Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

# Лабораторні роботи №1А, 2А, 3А

З дисципліни «Методи оптимізації та планування»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-93
Євтушок О.М. - 9311

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

Мета роботи:
№1А – ознайомитись з основними принципами розкладання числа на прості множники з використанням різних алгоритмів факторизації.
№2А – ознайомлення з принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрон(Perceptron). Змоделювати роботу нейронної мережі та дослідити вплив параметрів на час виконання та точність результату.
№3А – ознайомлення з принципами реалізації генетичного алгоритму, вивчення та дослідження особливостей даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.

## Код алгоритму до лабораторної роботи №1А:

```
fun lab1(view: View) {
    val txtN = findViewById<EditText>(R.id.editText1)
    val txt1 = findViewById<TextView>(R.id.txt1)
    val txtRes = findViewById<TextView>(R.id.textView2)

    txt1.visibility = View.VISIBLE
    txtRes.visibility = View.VISIBLE

    try{
        val strN: String = txtN.text.toString()
        val N: Int = strN.toInt()

        if(N%2==0){
            txtRes.text = "Число має\nбути непарним"
        } else{

        val sqn = 1+ sqrt(N.toDouble())-sqrt(N.toDouble())%1
        var k =0
        val txt:MutableList<Any> = ArrayList()

        while (true){
            val y2 = (sqn + k).pow(2)-N
            if (sqrt(y2) %1 ==0.0){
                val x = sqn + k
                txtRes.text = " n = %.0f = %.0f² - %.0f² =\n= %.0f * %.0f".format(N *
1.0, x, sqrt(y2 * 1.0), x - sqrt(y2 * 1.0), x + sqrt(y2 * 1.0))
                break
            }
            else{
                k+=1
            }
        }}
    } catch (e: NumberFormatException){txtRes.text = "Дані введені невірно\nабо не
введені\nвзагалі"}

}
```

## Код алгоритму до лабораторної роботи №2А:

```
fun lab2(view: View) {
    val txt = findViewById<TextView>(R.id.txt333)
    val txtRes = findViewById<TextView>(R.id.textView222)

    txt.visibility = View.VISIBLE
    txtRes.visibility = View.VISIBLE
```

```
    try {
        val txtP = findViewById<EditText>(R.id.editText013)
        val txtAx = findViewById<EditText>(R.id.Ax)
        val txtAy = findViewById<EditText>(R.id.Ay)

        val txtBx = findViewById<EditText>(R.id.Bx)
        val txtBy = findViewById<EditText>(R.id.By)

        val txtCx = findViewById<EditText>(R.id.Cx)
        val txtCy = findViewById<EditText>(R.id.Cy)

        val txtDx = findViewById<EditText>(R.id.Dx)
        val txtDy = findViewById<EditText>(R.id.Dy)

        val txtSpd = findViewById<EditText>(R.id.spd1)
        val txtTime = findViewById<EditText>(R.id.time1)
        val txtIt = findViewById<EditText>(R.id.iter1)

        val pStr: String = txtP.text.toString();
        val axStr: String = txtAx.text.toString(); val ayStr: String =
txtAy.text.toString()
        val bxStr: String = txtBx.text.toString(); val byStr: String =
txtBy.text.toString()
        val cxStr: String = txtCx.text.toString(); val cyStr: String =
txtCy.text.toString()
        val dxStr: String = txtDx.text.toString(); val dyStr: String =
txtDy.text.toString()

        val spdStr: String = txtSpd.text.toString()
        val timeStr: String = txtTime.text.toString()
        val itStr: String = txtIt.text.toString()

        val p: Double = pStr.toDouble();
        val ax: Double = axStr.toDouble(); val ay: Double = ayStr.toDouble()
        val bx: Double = bxStr.toDouble(); val by: Double = byStr.toDouble()
        val cx: Double = cxStr.toDouble(); val cy: Double = cyStr.toDouble()
        val dx: Double = dxStr.toDouble(); val dy: Double = dyStr.toDouble()

        val speed: Double = spdStr.toDouble()
        val time: Double = timeStr.toDouble()
        val num: Double = itStr.toDouble()

        val pointLst = mutableListOf(mutableListOf(ax,ay), mutableListOf(bx,by),
mutableListOf(cx,cy), mutableListOf(dx,dy))

        var w1 = 0.0; var w2 = 0.0; var y =0.0
        var iter = 0; var lstIter = 0; var delt=0

        while (true){
            val start = System.currentTimeMillis()
            if (lstIter>=pointLst.size){
                lstIter=0
            }

            val x1 = pointLst[lstIter][0]
            val x2 = pointLst[lstIter][1]

            y = x1*w1 + x2*w2

            delt = (p - y).roundToInt()

            w1 += delt * x1 * speed
```

```
            w2+= delt * x2 * speed

            if ((pointLst[0][0]*w1+pointLst[0][1]>p) and
(pointLst[1][0]*w1+pointLst[1][1]<p) and (pointLst[2][0]*w1+pointLst[2][1]>p) and
(pointLst[3][0]*w1+pointLst[3][1]>p)){
                txtRes.text = String.format("Значення w1 і w2\nотримані за %.1f
ітерацій\nw1 = %.1f\nw2 = %.1f",iter, w1, w2)
                break
            }
            iter+=1
            lstIter+=1
            val timeProgram : Double = (System.currentTimeMillis() - start)/1000.0
            if (timeProgram>=time){
                txtRes.text = String.format("Значення w1 і w2\nотримані за %.1f
секунд\nw1 = %.1f\nw2 = %.1f", timeProgram, w1, w2)
                break
            }
            if (iter>=num){
                txtRes.text = String.format("Значення w1 і w2\nотримані за "+iter+"
ітерацій\n\nw1 = %.1f\nw2 = %.1f", w1, w2)
                break
            }
        }



    } catch (e: NumberFormatException){txtRes.text = "Дані введені невірно\nабо не
введені\nвзагалі"}
}
```

**Код алгоритму до лабораторної роботи №3А:**

```
    fun lab3(view: View) {

        val txt1 = findViewById<EditText>(R.id.editText01)
        val txt2 = findViewById<EditText>(R.id.editText11)
        val txt3 = findViewById<EditText>(R.id.editText12)
        val txt4 = findViewById<EditText>(R.id.editText13)

        val txt5 = findViewById<EditText>(R.id.editText14)

        val txt = findViewById<TextView>(R.id.txt33)
        val txtRes = findViewById<TextView>(R.id.textView22)
        val txtRes2 = findViewById<TextView>(R.id.textView233)

        txt.visibility = View.VISIBLE
        txtRes.visibility = View.VISIBLE
        txtRes2.visibility = View.VISIBLE


        try{

            val table = findViewById<TableLayout>(R.id.tableLayout)
            val table2 = findViewById<TableLayout>(R.id.tableLayout2)
            val table3 = findViewById<TableLayout>(R.id.tableLayout3)
            val table4 = findViewById<TableLayout>(R.id.tableLayout4)
            val table5 = findViewById<TableLayout>(R.id.tableLayout5)
            val table6 = findViewById<TableLayout>(R.id.tableLayout6)
            val table7 = findViewById<TableLayout>(R.id.tableLayout7)

            clearTable(table)
            clearTable(table2)
```

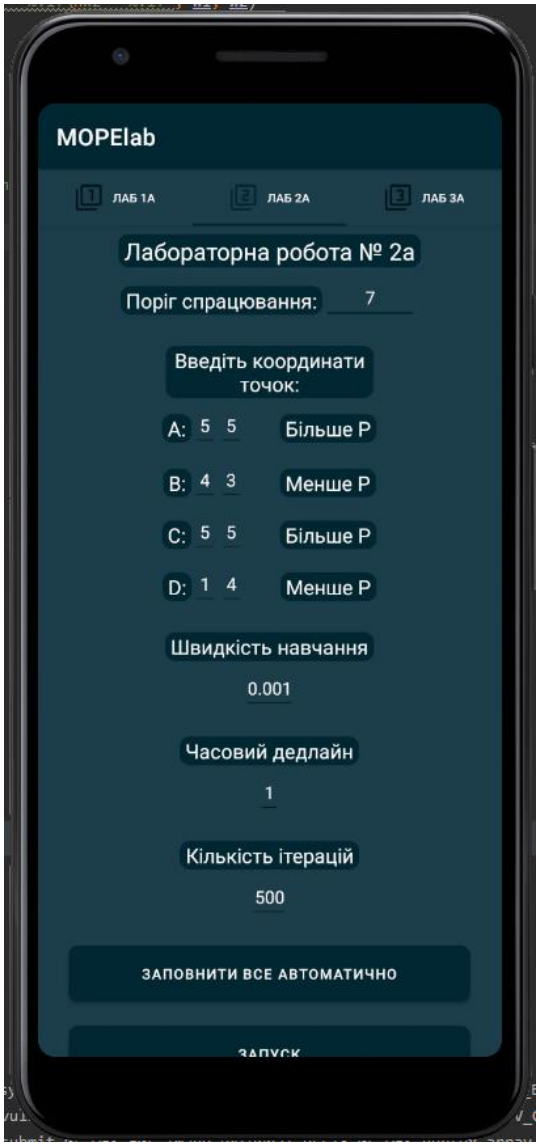
```
            clearTable(table3)
            clearTable(table4)
            clearTable(table5)
            clearTable(table6)
            clearTable(table7)

            val x1Str: String = txt1.text.toString(); val x2Str: String =
txt2.text.toString()
            val x3Str: String = txt3.text.toString() ; val x4Str: String =
txt4.text.toString()

            val yStr: String = txt5.text.toString()

            val x1: Double = x1Str.toDouble(); val x2: Double = x2Str.toDouble()
            val x3: Double = x3Str.toDouble(); val x4: Double = x4Str.toDouble()

            val y: Double = yStr.toDouble()

            var vList:MutableList<Double> = mutableListOf(x1, x2, x3, x4)
            var koefList:MutableList<Double> = ArrayList()
            var percentLst:MutableList<Double> = ArrayList()
            var coupleProb:MutableList<Double> = ArrayList()
            var couples:MutableList<MutableList<Double>> = ArrayList()
            var population:MutableList<MutableList<Int>> = ArrayList()
            var kLst:MutableList<Double> = ArrayList()
            var kLst2:MutableList<Double> = ArrayList()
            var babies:MutableList<MutableList<MutableList<Int>>> = ArrayList()
            var babies2:MutableList<MutableList<Int>> = ArrayList()
            var couples3:MutableList<MutableList<Double>> = ArrayList()
            var abcdLst:MutableList<Int> = ArrayList()

            addRow(table,"Хромосома", "a,b,c,d")
            addRow(table2,"Хромосома\n", "Коефіцієнт\nпристосованості")
            addRow(table3,"Хромосома", "Відповідність")
            addRow(table4,"Хромосома\nбатька", "Хромосом\nматері")
            addRow(table5,"Хромосома\nбатько", "Хромосома\nматір")
            addRow01(table6, "Хромосома-нащадок")
            addRow(table7, "Хромосома\nнащадок", "Коефіцієнт\nпристосованості")

            var k:Double = 1.0
            while (true) {

                vList = mutableListOf(x1, x2, x3, x4)
                koefList = ArrayList()
                percentLst = ArrayList()
                coupleProb = ArrayList()
                kLst=ArrayList()
                kLst2=ArrayList()
                couples = ArrayList()
                babies = ArrayList()
                babies2 = ArrayList()

                population =
(1..5).map{(1..4).map{(1..(y/2).toInt()).random()}.toMutableList()}.toMutableList()

                for(i in 0 until (population.size)){
                    k = 0.0
                    for(j in 0 until (population.size-1)){
                        k += abs(vList[j] * population[i][j])
                    }
                    k-=y
                    k = abs(k)
```

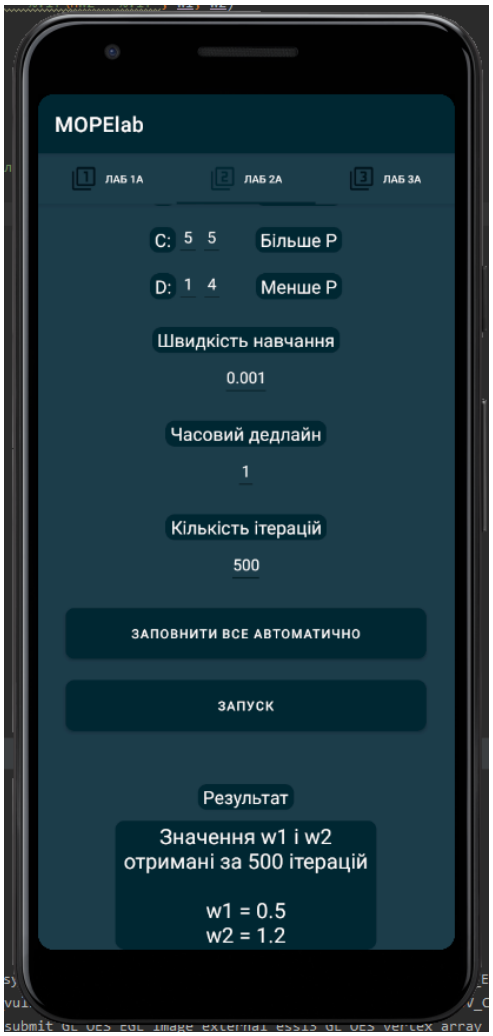
```
                    if (k == 0.0){
                        koefList.add(0.1)
                        kLst.add(0.1)
                    }else{
                        koefList.add(k)
                        kLst.add(k)
                    }
                }

                var koef2 = 0.0
                var percent = 0.0
                for(i in 0 until (koefList.size)){
                    koef2+=1/koefList[i]
                }
                for(i in 0 until (koefList.size)){
                    percent = ((1/koefList[i])/koef2)*100
                    percentLst.add(round((percent)*100.0)/100.0)
                }

                for(j in 0 until (percentLst[0]*10).toInt()){
                    coupleProb.add(1.0)
                }
                for(j in 0 until (percentLst[1]*10).toInt()){
                    coupleProb.add(2.0)
                }
                for(j in 0 until (percentLst[2]*10).toInt()){
                    coupleProb.add(3.0)
                }
                for(j in 0 until (percentLst[3]*10).toInt()){
                    coupleProb.add(4.0)
                }
                for(j in 0 until (percentLst[4]*10).toInt()){
                    coupleProb.add(5.0)
                }

                couples =
(1..5).map{(1..2).map{(coupleProb).random()}.toMutableList()}.toMutableList()
                //println(couples)
                if (couples.distinct().size==1){continue}
                for(i in 0 until couples.size){
                    while (couples[i][0] == couples[i][1]){
                        couples[i][0] = coupleProb.random()
                    }
                }
                //println("START C3")

                couples3 = fn(couples.distinct().toMutableList())
                //println(couples3)

//                for(i in 0..(couples.size-1)){
//                    addRow(table4, couples[i][0].toString(),
couples[i][1].toString())
//                }
                //println("STER 4")
                //println(couples3)
                for(i in 0 until couples3.size){
                    //addRow(table5, population[(couples3[i][0]).toInt()-
1].toString(), population[(couples3[i][1]).toInt()-1].toString())
                    babies.add(mutableListOf(population[(couples3[i][0]).toInt()-
1],population[(couples3[i][1]).toInt()-1]))
                }
                //println("STER 5")
```

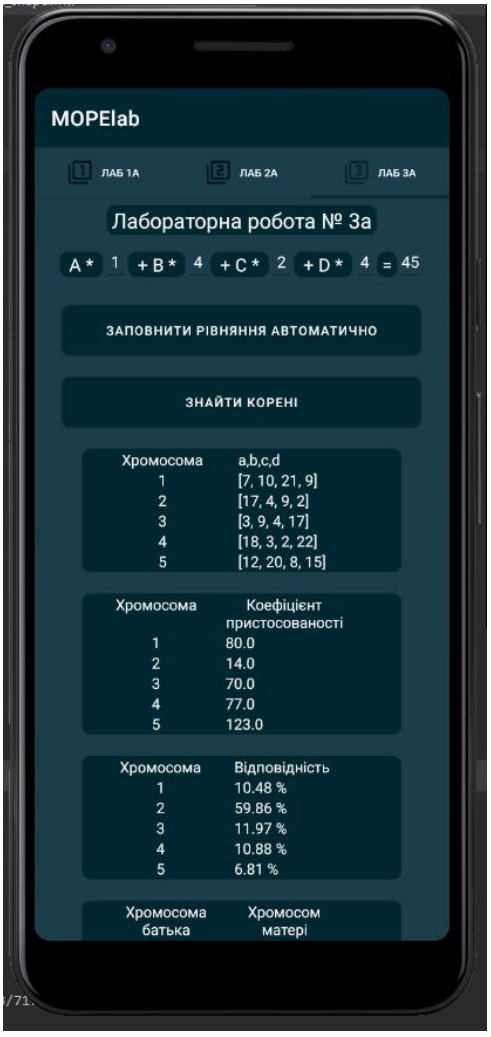
```
                //println(babies)

                for (element in babies){
                    //addRow01(table6, crossover(element[0],
element[1])[0].toString()+" a6o "+crossover(element[0], element[1])[1])
                    babies2.add(mutableListOf(crossover(element[0],
element[1])[0],crossover(element[0], element[1])[1]).random())
                }
                //println(babies2)
                //println("STER 6")

                k = 0.0

                for(i in 0 until (babies2.size)){
                    k = 0.0
                    for(j in 0 until (vList.size)){
                        k += abs(vList[j] * babies2[i][j])
                    }
                    k-=y
                    //addRow(table7, babies2[i].toString(), k.toString())
                    //println(k)
                    kLst2.add(k)
                }
                println("STER 7")
                for (item in kLst2){
                    if (item==0.0){
                        abcdLst = babies2[kLst2.indexOf(0.0)]
                        txtRes.text = "A, B, C, D = "+abcdLst
                        //val ind = kLst.indexOf(item)
                        txtRes2.text =String.format("A * %.1f + B * %.1f + C * %.1f +
D * %.1f = %.1f\n||\n\\/\n" +
                                "%.1f * %.1f + %.1f * %.1f + %.1f * %.1f + %.1f *
%.1f = %.1f",
                                vList[0]*1.0, vList[1]*1.0, vList[2]*1.0,
vList[3]*1.0, y*1.0, abcdLst[0]*1.0, vList[0]*1.0,
                                abcdLst[1]*1.0, vList[1]*1.0, abcdLst[2]*1.0,
vList[2]*1.0, abcdLst[3]*1.0, vList[3]*1.0, y*1.0)
                        //vList
                        //addRow(table7, babies2[i].toString(), kLst2[i].toString())
                        break
                    }
                }
                if (k==0.0){break}
            } //while (k != 0.0 )

            for(i in 0 until (population.size)) {
                addRow(table,(i+1).toString(), population[i].toString())
                addRow(table2, (i+1).toString(), kLst[i].toString())
            }
            for(i in 0 until (koefList.size)){
                //percent = ((1/koefList[i])/koef2)*100
                //percentLst.add(round((percent)*100.0)/100.0)
                addRow(table3,(i+1).toString(),String.format("%.2f %%",
percentLst[i]))
            }
            for(i in 0..(couples.size-1)){
                addRow(table4, couples[i][0].toString(), couples[i][1].toString())
            }
            for(i in 0 until couples3.size){
                addRow(table5, population[(couples3[i][0]).toInt()-1].toString(),
population[(couples3[i][1]).toInt()-1].toString())
                //babies.add(mutableListOf(population[(couples3[i][0]).toInt()-
```

```
1],population[(couples3[i][1]).toInt()-1]))
            }
            for (element in babies){
                addRow01(table6, crossover(element[0], element[1])[0].toString()+"
або    "+crossover(element[0], element[1])[1])
                //babies2.add(mutableListOf(crossover(element[0],
element[1])[0],crossover(element[0], element[1])[1]).random())
            }
            for(i in 0 until (babies2.size)){
                addRow(table7, babies2[i].toString(), kLst2[i].toString())
            }
            //println("klst ----------------------"+kLst2)

        } catch (e: NumberFormatException){txtRes.text = "Дані введені невірно\nабо
не  введені\nвзагалі"}
        catch (e: ArithmeticException){txtRes.text = "Дані введені невірно\nабо не
введені\nвзагалі"}
    }
```

## Скріншоти роботи:


MOPElab
ЛАБ 1А
ЛАБ 2А
ЛАБ 3А
C: 5 5
Більше P
D: 1 4
Менше P
Швидкість навчання
0.001
Часовий дедлайн
1
Кількість ітерацій
500
ЗАПОВНИТИ ВСЕ АВТОМАТИЧНО
ЗАПУСК
Результат
Значення w1 і w2
отримані за 500 ітерацій
w1 = 0.5
w2 = 1.2


MOPElab
ЛАБ 1А
ЛАБ 2А
ЛАБ 3А
Лабораторна робота № 3а
A * 1 + B * 4 + C * 2 + D * 4 = 45
ЗАПОВНИТИ РІВНЯННЯ АВТОМАТИЧНО
ЗНАЙТИ КОРЕНІ
Хромосома a,b,c,d
1 [7, 10, 21, 9]
2 [17, 4, 9, 2]
3 [3, 9, 4, 17]
4 [18, 3, 2, 22]
5 [12, 20, 8, 15]
Хромосома Коефіцієнт пристосованості
1 80.0
2 14.0
3 70.0
4 77.0
5 123.0
Хромосома Відповідність
1 10.48 %
2 59.86 %
3 11.97 %
4 10.88 %
5 6.81 %
Хромосома батька
Хромосом матері


**Висновок:**

В даній лабораторній роботі я ознайомився з основними принципами розкладання числа на прості множники з використанням різних алгоритмів факторизації, з принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрон(Perceptron).

Змоделював роботу нейронної мережі та дослідив вплив параметрів на час виконання та точність результату.

Ознайомтвся з принципами реалізації генетичного алгоритму, вивчив та дослідив особливості даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.