① 3ª Aula

Princípio de indução matemática (Fraco)

A prova de uma afirmação por indução matemática é feita em dois passos:

(1) Passo base: É provado que $P(n_0)$ é verdade para um dado n_0 específico.

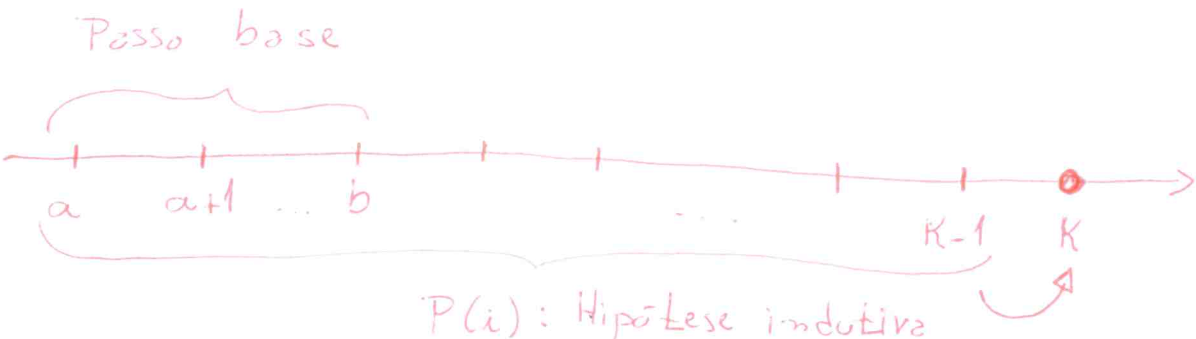
(2) Passo indutivo: É provado que para todo os valores $K \geq n_0$, se $P(K)$ é verdade então $P(K+1)$ é verdade

② Princípio da indução matemática (Forte)

Seja $P(n)$ um predicado que é definido para inteiros n, e seja a e b inteiros fixos, com $a \leq b$. A prova de $P(n)$ consiste em verificar a veracidade das seguintes afirmações:

(1) $P(a), P(a+1), \ldots, P(b)$ são verdades (Passo base)

(2) Para qualquer $K \geq b$, se $P(i)$ é verdade para $a \leq i < K$, então $P(K)$ é verdade. (Passo indutivo)



③ Corretude de algoritmos iterativos

Definição

Um invariante de um laço é uma propriedade que relaciona as variáveis de um algoritmo a cada execução completa daquele laço.

Estratégia "típica" para mostrar a corretude de um algoritmo iterativo através de invariantes:

(1) Mostre que o invariante vale no início da primeira iteração (trivial, em geral).

④ (2) Suponha que o invariante vale no início de uma iteração qualquer e prove que ele vale no início da próxima iteração.

(3) Conclua que se o algoritmo pára e o invariante vale no início da última iteração, então o algoritmo está correto.

Note que (1) e (2) implicam que o invariante vale no início de qualquer iteração do algoritmo. Isto corresponde ao método de indução matemática.

(5) Algoritmo que soma n elementos

```
Soma-Vetor(A)
1   S = 0
2   para j = 1 até n faça
3       S = S + A[j]
4   devolva S
```

```
S = 0
i = 1
while (i ≤ n) faça
    S = S + A[i]
    i = i + 1
retorne S
```

Corretude do algoritmo:

Invariante da linha 2:

No início da iteração j, vale que $S = \sum_{i=1}^{j-1} A[i]$

(6)

- Na primeira iteração temos $j=1$ e portanto $S = 0 = \sum_{i=1}^{0} A[i]$. Ou seja, o invariante vale.
- Suponha que no início da iteração j o invariante vale, ou seja, $S = \sum_{i=1}^{j-1} A[i]$.
  Então o algoritmo adiciona $A[j]$ a S e portanto, no início da iteração $j+1$ temos que $S = \sum_{i=1}^{j} A[i]$.
  Este é exatamente o invariante na iteração $j+1$.
- Na última iteração temos $j = n+1$ (laço pára) e a correção do algoritmo é evidente, pois o invariante diz que $S = \sum_{i=1}^{n} A[i]$

7) Algoritmo que calcula fatorial de n

```
Fatorial (n)
1   fat = 1
2   i = 1
3   while (i ≤ n) faça
4       fat = fat * i
5       i = i + 1
6   retorna fat
```

Qual é o invariante na linha 3?

Invariante: $fat = \prod_{k=1}^{i-1} k$

8)

→ Início

Antes do início do laço $i=1$, assim, $fat=1$. Isto mostra que o invariante está correto no início da primeira iteração.

→ Invariância (durante execução do laço)

Suponha que o invariante está correto no início da iteração i, i.e., $fat = \sum_{k=1}^{i-1} k$.

O algoritmo multiplica este valor por i, obtendo $\sum_{k=1}^{i} k$, e logo após incrementa i de 1. Portanto, isto mostra que depois da iteração o invariante se mantém.

9 → Término

O laço termina quando $i > n$, i.é, $i = n+1$. Substituindo i por $n+1$ no invariante, temos que:

$$fat = \prod_{K=1}^{n} K = 1 * 2 * \dots * n = n!$$

Portanto, o algoritmo está correto.

Laços aninhados

- → Analisar um laço por vez, começando pelo mais interno
- → Para cada laço, determinar um invariante
- → Provar que o invariante é válido
- → Mostrar que o algoritmo termina
- → Usar o invariante para provar que o algoritmo retorna o valor desejado.

10 Análise de corretude do Insertion-Sort

```
Insertion-Sort (A)
→ for j = 2 to n
      chave = A[j]
      // insere chave na sequência ordenada A[1..j-1]
      i = j-1
→     while i > 0 e A[i] > chave
            A[i+1] = A[i]
            i = i-1
      A[i+1] = chave
```

(11)

Invariante principal (i1):

No começo de cada iteração do laço da linha (1) o subvetor $A[1..j-1]$ está ordenado.

Suponha que o invariante é valido. Então a corretude do algoritmo é "evidente".

No início da última iteração temos $j = n+1$. Assim, do invariante segue que o (sub)vetor $A[1..n]$ está ordenado.

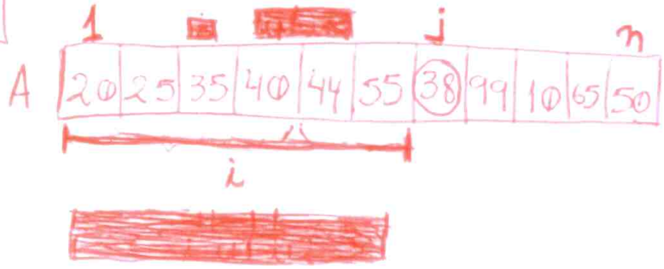
(12) Invariantes auxiliares

No início da linha 5 valem os seguintes invariantes:

(i2) $A[1..i]$ e $A[i+2..j]$ é crescente

(i3) $A[1..i] \leq A[i+2..j]$

(i4) $A[i+2..j] >$ chave



Invariante forte (i1'):

No começo de cada iteração do laço da linha 1, o subvetor $A[1..j-1]$ é uma permutação do subvetor original $A[1..j-1]$.

(13)

Invariantes (i2), (i3), (i4)
+ condição de parada da linha 5
+ atribuição da linha 7

$\Rightarrow$ invariante (i1')

Esboço da demonstração (de i1')

- Validade na primeira iteração: neste caso, temos $j=2$ e o invariante simplesmente afirma que $A[1..1]$ está ordenado, o que é evidente.
- Validade de uma iteração ($j>2$): segue da discussão anterior, os elementos maiores que a chave são "empurrados" para seus lugares corretos e esta é colocada no espaço vazio. Mas, uma demonstração mais formal deste fato exigiria uma prova dos invariantes auxiliares do laço interno.

(14)

- Na última iteração: temos $j=n+1$ e logo $A[1..n]$ está ordenado. Portanto, o algoritmo está correto.

Análise de corretude do algoritmo (iterativo) de Euclides para cálculo do MDC

(Exercício!)