به نام خدا

پروژه درس شبکه های تلفن همراه

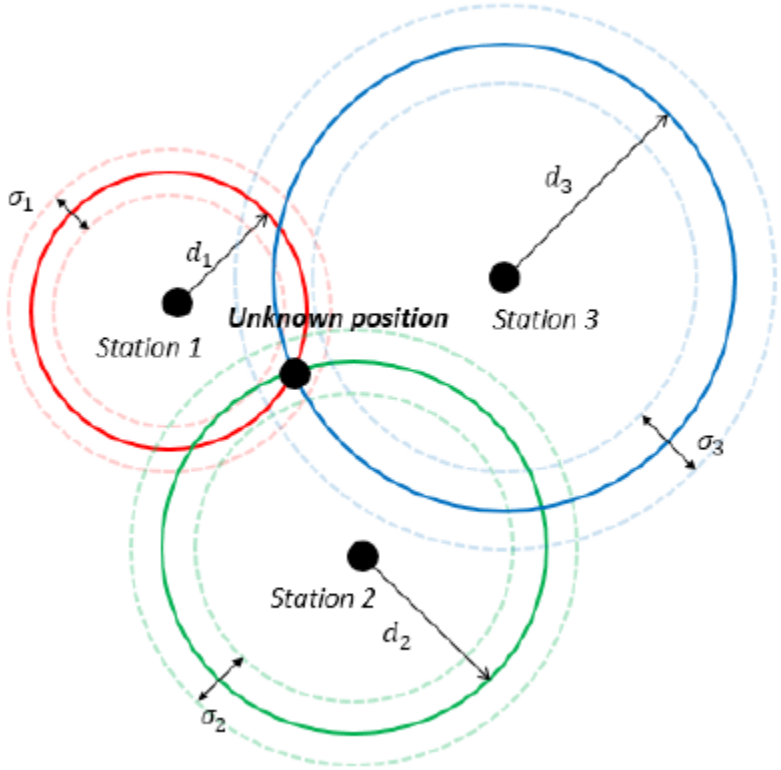
سید یاسین قرشی کرانی _ هانیه حق شناس

پروژه اسمیم:

هدف پروژه ساخت یک برنامه اندروید بود که میتواند سلول های شبکه تلفن همراه را با استفاده از الگوریتم پهلوبندی دایره اس پیدا کند .

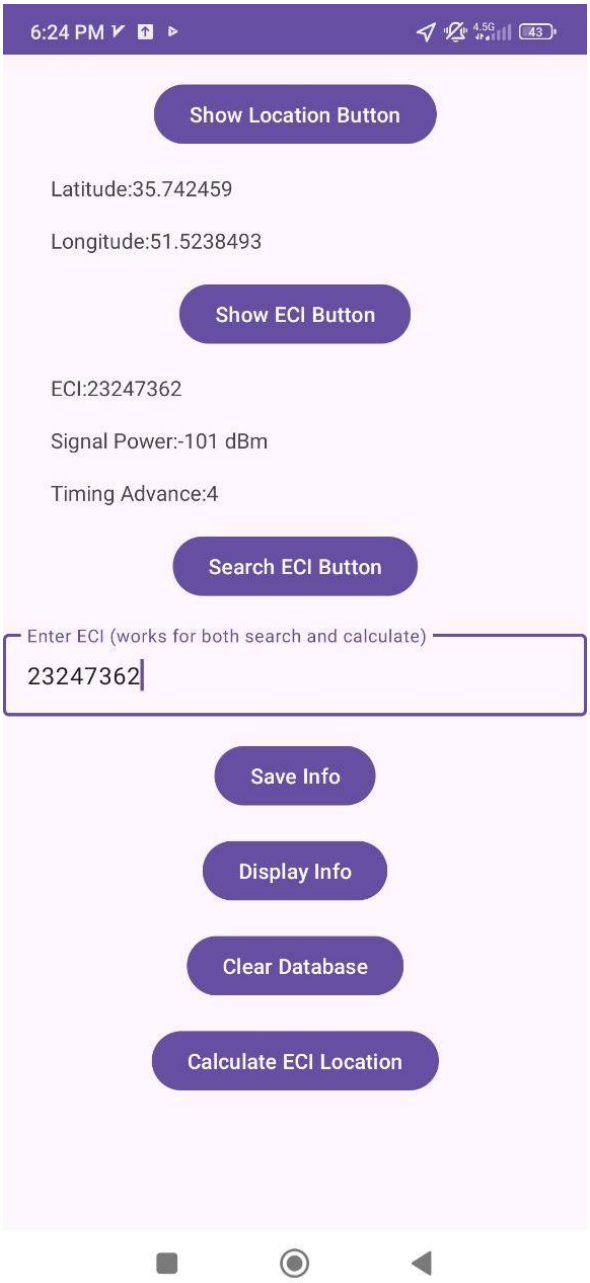
پهلوبندی دایره ای :

فرض میکنیم مختصات 3 نقطه a b c و همچنین فاصله آن ها از نقطه n را داریم . مختصات نقطه n را نمیدانیم



مطابق شکل 3 دایره به شعاع فاصله هر نقطه از n به مرکزیت همان نقطه میکشیم و نقطه تداخل آن ها برابر با موقعیت مکانی نقطه n یا همان سلول دلخواه است.

طراحی برنامه :



- مطابق تصویر دکمه اول (show location butten) مختصات کاربر را گرفته و نمایش میدهد

```
}

private fun getLocation() {
    locationManager = getSystemService(LOCATION_SERVICE) as LocationManager

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSI
        ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMIS

        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  minTimeMs: 0L,  minDistanceM: 0f,  listener: this)
        locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,  minTimeMs: 0L,  minDistanceM: 0f,  listener: this)
    }
}
```

- دکمه (show eci buttun) شماره eci سلول خدمتگزار و قدرت سیگنال و timing advance سلولی که کاربر به ان متصل است را نمایش میدهد (اطلاعات مربوط به سلول lte هستند)

```
private fun getCellInfo() {
    val telephonyManager = getSystemService(TELEPHONY_SERVICE) as TelephonyManager

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERM
        val cellInfoList = telephonyManager.allCellInfo

        for (cellInfo in cellInfoList) {
            if (cellInfo is CellInfoLte) {
                val cellIdentityLte = cellInfo.cellIdentity
                val eci = cellIdentityLte.ci
                val cellSignalStrengthLte = cellInfo.cellSignalStrength
                val rsrp = cellSignalStrengthLte.rsrp
                val timingAdvance = cellSignalStrengthLte.timingAdvance

                eciTextView.text = eci.toString()
                signalPowerTextView.text = "$rsrp dBm"
                taTextView.text = timingAdvance.toString()
                break
            }
        }
```

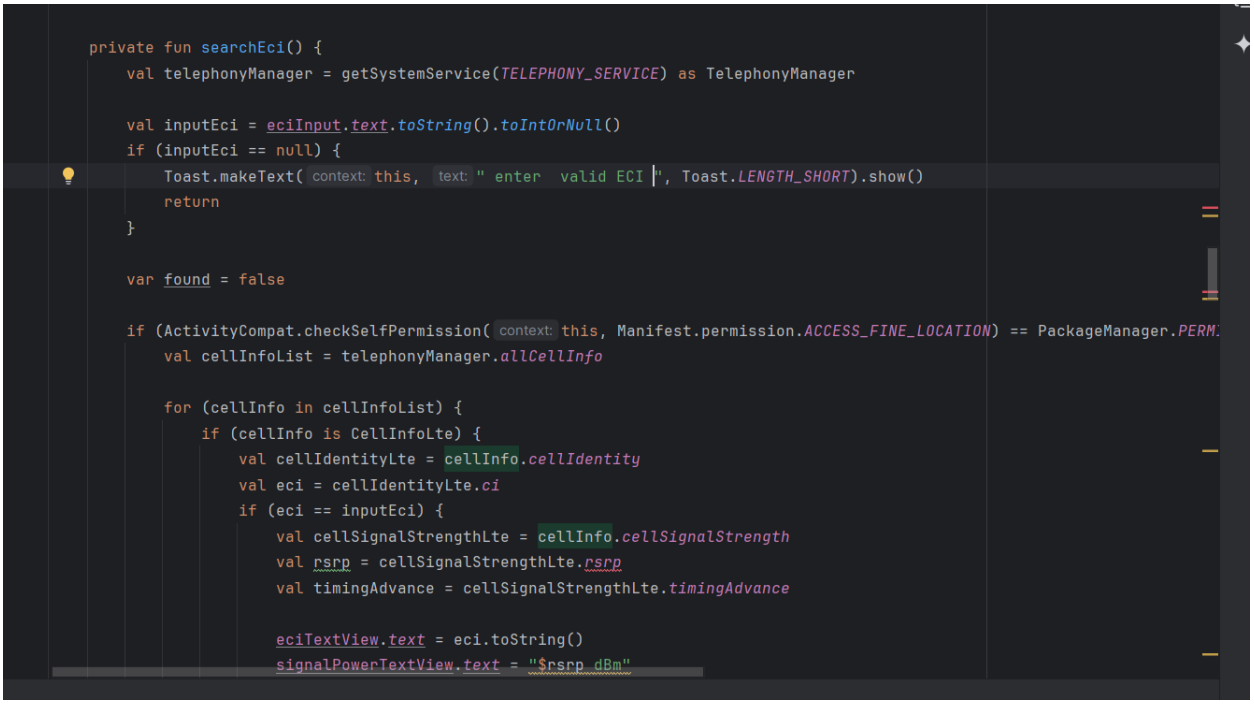
-

Timing advance چیست :

مدت زمانی که طول میکشد تا ارتباط بین کاربر و سلول برقرار شود (یک رفت و برگشت)

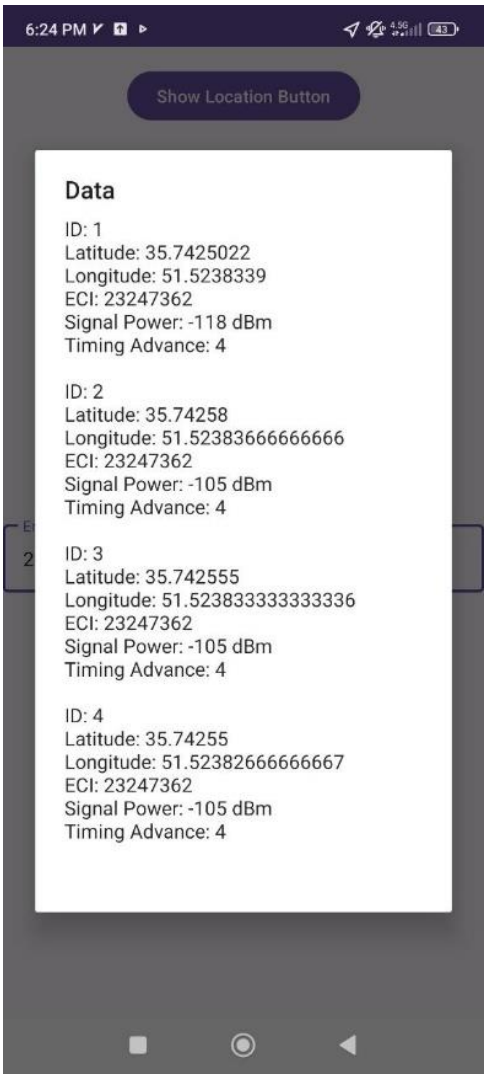
که از آن برای محاسبه فاصله استفاده کردم (با توجه به سرعت نور هر واحد آن برابر با 78 متر است که این برای یک رفت و برگشت است. در نتیجه فاصله تقریبی برابر با 39 متر براورد میشود)

- دکمه search eci buttun یک eci از کاربر گرفته و درصورت وجود سلول در سلول های همسایه و یا سلول خدمتگزار اطلاعات مربوط به آن را نشان میدهد.



```
private fun searchEci() {
    val telephonyManager = getSystemService(TELEPHONY_SERVICE) as TelephonyManager

    val inputEci = eciInput.text.toString().toIntOrNull()
    if (inputEci == null) {
        Toast.makeText( context: this,  text: " enter  valid ECI ", Toast.LENGTH_SHORT).show()
        return
    }

    var found = false

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERM
        val cellInfoList = telephonyManager.allCellInfo

        for (cellInfo in cellInfoList) {
            if (cellInfo is CellInfoLte) {
                val cellIdentityLte = cellInfo.cellIdentity
                val eci = cellIdentityLte.ci
                if (eci == inputEci) {
                    val cellSignalStrengthLte = cellInfo.cellSignalStrength
                    val rsrp = cellSignalStrengthLte.rsrp
                    val timingAdvance = cellSignalStrengthLte.timingAdvance

                    eciTextView.text = eci.toString()
                    signalPowerTextView.text = "$rsrp dBm"
```

- دکمه save info اطلاعات استخراج شده (مکان کاربر و ta) را در پایگاه داده ذخیره میکند
- دکمه display info هم این اطلاعات را نشان میدهد :



- دکمه calculate location موقعیت مکانی سلول با eci داده شده (در صورت وجود حداقل داده موجود) با استفاده از الگوریتم پهلوبندی دایره ای (Circular-lateration) را محاسبه میکند.

داده های مربوط به eci داده شده (مکان کاربر و timing advance) در یک لیست به تابع داده میشود .

```
private fun circularlateration(locations: List<Triple<Double, Double, Int>>): Pair<Double, Double> {
    val R = 6371e3

    val (lat1, lon1, q1) = locations[0]
    val (lat2, lon2, q2) = locations[1]
    val (lat3, lon3, q3) = locations[2]
    val dist1=q1*39
    val dist2=q2*39
    val dist3=q3*39
```

سپس index 3 اول از لیست برای استفاده در الگوریتم استخراج میشود.

همینطور فاصله ها به متر تبدیل میشوند.

پس از آن موقعیت های مکانی که به صورت longitude و latitude داده شده اند به رادیان (با استفاده از شعاع زمین) و سپس به مختصات دکارتی تبدیل میشوند :

```
    val radLat1 = Math.toRadians(lat1)
    val radLon1 = Math.toRadians(lon1)
    val radLat2 = Math.toRadians(lat2)
    val radLon2 = Math.toRadians(lon2)
    val radLat3 = Math.toRadians(lat3)
    val radLon3 = Math.toRadians(lon3)


    val x1 = R * cos(radLat1) * cos(radLon1)
    val y1 = R * cos(radLat1) * sin(radLon1)
    val x2 = R * cos(radLat2) * cos(radLon2)
    val y2 = R * cos(radLat2) * sin(radLon2)
    val x3 = R * cos(radLat3) * cos(radLon3)
    val y3 = R * cos(radLat3) * sin(radLon3)
```

پس از آن بر اساس فاصله ها و موقعیت مکانی سیستم معادلات را تشکیل میدهیم و سپس آن را حل میکنیم.

باید معادلات مربوط به موقعیت مکانی نقطه ناشناخته (x, y) را از روی سه نقطه شناخته شده(x1, y1) ، (x2, y2)و (x3, y3) و فاصله‌هایشان به نقطه ناشناخته به دست آوریم.

باید معادله b=Ax را حل کنیم :

- Aماتریسی است که ضرایب متغیرهای x و y را در معادلات خطی نشان می‌دهد.
- Xبردار متغیرها است (یعنی مختصات x و y نقطه ناشناخته)
- b بردار ثوابت معادلات است

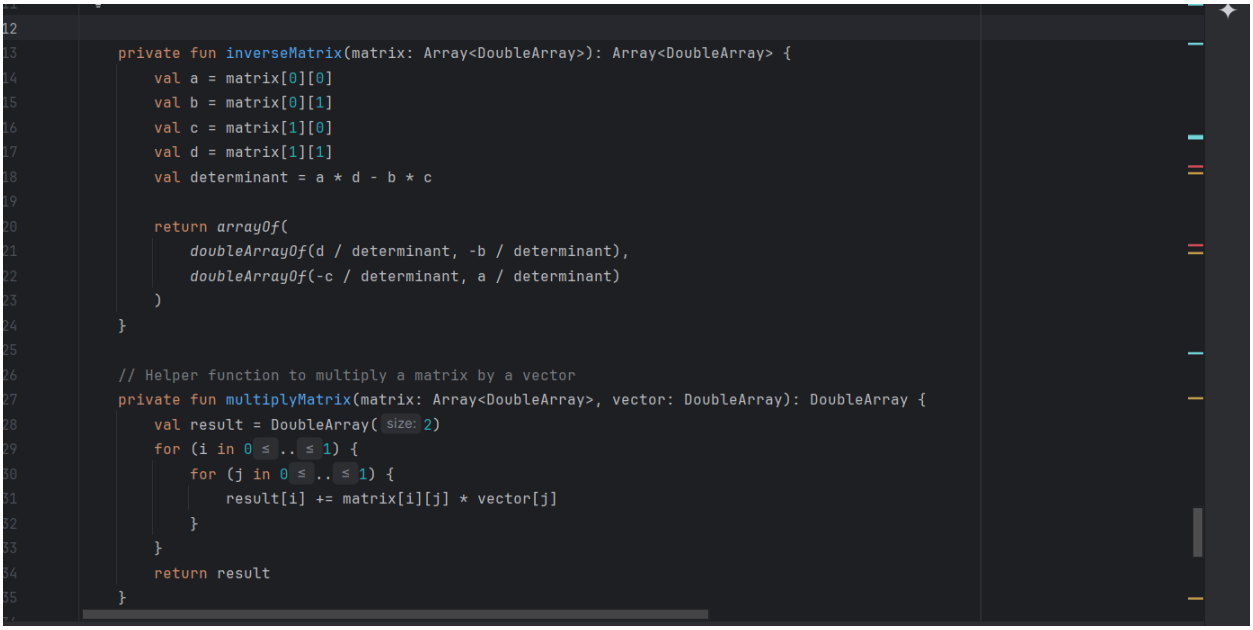
که برای حل آن معکوس ماتریس A را در b ضرب میکنیم.

سپس مختصات بدست آمده را به longitude و latitude تبدیل میکنیم.

:

```
class MainActivity : AppCompatActivity(), LocationListener {
    private fun circularlateration(locations: List<Triple<Double, Double, Int>>): Pair<Double, Double> {
        val x2 = R * cos(radLat2) * cos(radLon2)
        val y2 = R * cos(radLat2) * sin(radLon2)
        val x3 = R * cos(radLat3) * cos(radLon3)
        val y3 = R * cos(radLat3) * sin(radLon3)

        val A = arrayOf(
            doubleArrayOf(2 * (x2 - x1), 2 * (y2 - y1)),
            doubleArrayOf(2 * (x3 - x1), 2 * (y3 - y1))
        )
        val b = doubleArrayOf(
            dist1.toDouble().pow( n: 2) - dist2.toDouble().pow( n: 2) - x1.pow( n: 2) + x2.pow( n: 2) - y1.pow( n: 2) + y2.pow(
            dist1.toDouble().pow( n: 2) - dist3.toDouble().pow( n: 2) - x1.pow( n: 2) + x3.pow( n: 2) - y1.pow( n: 2) + y3.pow( n:
        )

        val Ainv = inverseMatrix(A)
        val result = multiplyMatrix(Ainv, b)

        val x = result[0]
        val y = result[1]

        // Convert Cartesian coordinates back to lat/lon
        val lat = Math.toDegrees(atan2(y, sqrt( x: x.pow( n: 2) + y.pow( n: 2))))
        val lon = Math.toDegrees(atan2(y, x))

        return Pair(lat, lon)
    }
```

توابع کمکی محاسبه معکوس و ضرب ماتریس



```
12
13      private fun inverseMatrix(matrix: Array<DoubleArray>): Array<DoubleArray> {
14          val a = matrix[0][0]
15          val b = matrix[0][1]
16          val c = matrix[1][0]
17          val d = matrix[1][1]
18          val determinant = a * d - b * c
19
20          return arrayOf(
21              doubleArrayOf(d / determinant, -b / determinant),
22              doubleArrayOf(-c / determinant, a / determinant)
23          )
24      }
25
26      // Helper function to multiply a matrix by a vector
27      private fun multiplyMatrix(matrix: Array<DoubleArray>, vector: DoubleArray): DoubleArray {
28          val result = DoubleArray( size: 2)
29          for (i in 0 ≤ .. ≤ 1) {
30              for (j in 0 ≤ .. ≤ 1) {
31                  result[i] += matrix[i][j] * vector[j]
32              }
33          }
34          return result
35      }
```