T1

May 2, 2016

1 T1 Estruturas Discretas 2016.1

1.1 Grupo

- Hugo Grochau - 1310486
- Gabriel Maia Guzowski - 1312681

1.2 Questão 1

O teorema acima resolve o problema de determinar o quociente entre $x^k - y^k$ e $x - y$. Assim deseja-se um algoritmo que dados x, y, e k inteiros determine esse quociente.

(a) Escreva o algoritmo resultante da prova acima.

(b) Implemente este algoritmo e teste para vários valores de x, y, e k.

1.2.1 Resposta

Pela prova temos que:

$$x^{k+1} - y^{k+1} = (x^k + y \cdot q_k) \cdot (x - y) = q_{k+1} \cdot (x - y)$$

A partir disso conseguimos a formula para conseguir o quociente q_{k+1} dado o quociente q:

$$q_{k+1} = x^k + y \cdot q_k$$

Logo basta começar do caso base $i = 1$, $k_i = 1$, $q_i = 1$ e ir calculando o q_{i+1} com a formula acima até chegar em $i = k$

```
In [5]: from datetime import datetime, timedelta

        def time(t, f, **args):
            i = 0
            end = datetime.now() + timedelta(milliseconds=t)
            while (datetime.now() < end):
                f(**args)
                i += 1
            return t/i

        def test(f, expected, **args):
            print("\n#############")
            print("Testing {}({}) = {}".format(f.__name__, args, expected))
            assert(f(**args) == expected)
            print("Passed")
            print("Timing...")
            print("{} ms average".format(time(5000, f, **args)))
```

```
        def quotient(x, y, k):
            # CB
            qi = 1

            # PI
            for ki in range(2, k+1):
                # q_{i+1} = x^{k_i} + y*q_i
                qi = pow(x, ki-1) + y*qi
            return qi


        test(quotient, 1, x=1, y=1, k=1)
        test(quotient, 19, x=2, y=3, k=3)
        test(quotient, 12185736200309627046 1, x=25, y=44, k=13)
        test(quotient, 16235887778794635975975966387492720966650659786893373459967485932898863385801955
             x=13, y=25934, k=45)
```

```
############
Testing quotient({'y': 1, 'k': 1, 'x': 1}) = 1
Passed
Timing...
0.001978122753594348 ms average

############
Testing quotient({'y': 3, 'k': 3, 'x': 2}) = 19
Passed
Timing...
0.0027780463222555956 ms average

############
Testing quotient({'y': 44, 'k': 13, 'x': 25}) = 12185736200309627046 1
Passed
Timing...
0.007125369450406004 ms average

############
Testing quotient({'y': 25934, 'k': 45, 'x': 13}) = 16235887778794635975975966387492720966650659786893373
Passed
Timing...
0.02417514408385874 ms average
```

1.2.2 Resultados

Teste	x	y	k	tempo de execução médio
1	1	1	1	0.0018934631592660027ms
2	2	3	3	0.002644323193411616ms
3	25	44	13	0.006969233621255605ms
4	13	25934	45	0.024239486122894196ms

1.3 Questão 2

Teorema 2: O número de números inteiros cujos dígitos pertencem ao conjunto $1, 2, ..., m$ de k dígitos diferentes é dado pelo produto $m \cdot (m-1)... \cdot (m-k+1)$.

(a) Enuncie o teorema de que sabe-se enumerar todos estes números especificando seu parâmetro indutivo e prove-o por indução matemática (simples).

(b) Apresente o algoritmo resultante da sua prova, que enumera todos os $m.(m-1).....(m-k+1)$ números (o que permite contá-los).

(c) Implemente este algoritmo e apresente os números inteiros (a sequência de dígitos, em especial para quando m é maior que nove) impressos para pequenos valores de m e k. Para valores maiores apresente o tempo de CPU e indique até que valores de m e k sua implementação (e computador) foi capaz de fazer a enumeração.

Observe que m pode ser maior que 10

1.3.1 Resposta

TCB $m = 1 \Rightarrow \{1\}$
$k = 1 \Rightarrow \{1\}$
$m = 2 \Rightarrow \{1, 2\}$
$k = 1 \Rightarrow \{1\}\{2\}$
$k = 2 \Rightarrow \{1, 2\}\{2, 1\}$
$m = 3 \Rightarrow \{1, 2, 3\}$
$k = 1 \Rightarrow \{1\}\{2\}\{3\}$
$k = 2 \Rightarrow \{1, 2\}\{1, 3\}\{2, 1\}\{2, 3\}\{3, 1\}\{3, 2\}$
$k = 3 \Rightarrow \{1, 2, 3\}\{1, 3, 2\}\{2, 1, 3\}\{2, 3, 1\}\{3, 1, 2\}\{3, 2, 1\}$

TPI Se é válido para k, então é válido para k+1
Dado um conjunto: $\{1, 2, ..., m\}$
$k = 1 \Rightarrow \{1\}\{2\}...\{m\}$
Observamos que para construir o subconjunto de k+1, precisamos pegar o subconjunto de k, analisarmos cada um de forma individual, e acrecentarmos a cada um deles 1 número pertencente ao conjunto mas ainda não pertencente ao subconjunto analisado.
$k + 1 = 2 \Rightarrow \{1, 2\}\{1, m\}\{2, 1\}\{2, m\}...\{m, 1\}\{m, 2\}$
$k + 1 = 3 \Rightarrow \{1, 2, m\}\{1, m, 2\}\{2, 1, m\}\{2, m, 1\}...\{m, 1, 2\}\{m, 2, 1\}$
.
.
.
$k + 1 = m \Rightarrow \{1, 2, ..., m\}\{1, m, ..., 2\}\{2, 1, ..., m\}\{2, m, ..., 1\}...\{m, 1, ..., 2\}\{m, 2, ..., 1\}$
Concluí-se então, que é possível construir qualquer subconjunto k+1 a partir de seu subconjunto k, pegando cada subconjunto de k de forma idividual e criando todos os novos subconjuntos com um número pertencente ao conjunto ainda não contido.

```
In [6]: from datetime import datetime, timedelta

        def time(t, f, **args):
            i = 0
            end = datetime.now() + timedelta(milliseconds=t)
            while (datetime.now() < end):
                f(**args)
                i += 1
            return t/i
```

```
def test(f, expected, **args):
    print("\n#############")
    print("Testing {}({}) = {}".format(f.__name__, args, expected))
    assert(f(**args) == expected)
    print("Passed")
    print("Timing...")
    print("{} ms average".format(time(5000, f, **args)))

def duration(f, **args):
    start = datetime.now()
    f(**args)
    return datetime.now() - start


def number_of_numbers(m, k):
    numbers = []

    # CB
    if k == 1:
        # just create m elements with each digit
        for i in range(m):
            numbers.append([i + 1])
        return numbers

    # PI
    old_numbers = number_of_numbers(m, k - 1)

    # for each of the numbers from the previous call (k - 1)
    for i in range(len(old_numbers)):
        # for each digit
        for j in range(m):
            # ignore repeated digits
            if (j + 1) not in old_numbers[i]:
                # append the old numbers together with the new digit
                numbers.append(list(old_numbers[i]) + [j + 1])
    return numbers

test(number_of_numbers, [[1]], m=1, k=1)
test(number_of_numbers, [[1],[2]], m=2, k=1)
test(number_of_numbers, [[1, 2],[2, 1]], m=2, k=2)
test(number_of_numbers, [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]], m=3
test(number_of_numbers, [[1, 2, 3], [1, 2, 4], [1, 2, 5], [1, 3, 2], [1, 3, 4], [1, 3, 5],
                         [1, 4, 2], [1, 4, 3], [1, 4, 5], [1, 5, 2], [1, 5, 3], [1, 5, 4],
                         [2, 1, 3], [2, 1, 4], [2, 1, 5], [2, 3, 1], [2, 3, 4], [2, 3, 5],
                         [2, 4, 1], [2, 4, 3], [2, 4, 5], [2, 5, 1], [2, 5, 3], [2, 5, 4],
                         [3, 1, 2], [3, 1, 4], [3, 1, 5], [3, 2, 1], [3, 2, 4], [3, 2, 5],
                         [3, 4, 1], [3, 4, 2], [3, 4, 5], [3, 5, 1], [3, 5, 2], [3, 5, 4],
                         [4, 1, 2], [4, 1, 3], [4, 1, 5], [4, 2, 1], [4, 2, 3], [4, 2, 5],
                         [4, 3, 1], [4, 3, 2], [4, 3, 5], [4, 5, 1], [4, 5, 2], [4, 5, 3],
                         [5, 1, 2], [5, 1, 3], [5, 1, 4], [5, 2, 1], [5, 2, 3], [5, 2, 4],
                         [5, 3, 1], [5, 3, 2], [5, 3, 4], [5, 4, 1], [5, 4, 2], [5, 4, 3]], m=5

tests = [(5,3),(5,4),(5,5),(7,5),(10,5),(11,4),(11,5),(11,6),(12,6),(13,7)]
for test in tests:
```

```
            print("Timing number_of_numbers with m={} k={}...".format(test[0], test[1]))
            print("{} ms average".format(time(5000, number_of_numbers, m=test[0], k=test[1])))

        tests = [(13,7)]
        for test in tests:
            print("Timing number_of_numbers with m={} k={}...".format(test[0], test[1]))
            print("{} s ".format(duration(number_of_numbers, m=test[0], k=test[1]).total_seconds()))
```

```
############
Testing number_of_numbers({'k': 1, 'm': 1}) = [[1]]
Passed
Timing...
0.00227942463675089 ms average

############
Testing number_of_numbers({'k': 1, 'm': 2}) = [[1], [2]]
Passed
Timing...
0.002415914109421582 ms average

############
Testing number_of_numbers({'k': 2, 'm': 2}) = [[1, 2], [2, 1]]
Passed
Timing...
0.0047157354661032935 ms average

############
Testing number_of_numbers({'k': 3, 'm': 3}) = [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3
Passed
Timing...
0.013166314248321953 ms average

############
Testing number_of_numbers({'k': 3, 'm': 5}) = [[1, 2, 3], [1, 2, 4], [1, 2, 5], [1, 3, 2], [1, 3, 4], [1
Passed
Timing...
0.0516459566380548 ms average
Timing number_of_numbers with m=5 k=3...
0.05170363476552402 ms average
Timing number_of_numbers with m=5 k=4...
0.1427062819305306 ms average
Timing number_of_numbers with m=5 k=5...
0.2812623052258536 ms average
Timing number_of_numbers with m=7 k=5...
2.875215641173088 ms average
Timing number_of_numbers with m=10 k=5...
28.089887640449437 ms average
Timing number_of_numbers with m=11 k=4...
6.20347394540943 ms average
Timing number_of_numbers with m=11 k=5...
50.505050505050505 ms average
Timing number_of_numbers with m=11 k=6...
```

```
384.61538461538464 ms average
Timing number_of_numbers with m=12 k=6...
714.2857142857143 ms average
Timing number_of_numbers with m=13 k=7...
5000.0 ms average
Timing number_of_numbers with m=13 k=7...
10.653366 s
```

1.3.2 Resultados

m	k	time
1	1	0.0023481443319787673 ms average
2	2	0.0047477248902326 ms average
3	3	0.013017104475280518 ms average
5	3	0.051864529848036925 ms average
5	4	0.1410636196924813 ms average
5	5	0.27577077932822236 ms average
7	5	3.109452736318408 ms average
10	5	30.303030303030305 ms average
11	4	6.802721088435374 ms average
11	5	54.34782608695652 ms average
11	6	384.61538461538464 ms average
12	6	714.2857142857143 ms average
13	7	10.592981 s

Tabela de tempo de execução

1.4 Questão 3

Seja um conjunto de n de equipes $e_1, e_2, ..., en$. Deseja-se construir as n − 1 rodadas de um campeonato onde todos jogam contra todos. Assuma que $n = 2^k$ para algum k. Enuncia-se abaixo o teorema de que sabe-se construir as $n - 1$ rodadas de $n/2$ jogos cada.

Teorema 3 (k): Sabe-se construir $2^k - 1$ rodadas de 2^{k-1} jogos onde cada equipe enfrenta uma equipe diferente em cada rodada.

1.4.1 Resposta

Sabe-se construir 2^{k-1} rodadas de 2^{k-1} jogos onde cada equipe enfrenta outra equipe diferente em cada rodada

TCB $k = 1$
$n = 2^1 = 2$ times
$r = 2^1 - 1 = 1$ rodada
$j = 2^{1-1} = 1$ jogo por rodada
T1 —- T2
$k = 2$
$n = 2^2 = 4$ times
$r = 2^2 - 1 = 3$ rodadas
$j = 2^{2-1} = 2$ jogos por rodada
T1 —- T2 | T1 —- T3 | T1 —- T4
T3 —- T4 | T2 —- T4 | T2 —- T3

TPI Se é válido para k, é válido para k + 1

$n = 2^{k+1} = 2^k \cdot 2 = 2^k + 2^k$

$r = 2^{k+1} - 1 = 2 \cdot 2^k - 1 = 2^k + 2^k - 1$

Percebemos aqui que $2^k - 1 = Teo(k)$

$j = 2^{k+1-1} = 2^k$

G1

k = 1

n = 2 times

r = 1 rodada

j = 1 jogo

T1 —- T2

G2

k = 1

n = 2 times

r = 1 rodada

j = 1 jogo

T3 —- T4

G1 + G2

k = 2

n = G1(n) + G2(n) = 2 + 2 = 4

r = G1(n) + G2(r) = 2 + 1 = 3

j = T1(n) = 2

T1 —- T2 | T1 —- T3 | T1 —- T4

T3 —- T4 | T2 —- T4 | T2 —- T3

Logo, concluimos que para gerar as partidas para $k + 1$, precisamos construir para k e obtermos duas construções de k diferentes, e ao aplicarmos o teorema para $k + 1$, vemos que estamos somando os números de times de k $n(k+1) = n(k)+n(k)$, fixando um time e perumutando o resto, obtemos o número de rodadas e jogos.

```
In [4]: import collections
        from datetime import datetime, timedelta

        def time(t, f, **args):
            i = 0
            end = datetime.now() + timedelta(milliseconds=t)
            while (datetime.now() < end):
                f(**args)
                i += 1
            return t/i

        def duration(f, **args):
            start = datetime.now()
            f(**args)
            return datetime.now() - start

        def test(f, expected, **args):
            print("\n#############")
            print("Testing {}({}) = {}".format(f.__name__, args, expected))
            assert(f(**args) == expected)
            print("Passed")
            print("Formatted output:")
            pretty_print_rounds(f(**args))
            print("Timing...")
            print("{} ms average".format(time(5000, f, **args)))
```

```
def pretty_print_rounds(t):
    i = 1
    for r in t:
        print("### Round {} ###".format(i))
        for g in r:
            print("Team {} vs Team {}".format(g[0], g[1]))
        i += 1

def tournament_generator(k, s):
    # cb
    if (k == 1):
        return [[(s, s + 1)]]

    n = 2**k
    r = n - 1
    num_games = 2**(k-1)

    # generate all group A vs group A and group B vs group B rounds
    rounds_1 = tournament_generator(k-1, s)
    rounds_2 = tournament_generator(k-1, s + 2**(k-1))
    # merge them
    rounds = []
    for i in range(len(rounds_1)):
        rounds.append(rounds_1[i] + rounds_2[i])


    # divide teams into two groups
    A = list(range(s, s + int(n/2)))
    B = list(range(s + int(n/2), s + n))
    # make b a rotating list
    B = collections.deque(B)


    # fix A and rotate B teams for remaining rounds
    for i in range(int(r/2) + 1):
        games = []

        # pair each of the teams in the list
        for j in range(num_games):
            games.append((A[j], B[j]))
        rounds.append(games)

        # rotate the B team list
        B.rotate(1)

    return rounds

test(tournament_generator, [[(1,2)]], k=1, s=1)
test(tournament_generator, [[(1, 2), (3, 4)], [(1, 3), (2, 4)], [(1, 4), (2, 3)]], k=2, s=1)
test(tournament_generator, [[(1, 2), (3, 4), (5, 6), (7, 8)], [(1, 3), (2, 4), (5, 7), (6, 8)],
, k=3, s=1)

for i in range(4,12):
```

```
            print("Timing tournament_generator with k={}...".format(i))
            print("{} ms average".format(time(5000, tournament_generator, k=i, s=1)))

        for i in range(12,14):
            print("Timing tournament_generator with k={}...".format(i))
            print("{} s ".format(duration(tournament_generator, k=i, s=1).total_seconds()))
```

```
############
Testing tournament_generator({'s': 1, 'k': 1}) = [[(1, 2)]]
Passed
Formatted output:
### Round 1 ###
Team 1 vs Team 2
Timing...
0.0019077700038269867 ms average

############
Testing tournament_generator({'s': 1, 'k': 2}) = [[(1, 2), (3, 4)], [(1, 3), (2, 4)], [(1, 4), (2, 3)]]
Passed
Formatted output:
### Round 1 ###
Team 1 vs Team 2
Team 3 vs Team 4
### Round 2 ###
Team 1 vs Team 3
Team 2 vs Team 4
### Round 3 ###
Team 1 vs Team 4
Team 2 vs Team 3
Timing...
0.007429685456836499 ms average

############
Testing tournament_generator({'s': 1, 'k': 3}) = [[(1, 2), (3, 4), (5, 6), (7, 8)], [(1, 3), (2, 4), (5,
Passed
Formatted output:
### Round 1 ###
Team 1 vs Team 2
Team 3 vs Team 4
Team 5 vs Team 6
Team 7 vs Team 8
### Round 2 ###
Team 1 vs Team 3
Team 2 vs Team 4
Team 5 vs Team 7
Team 6 vs Team 8
### Round 3 ###
Team 1 vs Team 4
Team 2 vs Team 3
Team 5 vs Team 8
Team 6 vs Team 7
### Round 4 ###
Team 1 vs Team 5
Team 2 vs Team 6
```

```
Team 3 vs Team 7
Team 4 vs Team 8
### Round 5 ###
Team 1 vs Team 8
Team 2 vs Team 5
Team 3 vs Team 6
Team 4 vs Team 7
### Round 6 ###
Team 1 vs Team 7
Team 2 vs Team 8
Team 3 vs Team 5
Team 4 vs Team 6
### Round 7 ###
Team 1 vs Team 6
Team 2 vs Team 7
Team 3 vs Team 8
Team 4 vs Team 5
Timing...
0.02091945174300872 ms average
Timing tournament_generator with k=4...
0.0573855158957879 ms average
Timing tournament_generator with k=5...
0.1621060822202049 ms average
Timing tournament_generator with k=6...
0.4801690194948622 ms average
Timing tournament_generator with k=7...
1.9364833462432223 ms average
Timing tournament_generator with k=8...
7.898894154818326 ms average
Timing tournament_generator with k=9...
31.25 ms average
Timing tournament_generator with k=10...
131.57894736842104 ms average
Timing tournament_generator with k=11...
500.0 ms average
Timing tournament_generator with k=12...
2.087176 s
Timing tournament_generator with k=13...
8.682202 s
```

1.4.2 Resultados

k	time
1	0.0018250839538618776 msaverage
2	0.007812268073291574 ms average
3	0.0214840309198173 ms average
4	0.05655788699734178 ms average
5	0.15794794035885773 ms average
6	0.4722773212430339 ms average
7	2.02757502027575 ms average
8	7.987220447284345 ms average
9	32.05128205128205 ms average
10	128.2051282051282 ms average

k	time
11	500.0 ms average
12	2.167275 s
13	8.976655 s

Tabela de tempo de execução

Resultado formatado

```
### Round 1 ###
Team 1 vs Team 2
Team 3 vs Team 4
Team 5 vs Team 6
Team 7 vs Team 8
### Round 2 ###
Team 1 vs Team 3
Team 2 vs Team 4
Team 5 vs Team 7
Team 6 vs Team 8
### Round 3 ###
Team 1 vs Team 4
Team 2 vs Team 3
Team 5 vs Team 8
Team 6 vs Team 7
### Round 4 ###
Team 1 vs Team 5
Team 2 vs Team 6
Team 3 vs Team 7
Team 4 vs Team 8
### Round 5 ###
Team 1 vs Team 8
Team 2 vs Team 5
Team 3 vs Team 6
Team 4 vs Team 7
### Round 6 ###
Team 1 vs Team 7
Team 2 vs Team 8
Team 3 vs Team 5
Team 4 vs Team 6
### Round 7 ###
Team 1 vs Team 6
Team 2 vs Team 7
Team 3 vs Team 8
Team 4 vs Team 5
```