UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Seminario Para la Solución de Problemas de Algoritmia

Reporte de práctica

Nombre del alumno:	Carlos Uriel Salcido Aviña
Profesor:	Erasmo Gabriel Martínez Soltero
Título de la práctica:	"Tarea 9. Algoritmo de Prim"
Fecha:	4 de mayo 2022

Introducción

El objetivo es trazar lineas uno de los archivos de mapas proporcionados por el profesor para implementarle el algoritmo de Prim. Antes de esto, se deben detectar las esquinas de las calles, marcar con un punto azul, y ahora sí aplicar Prim, con la condición de que los cables no pueden atravesar los edificios, y así conseguir un cableado mas eficiente.

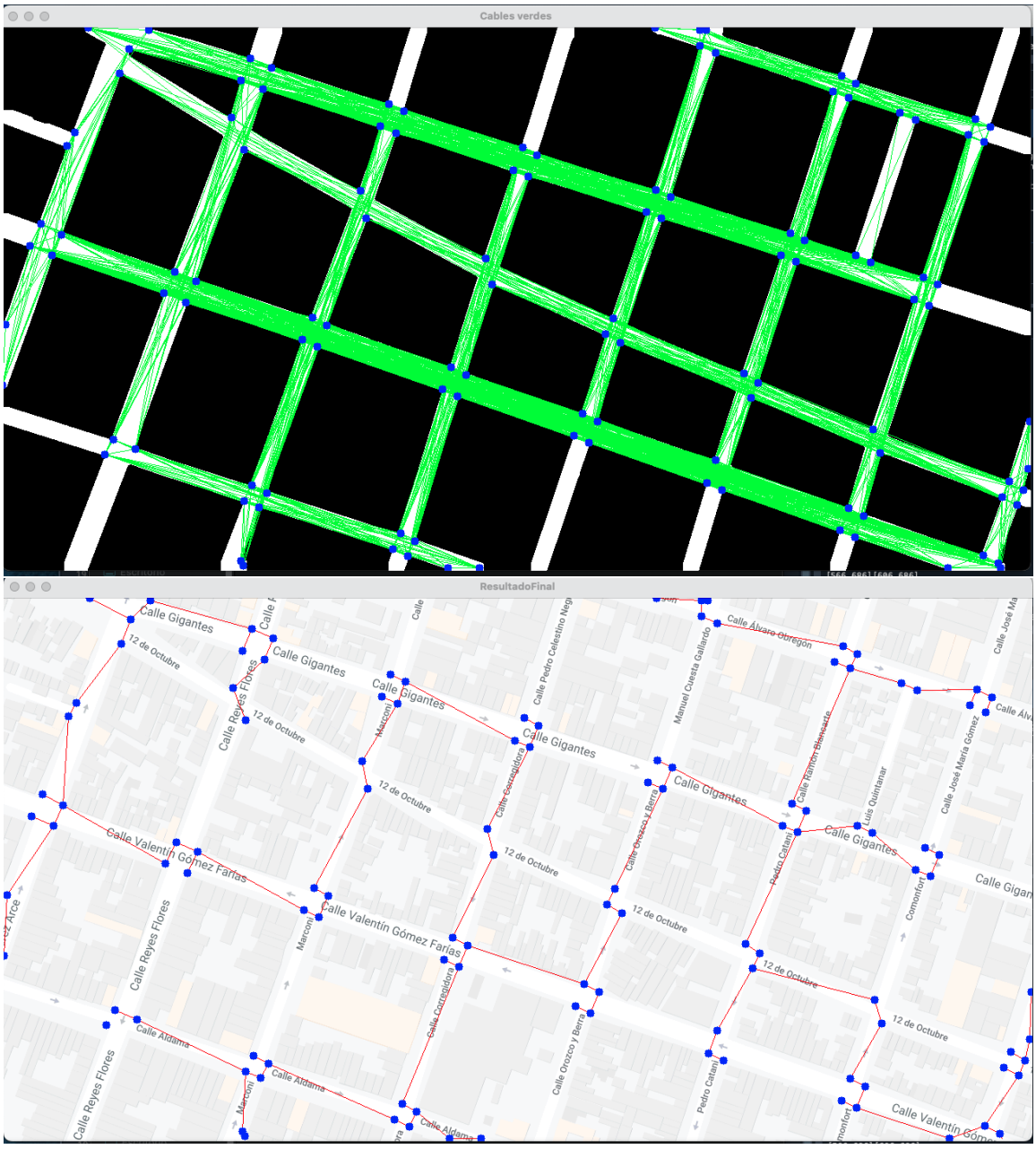
Metodología

Para cumplir el objetivo de la práctica, desarrollé el siguiente código:

```
    for arista in aristas:
        cv2.line(th2, tuple(arista[0]), tuple(arista[1]), (0,255,0), 1)

        #aqui pinto los puntos de las esquinas que son circulos de de radio de 5 pixeles, el -1 indica que van rellenados
        #point tiene la forma [fila, columna]
    for point in vertices:
        cv2.circle(th2,(point[0], point[1]), 5, (255,0,0), -1)
        cv2.waitKey(1)

    cv2.imshow('Cables verdes',th2)


    #Aplicar el algoritmo de Prim
    grafo = []
    listaVisitados = []
    listaVisitados.append(verticesConectados[0])
    while len(listaVisitados)!=len(verticesConectados):
        aristasNew=[]
        for a in listaVisitados:
            for arista in aristas:
                if np.array_equal(a,arista[0]) and (not isInTheList(arista[1],listaVisitados)):
                    aristasNew.append([a,arista[1],arista[2]])
                elif np.array_equal(a,arista[1]) and not isInTheList(arista[0],listaVisitados):
                    aristasNew.append([a,arista[0],arista[2]])
        edge = sorted(aristasNew,key=lambda element:element[2])[0]
        grafo.append(edge)

        listaVisitados.append(edge[1])

    for arista in grafo:
        cv2.line(mapa, tuple(arista[0]), tuple(arista[1]), (0,0,255), 1)

    for point in verticesConectados:
        cv2.circle(mapa,(point[0], point[1]), 5, (255,0,0), -1)
    cv2.imshow('ResultadoFinal',mapa)
    break

#aqui muestro como quedo de chingon el grafo
#cv2.imshow('points',th2)
```

```
for k in range(h,len(aux2)):
    j=aux2[k]
    if not (i==j).all():
        cont+=1
        print(cont)

        print(i,end='')
        print(j)

        promx = (i[0] + j[0]) / 2
        #int(promx)

        promy = (i[1] + j[1]) /2
        #int(promy)

        #prom = promx, promy
        #print(cont)

        x1 = int((i[0] + promx)/2)
        y1 = int((i[1] + promy)/2)
        x2 = int((i[0] + x1)/2)
        y2 = int((i[1] + y1)/2)

        x3 = int((j[0] + x2)/2)
        y3 = int((j[1] + y2)/2)
        x4 = int((j[0] + x3)/2)
        y4 = int((j[1] + y3)/2)


        m = th2[int(promy)][int(promx)]
        m1 = th2[y1][x1]
        m2 = th2[y2][x2]
        m3 = th2[y3][x3]
        m4 = th2[y4][x4]

        #Verificar si son blancos
        if((m[0]>250 and m[1]>250 and m[2]>250) and (m1[0]>250 and
            m1[1]>250 and m1[2]>250) and (m2[0]>250 and m2[1]>250 and
            m3[2]>250) and (m3[0]>250 and m3[1]>250 and m3[2]>250) and
            (m4[0]>250 and m4[1]>250 and m4[2]>250)):

            distancia = int(sqrt((i[0] - j[0])**2 + (i[1] - j[1])**2))
            aristas.append((i,j,distancia))

            if not isInTheList(i,verticesConectados):
                verticesConectados.append(i)

                if not isInTheList(j,verticesConectados):
                    verticesConectados.append(j)


        """
        print (str(th2[int(promx), int(promy)]))

        if( (str(th2[int(promx * .25), int(promy * .25)])) == 0,0,0 and
           (str(th2[int(promx * .5), int(promy * .5)])) == 0,0,0 and
```

Resultados



Conclusiones

Este reporte es una corrección del primero que envié, puesto que no me salió la primera vez que hice la tarea.

Referencias

La clase impartida por el profesor.