Algoritmia e Estruturas de Dados

Métodos de Programação 1

Helena Rodrigues
Departamento de Sistemas de Informação
helena@dsi.uminho.pt

Outubro/2002

Algoritmia e Estruturas de dados

1. **Construção de Algoritmos**
2. **Representação de Algoritmos**
3. **Codificação de Algoritmos e Linguagens de Programação**

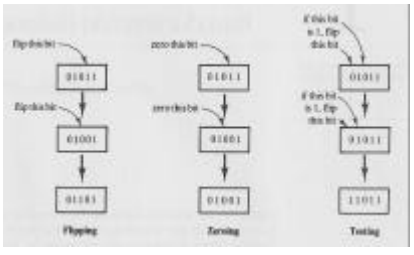
1. Construção de Algoritmos

O computador

[har92]

- pode ser visto com uma enorme colecção de *bits*
- executa um número muito pequeno de operações muito simples
- o programa de computador e o algoritmo por este representado, são responsáveis por transformar este conjunto de operações nas operações mais complexas que o computador pode executar

O computador



Perspectiva histórica

- a palavra "algoritmo" tem origem no nome do matemático árabe/persa (Mohammed *al-Khowârizmi*) do século IX que definiu as regras passo-a-passo para as operações básicas da aritmética
- em Latin, o nome transformou-se em *Algoritmus*, de onde "algoritmo" derivou directamente
- historicamente, o primeiro algoritmo não trivial foi inventado (entre 400 e 300 a.c.) pelo matemático grego Euclides
- esse primeiro algoritmo é o da determinação do máximo divisor comum de dois números inteiros e é designado por "algoritmo de Euclides"

Noção de algoritmo

- **Noção de algoritmo**
  (*Algorítmia Básica 1, Ricardo Machado, 2001/02*)
  - na ciência da computação, um algoritmo pode ser considerado
    uma sequência não ambígua de instruções elementares conducentes à solução de um dado problema
    ou
    um conjunto de instruções que pode ser executado mecanicamente numa quantidade finita de tempo e que resolve algum problema
  - um algoritmo pode também referir um processo resolvente de um problema em qualquer outro domínio distinto da computação

T.P.C.

- Encontrar definições de algoritmos
- Exemplos
  - Clássicos
  - Criativos

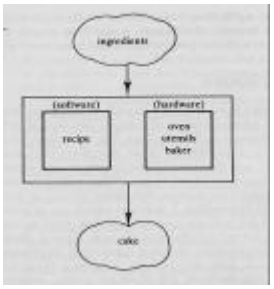
Exemplo da culinária

- imagine-se uma cozinha onde se encontra
  - ingredientes alimentares (farinha, ovos, açucar, chocolate,...)
  - utensílios de culinária (talheres, panelas, formas, ...)
  - aparelhos de cozinha (forno, micro-ondas, batedeira,...)
  - um cozinheiro
- a confecção de um bolo é um processo que
  - produz um bolo
  - a partir de ingredientes
  - pelo cozinheiro
  - com o auxílio dos utensílios e dos aparelhos
  - de acordo com uma receita

Exemplo da culinária – analogia com a informática

- os ingredientes constituem a entrada (*input*) do processo, o bolo consiste na saída (*output*)do processo e a receita corresponde ao algoritmo
- As receitas, ou algoritmos, são vistas normalmente como fazendo parte do *software* de um sistema, e os utensílios, aparelhos e cozinheiro representam o *hardware* do sistema

Exemplo da culinária – analogia com a informática



Exemplo da culinária: a receita e as operações binárias

- Tal como os computadores que só executam operações binárias, também o cozinheiro com o auxílio dos utensílios e aparelhos está limitado a realizar um conjunto finito e bem definido de operações, tais como misturar, bater, aquecer, cozer, abrir o forno, fechar o forno, ...
- É a receita, ou algoritmo (*software*), que transforma as capacidades limitadas do cozinheiro e dos utensílios de cozinha sobre os ingredientes (*hardware*) em bolos

Exemplo da culinária: receita da mousse de chocolate

- **cozinhado**: *6 taças de mousse de chocolate*
- **ingredientes**: *uma tablete de chocolate, 2 colheres de sopa com água, ¼ de uma chávena de pequeno almoço com açucar, 6 ovos, ...*
- **preparado**: *derreter em lume brando o chocolate juntamente com as duas colheres de água. Depois de derretido, misturar 2 colheres de sopa de açucar e adicionar a manteiga aos poucos de cada vez. Deixar tudo ao lume. Bater as seis gemas de ovos durante 5 minutos. Juntá-las lentamente ao chocolate. Mexer devagar se o chocolate não estiver todo derretido. ....*

Instruções básicas ou não?

- uma das instruções básicas presente no texto da receita é
  - *"misturar 2 colheres de sopa de açucar"*
- por que é que a receita não diz
  - *"medir ½ colher de sopa de açucar, deitá-lo no chocolate derretido, misturar bem, medir mais ½ colher de sopa de açucar, deitá-lo no chocolate derretido, misturar bem,..."*?
- ou mais pormenorizadamente ainda
  - *"medir 20,75 g de açucar, deitá-lo no chocolate derretido, usar uma colher para realizar movimentos circulares ao misturar o preparado, ..."*?
- ou ainda
  - *"deslocar o braço em direcção ao preparado, num ângulo de 14 graus, a uma velocidade de, ..."*?

Instruções básicas ou não?

- a resposta é: o *hardware* (cozinheiro, utensílios e aparelhos) sabe como "*misturar 2 colheres de sopa de açúcar*" e não necessita de mais pormenores
- mas será que sabe "*preparar uma mistura de chocolate com manteiga e açúcar*"? Se sim, a primeira parte da receita poderia ser substituída pela instrução "*preparar mistura de chocolate*"
- No extremo, talvez o hardware saiba como preparar uma mousse de chocolate. Neste caso, poderíamos substituir toda a receita pela instrução "preparar mousse de chocolate"

Nível de detalhe das instruções básicas

- decidir o nível de detalhe de uma instrução é uma questão importante quando se pretende decidir que instruções básicas cada passo do algoritmo deve conter
- as acções que cada passo do algoritmo refere que devem ser realizadas têm que estar de acordo com as capacidades do hardware que vai executar as instruções
- adicionalmente, as acções devem ser entendíveis pelos humanos, uma vez que (a) os algoritmos são escritos por humanos, (b) os humanos têm que estar convencidos de que os algoritmos resolvem correctamente os problemas para os quais forma construídos e (c) os humanos podem ter que vir a modificá-los no futuro (*Algoritmia básica 1*, Ricardo machado, 2001/02)

Nível de detalhe das instruções básicas

- na verdade, estes exemplos podem ser confusos, uma vez que dissemos que o computador apenas executa operações binárias
- assim, estas instruções seriam as únicas instruções básicas que poderíamos utilizar na definição de um algoritmo
- vamos ver mais à frente como é que fazemos a ligação entre os algoritmos e os computadores

Construção de Algoritmos

- **Exemplo do algoritmo da multiplicação**
  - o cálculo manual do produto de dois números inteiros é um problema algorítmico
  - se se pretende multiplicar 528 por 46, deve-se
    - multiplicar 6 (unidades de 46) por 8 (unidades de 528), resultando 48
    - escrever as unidades do resultado anterior obtido (8)
    - memorizar as dezenas do resultado anterior obtido (4)
    - multiplicar 6 (unidades de 46) por 2 (dezenas de 528), resultando 12
    - adicionar o resultado anterior (12) às dezenas obtidas na primeira multiplicação (4), resultando 16
    - escrever as unidades do resulatdo anterior obtido (6)
    - memorizar as dezenas do resultado anterior obtido (1)
    - ...

Construção de Algoritmos

- **Exemplo do algoritmo da multiplicação (cont.)**
  - porquê escrever no algoritmo *"multiplicar 6 ...por 8..."*?
  - porque não escrever *"adicionar 8 vezes o número 6"*?
  - porque não resolver o problema de uma só vez escrevendo *"multiplicar 528 por 46"*?
  - porque é permitido conceber uma instrução básica como sendo *"multiplicar 6 ... por 8 ..."* e a instrução *"multiplicar 528 por 46"* já não é considerada básica e como tal não permitida no algoritmo?
  - porque se assume que o hardware, neste caso um humano porque se disse que seria manual, é capaz de executar 6x8 directamente, mas não 528x46
  - ou seja, 528x46 deve ser refinado em instruções mais simples

Exemplo do cálculo do MDC

- o problema numérico do cálculo do máximo divisor comum de dois números inteiros *m* e *n* diferentes de 0 pode ser resoilvido com o seguinte algoritmo
  1. ler *m* e *n*
  2. se *m* < *n*, então tornar *min* = *m*, senão tornar *min* = *n*
  3. tornar *mdc* = *min*
  4. tornar *r1* = *m mod mdc* e *r2* = *n mod mdc*
  5. Se *r1* = *0* e *r2* = *0*, então devolver como resultado o valor de *mdc* e terminar, senão tornar *mdc* = *mdc –1* e ir para o passo 4
- Nesta solução, assumiu-se que para o hardware que vai executar uma implementação do algoritmo a instrução *"n mod mdc"* é básica

Exemplo do cálculo do MDC

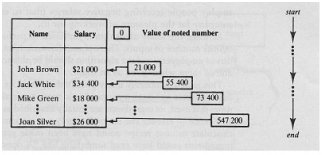
- o problema numérico do cálculo do máximo divisor comum de dois números inteiros *m* e *n* diferentes de 0 pode ser resoilvido com o seguinte algoritmo
  1. ler *m* e *n*
  2. se *m* < *n*, então tornar *min* = *m*, senão tornar *min* = *n*
  3. tornar *mdc* = *min*
  4. tornar *r1* = *m mod mdc* e *r2* = *n mod mdc*
  5. Se *r1* = *0* e *r2* = *0*, então devolver como resultado o valor de *mdc* e terminar, senão tornar *mdc* = *mdc –1* e ir para o passo 4
- Nesta solução, assumiu-se que para o hardware que vai executar uma implementação do algoritmo a instrução *"n mod mdc"* é básica

Tamanho do algoritmo vs tamanho do processo

- o exemplo da lista de funcionários
  - o registo de cada um dos funcionários contém o seu nome e o valor do salário, para além de outras informações
  - admita-se que é fornecida uma lista com registos de funcionários de uma determinada empresa
  - pretende-se calcular a soma dos salários de todos os funcionários da empresa. Um possível algoritmo é o seguinte:
    1. anotar o valor 0
    2. percorrer a lista de registos e adicionar ao valor anotado o valor do salário de cada funcionário
    3. depois de chegar ao fim da lista, produzir o valor anotado como saída

Tamanho do algoritmo vs tamanho do processo

- o exemplo da lista de funcionários (cont.)



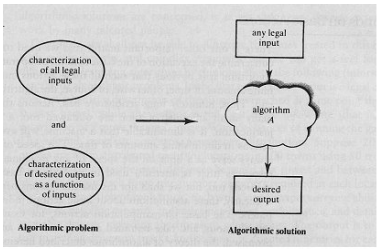
Tamanho do algoritmo vs tamanho do processo

- é importante constatar que o texto do algoritmo do exemplo da lista de funcionários é pequeno e constante no seu tamanho à medida que a lista de funcionários aumenta, se a empresa em causa contratar mais funcionários
- ou seja, apesar do processo que o algoritmo descreve aumentar linearmente com o tamanho da lista de valores de entrada (registos de funcionários) o algoritmo propriamente dito mantém-se imutável
- exemplo: duas empresas, uma com 10 funcionários e outra com 1 milhão, poderão utilizar exactamente o mesmo algoritmo para calcular a soma dos salários dos seus funcionários e exigindo ambas apenas um valor anotado, apesar do processo ser muito mais rápido a executar para o primeiro caso do que para o segundo

O problema algorítmico

- algoritmos (receitas) são soluções para problemas algorítmicos ou computacionais
  - no exemplo da lista de funcionários, o problema pode ser definido na forma do pedido da soma total de salários da lista de funcionários de uma organização
- este problema pode ser visto como a procura do conteúdo de uma "caixa negra", especificado pela definição precisa de todas as entradas válidas, possivelmente infinitas, e a definição precisa de todas as saídas possíveis como uma função dessas entradas

O problema algorítmico



O problema algorítmico

- o problema algoritmico está resolvido quando for encontrado um algoritmo (o conteúdo da caixa negra) apropriado: a caixa negra tem a capacidade de produzir/gerar/calcular a saída (output) apropriada/correcta a partir de qualquer entrada (input) válida, executando o processo descrito e controlado pelo algoritmo
- um algoritmo que só funcione correctamente para alguns conjuntos de entradas válidas, não é um algoritmo suficientemente correcto, porque está incompleto
  - como exemplo extremo, o algoritmo 1. produz 0 como saída funciona correctamente para empresas sem funcionários, empresas com salários a 0, mas ...

O problema algorítmico

- um algoritmo pode no entanto especificar qual o comportamento a adoptar para entradas indesejáveis
  - o algoritmo da soma de salários pode conter o requisito de que caso a informação sobre um determinado funcionário não apresente um valor numérico (ou até um valor que não se encontre dentro de um dado intervalo), a informação sobre esse funcionário deverá ser enviada para a secção de pessoal. Neste caso, a lista de funcionários é válida, mas parte da sua informação necessita de um tratamento especial
- a caracterização de todas as entradas legais é da responsabilidade da formulação do problema algoritmico, enquanto que o tratamento de classes especiais de entrada indesejáveis é da responsabilidade do algoritmo.

Características dos algoritmos

(*Algoritmia básica 1*, Ricardo Machado, 2001/02)

- Entradas
  - um algoritmo tem zero ou mais entradas (habitualmente acessíveis do exterior através de instruções de leitura)
  - a quantidade de entradas tem que ser especificada previamente
    - exemplo: o algoritmo MDC tem duas entradas *m* e *n*, ambas do conjunto dos números inteiros positivos
- Saídas
  - um algoritmo tem uma ou mais saídas (habitualmente acessíveis do exterior através de instruções de escrita)
  - no caso de algoritmos cujo objectivo é efectuar uma dada acção (tocar uma campainha, abrir uma porta, ...) consideram-se como saídas os conjuntos de dados que desencadeiam tais acções
    - exemplo: no algoritmo MDC a saída corresponde ao valor *mdc*

Características dos algoritmos

- Finitude
  - a execução de um algoritmo deve terminar sempre num número finito de passos
    - exemplo: o algoritmo MDC termina obrigatoriamente no passo 5, uma vez que o valor de *mdc* é decrescente
- Definibilidade
  - todos os passos de um algoritmo devem possuir um significado preciso e não ambíguo
  - num algoritmo devem ser rejeitados formulações do tipo
    - *"se n for grande"* (especificação qualitativa)
    - *"Para o maior inteiro n tal que* $x^n+y^n=z^n$, *com x,y,z inteiros ..."* (solução matemática desconhecida)

Características dos algoritmos

- Eficácia
  - os passos de um algoritmo devem conduzir à solução do problema proposto
  - no entanto, por vezes, não basta que o algoritmo seja eficaz, pois levantam-se questões de eficiência
  - a eficiência depende do método de resolução adoptado
  - na ausência de um conhecimento aprofundado sobre métodos resolventes óptimos (algoritmos eficientes) dever-se-á recorrer às definições matemáticas (tal como no exemplo do MDC) se existirem

Características dos algoritmos

Complexidade de algoritmos

- é importante reconhecer a dificuldade inerente à resolução satisfatória de problemas algorítmicos
- o exemplo da "receita de mousse de chocolate" e "lista de funcionários" fazem com que o problema algorítmico pareça um problema simples, mas
- os problemas algoritmos na prática podem ser muito complexos e exigir anos de esforço para serem resolvidos com sucesso, e
- não existem soluções satisfatórias para alguns problemas, e às vezes, não existe mesmo nenhuma solução para outros

Características dos algoritmos

- exemplo: O problema da distribuição de jornais

  20.000 jornais terão de ser distribuídos por 1.000 localidades em 100 cidades, utilizando 50 camiões.

  input: a distância entre as cidades e entre localizações dentro da mesma cidade; o requisito do número de jornais para cada localidade, a posição corrente de cada camião; a capacidade de cada camião; detalhes dos condutores disponíveis

  output: uma lista que associa condutores a camiões e contém o itinerário detalhado para cada camião, de modo que o total número de Km percorridos seja minimizado

Formalização da noção de algoritmo

- um algoritmo é um processo
  - *discreto* (sequência de acções indivisíveis)
  - *determinístico* (para cada passo da sequência e para cada conjunto válido de entradas corresponde uma e só uma acção)
  - *fiinito* (ou seja, o algoritmo deve terminar para quaisquer que sejam as entradas pertencentes aos domínios pré-definidos)
- um algoritmo é estruturalmente constituído por passos, ordenados segundo regras básicas do fluxo de execução dos algoritmos, sendo cada passo descrito à custa de uma ou mais instruções básicas para o nível de abstração adoptado

Estruturas e instruções algorítmicas

- ou como é que as acções básicas de um algoritmo são estruturadas de modo que uma pessoa ou o computador apreendam a ordem precisa pela qual as instruções devem ser executadas
- o controlo da execução de um algoritmo é normalmente feito através da combinação de instruções chamadas de "controlo de fluxo" (*control-flow structures*).

Estruturas e instruções algorítmicas

- Sequenciação
  - corresponde a instruções do tipo "fazer *A* seguido *de B*" (normalmente uma sequência de instruções é apresentada com uma instrução em cada linha, ou separada por um "." ou ";")
- Teste de condições
  - corresponde a instruções do tipo "se *Q* então *A* senão *B*" ou apenas "se *Q* então *A*"
  - a condição *Q* (proposição booleana simples ou composta) é avaliada com base nos valores correntes envolvidos na proposição
  - tendo em conta a resposta obtida (a condição ou é avaliada como *verdadeira* ou como *falsa*), uma determinada acção deve ser executada (*A* ou *B*)

Estruturas e instruções algorítmicas

- Ciclos
  - o tipo de instruções analisados até agora não explica como é que um algoritmo de tamanho fixo pode descrever processos longos, dependendo da entrada do algoritmo
  - as intruções de ciclos são do tipo "fazer *A* exactamente *N* vezes", ou "repetir *A* até *Q*", ou "enquanto *Q* fazer *A*", ou "fazer *A* enquanto *Q*", onde *Q* é uma condição
  - a condição *Q* é avaliada no início de cada ciclo, com base nos valores correntes envolvidos na proposição
  - tendo em conta a resposta obtida, a condição ou é avaliada como *verdadeira* ou como *falsa*, o ciclo continua ou não respectivamente

Estruturas e instruções algorítmicas

- Ciclos (cont.)
  - exemplo: lista de funcionários

(1) anotar o valor 0; apontar para o primeiro salário na lista
(2) fazer
  (2.1) somar o salário apontado ao valor anotado;
  (2.2) apontar para o próximo salário
  N-1 vezes
(3) somar o salário apontado ao valor anotado
(4) produzir o valor anotado como saída

Nota: E se o número de empregados não fosse disponibilizado?

Estruturas e instruções algorítmicas

- Agrupar instruções
  - instruções de sequenciação, de teste e de ciclos podem ser agrupadas e aninhadas umas nas outras
  - exemplo da "lista de funcionários" modificado: calcular a soma total dos salários dos funcionários cujo salário é superior ao salário do seu gestor (assumir que cada elemento da lista contém o nome do gestor do funcionário). Quando o gestor não pertencer à lista, considerar o salário do funcionário menor que este.

Estruturas e instruções algorítmicas

- Agrupar instruções (cont.) - exemplo (cont.)

(1) anotar o valor 0; apontar para o primeiro funcionário na lista
(2) fazer
  (2.1) anotar o salário apontado; anotar o nome do gestor apontado;
  (2.2) apontar para o primeiro funcionário na lista;
  (2.3) enquanto (nome apontado ≠ nome do gestor anotado fazer e ¬ (fim da lista))
  (2.3.1) apontar para o próximo funcionário
  (2.4) se (nome apontado = nome gestor anotado e salário anotado > salário apontado) então
    somar o salário anotado ao valor anotado;
  (2.5) apontar para o próximo funcionário
  N-1 vezes
(3) apontar para o primeiro funcionário na lista
(4) enquanto (nome apontado ≠ nome do gestor anotado fazer e ¬ (fim da lista))
  (4.1) apontar para o próximo funcionário
(5) se (nome apontado = nome gestor anotado e salário anotado > salário apontado) então
  somar o salário anotado ao valor anotado;
(6) produzir o valor anotado como saída

Estruturas e instruções algorítmicas

- Saltos
  - correspondem a instruções do tipo "ir para o passo ..."
  - redireccionam a sequência de execução dos passos do algoritmo,
  - tendem a tornar um algoritmo difícil de ler
  - tendem a fazer uma associação entre o texto do algoritmo e o processo que é descrito por este, no entanto, uma vez que algoritmos fixos descrevem processos de tamanho variável, uma única etapa no texto do algoritmo corresponde a muitas etapas na execução do processo correspondente
- Paragem
  - correspondem a instruções do tipo "terminar"
  - indicam que o fluxo de execução do algoritmo chegou ao fim

Tipos e estruturas de dados

- Dados (definição)
  - correspondem aos objectos manipulados pelo algoritmo. Estes objectos representam uma abstracção da realidade, na medida em que, certas características ou propriedades dos objectos reais são ignorados uma vez que são irrelevantes para o problema em particular
    - exemplo: no exemplo da lista dos funcionários, cada funcionário é representado pelo conjunto de dados relevante para o processo de cálculo de salários. Este conjunto inclui normalmente a identificação do funcionário e o salário. No entanto, não irá incluir em princípio a cor dos olhos, altura e peso.

Tipos e estruturas de dados

- Dados (definição)
  - na resolução de um problema algorítmico, o primeiro passo é escolher uma abstracção da realidade, i.e, definir o conjunto de dados que representa a situação real. Esta escolha tem de ser guiada pelo problema a resolver. O segundo passo é escolher uma representação da informação anterior. Na maior parte dos casos, estes dois passos não são totalmente independentes .

Tipos e estruturas de dados

- Tipos de dados
  - determinam a representação dos dados e o conjunto de operações que podem ser efectuadas nesses dados
    exemplo: podemos somar números, mas não podemos procurar vogais num número
  - tipos básicos:
    exemplo: números e as operações aritméticas e de comparação, palavras e as operações de comparação, valores booleanos, caracteres
  - tipos compostos ou estruturas de dados definem a forma como um algoritmo organisa, mantém, modifica e acede coleçções de dados
    exemplo: a lista de funcionários e a estrutura de cada membro

Tipos e estruturas de dados

- Estrutura de dados **lista** (ou vector)
  - corresponde a toda e qualquer informação representável por uma colecção de dados que podem ser referenciados de uma forma uniforme e que podem ser ordenados segundo um determinado critério
  - a estrutura de dados **lista** pode ser modelada pela estrutura matemática *sequência*

Nota: Vamos utilizar como formas de agregar dados as estruturas matemáticas e as operações correspondentes apresentadas em *Fundamentos da Computação*, Ricardo Machado, 2001/02

Tipos e estruturas de dados

- Estrutura de dados **registo**
  - corresponde à agregação de um conjunto de dados correspondentes a uma entidade do mundo real
  - a estrutura de dados **registo** pode ser modelada pela estrutura matemática *tuplo ordenado*

Nota: Vamos utilizar como formas de agregar dados as estruturas matemáticas e as operações correspondentes apresentadas em *Fundamentos da Computação*, Ricardo Machado, 2001/02

Tipos e estruturas de dados

- Variáveis
  - correspondem a células que contém dados. São normalmente identificadas por um nome.
    exemplo: o "valor anotado" no algoritmo da lista de funcionários, inicialmente 0 e utilizado posteriormente para acumular a soma dos salários é uma variável.
  - atribuição de valores
    - corresponde a instruções conducentres à atribuição de um valor (de dados) a uma variável
    - normalmente, em linguagens algoritmicas baseadas em liguangem natural, utiliza-se a seta (←) para representar atribuições

Tipos e estruturas de dados

- Variáveis - Exemplo da "lista de funcionários" modificado, utilizando variáveis

Dados de entrada: $l = <(n_1,s_1,g_1),(n_2,s_2,g_2), ..., (n_n,s_n,g_n)>$ com $n_n \in$ NOMES, $s_n \in$ N, $g_n \in$ NOMES; TAM $\in$ N (tamanho da lista)
Dados de saída: soma $\in$ N;
(1) contador ← 0; i ←1;
(2) fazer
  (2.1) salario ← Π_2l[i]; gestor ← Π_3l[i]
  (2.2) j ←1
  (2.3) enquanto (Π_2l[j] ≠ gestor e j < TAM) fazer
    (2.3.1) j ← j + 1
  (2.4) se (Π_1l[j] = gestor e salário > Π_2l[j]) então contador = contador + salario
  (2.5) i = i + 1
TAM-1 vezes
(3) j ←1
(4) enquanto (Π_1l[j] ≠ gestor e j < TAM) fazer
  (4.1) j = j +1
(5) se (Π_1l[j] = gestor e salário > Π_2l[j]) então contador = contador + salario
(6) soma ← contador

O processo de descoberta de um algoritmo (Meta-algoritmo)

— Meta –algoritmo (algoritmo para construir algoritmos) *(adaptado de Algoritmia básica 1,* Ricardo Machado, 2001/02*)*

1. ler, atentamente, o texto do problema algorítmico
   1.1 procurar o significado de todas as palavras e expressões
   1.2 compreender clara e completamente o problema descrito
   1.3 eliminar detalhes supérfluos

O processo de descoberta de um algoritmo (Meta-algoritmo)

2. especificar simbolicamente os dados do problema algorítmico
   2.1 retirar do texto do enunciado, e representar simbolicamente por compreensão, os conjuntos de dados de entrada (*input*)
   2.2 retirar do texto do enunciado, e representar simbolicamente por compreensão, os conjuntos de dados de saída (*output*)
   2.3 retirar do texto do enunciado, e representar simbolicamente, as relações entre os conjuntos de dados de saída e os de entrada

O processo de descoberta de um algoritmo (Meta-algoritmo)

3. determinar que acções devem ser descritas pelo algoritmo
   3.1 decidir que problema algorítmico vai ser tratado
   3.2 identificar as acções a descrever pelo algoritmo
   3.3 especificar o algoritmo, recorrendo ao princípio do rigor e formalismo
   3.4 verificar se as acções identificadas no passo 3.2 deram origem a instruções elementares; se sim, então ir para o passo 4, senão decidir o nível de detalhe a adoptar e ir para o passo 3.2

O processo de descoberta de um algoritmo (Meta-algoritmo)

— 3.3 especificar o algoritmo, recorrendo ao princípio do rigor e formalismo
   3.3.1 decidir que linguagem (natural, simbólica, de modelação) vai ser utilizada para descrever as acções do algoritmo
   3.3.2 agrupar as acções em passos discretos de execução
   3.3.3 ordenar os passos segundo um fluxo essencialmente sequencial de execução das acções
   3.3.4 identificar quais as violações (saltos) ao fluxo sequencial de execução das acções
   3.3.5 descrever, ordenadamente, as acções recorrendo à linguagem adoptada

O processo de descoberta de um algoritmo (Meta-algoritmo)

4. executar, manualmente, o algoritmo
   4.1 seleccionar um conjunto de entradas válidas
   4.2 aplicar o conjunto de entradas ao algoritmo resultante da última execução do passo 3.3
   4.3 verificar a validade dos dados de saída; se dados de saída válidos então terminar, senão identificar incorrecção e ir para o passo 1