cslsi-05-mueller

November 13, 2018

1 Sheet 05

- Student: Simon Mueller
- Mail: s.mueller1995@gmail.com / s6siume2@uni-bonn.de


```
In [1]: def matrix_get_submatrix(m, ii, jj):
            rows = len(m)
            cols = len(m[0])
            assert all([len(m[i]) == len(m[0]) for i in range(rows)])
            assert 0 <= ii < rows
            assert 0 <= jj < cols

            return [m[i][0:jj] + m[i][jj+1:] for i in range(rows) if i != ii]

        def matrix_det(m):
            rows = len(m)      # number of rows of m
            assert rows > 0
            cols = len(m[0])  # number of cols of m
            if rows != cols:
                raise Exception("matrix must be square")
            if rows == 1:
                return m[0][0]
            elif rows == 2:
                return m[0][0] * m[1][1] - m[0][1] * m[1][0]
            elif rows == 3:
                return m[0][0]*m[1][1]*m[2][2] + m[0][1]*m[1][2]*m[2][0] + m[0][2]*m[1][0]*m[2][
            sgn = -1      #Fixed sign problem
            sum = 0
            for i in range(cols): #idx had to start at 1, not 0
                sum += sgn**(i) * m[0][i] * matrix_det(matrix_get_submatrix(m, 0, i)) # added co
            return sum
```

1.1 Exercise 1

```
In [4]: import unittest

        class TestMatrixDet(unittest.TestCase):
```

```
            def test_submatrix(self):
                testMat = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
                newMat = matrix_get_submatrix(testMat, 0, 0)

                self.assertEqual(len(newMat), len(testMat) - 1, 'Test cols')
                self.assertEqual(len(newMat[0]), len(testMat[0]) - 1, 'Test rows')
                self.assertEqual(newMat[0][0], testMat[1][1])
                self.assertEqual(newMat[0][1], testMat[1][2])

                testMat = [[1, 2], [4, 5, 6], [7, 8, 9]]
                with self.assertRaises(AssertionError):
                    newMat = matrix_get_submatrix(testMat, 0, 0)

            def test_determinate(self):
                testMat = [[1, 2, 3, 4], [5, 1, 7, 8], [10, 10, 11, 12], [13, 14, 15, 16]]
                det = matrix_det(testMat)

                self.assertEqual(det, -60)
```

```
In [5]: unittest.main(argv=['first-arg-is-ignored'], exit=False)

..
----------------------------------------------------------------------
Ran 2 tests in 0.002s

OK


Out[5]: <unittest.main.TestProgram at 0x7fda7825a0b8>
```

```
In [172]: import random

          for d in range(1, 11):
              print("Testing dimension: " + str(d))
              testMat = [[random.randrange(-5, 5) for x in range(d)] for y in range(d)]
              %timeit -n 3 matrix_det(testMat)
```

```
Testing dimension: 1
808 ns ± 295 ns per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 2
1.22 µs ± 301 ns per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 3
2.79 µs ± 351 ns per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 4
30.1 µs ± 1.2 µs per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 5
154 µs ± 11.6 µs per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 6
639 µs ± 167 µs per loop (mean ± std. dev. of 7 runs, 3 loops each)
```

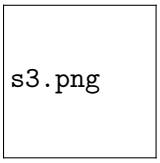
```
Testing dimension: 7
3.04 ms ± 354 µs per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 8
17.7 ms ± 980 µs per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 9
144 ms ± 9.12 ms per loop (mean ± std. dev. of 7 runs, 3 loops each)
Testing dimension: 10
1.24 s ± 18.9 ms per loop (mean ± std. dev. of 7 runs, 3 loops each)
```

1.2 Exercise 2

- Alphonso: (Person)
- Natalia: Gangster
- Paolo: Gangster
- Cosimo: (Person)
- Giovanni: Gangster
- Carla: (Person)
- Rosa: Gangster
- Jake: (Person)

You can only state that you assume that every person is normal who is not certainly a gangster. The inheritane relationship implies that every object in the diagram could be a gangster and only some of them do gangster specfic actions (stealing, denounicing). Therefore the person assignment is only fuzzy and not set in stone.

1.3 Exercise 3



Shitty Student System (S3)

Used classes: - Publishing a list of offered classes for a given semester (Class, filter for semester variable) - Keeping track of registrations to classes (Filter students for class, Student) - Keeping track of scores from exercise sheets (Saved in the student, Student) - Keeping track of registrations to exams and their results (Student, Class, Exam) - Printing a transcript of records for a specific student (Student, calcAverageOverExamGrades)

```
In [ ]:
```