Aluno: Lenington do C. Rios, 13211187

1 Prim's

Algorithm 1: prims(G)

```
input  : The original graph G
output: The minimum spanning tree of G (SPT)
SPT ← smallestEdge(G);
while |edges(SPT))|/2 < |vertices(G)| − 1 do
    cost ← maxIntNumber;
    for v in vertices(SPT) do
        for e in adjacents(G, v) do
            if getEdge(G, v, e) < cost and e ∉ vertices(SPT) then
                vertex ← v;
                minAdjascent ← e;
                cost ← getEdge(G, v, e);
            end
        end
    end
    addVertex(SPT, minAdjascent);
    addEdge(SPT, vertex, minAdjascent, cost);
end
return SPT;
```

2 Kruskal's

Algorithm 2: kruskals(G)

input : The original graph G
output: The minimum spanning tree of G (SPT)

$SPT \leftarrow smallestEdge(G)$;
$H = MinHeap()$;
$H \leftarrow H + smallestEdge(G)$;
$edgesCycles \leftarrow set()$;
while $|edges(SPT))|/2 < |vertices(G)| - 1$ **do**
    $(o, d, cost_{(o,d)}) \leftarrow H.pop()$;
    if $(o, d) \notin edges(SPT)$ **then**
        if $o \in vertices(SPT)$ **then**
            $path \leftarrow path(SPT, o)$;
            for p ***in*** $path$ **do**
                if $p \in d$ **then**
                    $edgesCycles.add((o, d))$
                end
            end
        end
        if $(o, d) \notin edgesCycles$ ***or*** $(o, d) \notin vertices(SPT)$ **then**
            $addVertexs(o, d)$;
            $addEdge(SPT, o, d, cost_{(o,d)})$;
        end
    end
end
return SPT;