Biblioteca de Grafos

Parte 2

Teoria dos Grafos
2014/2

Patrícia Kovaleski
Victor Maia

Programação Funcional para Dijkstra e Prim

- Prim e Dijkstra possuem códigos muito semelhantes;
- Utilizar funções de alta ordem para instanciar duas funções baseadas em um único modelo.

```
Graph.prototype.walk = function(distance){
    // Walks through a graph, visiting the nodes, ordered by distance.
    return function(n){
        this.clearState();
        for (var i = 1; i <= this.size; ++i)
            this.distance[i-1] = Infinity;
        this.distance[n-1] = 0;
        this.parent[n-1]   = 0;
        this.pqueue.add(0,n);
        while (node = this.pqueue.get()){
            if (this.marked[node-1]) continue;
            this.marked[node-1] = 1;
            var neighbors     = this.neighbors(node);
            var weights       = this.weights(node);
            for (var i=0, l=neighbors.length; i<l; ++i){
                var neig = neighbors[i];
                var weig = weights[i];
                if (!this.marked[neighbors[i]-1]){
                    var dist = distance.call(this,node,neig,weig);
                    if (this.distance[neig-1] > dist){
                        this.distance[neig-1] = dist;
                        this.parent[neig-1]   = node;
                        this.pqueue.add(dist,neig); }; }; }; }; }; };
```

- Com a função "*walk*" implementada, dijkstra e prim diferem em apenas uma linha:

```
Graph.prototype.dijkstra = Graph.prototype.walk(function(node,neig,weig){
   return this.distance[node-1] + weig;
});
```

```
Graph.prototype.prim = Graph.prototype.walk(function(node,neig,weig){
   return weig;
});
```

- Reutilização de código!

Resultados

	Memória Utilizada	Peso total da MST	Distância Média	Tempo de execução
Grafo 1	2.625 MBs	336	13.0192	0.009s
Grafo 2	63.289 MBs	999	2.0859	9.897s
Grafo 3	27.922 MBs	31947	16.0108	53.232s
Grafo 4	81.4141 MBs	216236	24.6289	1265.527s (aprox. 21min)
Grafo 5	97.891 MBs	608677	57.1552	4246.223s (aprox 1h 10min)