map the internet based on if i get a ping

- map maker v3.py (functioning?) last ping 1.0.212.120 white

map maker v3.py (functioning?) last ping 1.0.212.120 white

```
import os
import socket
import struct
from ping3 import ping
from PIL import Image

print("start running")



def ip_to_int(ip):
    int_ip = struct.unpack("!I", socket.inet_aton(ip))[0]
    return int_ip
def int_to_ip(i):
    ip = socket.inet_ntoa(struct.pack("!I", i))
    return ip
def hilbert_curve(n):
    points = [(0, 0)]
    for i in range(n):
        gray = [((k >> i) ^ (k >> (i + 1))) & 1 for k in range(2 ** i)]
        points = _rot(points, gray)
    return points

def _rot(points, gray):
    rot = [(1, 0), (0, 1), (-1, 0), (0, -1)]
    last = points[-1]
    for code in gray:
        last = (last[0] + rot[code][0], last[1] + rot[code][1])
        points.append(last)
    return points

def ping_ip(ip):
    try:
        response_time = ping(ip, timeout=1)
        return 1 if response_time is not None else 0
    except OSError:
```

```
        return -1

def main():
    img_size = 65536
    n = 16
    print(f"img size ={img_size}, n={n} Creating a blank image with a white background")

    img = Image.new("1", (img_size, img_size), color="grey")
    pixels = img.load()
    curve_filename = "hilbert_curve v2.txt"
    if os.path.exists(curve_filename):
        with open(curve_filename, "r") as f:
            print("reading Hilbert curve coordinates file")
            curve_points = [tuple(map(int, line.strip().split())) for line in f.readlines()]
    else:
        print("Generating Hilbert curve coordinates file")
        curve_points = hilbert_curve(n)
        with open(curve_filename, "w") as f:
            for x, y in curve_points:
                f.write(f"{x} {y}\n")

    ip_range = 2 ** 32
    start_ip_int = 0
    if os.path.exists('ping_status.txt'):
        with open('ping_status.txt', 'r') as f:
            lines = f.readlines()
            if len(lines) > 0:
                last_line = lines[-1].strip()
                last_ip, last_result = last_line.split()
                start_ip_int = ip_to_int(last_ip) + 1
                print(f"Resuming from IP address {last_ip}, result = {last_result}")
    with open('ping_status.txt', 'a') as f:
        print("pinging and writing to ping status.txt")
        for i in range(start_ip_int, ip_range):
            ip = int_to_ip(i)
            x, y = curve_points[i % len(curve_points)]
            result = ping_ip(ip)
            print(f"result {result} for ip {ip}")
            if result == -1:
                f.write(f"{ip} black\n")
                pixels[x, y] = 0
            else:
                pixels[x, y] = result
                if result == 0:
                    f.write(f"{ip} white\n")
                else:
                    f.write(f"{ip} black\n")
```

```
            if (i + 1) % (2 ** 24) == 0:
                img.save("ping_map.png")

    img.save("ping_map.png")
    print("Finished pinging IP addresses and saved final image")
if __name__ == '__main__':
    main()
```