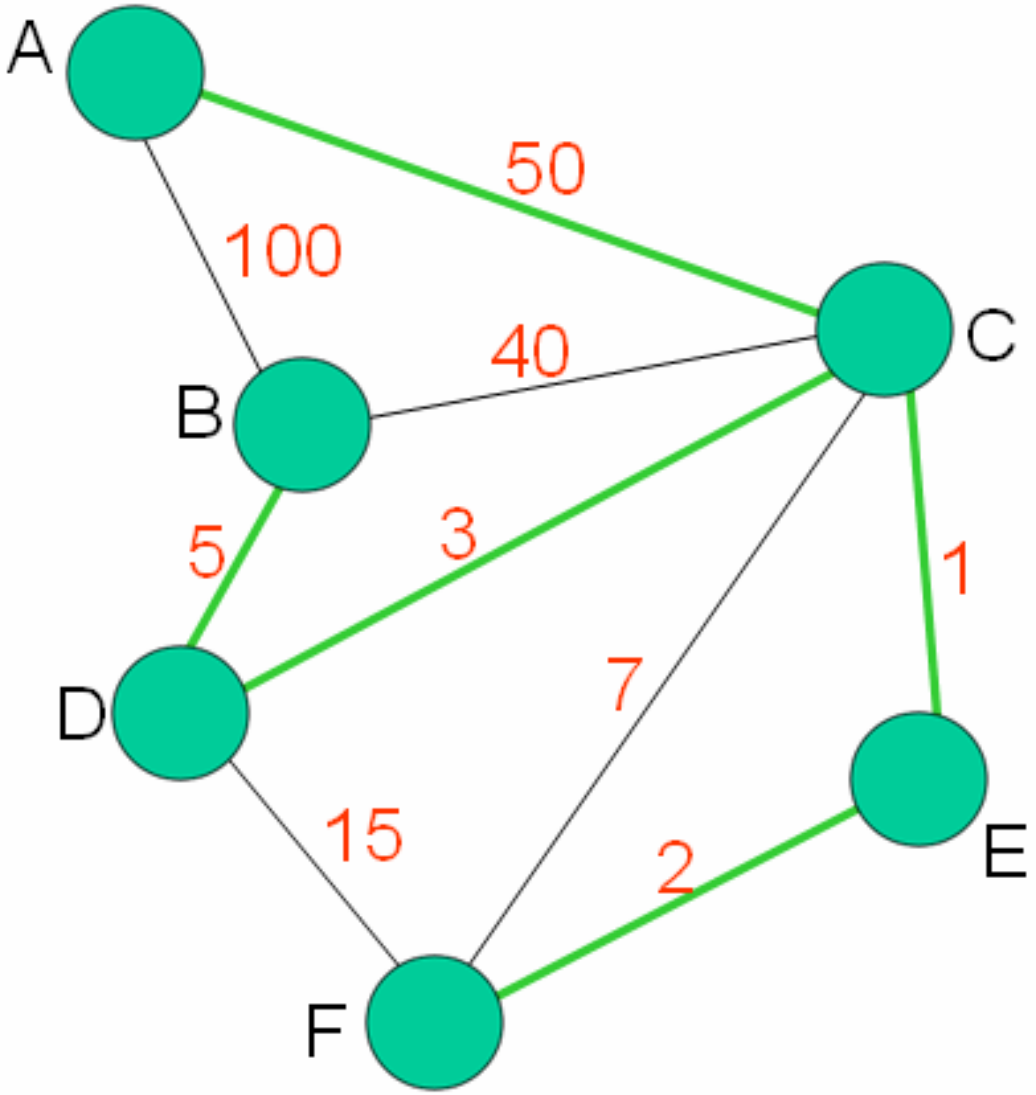
Ford-Fulkerson





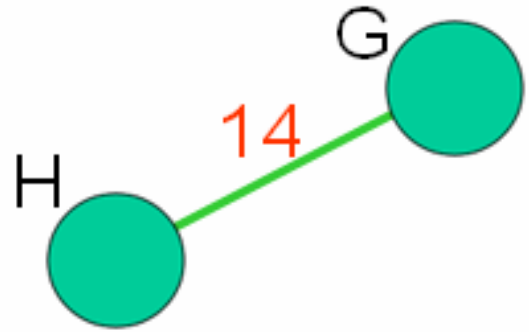
Resultado (A, E): 9

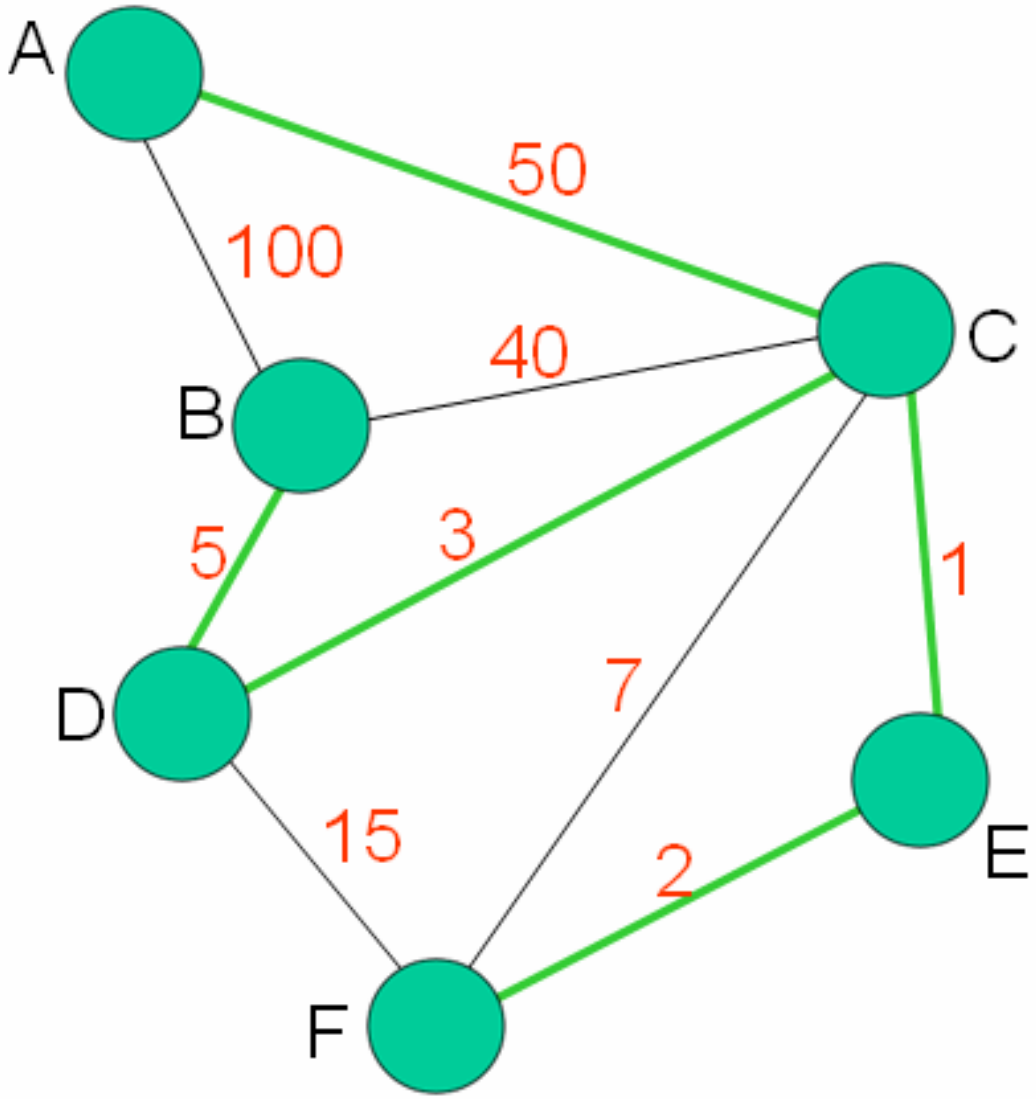
Prim



Resultado: 61

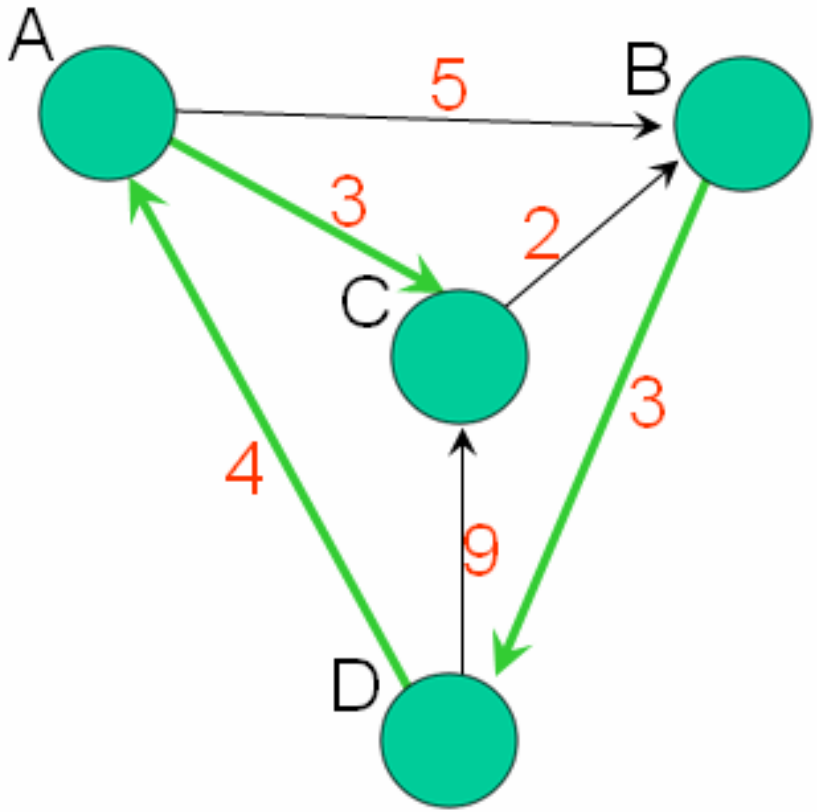
Kruskal





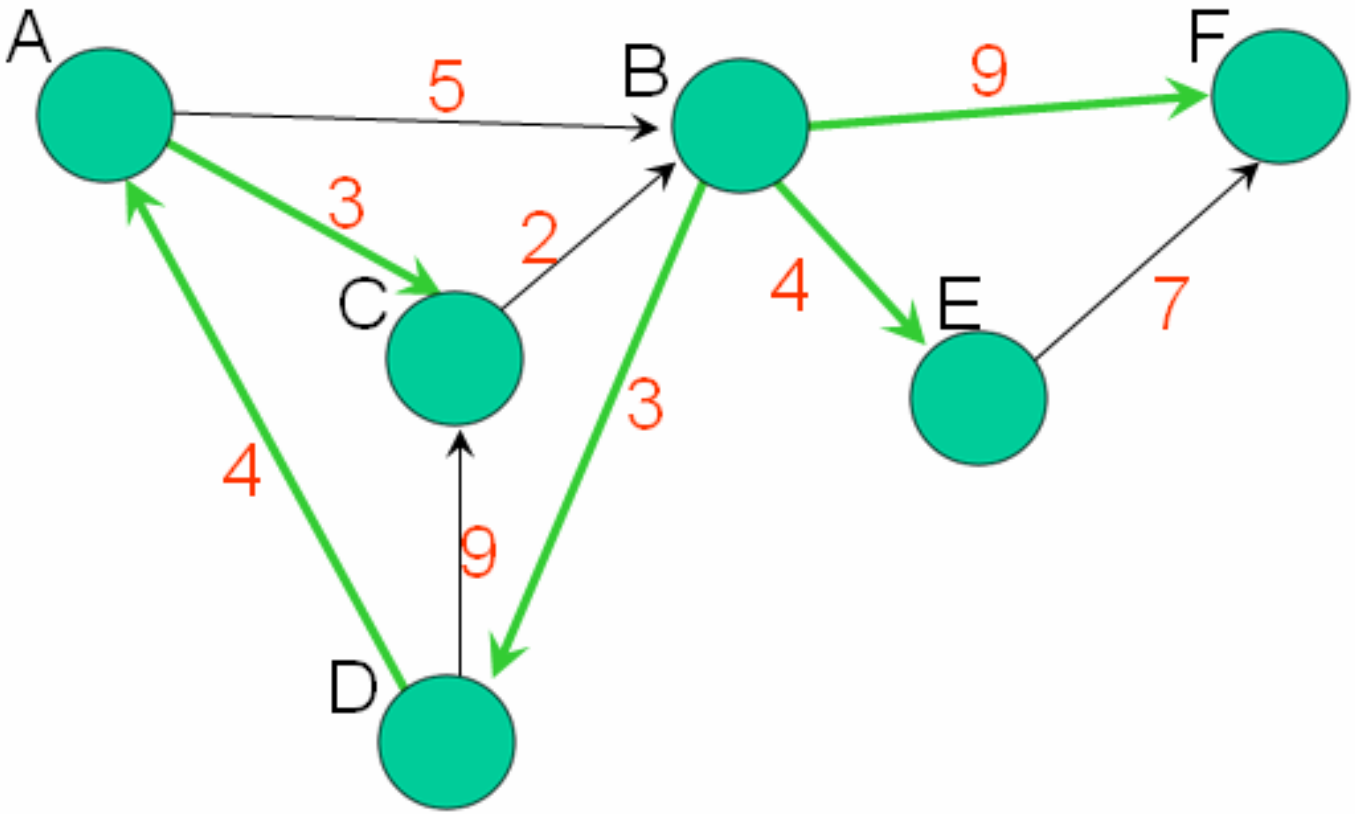
Resultado: 75

Dijkstra



Resultado (B,C): 10 Caminho: [2 > 4 > 1 > 3]

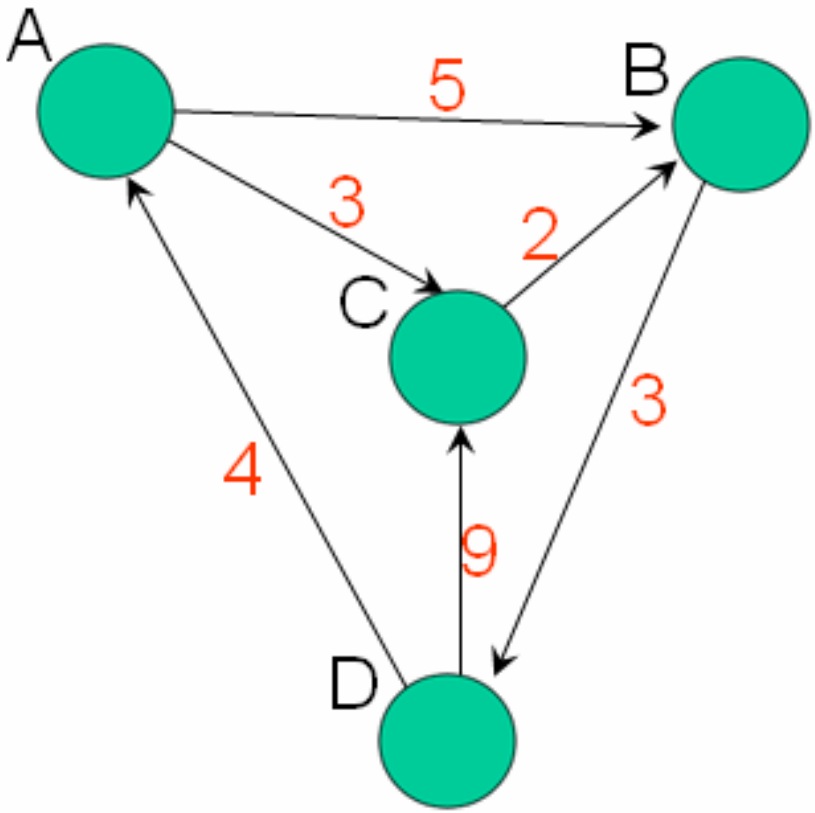
Bellman-Ford



Resultado (B)

	A	B	C	D	E	F
B	7	0	10	3	4	9

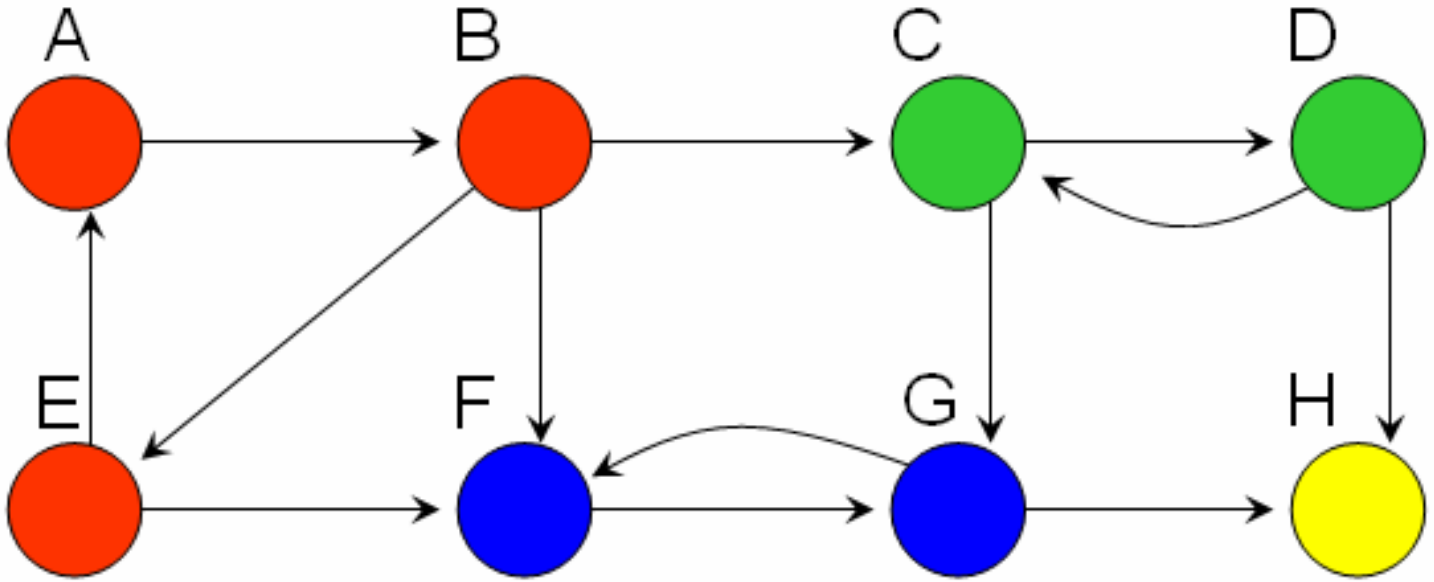
Floyd-Warshall

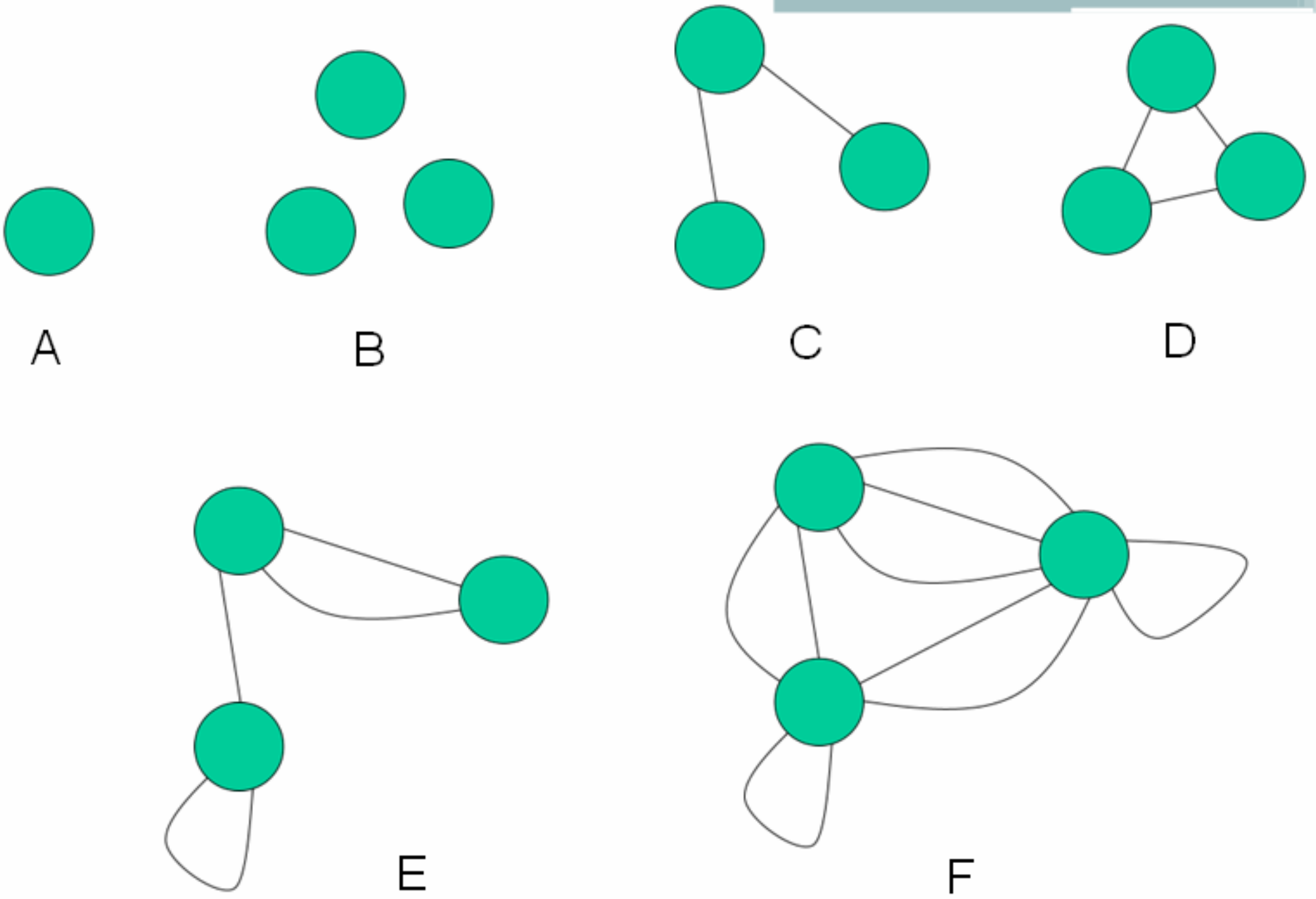


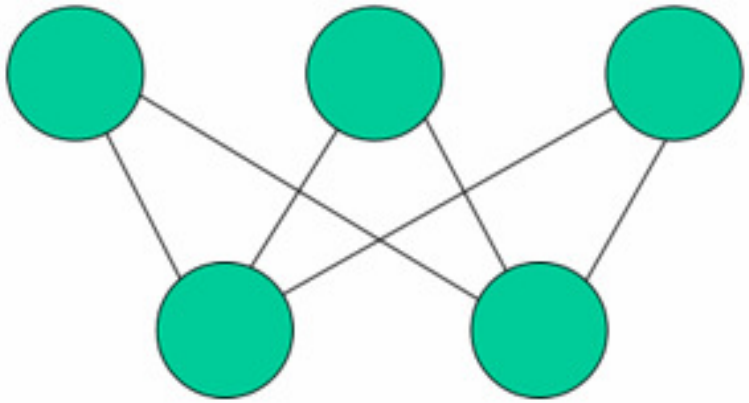
Resultado

	A	B	C	D
A	0	5	3	8
B	7	0	10	3
C	9	2	0	5
D	4	9	7	0

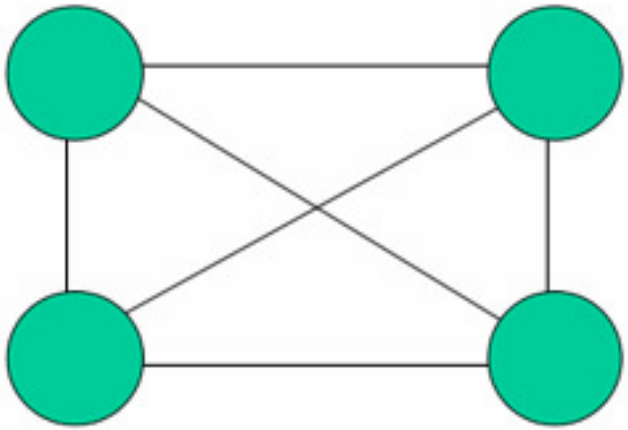
Hopcroft-Tarjan



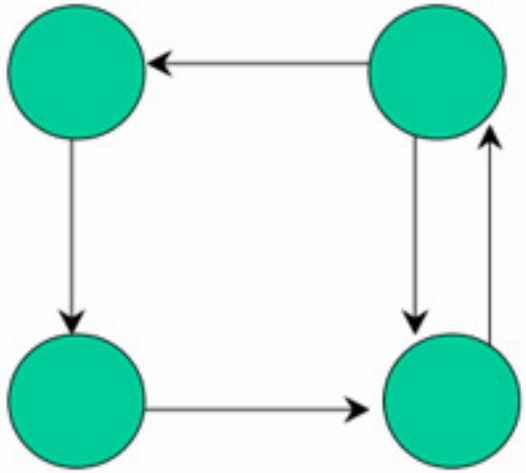

A
B
C
D
E
F



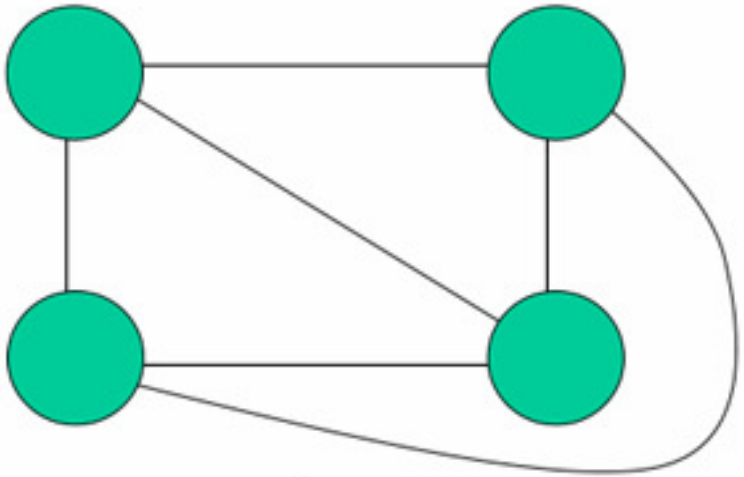
G



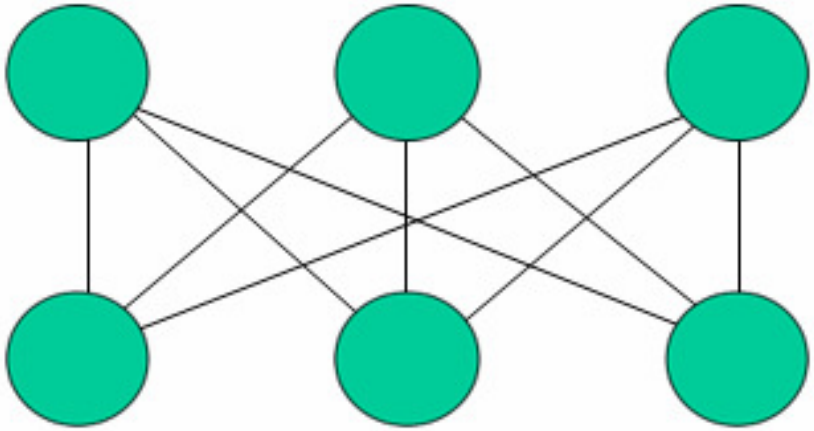
H



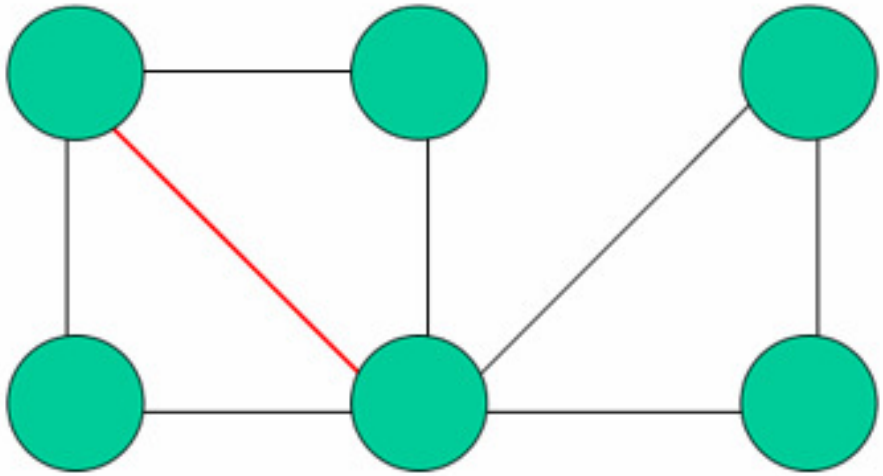
I



J



K



L

Desenvolvimento do trabalho

- Levantamento dos requisitos;
- Especificação do *framework* através de diagramas da UML;
- Especificação do modelo de persistência de grafos;
- Implementação do *framework*;
- Documentação do *framework*.

Requisitos do framework

Requisitos funcionais

- Disponibilizar funções para criação e edição de grafos
- Disponibilizar funções para gerar grafos
- Permitir extrair propriedades de grafos
- Disponibilizar um subconjunto de algoritmos clássicos
- Permitir persistir e carregar grafos

Requisitos não-funcionais

- Ser implementado utilizando o ambiente Fedora Eclipse 3.4.1 e a linguagem Java versão 6
- Conter documentação detalhada sobre os recursos oferecidos pelo *framework*

Diagrama de casos de uso

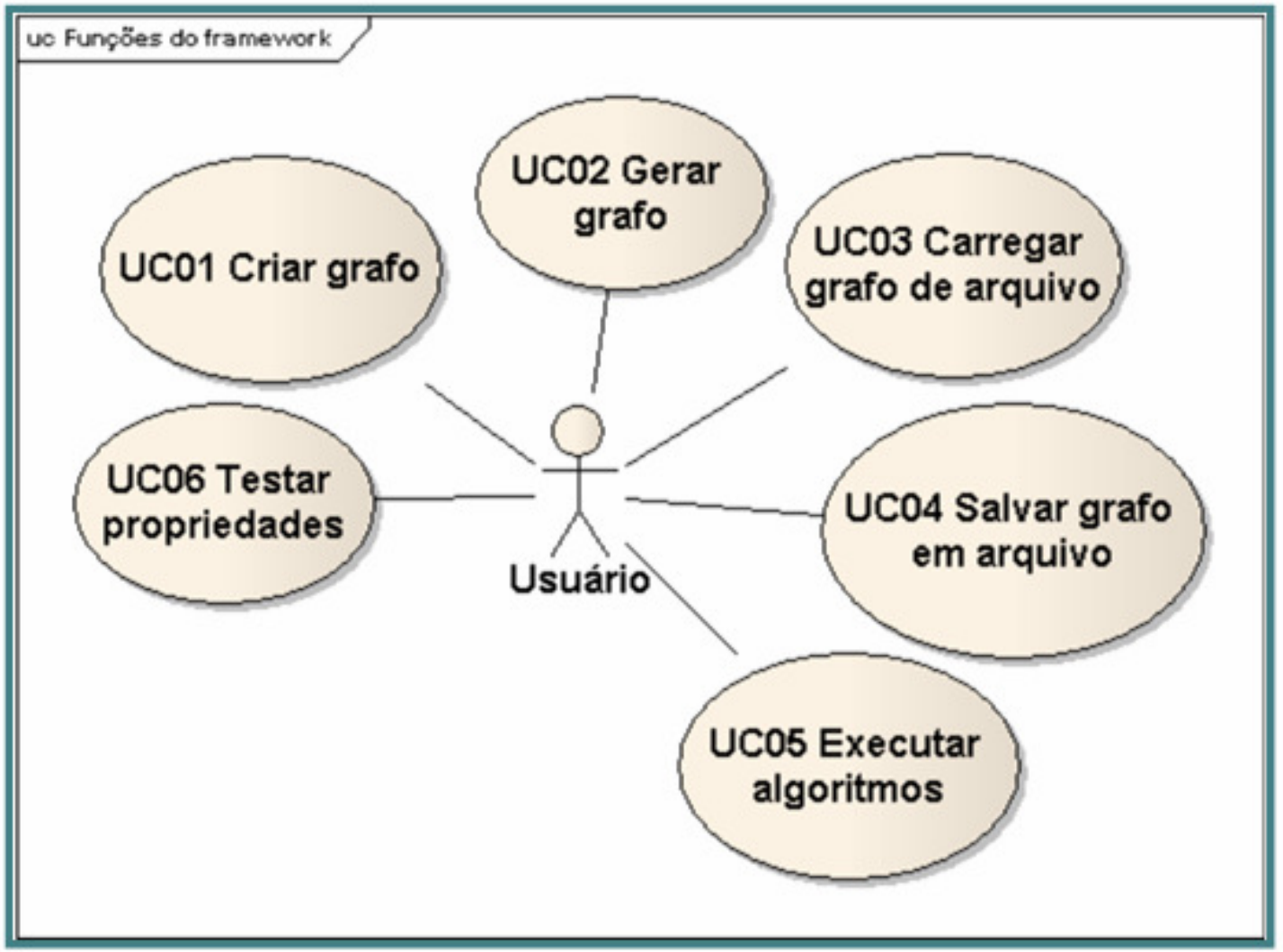


Diagrama de pacotes

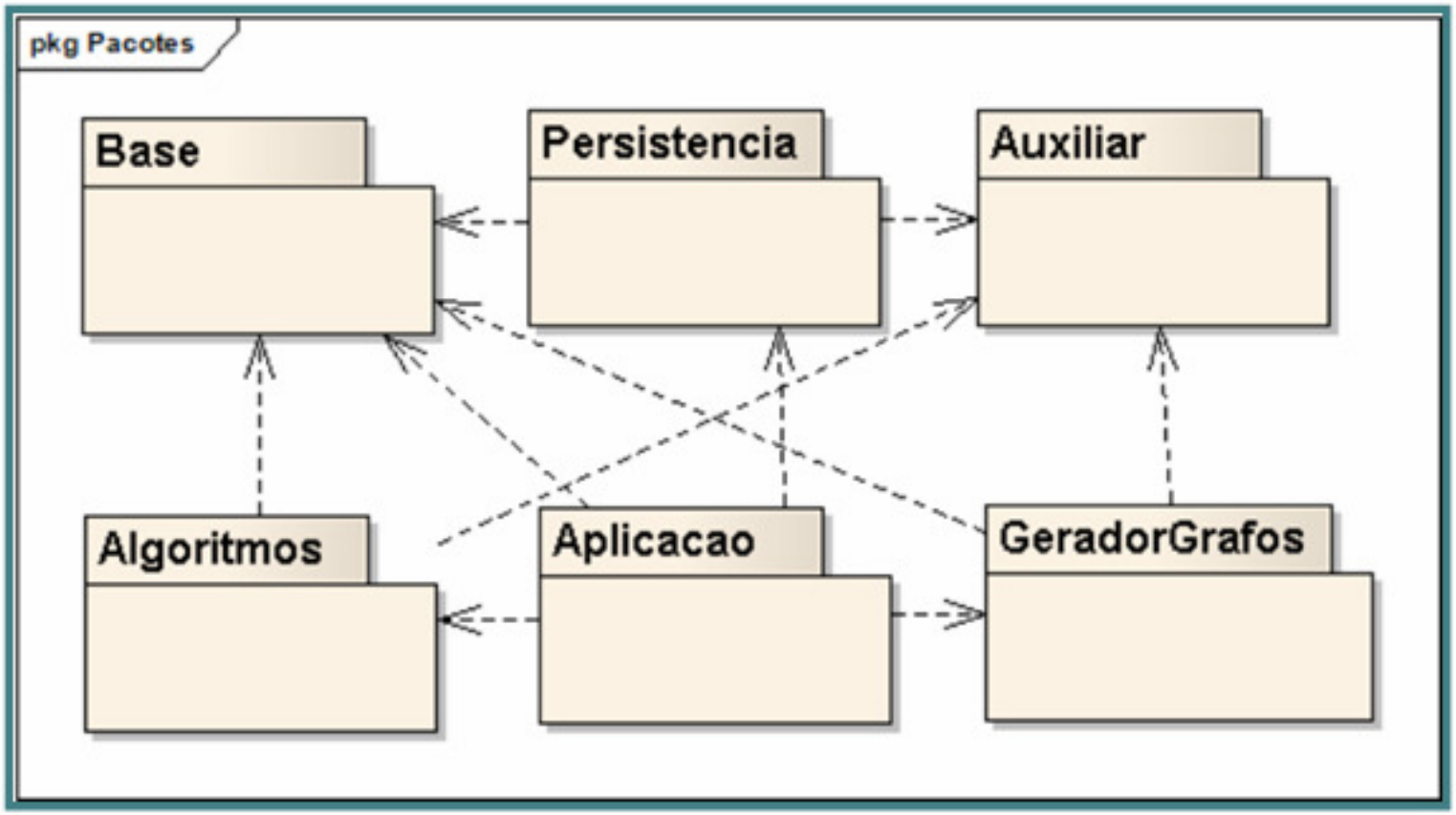


Diagrama de classes

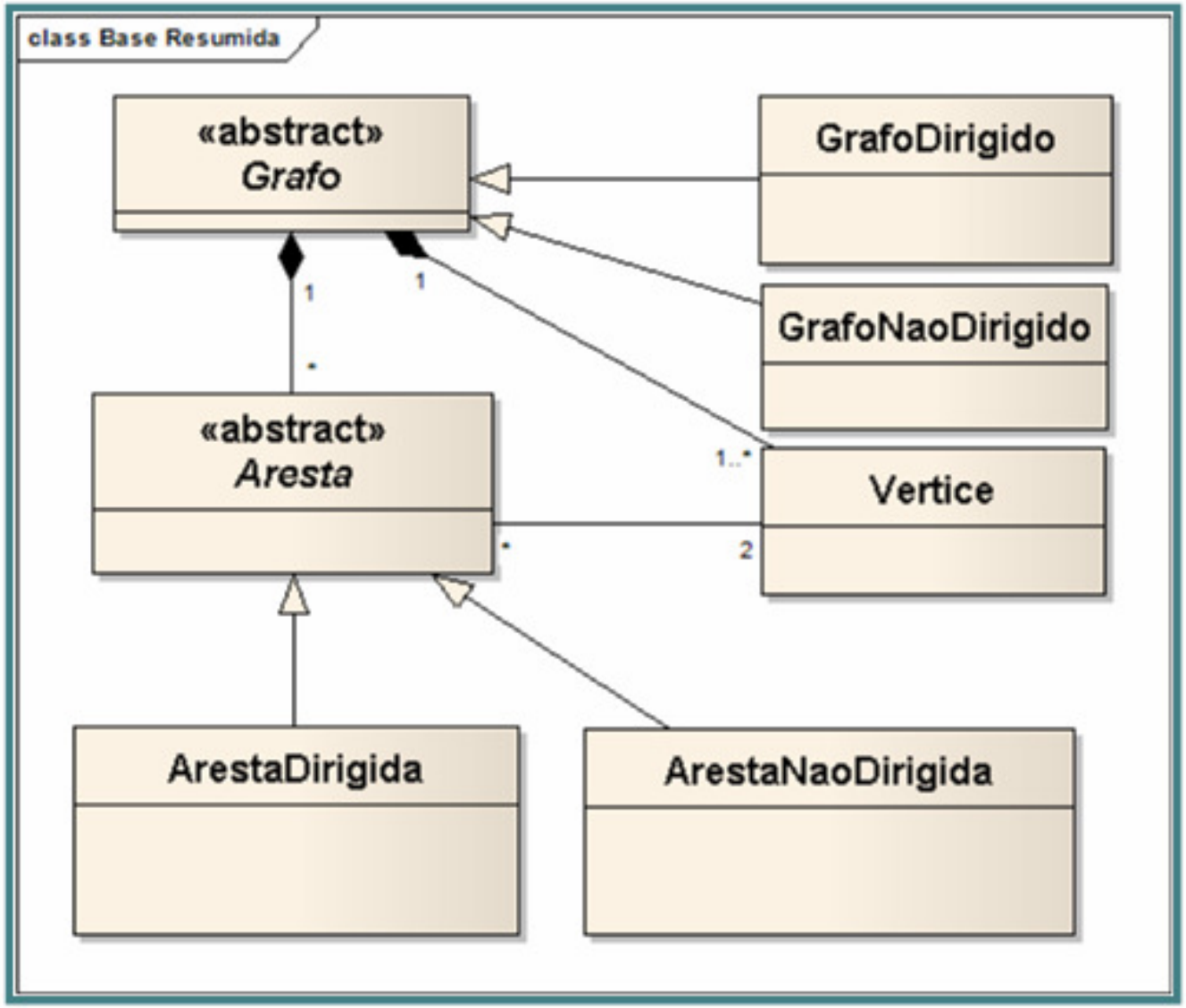


Diagrama de classes

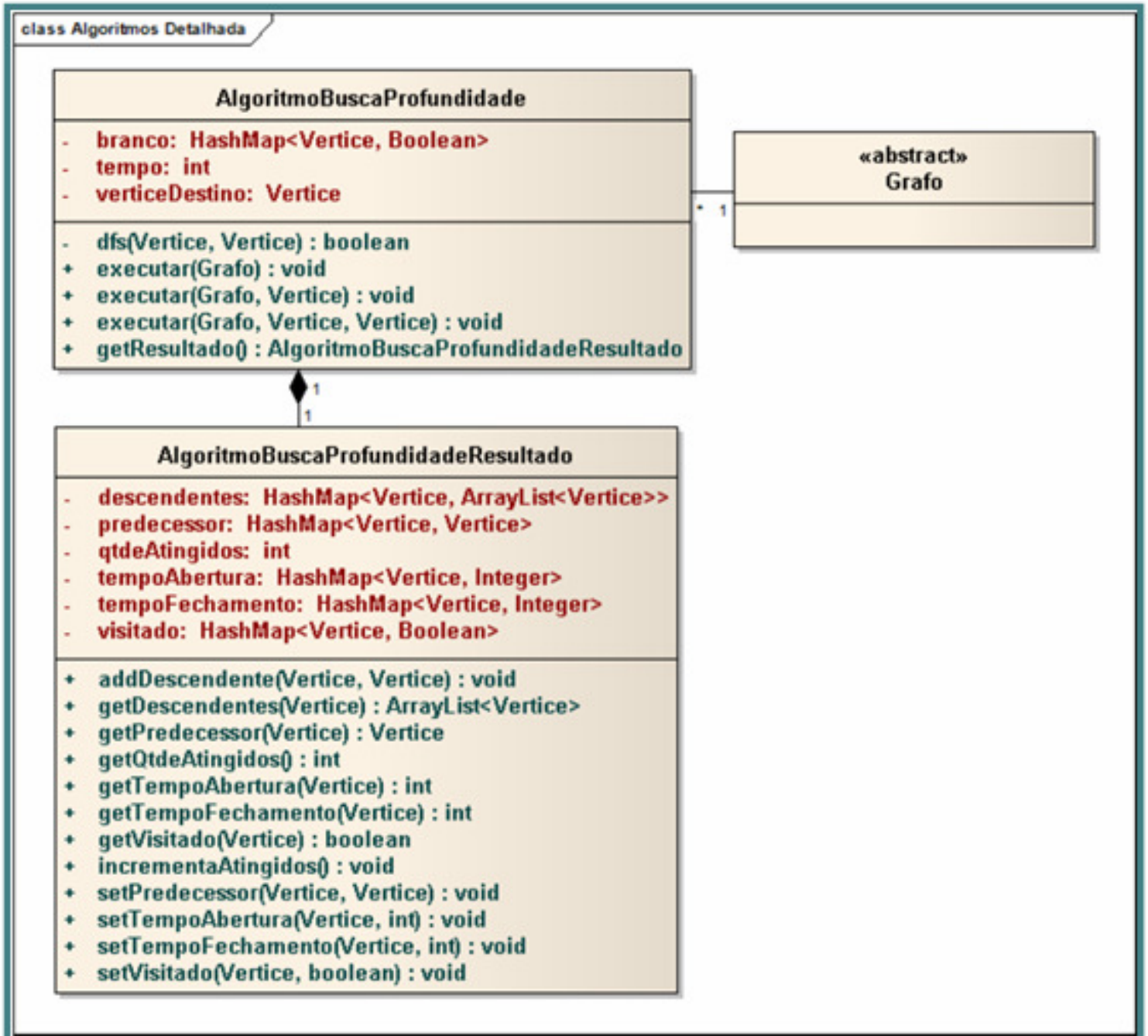


Diagrama de classes

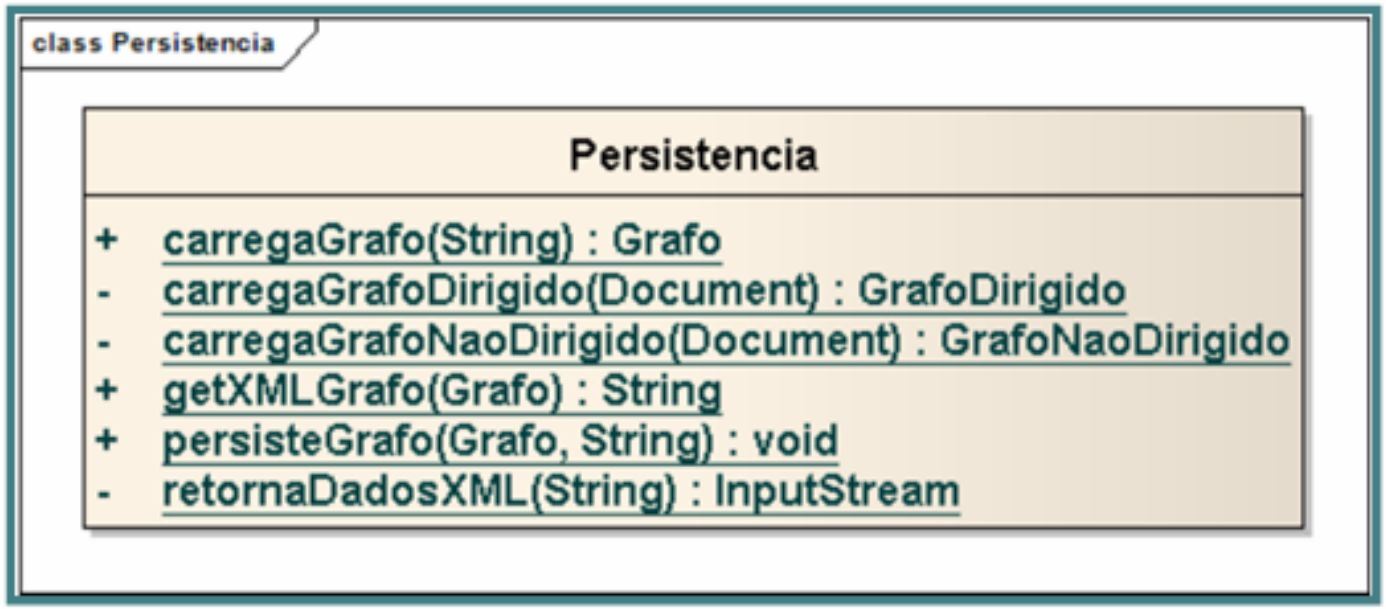


Diagrama de classes

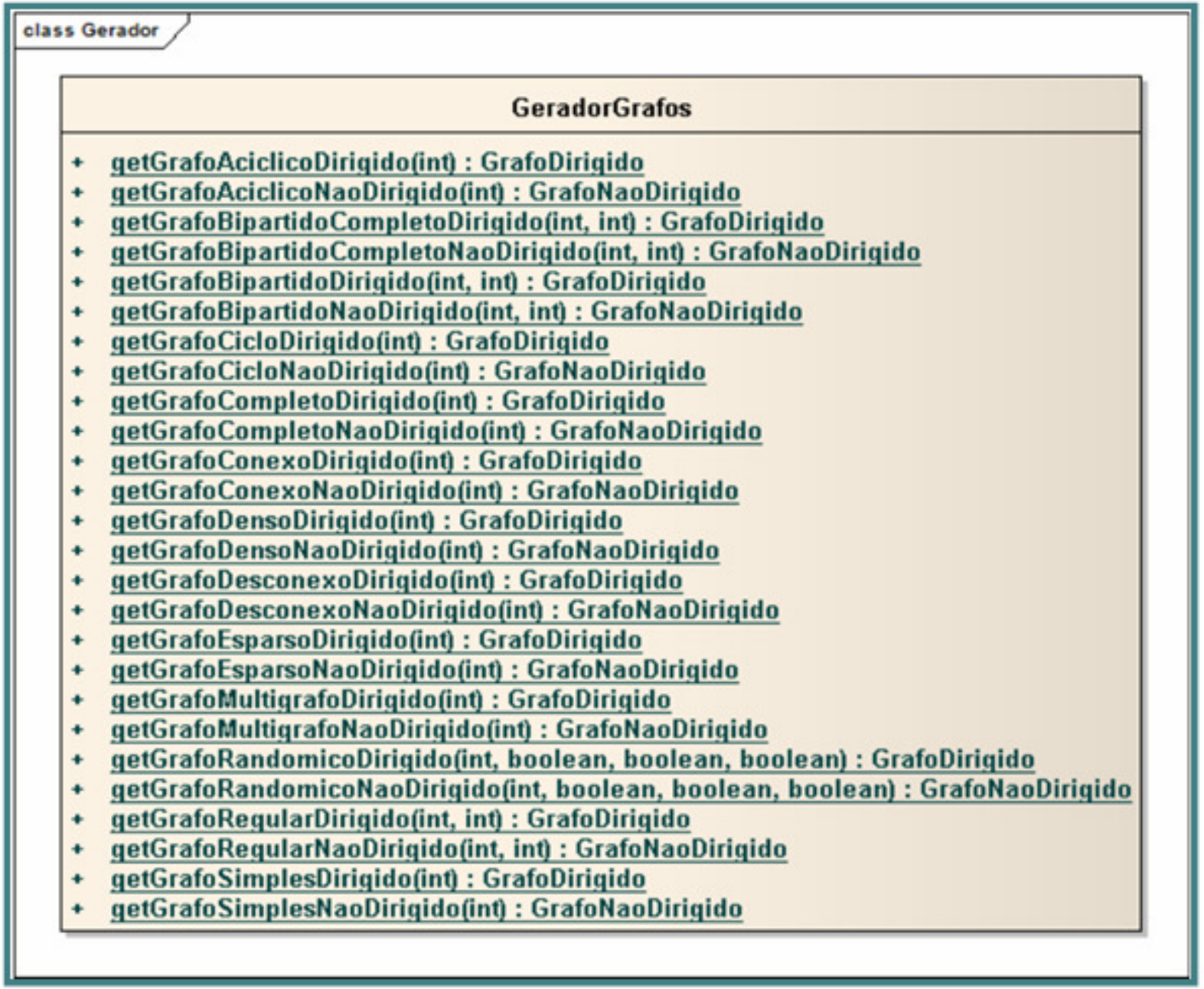


Diagrama de atividades

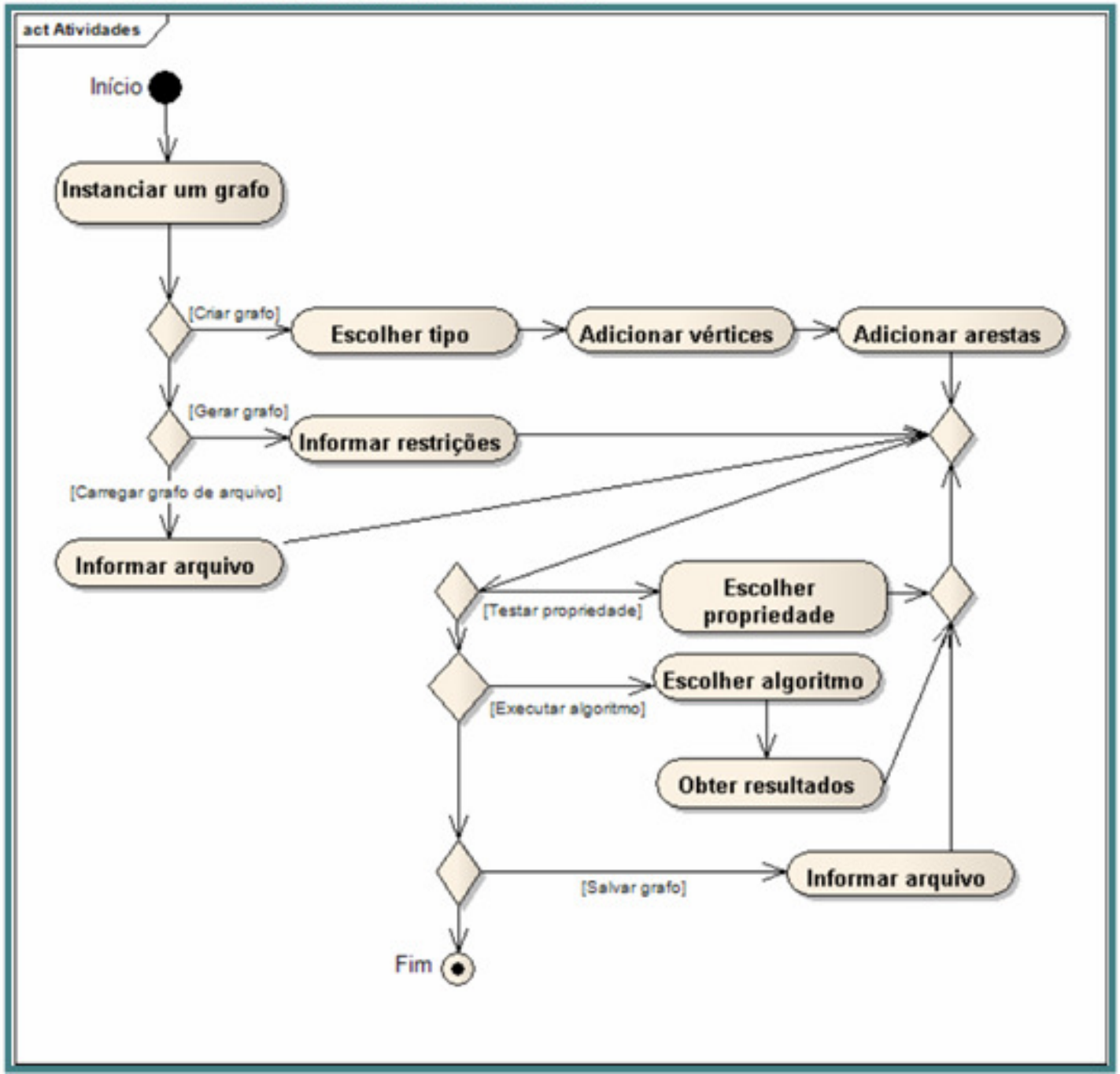
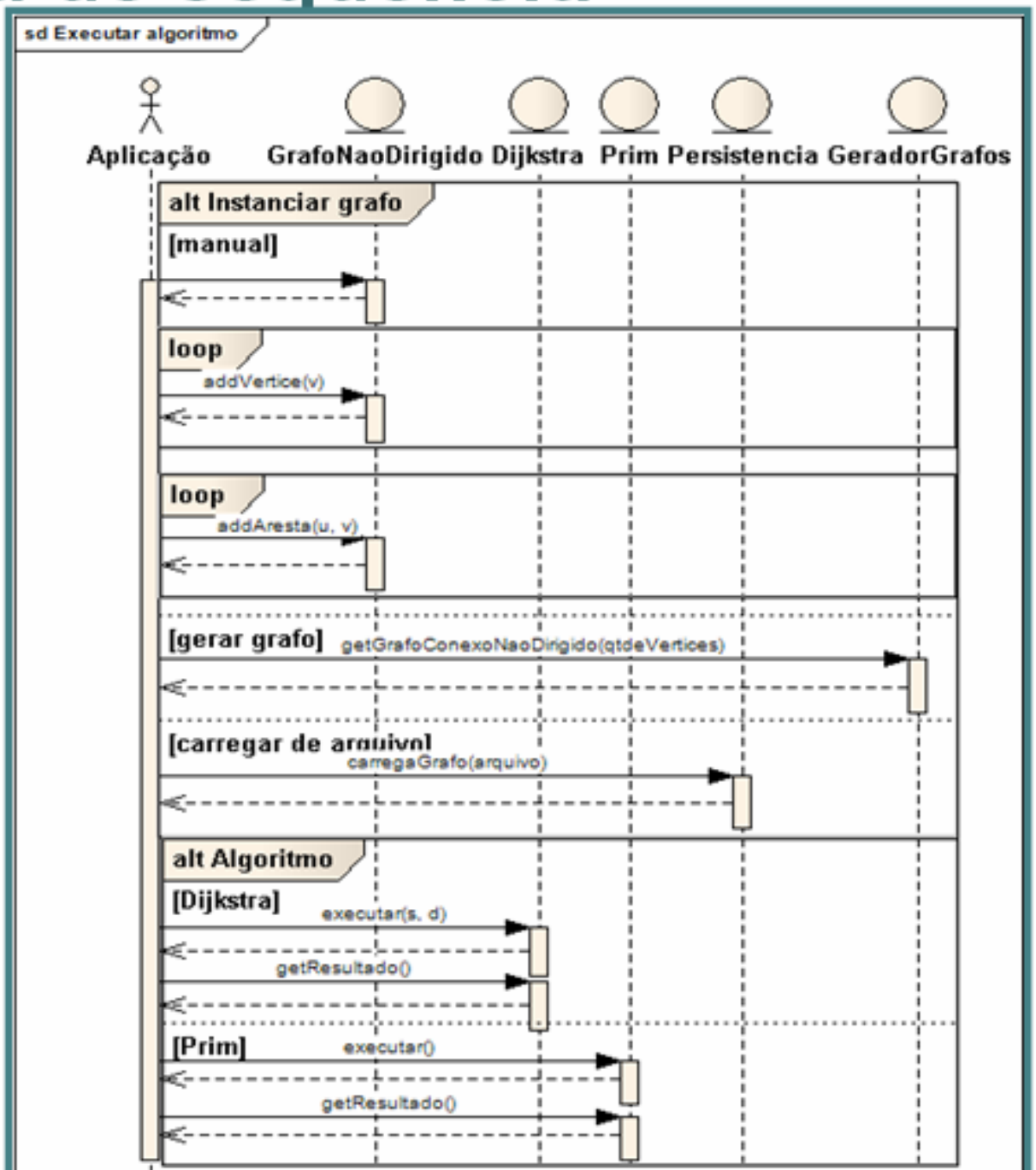


Diagrama de sequência



Modelo de persistência

XML

```
-<Grafo>
   <Tipo>2</Tipo>
  -<Vertices>
   -<Vertice>
      <Id>9</Id>
    -<Cor>
       <R>255</R>
       <G>100</G>
       <B>100</B>
     </Cor>
     <Dado>Vertice 9</Dado>
   </Vertice>
  +<Vertice></Vertice>
   </Vertices>
  -<Arestas>
   -<Aresta>
      <Id>1</Id>
      <Vi>1</Vi>
      <Vj>2</Vj>
      <Valor>4.23</Valor>
      <Capacidade>1.0</Capacidade>
      <Dado>Aresta 1</Dado>
   </Aresta>
  +<Aresta></Aresta>
   </Arestas>
 </Grafo>
```

Implementação

Técnicas e ferramentas utilizadas

- Java versão 6
- Fedora Eclipse 3.4.1
- DOM
- JavaDoc

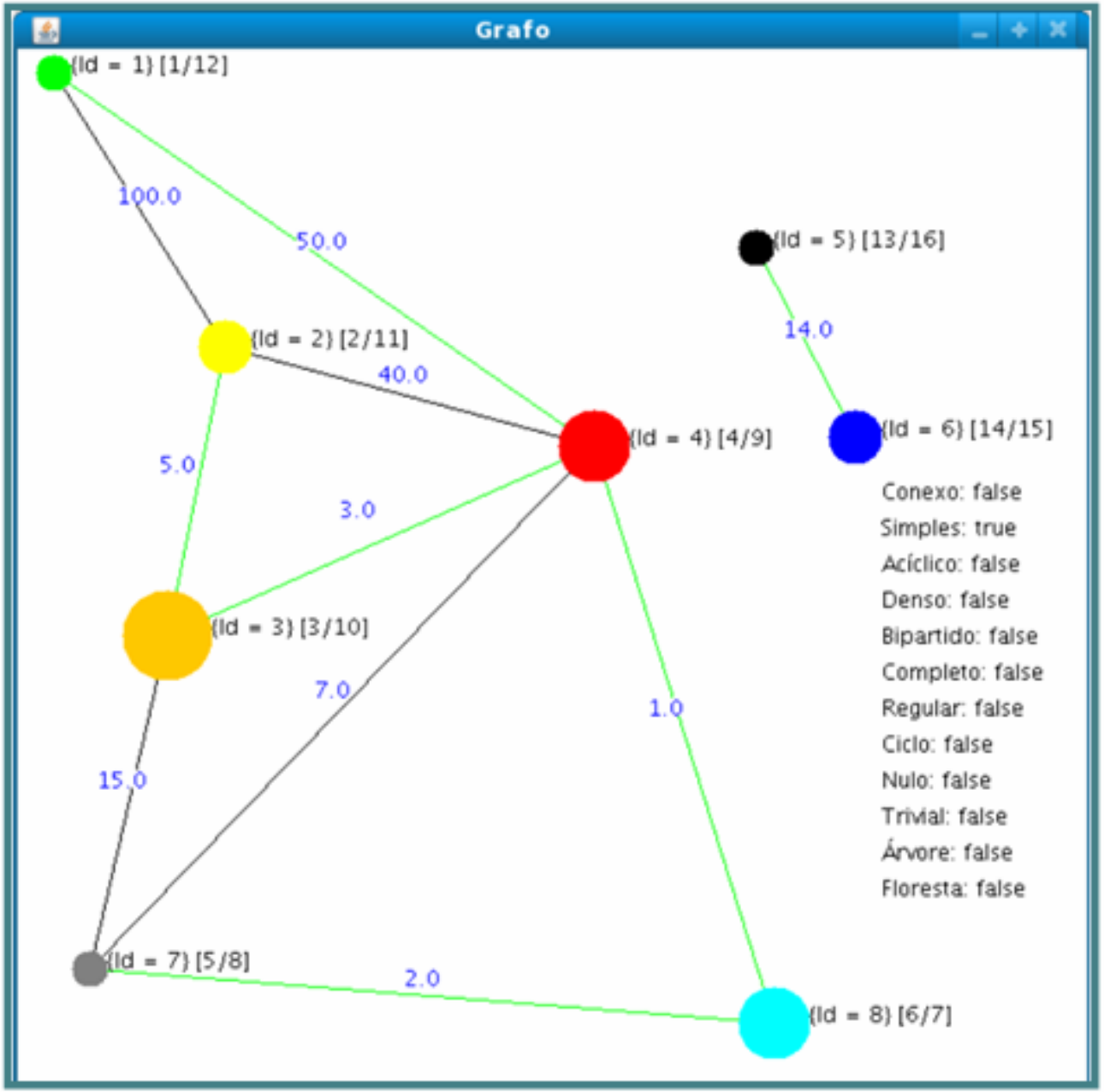
Implementação

- Estrutura do grafo
- Persistência
- Algoritmos de grafos
- Geradores de grafos
- Aplicação de exemplo

Operacionalidade

- Estender classe Vertice;
- Instanciar grafo;
- Criar vértices e arestas;
- Executar algoritmos;
- Obter resultados.

Operacionalidade - Demonstração



Resultados e discussões

Comparação entre algoritmos da teoria dos grafos

O algoritmo ideal para cada tipo problema

Implementação de parte do *framework* na linguagem Objective-C

Resultados e discussões

Vértices	Dijkstra Tempo (ms)	Bellman-Ford Tempo (ms)	Floyd-Warshall Tempo (ms)
10	3	2	11
30	9	15	53
60	42	69	137
100	46	54	446
200	80	94	3775
500	1133	4531	77766

Resultados e discussões

Vértices	Consultas	Dijkstra Tempo (ms)	Bellman-Ford Tempo (ms)	Floyd-Warshall Tempo (ms)
10	5	4	1	6
30	15	107	40	56
60	30	201	47	171
100	50	631	65	439
200	100	6653	233	3882
500	250	158305	2318	75219

Resultados e discussões

Vértices	Consultas	Dijkstra Tempo (ms)	Bellman-Ford Tempo (ms)	Floyd-Warshall Tempo (ms)
10	5	8	11	13
30	15	79	77	57
60	30	214	326	151
100	50	1014	1657	436
200	100	13870	36652	4352
500	250	546846	1908240	85741

OBJECTIVE-C

ALGORITMO DE PRIM

```
#ifndef _ALGORITMOPRIM_
#define _ALGORITMOPRIM_
#import <Foundation/Foundation.h>
#import <Foundation/NSArray.h>
#import "AlgoritmoPrimResultado.h"
#import "GrafoNaoDirigido.h"
#import "PriorityQueue.h"

@interface AlgoritmoPrim: NSObject {
@private
    AlgoritmoPrimResultado* resultado;
    NSMapTable* naArvore;
    PriorityQueue* fila;
}
-(AlgoritmoPrimResultado*) getResultado;
-(void) executar: (GrafoNaoDirigido*) g;
@end
#endif
```

```
#import "AlgoritmoPrim.h"
#import "Constante.h"

@implementation AlgoritmoPrim
-(AlgoritmoPrimResultado*) getResultado {
    return resultado;
}

-(void) executar: (GrafoNaoDirigido*) g {
    int tamanhoGrafo = [g getTamanho];
    resultado = [[AlgoritmoPrimResultado alloc] init];
    naArvore = [NSMapTable mapTableWithStrongToStrongObjects];
    fila = [[PriorityQueue alloc] init: tamanhoGrafo];
```

Comparativo

	JGRAPHT	HACKBART	BRAUN	FRAMEWORK
Criação de grafos				
Estender classes				
Gerar grafos				
Persistir grafos				
Verificar propriedades				
Disponibiliza algoritmos				
Permite criar novos algoritmos				
Permite acompanhar execução de algoritmos				

Não possui a funcionalidade

Possui a funcionalidade

Conclusão

- Requisitos foram cumpridos;
  - **Novos recursos adicionados**
- Comparação de desempenho dos algoritmos;
- Criação de instâncias de grafos com muitos vértices e arestas;
- Verificação de propriedades dos grafos;
- Possibilidade de persistência dos grafos;
  - **DOM**
- Documentação gerada.

Conclusão

Limitações

- Baixo desempenho para trabalhar com grafos grandes;
- Muito consumo de memória para grafos com muitos vértices e arestas;
- Não está construída na aplicação de exemplo uma forma de ter uma representação visual do grafo criado.

Extensões

- Verificar outras propriedades de grafos: cordal, hipercubo, perfeito, cactos, planar, isomorfo a outro grafo;
- Gerar os tipos de grafos acima comentados;
- Implementar recursos para trabalhar com o grafo em modo visual;
- Possibilitar a exportação do grafo para outros formatos;
- Implementar outros algoritmos da teoria dos grafos: emparelhamento perfeito, clique máximo, ciclo hamiltoniano, ciclo euleriano, relabel-to-front, Boruvka.