

Análise de Algoritmos

Modelos de Computação

Introdução

- Dado um problema, como encontramos um algoritmo eficiente para solucioná-lo ?
- Encontrado um algoritmo, como o comparamos a outros algoritmos que solucionam o mesmo problema ?
- Como podemos julgar os benefícios de um algoritmo ?

Questões como essas são de interesse tanto de programadores como de cientistas da computação teoricamente orientados.

Introdução

- Nessa disciplina vamos
 - estudar várias técnicas que tentam responder perguntas como as anteriores.
 - considerar vários modelos de um computador
 - A máquina de acesso aleatório
 - Máquina de acesso aleatório com programa armazenado
 - Máquina de Turing
 - comparar esses modelos na base de suas habilidades para refletir a complexidade de um algoritmo, e assim derivar desses vários modelos de computação mais especializados.

Algoritmos e suas complexidades.

- Algoritmos podem ser avaliados por diversos critérios.
- Estamos interessados na **taxa de crescimento de tempo ou espaço (quantidade de memória)** para solucionar grandes instâncias do problema. De preferência, que esta taxa de crescimento esteja relacionada a um número inteiro chamado de **tamanho da entrada**, representado por **n**.

Algoritmos e suas complexidades

- O tempo necessário por um algoritmo expresso como uma função do tamanho do problema é chamado de **complexidade de tempo** ,
- O comportamento limítrofe da complexidade conforme o tamanho aumenta é chamado de **complexidade assintótica de tempo**.
- De forma análoga temos a **complexidade de espaço** e **complexidade assintótica de espaço**.

Algoritmos e suas complexidades

- O crescimento no número de cálculos que os computadores apresentam, de uma geração para outra, poderia diminuir a importância de algoritmos eficientes.
- A complexidade de um algoritmo determina o tamanho do problema que pode ser alcançado com o crescimento da velocidade dos computadores.
- Para definirmos complexidades de algoritmos é preciso antes definir para qual modelo de computação esses algoritmos serão construídos.

Modelo de Computação

- um modelo de computação é a definição de um conjunto de operações que podem ser usadas numa computação e seus respectivos custos.
- Somente assumindo certo modelo de computação é possível analisar os recursos computacionais requeridos, como tempo de execução e espaço de armazenamento, ou discutir as limitações de algoritmos ou computadores.

Modelo de Computação

- Os principais modelos de computação são :
 - Máquina de Acesso Aleatório (RAM)
 - Máquina de Acesso Aleatório com Programa Armazenado
 - Máquina de Turing.

Modelo de Acesso Aleatório (RAM)

- Este modelo consiste de :
 - Uma fita somente de entrada;
 - Uma fita somente de saída;
 - Um programa; e
 - Uma memória (acumulador).
- As fitas são divididas em quadros onde cabe um inteiro
- A memória consiste em uma sequencia de registradores r_0, r_1, \dots, r_n , onde cada uma é capaz de armazenar um inteiro de tamanho arbitrário. Não há um limite superior no número de registradores a serem usados.
 - Com esta abstração temos que : o tamanho do problema cabe na memória e os inteiros são pequenos o suficiente para caber em uma word do computador.

Modelo de Acesso Aleatório (RAM)

- Instruções
 - A natureza exata das instruções não é importante,
 - Podemos ter o mesmo número de instruções que um computador tenha,
 - Um conjunto mínimo de instruções está no quadro a seguir, mostrando o código da instrução e um endereço.

Modelo de Acesso Aleatório

Código da Instrução

Endereço

1 LOAD

operando

2 STORE

operando

3 ADD

operando

4 SUB

operando

5 MULT

operando

6 DIV

operando

7 READ

operando

8 WRITE

operando

9 JUMP

operando

10 JGTZ

operando

11 JZERO

operando

12 HALT

operando

Modelo de Acesso Aleatório

- Um operando pode ser:
 - $= i$, indicando o próprio operando i ;
 - um inteiro não-negativo i , indicando o conteúdo do registro r_i ;
 - $* i$, indicando um endereçamento indireto. Isto é, o operando é o conteúdo de um registrador j (r_j), onde o inteiro j é encontrado no registrador r_i . Se $j < 0$ então a máquina para.

Modelo de Acesso Aleatório

- A maioria das instruções utilizam o registro r0, chamado de acumulador.
- Para explicar o significado das instruções utilizamos duas funções:
 - Um mapeamento da memória $c(i)$ que é o inteiro armazenado (o conteúdo) em r_i ; e
 - $v(a)$, o valor do operando a , assim
 - $v(=i) = i$,
 - $v(i) = c(i)$,
 - $v(*i) = c(c(i))$.

Modelo de Acesso Aleatório

1	LOAD a	$c(0) \leftarrow v(a)$
2	STORE i	$c(i) \leftarrow c(0)$
	STORE *i	$c(c(i)) \leftarrow c(0)$
3	ADD a	$c(0) \leftarrow c(0) + v(a)$
4	SUB a	$c(0) \leftarrow c(0) - v(a)$
5	MULT a	$c(0) \leftarrow c(0) \times v(a)$
6	DIV a	$c(0) \leftarrow c(0) / v(a)$ (piso)
7	READ i	$c(i) \leftarrow$ símbolo corrente da entrada
	READ *i	$c(c(i)) \leftarrow$ símbolo corrente da entrada, e avança a fita
8	WRITE a	$v(a)$ é escrito na saída, e a fita avança
9	JUMP b	o counter location aponta para a instrução rotulada b
10	JGTZ b	se $c(0) > 0$, o counter location aponta para a instrução rotulada b, senão aponta para a próxima instrução
11	JZERO b	se $c(0) = 0$, o counter location aponta para a instrução rotulada b senão p/ próxima

Modelo de Acesso Aleatório

- O programa:
 - Em geral, o programa RAM define um mapeamento da fita de entrada para a fita de saída;
 - um mapeamento parcial, uma vez que para alguma entrada não haja uma saída.
 - O mapeamento pode ser interpretado de várias maneiras, sendo duas muito importantes:
 - Como função. $f(x_1, x_2, \dots, x_n) \rightarrow y$
 - Como linguagem. $s_1, s_2, \dots, s_k \rightarrow 0/1$, onde $s_i \in \Sigma$

Modelo de Acesso Aleatório

- Exemplo : o algoritmo abaixo determina uma função $f(n)$ que retorna n^n para todo inteiro $n \geq 1$ e no caso contrário retorna 0.

inicio

leia r1;

se $r1 \leq 0$ então escreva 0

senão

r2 \leftarrow r1;

r3 \leftarrow r1 -1;

enquanto r3 > 0 faça

r2 \leftarrow r2 * r1;

r3 \leftarrow r3 - 1;

fim_enquanto

escreva r2

fim_se

fim

Modelo de Acesso Aleatório

- O algoritmo abaixo reconhece a linguagem sobre o alfabeto {1,2} consistindo de todas as palavras com o mesmo número de 1's e 2's. O indicador de fim da entrada é o 0.

inicio

$d \leftarrow 0;$

leia x;

enquanto $x \neq 0$ **faça**

se $x \neq 1$ **então** $d \leftarrow d - 1$

senão $d \leftarrow d + 1;$

leia x;

fim_enquanto

se $d = 0$ **então escreva** 1

senão escreva 0;

fim

Modelo de Acesso Aleatório

- Exercícios:

1 – Escreva o programa (utilizando as instruções) RAM para o algoritmo de função acima.

2 – Escreva o programa RAM para o algoritmo reconhecedor de linguagem acima.

Modelo de Acesso Aleatório

- Complexidade Computacional de Programas RAM
 - Duas importantes medidas calculadas em função do tamanho da entrada.:
 - Complexidade de tempo; e
 - Complexidade de espaço.
 - Complexidade do pior caso :
 - para um dado tamanho, a complexidade é definida como a **máxima complexidade** sobre **todas as entradas daquele tamanho**.
 - Complexidade esperada :
 - para um dado tamanho, a complexidade é definida como a **complexidade média de todas as entradas daquele tamanho**.

Modelo de Acesso Aleatório

- Para especificar exatamente a complexidade de tempo e de espaço no Modelo RAM é preciso definir:
 - O tempo que cada instrução consome; e
 - O espaço que cada registrador ocupa.
- Serão utilizados dois critérios de custos:
 - Critério de Custo Uniforme; e
 - Critério de Custo Logarítmico.
- No critério de custo uniforme cada instrução RAM requer uma unidade de tempo e cada registro requer uma unidade de espaço.

Modelo de Acesso Aleatório

- No critério de custo logarítmico, definições mais realistas são consideradas.
- Seja $l(i)$ a função logarítmica de inteiros definida:

$$l(i) = \begin{cases} \log |i| + 1, & i \neq 0 \\ 1, & i = 0 \end{cases}$$

- Assim o custo $t(a)$ para um operando a fica:

operando a custo $t(a)$

$= i$ $l(i)$

i $l(i) + l(c(i))$

$*j$ $l(i) + l(c(i)) + l(c(c(i)))$

Modelo de Acesso Aleatório

- Exercício:
3 – Construa a tabela de instruções RAM e seus respectivos custos $t(a)$.

Modelo de Acesso Aleatório

1- LOAD a	$t(a)$
2- STORE i	$l(c(0)) + l(i)$
STORE *i	$l(c(0)) + l(i) + l(c(i))$
3- ADD a	$l(c(0)) + t(a)$
4- SUB a	$l(c(0)) + t(a)$
5- MULT a	$l(c(0)) + t(a)$
6- DIV a	$l(c(0)) + t(a)$
7- READ i	$l(\text{input}) + l(i)$
READ *i	$l(\text{input}) + l(i) + l(c(i))$
8- WRITE a	$t(a)$
9- JUMP b	1
10- JGTZ b	$l(c(0))$
11- JZERO b	$l(c(0))$
12- HALT	1

Modelo de Acesso Aleatório

- A complexidade logarítmica de espaço é definida sobre a soma de todos os registradores, incluindo o acumulador, de

$$l(x_i),$$

onde x_i é o inteiro da mais alta magnitude armazenado no registrador i em qualquer tempo durante a execução.

Modelo de Acesso Aleatório

- Exercício
- 4 - Calcule a complexidade esperada para os programas escritos nos exercícios 1 e 2.

Modelo de Programa Armazenado com Acesso Aleatório - RASP

- Uma máquina RASP é uma variação da máquina RAM onde **o programa** ocupa espaço na memória, **fica armazenado em registradores**.
 - A estrutura geral desta máquina é semelhante à da RAM, porém é permitido, e até necessário, **modificar o programa em tempo de execução**.
 - O conjunto de instruções é o mesmo, exceto que as **instruções de endereçamento indireto não são permitidas**.

Modelo RASP

- O Programa Armazenado
 - Ocupa espaço na memória.
 - Cada instrução RASP ocupa dois registradores consecutivos;
 - o primeiro armazena um código que define a instrução; e
 - o segundo armazena o endereço (i).

Modelo RASP

Instrução	Código	Instrução	Código
LOAD i	1	DIV i	10
LOAD =i	2	DIV =i	11
STORE i	3	READ i	12
ADD i	4	WRITE i	13
ADD =i	5	WRITE =i	14
SUB i	6	JUMP i	15
SUB =i	7	JGTZ i	16
MULT i	8	JZERO i	17
MULT =i	9	HALT	18

Modelo RASP

- A complexidade no Modelo RASP segue o mesmo cálculo aplicado nos programas do Modelo RAM.

Um modelo primitivo de computação: A MÁQUINA DE TURING

- Definição: Uma Máquina de Turing Multifitas(TM) deve conter um número k de fitas, as quais são finitas para a direita. Cada fita é dividida em células, onde cada uma armazena um de um número finito de símbolos de fita. Uma célula em cada fita é lida por uma cabeça de fita que pode ser e escrever. A operação da máquina de Turing é determinado por um programa primitivo chama de controle finito. O controle finito é sempre um número finito de estados, os quais podem ser considerados posições em um programa.

Máquina de Turing

- Um passo computacional de uma TM consiste do seguinte. De acordo com o estado corrente do controle finito e os símbolos de fita que estão sob cada cabeça de fita, a TM pode executar qualquer ou todas essas operações:
 - 1 – Mudar o estado do controle finito.
 - 2 – Escrever novos símbolos de fita sobre o símbolo corrente em qualquer ou todas as células sob as cabeças de fita.
 - 3 – Mover qualquer ou todas as cabeças de fita, independentemente, uma célula para a direita(r) ou esquerda(l) ou manter-se parada(s).

Máquina de Turing

- Formalmente, denota-se uma Máquina de Turing k-fitas pela séptupla

$$(Q, T, I, \delta, b, q_0, q_f),$$

onde:

Q é o conjunto de estados.

T é o conjunto de símbolos de fita.

I é o conjunto de símbolos de entrada; I está contido em T

b , em $T-I$, é o espaço em branco.

q_0 é o estado inicial.

q_f é o estado final

δ é a função de próximo-movimento.

Máquina de Turing

- A função de próximo-movimento mapeia um subconjunto de $Q \times T^k$ para $Q \times (T \times \{l, r, s\})^k$. Isto é, para alguma $(k+1)$ -tupla consistindo de um estado e k símbolos de fita, ela determina um novo estado e k pares, cada par consistindo de um novo símbolo de fita e uma direção para a cabeça da fita. Suponha que

$$\delta(q, a_1, a_2, \dots, a_k) = (q', (a_1', d_1), (a_2', d_2), \dots, (a_k', d_k))$$

A máquina está no estado q com as i -ésima cabeça de fita lendo o símbolo de fita a_i para $1 \leq i \leq k$. Então em um movimento a máquina entra no estado q' muda os símbolos a_i para a_i' e move a i -ésima cabeça de fita nas direções d_i para $1 \leq i \leq k$.