Alfredo's MAC0110 Journal

Alfredo Goldman

May 5, 2020

1 Programa do curso

1.1 Aula 16 *<2020-05-06 qua>*

1.1.1 Um pouco mais de sintaxe de Julia e vetores

Como é de se imaginar, testes automatizados são muito utilizados por bons desenvolvedores. Logo podemos imaginar que a linguagem Julia tem formas de ajudar a escrita de testes.

Isso é feito com o pacote de testes de Julia, para usá-lo precisamos usar o comando using. Vamos a um exemplo abaixo:

```
using Test
@test 1 == 1
@test 1 != 2
```

O comando @test avalia a expressão e verifica se o valor é verdadeiro (true), se for, não faz nada. Mas, se não for, aponta o erro.

Dessa forma, podemos escrever testes em Julia de forma mais compacta. Para começar vamos verificar se uma função que recebe um vetor e devolve a soma dos seus elementos funciona.

```
using Test
include("soma.jl")
function testaTudo()
  @test soma([]) == 0
  @test soma([1]) == 1
  @test soma([10, 20, 30]) == 60
  println("final dos testes")
end
testaTudo()
```

Que funciona verifica se a seguinte função funciona:

```
function soma(v)
  s = 0
  for el in v
    s = s + el
  end
  return s
end
```

Outro comando útil de Julia é a verificação aproximada, pois já vimos que operações com número reais nem sempre é exata. Essa comparação é dada com $\approx$ (barra approx)

```
0.2 + 0.2 + 0.2 \approx 0.6
```

1.1.2 Voltando a vetores

Vamos agora voltar à parte algorítmica, com o seguinte problema. Subsequência de soma máxima. Dado um vetor de inteiros, devolver a soma de elementos consecutivos que seja máxima.

Vamos começar pelos testes.

```
using Test
function verificaSoma()
  @test somasub([]) == 0
  @test somasub([1, 2, 3]) == 6
  @test somasub[-1, -2, -3]) == -1
  @test somasub([10, 5, -17, 20, 5, -1, 3, -30, 10]) == 72
  @test somasub([31, -41, 59, 26, -53, 58, 97, -93, -23, 84] == 187
  println("Final dos testes")
end
```

Vamos começar com a solução de força bruta, isso é, calcular a soma de todas a sub-sequências, procurando pela máxima.

```
# lugar para escrever o codigo
```

Agora vamos a um algoritmo mais elaborado. (Jay-Kadane)

```
function somasub(v)
  if length(v) == 0
    return 0
  end
  soma = 0
  somamax = v[1]
  for i in 1:length(v)
    if soma + v[i] < 0
      soma = 0
    else
      soma = soma + v[i]
    end
    if soma > somamax
      somamax = soma
    end
  end
  return somamax
end
```

Faça uma função, onde dados dois vetores u e v, devolve o seu produto escalar.

Faça uma função, onde dados dois vetores ordenados u e v, sem repetição, devolve o vetor ordenado com os elementos de u e v, sem repetição.