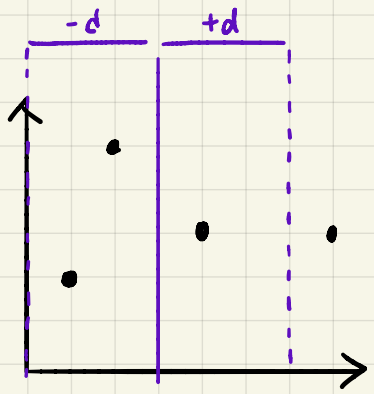
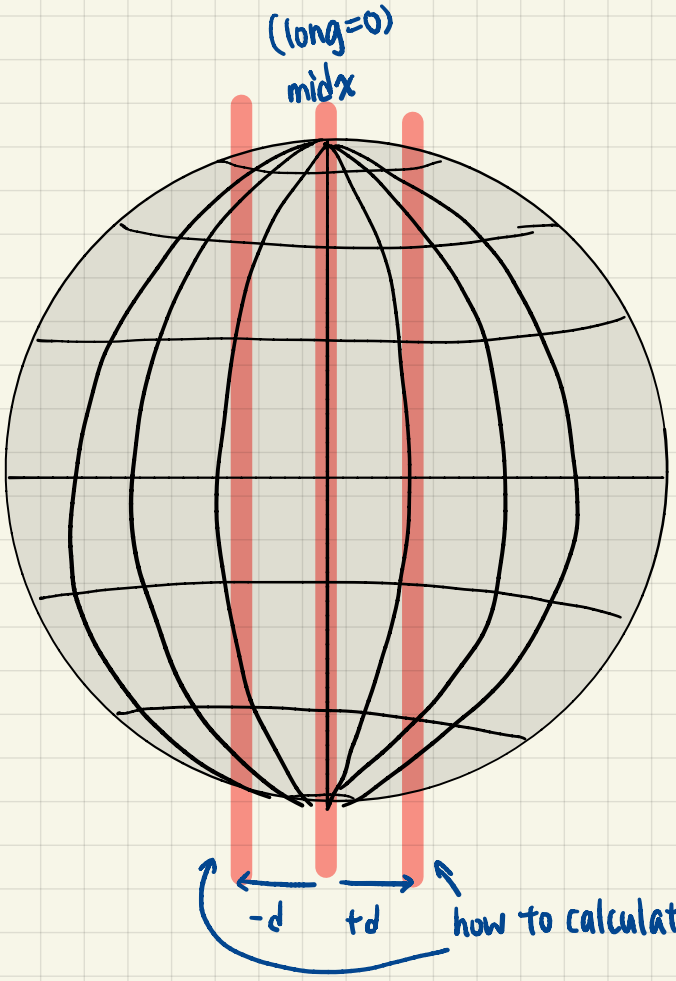
Closest Pair (2D)

pts = [[1,2], [2,5], [4,3], [7,3]]




```
function Closest_pair ( pts ):
    if len(pts) <= 3:
        return smallest dis
    else:
        left = pts [:len(pts)]
        right = pts [len(pts):]
        // recursive
        disleft = Closestpair (left)
        disright = Closestpair (right)
⊛       d = min (disleft, disright)
        // handle mid area.
        construct area of +d & -d base on mid x
        compare their distance
        ⋮
```

(divide)
// cut input in half and handle seperately.



how to calculate the long lat on the line?

(-90 ~ 90) y-axis (-180 ~ 180) x-axis

For (lat, long)

we get distance using

Haversine formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km); note that angles need to be in radians to pass to trig functions!

JavaScript:

```
const R = 6371e3; // metres
const φ1 = lat1 * Math.PI/180; // φ, λ in radians
const φ2 = lat2 * Math.PI/180;
const Δφ = (lat2-lat1) * Math.PI/180;
const Δλ = (lon2-lon1) * Math.PI/180;

const a = Math.sin(Δφ/2) * Math.sin(Δφ/2) +
          Math.cos(φ1) * Math.cos(φ2) *
          Math.sin(Δλ/2) * Math.sin(Δλ/2);
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

const d = R * c; // in metres
```

Ex: At ⊛, d = 6000 km

how to construct the mid area (+d, -d) to compare?