Inducao para desenho de algoritmos

Inducao para desenho de algoritmos

Indução pode servir como base para se criar um algoritmo

Inducao para desenho de algoritmos

Indução pode servir como base para se criar um algoritmo

$$\text{hipotese indutiva} \equiv \text{chamada recursiva}$$

Inducao para desenho de algoritmos

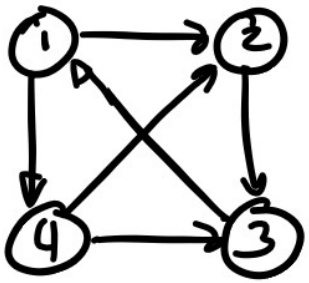
Proposição

Ao final de qualquer campeonato em que todos os $n \geq 2$ *participantes jogaram entre si, é possível, independentemente dos resultados, ordenar os participantes como* $p_1, p_2, \ldots, p_n$ *de forma que, para todo* $i \in \{1, 2, \ldots, n-1\}$, p_i *vence* p_{i+1}

Inducao para desenho de algoritmos

Proposição

Ao final de qualquer campeonato em que todos os $n \geq 2$ *participantes jogaram entre si, é possível, independentemente dos resultados, ordenar os participantes como* $p_1, p_2, \ldots, p_n$ *de forma que, para todo* $i \in \{1, 2, \ldots, n-1\}$, p_i *vence* p_{i+1}



Inducao para desenho de algoritmos

Proposição

Ao final de qualquer campeonato em que todos os $n \geq 2$ *participantes jogaram entre si, é possível, independentemente dos resultados, ordenar os participantes como* $p_1, p_2, \ldots, p_n$ *de forma que, para todo* $i \in \{1, 2, \ldots, n-1\}$, p_i *vence* p_{i+1}

Vamos primeiro provar isso por indução fraca no número de jogadores

Inducao para desenho de algoritmos

Proposição

Ao final de qualquer campeonato em que todos os $n \geq 2$ participantes jogaram entre si, é possível, independentemente dos resultados, ordenar os participantes como $p_1, p_2, \ldots, p_n$ de forma que, para todo $i \in \{1, 2, \ldots, n-1\}$, p_i vence p_{i+1}

Vamos primeiro provar isso por indução fraca no número de jogadores

Prova: Caso base: $n = 2$. Caso Jogador 1 ganhou do Jogador 2 usamos a ordem

$$\text{Jogador } 1 \rightarrow \text{Jogador } 2;$$

caso contrário usamos

$$\text{Jogador } 2 \rightarrow \text{Jogador } 1$$

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto.

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto. Podemos usar a **hipótese indutiva** nos n jogadores (e seus jogos) restantes para obter uma ordem

$$p_1 \to p_2 \to \ldots \to p_n$$

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto. Podemos usar a **hipótese indutiva** nos n jogadores (e seus jogos) restantes para obter uma ordem

$$p_1 \to p_2 \to \ldots \to p_n$$

Aonde inserir p^* nessa ordem?

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto. Podemos usar a **hipótese indutiva** nos n jogadores (e seus jogos) restantes para obter uma ordem

$$p_1 \to p_2 \to \ldots \to p_n$$

Caso 1: p^* perdeu todos os jogos

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto. Podemos usar a **hipótese indutiva** nos n jogadores (e seus jogos) restantes para obter uma ordem

$$p_1 \to p_2 \to \ldots \to p_n$$

Caso 1: p^* perdeu todos os jogos, coloque no fim:

$$p_1 \to \ldots \to p_n \to p^*$$

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto. Podemos usar a **hipótese indutiva** nos n jogadores (e seus jogos) restantes para obter uma ordem

$$p_1 \to p_2 \to \ldots \to p_n$$

Caso 1: p^* perdeu todos os jogos, coloque no fim:

$$p_1 \to \ldots \to p_n \to p^*$$

Caso 2: Caso contrário, seja p_i o primeiro jogador tal que p^* ganhou de p_i

Inducao para desenho de algoritmos

Passo indutivo. Seja $n \geq 2$, suponha que propriedade valha para n. Precisamos provar que vale para $n + 1$

Considere $n + 1$ jogadores e o resultado dos seus jogos

Pegue um jogador p^* e retire ele do conjunto. Podemos usar a **hipótese indutiva** nos n jogadores (e seus jogos) restantes para obter uma ordem

$$p_1 \to p_2 \to \ldots \to p_n$$

Caso 1: p^* perdeu todos os jogos, coloque no fim:

$$p_1 \to \ldots \to p_n \to p^*$$

Caso 2: Caso contrário, seja p_i o primeiro jogador tal que p^* ganhou de p_i

Coloque p^* logo antes de p_i:

$$p_1 \to \ldots \to p_{i-1} \to p^* \to p_i \to \ldots \to p_n$$

Inducao para desenho de algoritmos

Note que p^* está numa posição válida: o anterior ganhou dele e ele ganhou do próximo

Inducao para desenho de algoritmos

Note que p^* está numa posição válida: o anterior ganhou dele e ele ganhou do próximo

Por definição p^* ganhou de p_i

Inducao para desenho de algoritmos

Note que p^* está numa posição válida: o anterior ganhou dele e ele ganhou do próximo

Por definição p^* ganhou de p_i

E se $i > 1$, então p^* perdeu de p_{i-1}: caso contrário i não seria o menor índice tal que p^* ganhou de i

Inducao para desenho de algoritmos

Note que p^* está numa posição válida: o anterior ganhou dele e ele ganhou do próximo

Por definição p^* ganhou de p_i

E se $i > 1$, então p^* perdeu de p_{i-1}: caso contrário i não seria o menor índice tal que p^* ganhou de i

(Note que se $i = 1$ então p^* é o primeiro na ordem, e não precisamos mostrar que perdeu do anterior)

Inducao para desenho de algoritmos

Q: Baseado nessa prova, qual seria um algoritmo para criar essa ordem, dado n jogadores e seus jogos?

Inducao para desenho de algoritmos

Q: Baseado nessa prova, qual seria um algoritmo para criar essa ordem, dado n jogadores e seus jogos?

(Imaginamos que cada jogador é uma estrutura que guarda de quem ele ganhou e quem ganhou dele)

Inducao para desenho de algoritmos

Função Torneio($J1, J2, \ldots, Jn$) # conjunto de n jogadores
  If $n = 2$
    If $J1$ **ganhou de** $J2$
      Return sequencia $J1, J2$
    Else
      Return sequencia $J2, J1$
  Else
    seq $\leftarrow$ Torneio($J1, J2, \ldots, J_{n-1}$) # retirou jogador J_n
    If J_n não ganhou de nenhum jogador
      Insira J_n no fim da sequencia *seq*
    Else
      $p \leftarrow$ primeiro jogador em *seq* tal que J_n ganhou de p_i
      Insira J_n logo antes de p na sequencia *seq*
    Return seq

Corretude de algoritmos iterativos

Algoritmos iterativos

Podemos também usar indução para provar corretude de algoritmos iterativos (não-recursivos)

Algoritmos iterativos

Podemos também usar indução para provar corretude de algoritmos iterativos (não-recursivos)

$$\text{hipótese indutiva} \equiv \text{invariante do } \textbf{For}$$

Algoritmos iterativos

Podemos também usar indução para provar corretude de algoritmos iterativos (não-recursivos)

$$\text{hipótese indutiva} \equiv \text{invariante do } \textbf{For}$$

Vamos só fazer alguns exemplos simples

(O livro-texto do Rosen ve isso em muito mais profundidade)

Algoritmos iterativos

Função foo(n)
    $x \leftarrow 0$
    For $i = 1$ **to** n
        $x \leftarrow x + 3$ $(*)$
    End for
    Return x

Algoritmos iterativos

Função foo(n)
    $x \leftarrow 0$
    For $i = 1$ **to** n
        $x \leftarrow x + 3$ (*)
    End for
    Return x

Proposição (1)

Para todo inteiro $n \geq 1$, foo(n) retorna o número $3n$.

Algoritmos iterativos

Função foo(n)
    $x \leftarrow 0$
    For $i = 1$ **to** n
        $x \leftarrow x + 3$ $(*)$
    End for
    Return x

Proposição (1)

Para todo inteiro $n \geq 1$, *foo*(n) *retorna o número* $3n$.

Proposição (2)

Para todo $i \geq 1$, *ao fim da* i*-ésima iteração do* ***For*** *(após executar* $(*)$*),* $x = 3i$

Algoritmos iterativos

Função foo(n)
    $x \leftarrow 0$
    For $i = 1$ **to** n
        $x \leftarrow x + 3$ $(*)$
    End for
    Return x

Proposição (1)

Para todo inteiro $n \geq 1$, *foo*(n) *retorna o número* $3n$.

Proposição (2)

Para todo $i \geq 1$, *ao fim da* i*-ésima iteração do* ***For*** *(após executar* $(*)$*),* $x = 3i$

Proposição 2 implica na Proposição 1, pois *foo*(n) retorna valor de x logo após a n-ésima iteração do **For**

Proposição (2)

Para todo $i \geq 1$, ao fim da i-ésima iteração do ***For*** *(após executar $(*)$), $x = 3i$* *(caso essa iteração exista)*

Prova: Por indução fraca em i, número da iteração

Proposição (2)

Para todo $i \geq 1$, ao fim da i-ésima iteração do ***For*** *(após executar $(*)$), $x = 3i$* *(caso essa iteração exista)*

Prova: Por indução fraca em i, número da iteração

Caso base: $i = 1$. Como x inicia como 0, verificamos que ao fim da primeira iteração do **For** $x = 3$

Proposição (2)

Para todo $i \geq 1$, ao fim da i-ésima iteração do ***For*** *(após executar $(*)$), $x = 3i$* *(caso essa iteração exista)*

Prova: Por indução fraca em i, número da iteração

Caso base: $i = 1$. Como x inicia como 0, verificamos que ao fim da primeira iteração do **For** $x = 3$

Passo indutivo. Seja $i \geq 1$, suponha verdade para a i-ésima iteração. Precisamos provar para a iteração $i + 1$

Proposição (2)

*Para todo $i \geq 1$, ao fim da i-ésima iteração do **For** (após executar $(*)$), $x = 3i$* *(caso essa iteração exista)*

Prova: Por indução fraca em i, número da iteração

Caso base: $i = 1$. Como x inicia como 0, verificamos que ao fim da primeira iteração do **For** $x = 3$

Passo indutivo. Seja $i \geq 1$, suponha verdade para a i-ésima iteração. Precisamos provar para a iteração $i + 1$

Devido ao algoritmo

$$(x \text{ ao fim da iteração } i + 1) = (x \text{ ao fim da iteração } i) + 3$$

Proposição (2)

*Para todo $i \geq 1$, ao fim da i-ésima iteração do **For** (após executar $(*)$), $x = 3i$* *(caso essa iteração exista)*

Prova: Por indução fraca em i, número da iteração

Caso base: $i = 1$. Como x inicia como 0, verificamos que ao fim da primeira iteração do **For** $x = 3$

Passo indutivo. Seja $i \geq 1$, suponha verdade para a i-ésima iteração. Precisamos provar para a iteração $i + 1$

Devido ao algoritmo

$$(x \text{ ao fim da iteração } i+1) = (x \text{ ao fim da iteração } i) + 3$$
$$\overset{hip\ ind}{=} 3i + 3 \ = \ 3(i+1)$$

Isso conclui o passo indutivo. Fim de prova

Uma coisa interessante dessa análise é que precisamos criar uma proposição auxiliar para o **For**

Fibonacci

Definição

A sequencia de Fibonacci $F_0, F_1, \ldots,$ *é definida assim:*

- $F_0 = 0, \quad F_1 = 1$
- $F_i = F_{i-1} + F_{i-2}$ *para* $i \geq 2$

Considere o seguinte algoritmo para calcular F_n

Função Fib(n)
    $F \leftarrow$ vetor com posição $F[0]$ até $F[n]$
    $F[0] \leftarrow 0, \quad F[1] \leftarrow 1$
    For $i = 2$ to n
        $F[i] = F[i-1] + F[i-2]$
    End For
    Return $F[n]$

Fibonacci

Função Fib(n)
    $F \leftarrow$ vetor com posição $F[0]$ até $F[n]$
    $F[0] \leftarrow 0, \quad F[1] \leftarrow 1$
    For $i = 2$ to n
        $F[i] = F[i-1] + F[i-2]$
    End For
    Return $F[n]$

Proposição (1)

Para todo $n \geq 0$, $Fib(n)$ retorna o número de Fibonacci F_n

Fibonacci

Função Fib(n)
    $F \leftarrow$ vetor com posição $F[0]$ até $F[n]$
    $F[0] \leftarrow 0, \quad F[1] \leftarrow 1$
    For $i = 2$ to n
        $F[i] = F[i-1] + F[i-2]$
    End For
    Return $F[n]$

Proposição (1)

Para todo $n \geq 0$, $Fib(n)$ retorna o número de Fibonacci F_n

É suficiente provar a seguinte proposição

Fibonacci

Função Fib(n)
    $F \leftarrow$ vetor com posição $F[0]$ até $F[n]$
    $F[0] \leftarrow 0, \quad F[1] \leftarrow 1$
    For $i = 2$ to n
        $F[i] = F[i-1] + F[i-2]$
    End For
    Return $F[n]$

Proposição (1)

Para todo $n \geq 0$, $Fib(n)$ retorna o número de Fibonacci F_n

É suficiente provar a seguinte proposição

Proposição (2)

Para todo $0 \leq j \leq n$, ao fim do algoritmo $F[j]$ é igual ao número de Fibonacci F_j

Prova Proposição 2: Por indução forte em j

Casos base: $j = 0, 1$**.** Em ambos os casos, o algoritmo seta o valor $F[i]$ corretamente na segunda linha, e não modifica durante execução

Prova Proposição 2: Por indução forte em j
Casos base: $j = 0, 1$**.** Em ambos os casos, o algoritmo seta o valor $F[i]$ corretamente na segunda linha, e não modifica durante execução

Passo indutivo. Seja $1 \leq j \leq n-1$, suponha que a propriedade valha pra todo índice $\leq j$, ou seja
$F[0] = F_0,\ F[1] = F_1,\ \ldots,\ F[j] = F_j$.

Prova Proposição 2: Por indução forte em j
Casos base: $j = 0, 1$**.** Em ambos os casos, o algoritmo seta o valor $F[i]$ corretamente na segunda linha, e não modifica durante execução

Passo indutivo. Seja $1 \leq j \leq n-1$, suponha que a propriedade valha pra todo índice $\leq j$, ou seja
$F[0] = F_0,\ F[1] = F_1,\ \ldots,\ F[j] = F_j$.

Precisamos provar para $j+1$, ou seja que $F[j+1] = F_{j+1}$

Prova Proposição 2: Por indução forte em j
Casos base: $j = 0, 1$**.** Em ambos os casos, o algoritmo seta o valor $F[i]$ corretamente na segunda linha, e não modifica durante execução

Passo indutivo. Seja $1 \leq j \leq n-1$, suponha que a propriedade valha pra todo índice $\leq j$, ou seja
$F[0] = F_0,\ F[1] = F_1,\ \ldots,\ F[j] = F_j$.

Precisamos provar para $j+1$, ou seja que $F[j+1] = F_{j+1}$

Na iteração $i = j+1$ do **For** o algoritmo seta

$$F[\underbrace{j+1}_{i}] = F[\underbrace{\ j\ }_{i-1}] + F[\underbrace{j-1}_{i-2}]$$
$$\overset{hip\_ind}{=} F_j + F_{j-1} \overset{def}{=} F_{j+1}$$

Prova Proposição 2: Por indução forte em j
Casos base: $j = 0, 1$**.** Em ambos os casos, o algoritmo seta o valor $F[i]$ corretamente na segunda linha, e não modifica durante execução

Passo indutivo. Seja $1 \leq j \leq n-1$, suponha que a propriedade valha pra todo índice $\leq j$, ou seja
$F[0] = F_0,\ F[1] = F_1,\ \ldots,\ F[j] = F_j$.

Precisamos provar para $j+1$, ou seja que $F[j+1] = F_{j+1}$

Na iteração $i = j+1$ do **For** o algoritmo seta

$$F[\underbrace{j+1}_{i}] = F[\underbrace{\ j\ }_{i-1}] + F[\underbrace{j-1}_{i-2}]$$
$$\overset{hip\_ind}{=} F_j + F_{j-1} \overset{def}{=} F_{j+1}$$

Note que depois dessa iteracao o algoritmo nao modifica $F[j+1]$

Prova Proposição 2: Por indução forte em j
Casos base: $j = 0, 1$**.** Em ambos os casos, o algoritmo seta o valor $F[i]$ corretamente na segunda linha, e não modifica durante execução

Passo indutivo. Seja $1 \leq j \leq n-1$, suponha que a propriedade valha pra todo índice $\leq j$, ou seja
$F[0] = F_0,\ F[1] = F_1,\ \ldots,\ F[j] = F_j$.

Precisamos provar para $j+1$, ou seja que $F[j+1] = F_{j+1}$

Na iteração $i = j+1$ do **For** o algoritmo seta

$$F[\underbrace{j+1}_{i}] = F[\underbrace{\ j\ }_{i-1}] + F[\underbrace{j-1}_{i-2}]$$
$$\overset{hip\_ind}{=} F_j + F_{j-1} \overset{def}{=} F_{j+1}$$

Note que depois dessa iteracao o algoritmo nao modifica $F[j+1]$

Isso conclui o passo indutivo. Fim de prova