Alfredo's MAC0110 Journal

Alfredo Goldman

May 17, 2020

1 Programa do curso

1.1 Aula 17 *<2020-05-18 seg>*

1.1.1 Um pouco mais de manipulação de vetores (com merge e busca) e mudança de base

Na aula passada, vimos como fazer um dos problemas de merge, isso, é dados dois vetores ordenados sem repetição, devolver um vetor que corresponda a união deles sem repetição.

```
function merge(u, v)
  pu = 1   # ponteiro em u
  pv = 1   # ponteiro em v
  resp = []
  while pu <= length(u) && pv <= length(v)
    if u[pu] < v[pv]
       push!(resp, u[pu])
       pu = pu + 1
    elseif v[pv] < u[pu]
       push!(resp, v[pv])
       pv = pv + 1
    else
       push!(resp, v[pv])
       pu = pu + 1
       pv = pv + 1
    end
  end
  while pu <= length(u)
    push!(resp, u[pu])
    pu = pu + 1
  end
  while pv <= length(v)
    push!(resp, v[pv])
    pv = pv + 1
  end
  return resp
end
```

Com pequenas variações nesse código, podemos fazer outros tipos de merge:

- Fazer a intersecção
- Fazer a diferença (isso é, elementos devolver apenas elementos que estão em u, mas não e, v

Todas essas soluções se baseiam em variações do código acima. Tudo isso sabendo que os vetores originais estão ordenados e sem repetição.

Aproveitando, como podemos fazer para dado um vetor ordenado, com repetições, devolver um vetor ordenado sem repetições?

A ordenação também pode ser útil para verificar se um elemento está em um vetor, mas vamos começar com a versão simples, de percorrer todo o vetor.

Para isso, faremos uma função que recebe um vetor e um elemento, caso o elemento pertença ao vetor, devolvemos sua posição, caso contrário devolvemos 0.

```
function posicaonovetor(v, el)
  for i in 1:length(v)
    if el == v[i]
      return i
    end
  end
  return 0
end
```

A solução acima funciona, mas ela não considera que o vetor está ordenado. Para tentar entender uma solução melhor (busca binária), vamos fazer uma brincadeira. Um voluntário vai pensar em um número entre 0 e 1023 e vou advinhá-lo em até 10 tentativas, sendo que as respostas podem ser:

- Número encontrado

- O número que eu pensei é maior
- O número que eu pensei é menor

A lógica por trás dessa solução é que eu quero eliminar a maior quantidade de números a cada palpite, para fazer isso, o melhor é pensar em um palpite no "meio", que elimine metade dos números em questão.

Vamos agora desenvolver um algoritmo que faz a busca binária. Mas, como ele é um pouco mais complicado, vamos começar com os testes, validar os testes com a função anterior (posição no vetor) e finalmente escrever o nosso algoritmo de busca binária.

Na aula que vem veremos algumas formas de se ordenar um vetor, sim Julia tem as funções sort() e sort!(). Mas, a motivação por trás de aprender a ordenar é aprender como fazê-lo.

1.1.2 Mudança de base

Vamos começar com umas ideias intuitivas, na base 10, temos os dígitos de 0 a 9, e os números são agrupados de forma que o menos significativo corresponde à base 10^0, o segundo à base 10^1 e assim por diante.

Logo, um número como o 123 é na verdade igual a $3 * 10^0 + 2 * 10^1 + 1 * 10^2$. Isso é tão natural que usamos naturalmente, sem pensar em base quando falamos em base decimal.

O mesmo vale para outras bases, como a binária:

10010 é igual a $0 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 + 1 * 2^4$ ou seja é igual a 18 na base 10.

Dessa forma, podemos pensar em como devolver o valor na base 10, para um número qualquer na base 2.

```
function valorbase2(n)
  pot = 0
  soma = 0
  while n != 0
    dig = n % 10
    soma = soma + dig * 2 ^pot
    n = div(n, 10)
    pot = pot + 1
  end
  return soma
end
```

Há duas coisas a observar acima, o código pode ser melhorado e podemos pensar em outras bases que não sejam apenas a binária (a dificuldade de trabalhar com bases maiores do que 10 é devido às variáveis inteiras terem apenas os dígitos de 0 a 9).

Mas, vamos fazer isso de uma forma iterativa, usando testes automatizados.

Após vermos como ter um número em uma base menor, na base 10, vamos ver como transformar um número na base 10, em outra base. Comecemos com a binária:

Temos as potências: 1, 2, 4, 8, 16, 32, 64, ... Como escrever o número 99 em binário? Começamos pela parte menos significativa, isso é, 99 % 2, e continuamos com a sobra da parte mais significativa, isso é, 99 % 2 (= 1) e 99 ÷ 2 (= 49), e repetimos o mesmo procedimento.

Mas, não vamos esquecer dos nossos testes :)

Olhando os códigos com cuidado, dá para generalizar?