

Laboratório de Informática

Leopoldo Teixeira
leo@leopoldomt.com

O que é mesmo um
algoritmo?

Um algoritmo é uma série ordenada de passos finita, não-ambígua e executável.

Conceito de Algoritmo

- Esta definição exige que o conjunto de passos em um algoritmo seja ordenado, ou seja, os passos de um algoritmo devem possuir uma estrutura bem estabelecida em termos de ordem na qual são executadas.
- Não implica que os passos sejam executados em seqüência. Ex: algoritmos paralelos, algoritmos executados por circuitos (flip-flop).

Algoritmo?

- *Construir uma lista de todos os inteiros positivos.*
- Impossível realizar essa instrução. Nenhum conjunto de instruções que inclua esta instrução seria um algoritmo, pois ele não é executável (não-efetivo). Dizer que um passo de um algoritmo é efetivo significa que ele pode ser realizado.
- Definição também exige que um algoritmo defina um processo que termine. A execução de um algoritmo deve sempre levar o processamento a um estado de término.
- Quais são os limites reais dos algoritmos e das máquinas?
 - Teoria da Computação.

Natureza abstrata

- Diferença entre algoritmo e sua representação. O algoritmo é abstrato e distinto de suas representações. (possui várias formas)
 - $F = (9/5)C + 32$
 - Multiplicar a temperatura, lida em graus Celsius, por 9/5, e então somar 32 ao produto assim obtido.
 - Circuito eletrônico.
- A distinção entre um algoritmo e sua representação apresenta um problema quando tentamos comunicá-lo. (nível de detalhe)
- Diferença entre dois conceitos inter-relacionados – programa e processos.
 - programa é uma representação de um algoritmo.
 - processo é a atividade de executar um algoritmo.

Representação de Algoritmos - Primitivas

- Representação requer uma forma de linguagem.
- Primitivas - conjunto bem-definido de elementos funcionais básicos com os quais podem ser construídas representações de algoritmos.
- Linguagem de programação – conjunto de primitivas + regras
(maneira que as primitivas podem ser combinadas para representar idéias mais complexas)
- Cada primitiva consiste em duas partes: sua sintaxe e sua semântica.
 - Sintaxe refere-se à representação simbólica das primitiva.
 - Semântica refere-se ao conceito representado, ou seja ao significado da primitiva

Pseudocódigo

- Sistema notacional no qual as idéias podem ser informalmente expressas durante o processo de desenvolvimento do algoritmo.
- Nome \leftarrow expressão
- Fundos \leftarrow SaldoCC + Poupança
- Estrutura semântica
 - Se o produto interno bruto tiver aumentado, compre ações ordinárias; caso contrário, venda ações ordinárias.
 - **se** (condição) **então** (atividade) **senão** (atividade)

Pseudocódigo

- Enquanto houver ingressos para vender, permanecer vendendo ingressos.
 - **enquanto** (condição) **faça** (ação)
 - **enquanto** (há ingressos) **faça** (venda)
 - Indentação facilita a leitura do programa
 - se** (produto está sujeito a imposto)
 - então** (**se** (preço > limite)
 - então** (pagar x)
 - senão** (pagar y)
 - senão** (pagar z)
- é mais fácil de ser entendido do que:
- **se** (produto está sujeito a imposto) **então** (se (preço > limite) **então** (pagar x) **senão** (pagar y)) **senão** (pagar z)

Procedimentos

- Queremos usar nosso pseudocódigo para descrever as atividades que podem ser usados como ferramentas abstratas em outras aplicações.
- Há uma variedade de termos para tais unidades do programa, incluindo subprograma, sub-rotina, **procedimento**, módulo e **função**, cada um com sua própria variação de significado.
- Declaramos um procedimento da seguinte maneira:
 - **procedimento** nome
- onde nome é o nome específico do procedimento. Seguimos esta declaração introdutória com os comandos que definem a ação do procedimento.

Procedimentos

- Devemos conceber procedimentos da forma mais genérica possível.
- Um procedimento para ordenar listas de nomes deve ser projetado para ordenar qualquer lista, não uma lista em particular.
- Para tanto, utilizamos parâmetros, para identificar entradas aos procedimentos. Por exemplo:
 - **procedimento** Ordenar (Lista)

Desenvolvimento de programas

- Descobrir o algoritmo
- Representá-lo em forma de programa.
- O maior desafio em termos de desenvolvimento é encontrar um método de solucionar um problema. Entender como os algoritmos são descobertos é entender o próprio processo de resolução de problemas (idéias).
- A arte da resolução de problemas
 - Em última instância, deseja-se reduzir o processo de resolução de problemas a um algoritmo autônomo, mas comprovou-se que essa meta é impossível. (Existem problemas que não têm solução algorítmicas).

Processo de resolução de problemas (G. Polya)

- Fase 1. Entender o problema.
- Fase 2. Construir um plano para solucionar o problema.
- Fase 3. Colocar o plano em funcionamento.
- Fase 4. Avaliar a solução quanto à precisão e ao seu potencial como ferramenta para solucionar outros problemas.

Desenvolvendo programas...

- Fase 1. Compreender o problema
- Fase 2. Adquirir uma ideia da forma que um procedimento algorítmico poderia resolver o problema
- Fase 3. Formular o algoritmo e representá-lo na forma de um programa
- Fase 4. Avaliar o programa quanto à precisão e ao seu potencial como ferramenta para solucionar outros problemas

Descobrendo algoritmos

- As fases não são passos a seguir quanto se tenta resolver um problema, mas fases a serem seguidas algum dia, durante o processo de solução.
- As quatro fases de Polya não são necessariamente executadas em seqüência.
- Estamos discutindo sobre a forma como os problemas são resolvidos, e não sobre como gostaríamos que fossem resolvidos.
- Queremos eliminar a perda de tempo com o método de tentativa e erro

Um exemplo

- *Uma pessoa (X) é encarregada da tarefa de determinar as idades dos três filhos de Y. Y diz a X que o produto da idade das crianças é 36. Depois de considerar esta pista, X responde que outra pista é necessária, ao que Y diz a X a soma das idades dos filhos. Mais uma vez, X responde que outra pista é necessária, de modo que Y diz que seu filho mais velho toca piano. Depois de ouvir esta pista, X diz as idades dos três filhos.*
- Qual é a idade das crianças?

Solução

Produto 36

(1,1,36) (1,6,6)

(1,2,18) (2,2,9)

(1,3,12) (2,3,6)

(1,4,9) (3,3,4)

Soma das Idades

$$1 + 1 + 36 = 38$$

$$1 + 2 + 18 = 21$$

$$1 + 3 + 12 = 16$$

$$1 + 4 + 9 = 14$$

$$1 + 6 + 6 = 13$$

$$2 + 2 + 9 = 13$$

$$2 + 3 + 6 = 11$$

$$3 + 3 + 4 = 10$$

Outro exemplo

- Antes de A,B,C e D participarem de uma corrida, eles fizeram as seguintes previsões:
 - A previu que B ganharia.
 - B previu que D seria o último.
 - C previu que A seria o terceiro.
 - D previu que a previsão de A estaria correta.
- Apenas uma destas previsões deu certo, e esta foi feita pelo vencedor da corrida.
- Em que ordem A, B, C e D terminaram a corrida?

Solucionando...

- Previsões A e D são equivalentes, e só uma previsão era verdadeira.
 - Subimos o primeiro degrau e obter a solução completa para o nosso problema será simplesmente uma questão de estender a partir daí o nosso conhecimento
- Previsão A é falso, B também não pode ter sido o vencedor.
- A única escolha que resta para o vencedor é C.
- C ganhou a corrida logo A chegou em terceiro lugar.
 - Opção CBAD ou então CDAB.
- Como a previsão de B é falsa a solução é CDAB.

Diversos enfoques

- Trabalhar o problema de marcha a ré — consiste em encontrar um modo de produzir uma certa saída a partir de uma dada entrada.
- Procurar um problema relacionado que seja mais fácil de resolver ou já tenha sido resolvido antes, e então tentar solucionar o problema usando a mesma forma de resolução.
- Técnica do refinamento sucessivo – consiste essencialmente em não tentar realizar imediatamente uma tarefa inteira, em todos os detalhes.
 - Decompor o problema em vários sub-problemas.
 - Dividir e conquistar.
 - Metodologias: top-down – desenvolvimento leva do geral para o particular.
bottom-up – desenvolvimento leva do específico para o geral.
- A descoberta de algoritmos continua sendo uma ARTE desafiadora.

Estruturas iterativas - Busca Sequencial

procedure Search (List, TargetValue)

if (List empty)

then

(Declare search a failure)

else

(Select the first entry in List to be TestEntry;

while (TargetValue > TestEntry and
there remain entries to be considered)

do (Select the next entry in List as TestEntry.);

if (TargetValue = TestEntry)

then (Declare search a success.)

else (Declare search a failure.)

) end if

Componentes de Controle

- Iniciação:
 - Estabelecer um estado inicial que será modificado para atingir uma condição de terminação.
- Teste:
 - Comparar o estado corrente com a condição de terminação e finalizar a iteração se forem iguais.
- Modificação:
 - Alterar o estado de modo a convergir para a condição de terminação.

Controle Adequado

- Característica fundamental para que haja controle adequado da iteração:
 - Os passos de iniciação e de modificação conduzem a uma condição terminal apropriada.
 - É preciso ter certeza absoluta de sua correta aplicação sempre que se projetar uma estrutura iterativa.

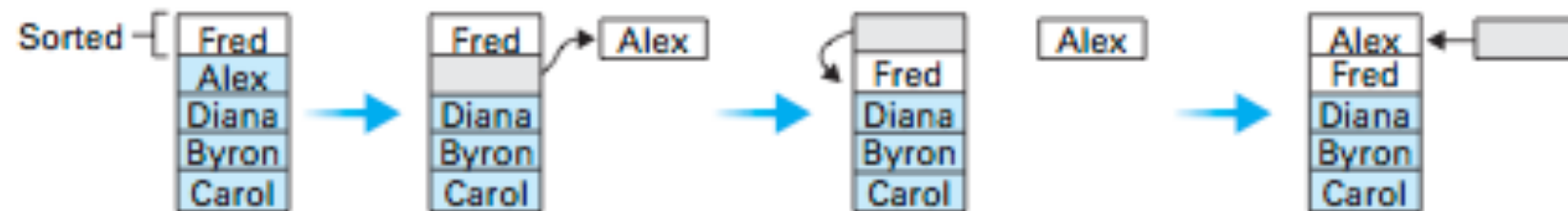
Número = 1;

enquanto (Número \neq 6) **faça**
(Número = Número + 2)

- Há duas estruturas iterativas comuns:
 - **enquanto** (condição) **faça** (ação) — laço pré-teste
 - **repete** (ação) **até** (condição) — laço pós-teste

Initial list:

Fred
Alex
Diana
Byron
Carol



procedure Sort (List)

$N \leftarrow 2$;

while (the value of N does not exceed the length of List) **do**

 (Select the N th entry in List as the pivot entry;

 Move the pivot entry to a temporary location leaving a hole in List;

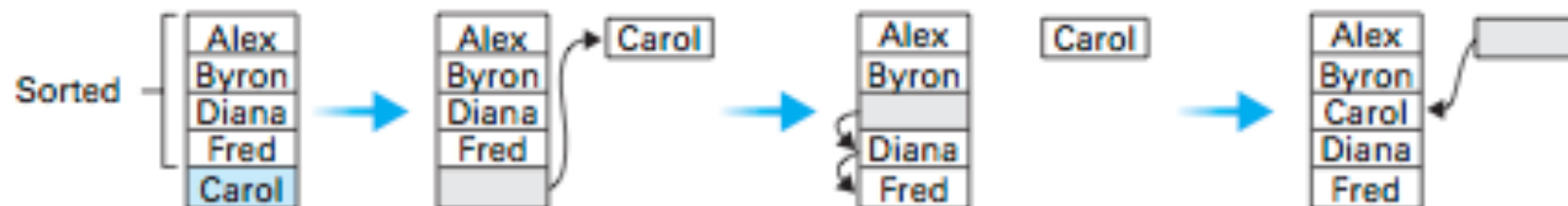
while (there is a name above the hole and that name is greater than the pivot)

do (move the name above the hole down into the hole
 leaving a hole above the name)

 Move the pivot entry into the hole in List;

$N \leftarrow N + 1$

)



Sorted list:

Alex
Byron
Carol
Diana
Fred

O que é recursão?

Estruturas recursivas

- Recursão envolve a repetição do conjunto de instruções como uma subtarefa de si própria.
- Laço envolve repetição de um conjunto de instruções de modo que o conjunto seja completado e então repetido.
- Importante definir o caso base, para que termine

Busca Binária

```
procedure Search (List, TargetValue)
if (List empty)
  then
    (Report that the search failed.)
  else
    [Select the "middle" entry in List to be the TestEntry;
    Execute the block of instructions below that is
    associated with the appropriate case.
    case 1: TargetValue = TestEntry
      (Report that the search succeeded.)
    case 2: TargetValue < TestEntry
      (Apply the procedure Search to see if TargetValue
       is in the portion of the List preceding TestEntry,
       and report the result of that search.)
    case 3: TargetValue > TestEntry
      (Apply the procedure Search to see if TargetValue
       is in the portion of List following TestEntry,
       and report the result of that search.)
    ] end if
```

We are here.

procedure Search (List, TargetValue)

if (List empty)

then (Report that the search failed.)

else

[Select the "middle" entry in List to be the TestEntry;
Execute the block of instructions below that is
associated with the appropriate case.

case 1: TargetValue = TestEntry

(Report that the search succeeded.)

case 2: TargetValue < TestEntry

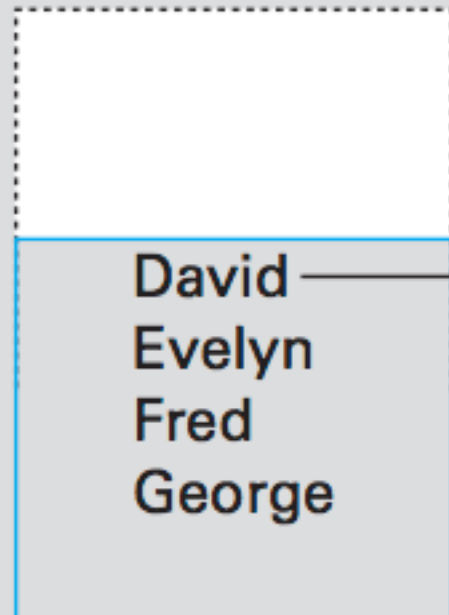
(Apply the procedure Search to see if TargetValue
is in the portion of the List preceding TestEntry,
and report the result of that search.)

case 3: TargetValue > TestEntry

(Apply the procedure Search to see if TargetValue
is in the portion of List following TestEntry,
and report the result of that search.)

] end if

List



(TestEntry)

procedure Search (List, TargetValue)

if (List empty)

then (Report that the search failed.)

else

[Select the "middle" entry in List to be the TestEntry;
Execute the block of instructions below that is
associated with the appropriate case.

case 1: TargetValue = TestEntry

(Report that the search succeeded.)

case 2: TargetValue < TestEntry

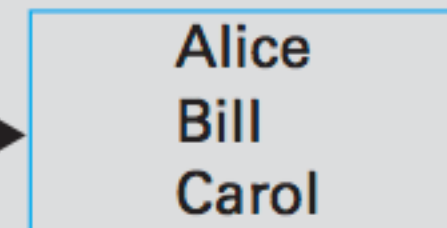
(Apply the procedure Search to see if TargetValue
is in the portion of the List preceding TestEntry,
and report the result of that search.)

case 3: TargetValue > TestEntry

(Apply the procedure Search to see if TargetValue
is in the portion of List following TestEntry,
and report the result of that search.)

] end if

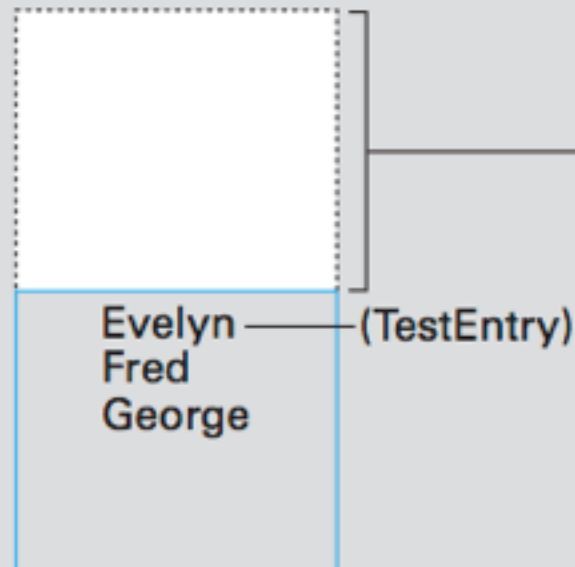
List



procedure Search (List, TargetValue)

```
if (List empty)
then (Report that the search failed.)
else
[Select the "middle" entry in List to be the TestEntry;
Execute the block of instructions below that is
associated with the appropriate case.
case 1: TargetValue = TestEntry
(Report that the search succeeded.)
case 2: TargetValue < TestEntry
(Apply the procedure Search to see if TargetValue
is in the portion of the List preceding TestEntry,
and report the result of that search.)
case 3: TargetValue > TestEntry
(Apply the procedure Search to see if TargetValue
is in the portion of List following TestEntry,
and report the result of that search.)
] end if
```

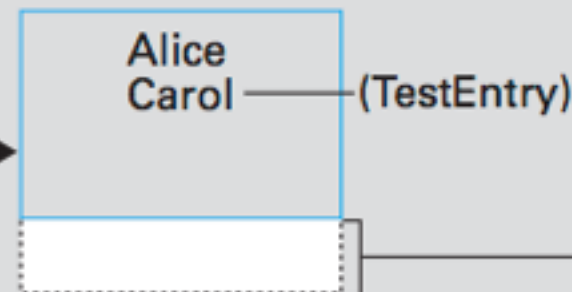
List



procedure Search (List, TargetValue)

```
if (List empty)
then (Report that the search failed.)
else
[Select the "middle" entry in List to be the TestEntry;
Execute the block of instructions below that is
associated with the appropriate case.
case 1: TargetValue = TestEntry
(Report that the search succeeded.)
case 2: TargetValue < TestEntry
(Apply the procedure Search to see if TargetValue
is in the portion of the List preceding TestEntry,
and report the result of that search.)
case 3: TargetValue > TestEntry
(Apply the procedure Search to see if TargetValue
is in the portion of List following TestEntry,
and report the result of that search.)
] end if
```

List

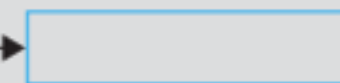


We are here.

procedure Search (List, TargetValue)

```
if (List empty)
then (Report that the search failed.)
else
[Select the "middle" entry in List to be the TestEntry;
Execute the block of instructions below that is
associated with the appropriate case.
case 1: TargetValue = TestEntry
(Report that the search succeeded.)
case 2: TargetValue < TestEntry
(Apply the procedure Search to see if TargetValue
is in the portion of the List preceding TestEntry,
and report the result of that search.)
case 3: TargetValue > TestEntry
(Apply the procedure Search to see if TargetValue
is in the portion of List following TestEntry,
and report the result of that search.)
] end if
```

List













Eficiência de Algoritmos

- Se considerarmos um registro de 30 mil estudantes em uma universidade
- Usando busca sequencial, na média teremos que investigar 15 mil registros. Se cada pesquisa dura 10 ms, a média de tempo seria 2.5 minutos
- Com busca binária, no máximo avaliamos 15 entradas. Se cada pesquisa dura 10 ms, a média de tempo seria 0,15 segundos
 - No entanto, temos que nos certificar da ordem...

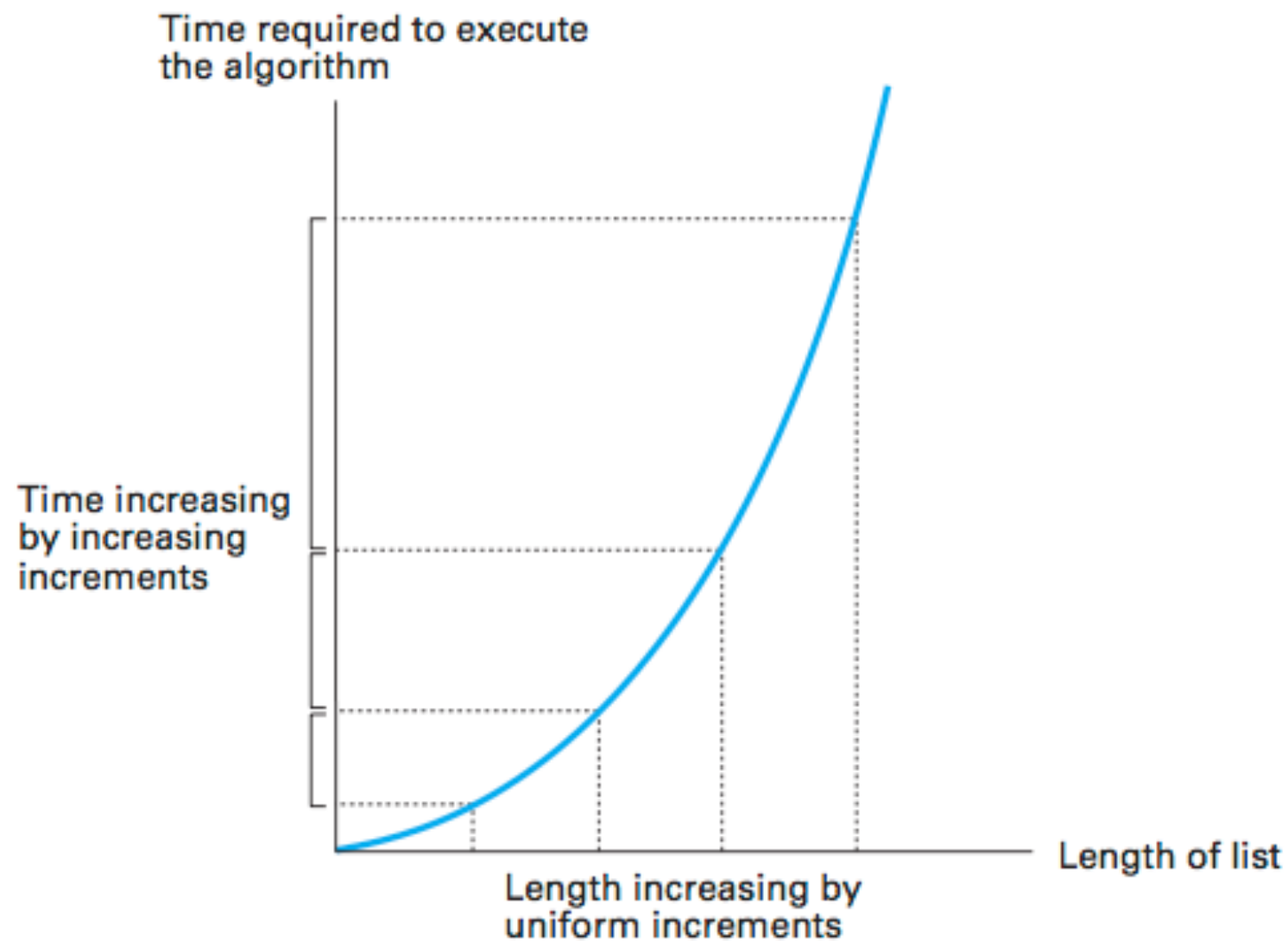
Análise de Algoritmos

- Envolve a avaliação do melhor caso, pior caso e caso médio dos algoritmos desenvolvidos
- Estas análises são feitas em contextos genéricos, para uma lista de tamanho N , por exemplo

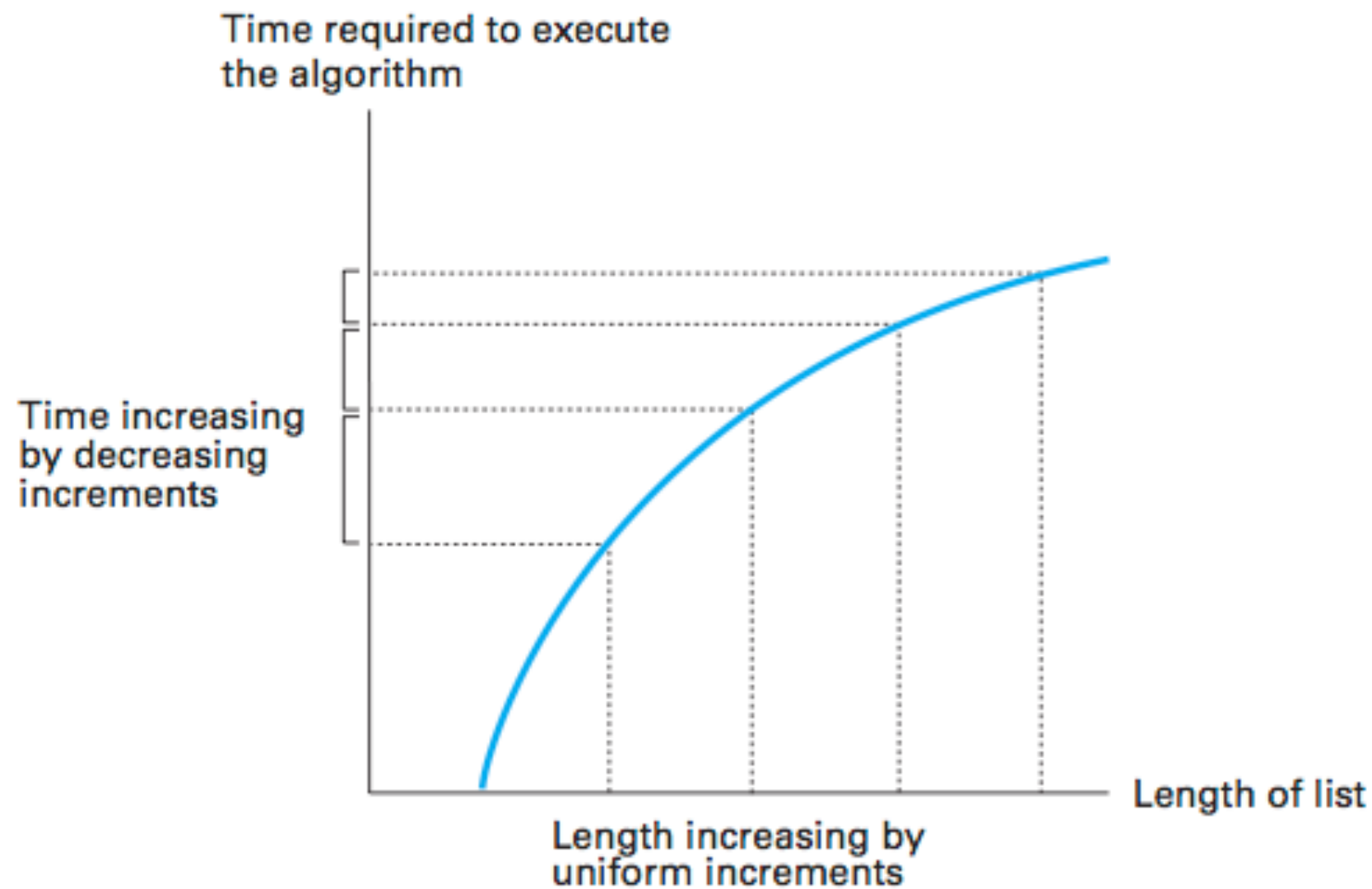
Insertion Sort - pior caso

Comparisons made for each pivot					
Initial list	1st pivot	2nd pivot	3rd pivot	4th pivot	Sorted list
Elaine David Carol Barbara Alfred	1  Elaine David Carol Barbara Alfred	3  David 2  Elaine Carol Barbara Alfred	6  Carol 5  David 4  Elaine Barbara Alfred	10  Barbara 9  Carol 8  David 7  Elaine Alfred	Alfred Barbara Carol David Elaine

Análise do pior caso - Insertion Sort



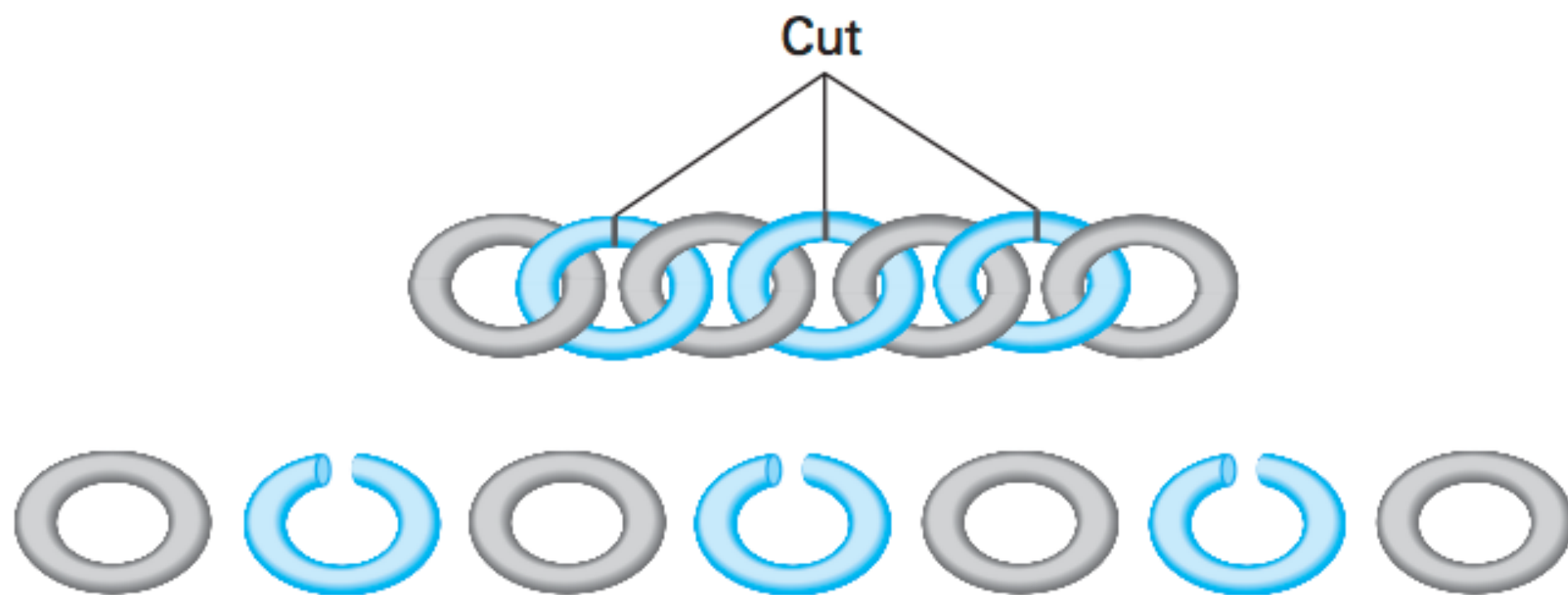
Análise do pior caso - Busca Binária



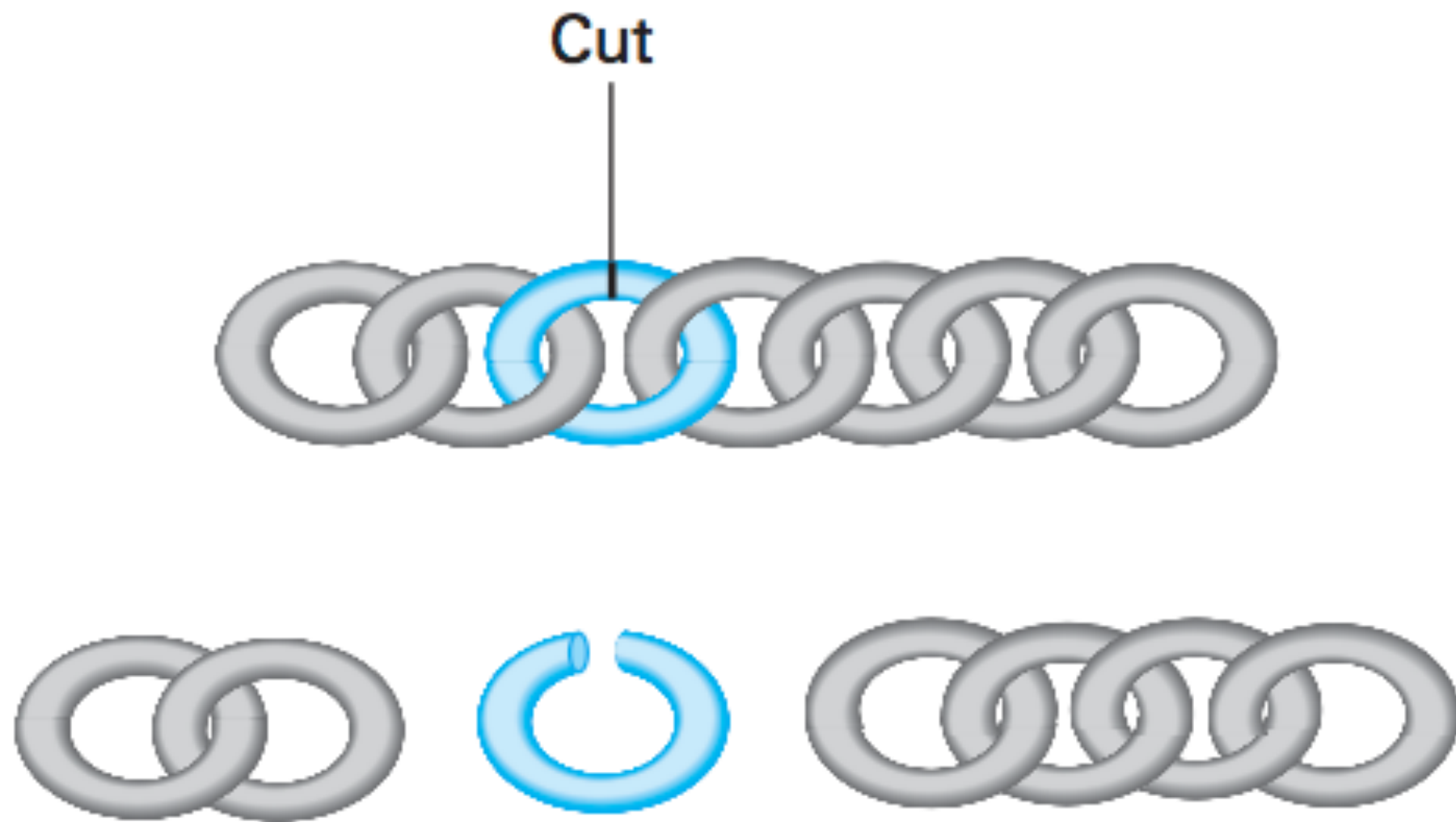
Corretude de Algoritmos

- Um viajante, com uma corrente de ouro formada por uma cadeia de sete argolas, deve ficar por sete noites em um hotel isolado.
- O aluguel por noite equivale a uma argola da corrente.
- Qual o número mínimo de argolas que devem ser cortadas de modo que o viajante possa pagar ao hoteleiro uma argola a cada manhã, sem antecipar o pagamento pela hospedagem?

Uma solução



Solução Correta



Como é impossível verificar completamente a precisão de programas complexos, em que circunstâncias, se houver, deve o criador de um programa ser responsabilizado por erros?

Suponha que um pacote de software é tão caro que ele está totalmente fora de sua faixa de preço. É ético copiá-lo para seu próprio uso?

(você não está 'roubando' o fornecedor de uma venda, pois não teria comprado o pacote de qualquer maneira.)

Trabalho em Grupo

- Escolher um tópico (sugestões a seguir) e fazer um estudo aprofundado do mesmo. Tal estudo aprofundado irá analisar a história, o estado da arte, desafios técnicos, o impacto social e uma visão para o futuro.
- Com base neste estudo, preparar uma apresentação de 20 minutos e um relatório no formato de ensaio.
- Os grupos podem ter até 5 pessoas. Após a formação do grupo e escolha do tema, devem enviar email para a lista da disciplina
- Datas das apresentações: **24/7** e **31/7**
- Prazo final para entrega do relatório: **31/7**

Sugestões de Temas

- Interação homem-máquina
- Engenharia de Software
- Sistemas de Banco de Dados
- Inteligência Artificial e Aprendizagem de Máquina
- Computação Gráfica
- Processamento Digital de Imagens
- Computação Distribuída/Paralela (Concorrência)
- Robótica
- Criptografia (Privacidade/Segurança/Autenticação)

Trabalho Individual

- Faça um ensaio sobre a vida e obra de um cientista que influenciou a computação. Lista em <https://docs.google.com/spreadsheets/d/1SgsSikg2At-2LpdrQOVBYobfsWvYfNfjcAVtelJ1GvI/pubhtml>
- Utilize diversas fontes independentes. Não confie em uma única fonte. Fontes utilizadas para quaisquer informações devem ser citadas.
- Além de informações biográficas, descreva como o trabalho do cientista se conecta com a computação. Aproveite para entender o trabalho realizado, o que você escreve deve ilustrar a sua compreensão.
- Discuta se as contribuições são relevantes ainda hoje (se aplicável).
- Ao concluir, envie por e-mail um arquivo PDF (5 páginas) nomeado como **EnsaioNomeSobrenome.pdf** até o **dia 2/8**.