Министерство образования Республики Беларусь  
Учреждения образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе 1

студент гр. 853504  
Лукашёнок В.Д.

Минск 2021

**Лабораторная работа 1.**

Написано на golang, пишу в vscode. Надеюсь, что и остальные лабораторные получится делать на go и не придётся использовать python. Код С-подобный и очень хорошо читается. Приложил файлик с инпутом и два файлика с кодом, один с дебагом и выводом действий, второй — без. Как и требовалось, умножение выполняется в 2 цикла for, что даст сложность O(n2). Читаю из файлика две матрицы, вектор и индекс, и с ними делаю нужные действия.

**Версия**



**Код**

package main

import (

"fmt"

"io/ioutil"

"strconv"

"strings"

"gonum.org/v1/gonum/mat"

)

func mulVec(m \*mat.Dense, v \*mat.VecDense) \*mat.VecDense {

n, sum := len(v.RawVector().Data), float64(0)

vector := mat.NewVecDense(n, nil)

for i := 0; i < n; i++ {

sum = 0

for j := 0; j < n; j++ {

sum += m.At(i, j) \* v.At(i, 0)

}

vector.SetVec(i, sum)

}

return vector

}

func eye(n int) \*mat.Dense {

d := make([]float64, n\*n)

for i := 0; i < n\*n; i += n + 1 {

d[i] = 1

}

return mat.NewDense(n, n, d)

}

func matPrint(X mat.Matrix) {

fa := mat.Formatted(X, mat.Prefix(""), mat.Squeeze())

fmt.Printf("%v\n", fa)

}

func readMatrixFromStringArray(n, shift int, lines []string) \*mat.Dense {

var matrix []float64

for i := shift \* n; i < (shift+1)\*n; i++ {

numbersRaw := strings.Split(lines[i], " ")

for j := 0; j < n; j++ {

number, \_ := strconv.ParseFloat(numbersRaw[j], 64)

matrix = append(matrix, number)

}

}

return mat.NewDense(n, n, matrix)

}

func readMatrixMatrixInvVectorIndex() (\*mat.Dense, \*mat.Dense, \*mat.VecDense, int) {

str, err := ioutil.ReadFile("input.txt")

if err != nil {

panic(err)

}

lines := strings.Split(string(str), "\n")

n := len(strings.Split(lines[0], " "))

// reading matrix

matrix := readMatrixFromStringArray(n, 0, lines)

// reading matrixInv

matrixInv := readMatrixFromStringArray(n, 1, lines)

// reading vector

vector := make([]float64, n)

for i := 2 \* n; i < 3\*n; i++ {

number, \_ := strconv.ParseFloat(lines[i], 64)

vector[i%n] = number

}

// reading vector

index, \_ := strconv.Atoi(lines[len(lines)-1])

return matrix, matrixInv, mat.NewVecDense(n, vector), index - 1

}

func mulOptimized(a, b \*mat.Dense, index int) \*mat.Dense {

n, \_ := a.Dims()

result, subSum := mat.NewDense(n, n, nil), float64(0)

for i := 0; i < n; i++ {

for j := 0; j < n; j++ {

if i != index {

subSum = a.At(i, i)\*b.At(i, j) + a.At(i, index)\*b.At(index, j)

} else {

subSum = a.At(i, index) \* b.At(index, j)

}

result.Set(i, j, subSum)

}

}

return result

}

func invOptimized(matrix, matrixInv \*mat.Dense, vector \*mat.VecDense, index int) \*mat.Dense {

n := len(vector.RawVector().Data)

// step 0

matrix.SetCol(index, vector.RawVector().Data)

// step 1

AInv := mat.NewDense(n, n, make([]float64, n\*n)) // clonning matrixInv

AInv.CloneFrom(matrixInv)

l := mulVec(AInv, vector)

if l.At(index, 0) == 0 {

panic("Matrix is uninversable!!!")

}

// step 2

storedNumber := l.At(index, 0)

l.SetVec(index, -1)

// step 3

l.ScaleVec((-1 / storedNumber), l)

// step 4

Q := eye(n)

Q.SetCol(index, l.RawVector().Data)

// step 5

result := mulOptimized(Q, AInv, index)

return result

}

func main() {

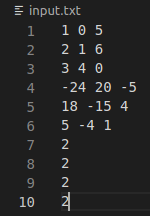
matrix, matrixInv, vector, index := readMatrixMatrixInvVectorIndex()

result := invOptimized(matrix, matrixInv, vector, index)

matPrint(result)

}

**Пример input.txt**



**Пример работы**

