Universidade Estadual de Maringá - Ciência Da Computação (DIN)
Aluno: Otávio Hideki Gonçalves Kochi RA:107635
Aluno: Thiago Issao Yasunaka RA:103069

Trabalho1 – Algoritmo em Grafos 6898
Árvores Aleatórias

Parte 3 – Função randomTreeKruskal(T)
e
MST-Kruskal()

IMPLEMENTACÃO UTILIZANDO A LINGUAGEM JAVASCRIPT

Repositório no GitHub:
https://github.com/thiagoissao/arvore_aleatorias_grafos/blob/master/kruskal.js

Código:

```
const assert = require('assert')

const make_set = (v, adj) => {
    return {
        E: adj,
        rank: 0,
        p: v
    }
}

const union = (x, y, G) => {
    link(find_set(G, x), find_set(G, y), G)
}

const find_set = (G,v) => {
    if(v != G[v].p) G[v].p = find_set(G, G[v].p)
    return G[v].p
}

const link = (x, y, G) => {
    if(G[x].rank  > G[y].rank){
        G[y].p = x
    } else{
        G[x].p = y
        if(G[x].rank == G[y].rank){
            G[y].rank += 1
        }
    }
```

```
}

const mst_kruskal = (G, w) => {
    let A = []
    Object.keys(G).forEach( v => G[v] = make_set(v, G[v]))
    const wTemp = Object.assign({}, w)
    Object.keys(w).forEach( v => {
        w[v].sort((a,b) => {
            if(b < a){
                const iA = wTemp[v].indexOf(a)
                let iB = wTemp[v].indexOf(b)
                iB = iB == -1 ? iA + 1 : iB
                const aux = G[v].E[iB]
                G[v].E[iB] = G[v].E[iA]
                G[v].E[iA] = aux
            }
           return a - b
        })
    })
    let arestaOrdenada = []
    Object.keys(G).forEach(u => {
        G[u].E.forEach(v => {
            u = parseInt(u)
            if(u > v){
                arestaOrdenada.push([w[u][G[u].E.indexOf(v)], u, v])
            }
        })
    })

    arestaOrdenada.sort((a,b) => {
        return a[0] - b[0]
    })

    for(let i = 0; i < arestaOrdenada.length; i ++){
        if(find_set(G, arestaOrdenada[i][1]) != find_set(G,
          arestaOrdenada[i][2])){
            A.push([arestaOrdenada[i][1], arestaOrdenada[i][2]])
            union(arestaOrdenada[i][1], arestaOrdenada[i][2], G)
        }
    }
    return A
}


const kruskal_graph_test = () => {
    const a = 0
    const b = 1
    const c = 2
    const d = 3
    const e = 4
    const f = 5
    const g = 6
    const h = 7
    const i = 8

    const G = [
        [b, h],          //a
        [a, c, h],       //b
        [b, d, i ,f],    //c
        [c, e, f],       //d
        [d, f],          //e
        [c, d, e, g],    //f
        [f, i, h],       //g
        [a, b, i, g],    //h
        [c, g, h],       //i
    ]

    const w = [
        [4, 8],
        [4, 8, 11],
        [8, 7, 2, 4],
        [7, 9, 14],
        [9, 10],
        [4, 14, 10, 2],
        [2, 6, 1],
        [8, 11, 7, 1],
        [2, 6, 7]
    ]

//RETORNA um vetor com as arestas u,v
/*
    Ex: [
        [u0, v0],
        [u1, v1],
        [u2, v2],
        ...
    ]

*/
const A = mst_kruskal(G, w)
    assert (A[0][0] == 7)
    assert (A[0][1] == 6)
    assert (A[1][0] == 6)
    assert (A[1][1] == 5)
    assert (A[2][0] == 8)
    assert (A[2][1] == 2)
    assert (A[3][0] == 1)
    assert (A[3][1] == 0)
    assert (A[4][0] == 5)
    assert (A[4][1] == 2)
    assert (A[5][0] == 3)
    assert (A[5][1] == 2)
    assert (A[6][0] == 2)
    assert (A[6][1] == 1)
    assert (A[7][0] == 4)
    assert (A[7][1] == 3)
}

const createEdges = (current, n) => {
    let e = Array()
    for(let i=0; i < n; i++)
        if(i != current) e.push(i)
    return e
}

const createWeights = (end) => {
    let w = Array()
    for(let i=0; i < end - 1; i++)
        w.push(Math.random())
    return w
}

const randomTreeKruskal = n => {
    const G = Array(n).fill(null)
    G.forEach((element,i) => G[i] = createEdges(i, n))

    const w = Array(n).fill(null)
    w.forEach((element, i) => w[i] = createWeights(n))

    return mst_kruskal(G, w)
}

console.log("randomTreeKruskal -> n = 5: ")
console.log(randomTreeKruskal(5))

console.log("\nrandomTreeKruskal -> n = 10")
console.log(randomTreeKruskal(10))

kruskal_graph_test()
```