Bioinformática

Ney Lemke

Departamento de Física e Biofísica

2009

Outline

Outline

Bibliografia

- JONES, N. C.; PEVZNER, P. A. “An Introduction to Bioinformatics Algorithms”, Cambridge,The MIT Press, 2004, 435pp.

Outline

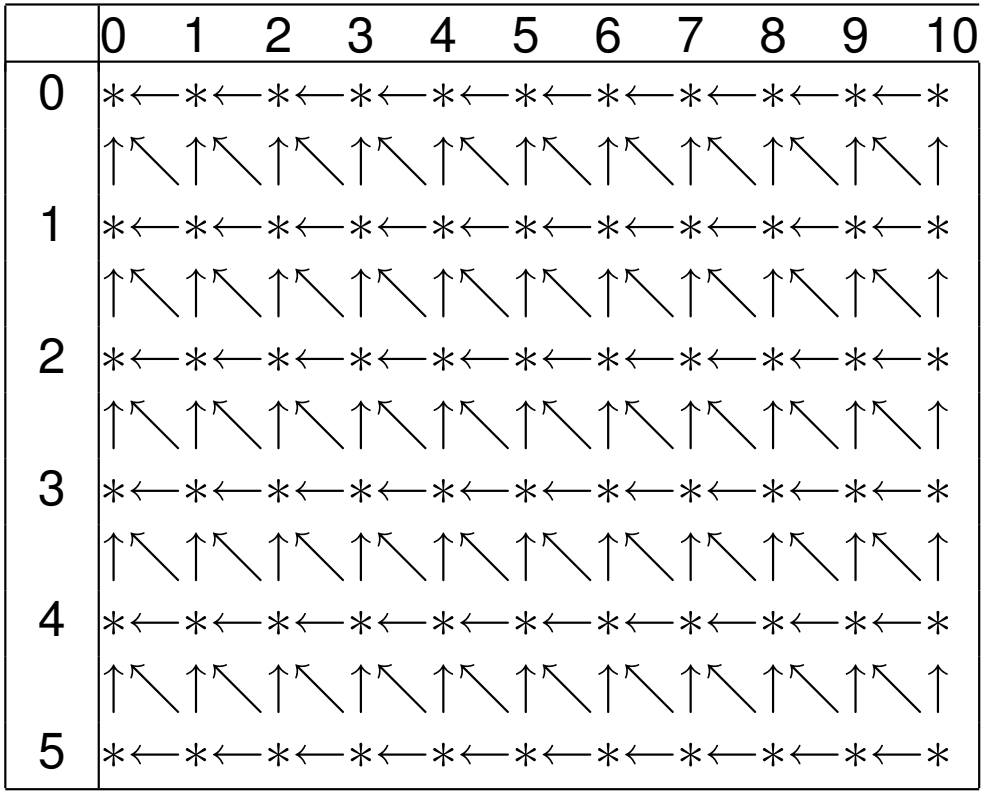
Introdução

- Importância do Raciocínio Algoritmico
- Compreensão das idéias Básicas
- Limitações e falhas

Exemplo

Alice e Bob estão envolvidos em um excitante jogo. Temos duas pilhas com 10 pedras, cada jogador pode ou retirar uma pedra de uma pilha ou uma pedra de cada uma das pilhas. Alice começa jogando e ganha quem retirar a última pedra.

Análise do Jogo



Análise do Jogo

	0	1	2	3	4	5	6	7	8	9	10
0	L	W	L	W	L	W	L	W	L	W	L
1	W	W	W	W	W	W	W	W	W	W	W
2	L	W	L	W	L	W	L	W	L	W	L
3	W	W	W	W	W	W	W	W	W	W	W
4	L	W	L	W	L	W	L	W	L	W	L
5	W	W	W	W	W	W	W	W	W	W	W
6	L	W	L	W	L	W	L	W	L	W	L
7	W	W	W	W	W	W	W	W	W	W	W
8	L	W	L	W	L	W	L	W	L	W	L
9	W	W	W	W	W	W	W	W	W	W	W
10	L	W	L	W	L	W	L	W	L	W	L

Outline

O que é um algoritmo

Def. Um algoritmo é uma seqüência de instruções que devemos utilizar para resolver um dado problema.
Para podermos descrever um algoritmo em geral fazemos uso de uma lingugem chamada de pseudo-código.

Atribuição

Atribuição $a \leftarrow b$

Efeito A variável a recebe o valor de b.

Aritmética

Formato $a + b$, $a - b$, a/b, $a \times b$

Efeito Realiza a operação.

Condicional

if A is true **then**
    B
else
    C
end if

For

for i←b to *n* **do**
  B
end for

while

while A é verdade **do**
    B
end while

Acesso à vetores

Exemplo:

1: $F_1 \leftarrow 1$
2: $F_2 \leftarrow 1$
3: **for** $i \leftarrow 3$ **to** n **do**
4: $\quad F_i \leftarrow F_{i-1} + F_{i-2}$
5: **end for**

Outline

Receita de Bolo

1: PREAQUECAFORNO(425)
2: *recheio*← BATARECHEIO(*abóbora, açucar, sal, temperos, ovos, leite*)
3: *torta* ← ASSENTE(*massa, recheio*)
4: **while** garfo sai molhado é verdade **do**
5: COZINHE(pie)
6: **end while**
7: **output** “Torta Pronta”
8: **return** *pie*

BATARECHEIO

1: *pote* ← Pegue um pote do armário
2: ADICIONE(*abóbora, pote*)
3: ADICIONE(*açucar, pote*)
4: ADICIONE(*sal, pote*)
5: ADICIONE(*temperos, pote*)
6: BATA(pote)
7: ADICIONE(*ovos, pote*)
8: ADICIONE(*leite, pote*)
9: BATA(pote)
10: *recheio*← Conteúdo de *pote*
11: **return** *recheio*

Algoritmos Biológicos

- Helicases se ligam a origem de replicação
- Helicase separa as duas fitas do DNA
- Ligação dos *primers*
- Ligação da DNA polimerase a cada uma das fitas do DNA
- DNA Polimerase complementa de forma contínua uma das fitas
- DNA Polimerase replica de forma descontínua a outra fita
- DNA ligase repare os buracos na segunda fita
- As duas fitas se separam

Algoritmos para Computadores

COPIASTRING(**s**,*n*)

for i← 1 to *n* **do**

$t_i \leftarrow s_i$

return t

end for

Exercício

Escrevam um algoritmo para escolha de qual revista um paper deve ser submetido.

Problema do Troco (USA)

Converter alguma quantidade de dinheiro M no menor número possível de moedas. Ou seja $M = 25q + 10d + 5n + p$ e $q + d + n + p$ deve ser mínimo.

UsChange(M)

while $M > 0$ **do**
  $c \leftarrow$ Maior moeda menor ou igual a M
  Entregue a moeda c ao cliente
  $M \leftarrow M - c$
end while

UsChange(M)

1: $r \leftarrow M$
2: $q \leftarrow r/25$
3: $r \leftarrow r - 25 * q$
4: $d \leftarrow r/10$
5: $r \leftarrow r - 10 * d$
6: $n \leftarrow r/5$
7: $r \leftarrow r - 5 * n$
8: $p \leftarrow r$
9: **return** (q,d,n,p)

Problema do Troco

Converter alguma quantidade de dinheiro M no menor número possível de moedas. Considere um vetor de moedas $(c_1, \ldots, c_d)$ em ordem descrescente de valor e o troco caracterizado pela lista de inteiros: $(i_1, \ldots, i_d)$ Ou seja $M = c_1 * i_1 + \ldots c_d * i_d$ e $i_1 + \ldots i_d$ deve ser mínimo.

MELHORTROCO(M,**c**,d)

1: $r \leftarrow M$
2: **for** $k \leftarrow 1$ to d **do**
3: $\quad i_k \leftarrow r/c_k$
4: $\quad r \leftarrow r - c_k * i_k$
5: **end for**
6: **return** ($i_1, \ldots i_d$)

Discussão

- Algoritmos Corretos
- Contra-exemplo: Caso Americano troco para 40 c (moedas: 25,20,10,5,1).
- Solução: Procurar por todas as soluções e escolher a melhor.

Outline

Algoritmos Recursivos

Def. Algoritmos recursivos são aqueles que chamam a si mesmos.
Exemplo: Fibonacci Recursivo FIBONACCIRECURSIVO

if $n = 1$ ou $n = 2$ **then**
  return 1
else
  $a \leftarrow$ FIBONACCIRECURSIVO$(n-1)$
  $b \leftarrow$ FIBONACCIRECURSIVO$(n-2)$
  return $a + b$
end if

Algoritmos Recursivos e Iterativos

- Algoritmos Recursivos podem ser eficazes em tempo de computação, mas em geral são gulosos em termos de memória.
- São mais difíceis de entender e de “debugar”.

Outline

Complexidade dos Algoritmos

Considere um problema caracterizado por um número n chamado de tamanho do problema.
A complexidade de um algoritmo é a dependência do tempo de computação em termos de n. Em geral só queremos saber a complexidade no limite de n grande e consideramos os casos

n ou $n \log n$ Problemas lineares e portanto eficientes, exemplos ordenação de um vetor.

n^2 Problemas quadráticos mais custosos, mas que podem ser resolvidos exatamente na maioria dos casos de interesse.

n^α Problemas ditos polinomiais, podem em tese ser resolvidos, mas rapidamente se tornam muito custosos.

Outros Problemas ditos não tratáveis algoritmicamente, ou seja a solução exata levaria em muitos casos milhares de anos.

Tipos de Algoritmos

Busca Exaustiva Procura por todos os casos a solução do problema.

Branch and Bound Divide o problema em classes e ataca as classes promissoras.

Divida e Conquiste Estratégia de devidir um problema em partes e atacar cada uma das partes em separaddo.

Algoritmos Aleatórios Utilizam números aleatórios para propor soluções para problemas (método predileto dos alunos).