Fundamentos de Análise de Algorítmos

Iyan Lucas Duarte Marques[1]

[1]Instituto de Ciências Exatas e Informática - Pontifícea Universid ade Católica Minas Gerais (PUC-MG)

1. Handbook

1.1. Notas da aula

- Em um comando de repetição, com i começando em 0, sendo repetido enquanto i $<$ n, é incrementado de 1 em 1 (i++), faremos n operações.
- Na computação, quando reduzimos o nosso escopo de busca, sistematicamente pela metade (i /= 2) teremos um custo logarítmico.
- Em AED2, a menos que dito o contrário, a operação mais relevante será a comparação entre elementos do array. if(array[i] $<$ array[i+1]).
- O número mínimo de comparações para ordenar um vetor é $\Theta n(\log_2 n)$ isso, para situações genéricas e sem "roubar".
- Um algoritmo é estável se os elementos com a mesma chave mantiverem a ordem original.
- Quando tivermos uma estrutura de repetição começado com a e repete enquanto menor do que n, e sendo incrementado de 1 em 1, teremos n - a repetições. Por exemplo, temos o caso normal que é $i = 0; (i < n)$, teremos n operações, mas se $i = 1; (i < n)$, teremos $n - 1$ operações.

1.2. Cenários Possíveis

1.2.1. Melhor Caso

Menor "tempo de execução"para todas as entradas possíveis de tamanho n.

1.2.2. Pior Caso

Maior "tempo de execução"para todas as entradas possíveis.

1.3. Contagem de Operações com Condicional

1.3.1. Melhor Caso

É a condição (o parêntese do if, por exemplo) mais o mínimo

1.3.2. Pior Caso

1.3.3. Melhor Caso

É a condição (o parêntese do if, por exemplo) mais o máximo

1.4. Contagem de operações com Repetição

Será o custo da condição mais o número de interações multiplicado pela soma dos custos da condição e da lista a ser repetida. Por exemplo, condição() $+ n *$ (lista()+condição())

1.5. Restrição de Algorítmos

- Restrições do computador: Capacidade computacional e armazenamento.
- Um algorítmo que leva séculos para terminar é uma opção inadequada.

1.6. Métricas (o que devemos analisar)

- Tempo de execução
- Espaço de memória ocupado
- Outros

2. Como Medir o Custo de um Algorítmo

2.1. Modelo Matemático

Determinamos e contamos as operações relevantes porque o custo total de um algorítmo é igual a soma das suas operações. Depois disso, desconsideramos sobrecargas de gerenciamento de memória ou E/S, após isso, é definir a função de complexidade. A menos que dito o contrário, consideramos o pior caso.

2.2. Funções de Complexidade

- Função de Complexidade de Tempo: Mede o tempo (número de execuções da operação relevante) de execução do algorítmo para problema de tamanho n.
- Função de Complexidade de espaço: Mede a quantidade de memória necessária para executar um algorítmo de tamanho n.

3. Como Calcular a Complexidade de um Algorítmo

3.1. Calculo de Complexidade para Condicional

Será o custo da condição mais ou o da lista de verdadeira ou da falsa.

3.2. Calculo de Complexidade para Repetição

Será o custo da condição mais o número de iterações multiplicadas pela soma dos custos a ser repetida.

4. Notações O, Ω e Θ

Nesta notação, consideramos apenas a maior potência e ignoramos os coeficientes. Sendo O o limite superior, Ω o limite inferior e Θ o limite justo.

4.1. Diferença entre as notações

4.1.1. O

É o limite superior, logo, se um algorítmo é $O(f(n))$, ele também será $O(g(n))$ para toda função $g(n)$ tal que seja maior que $f(n)$

4.1.2. Ω

É o limite inferior, logo, se um algorítmo é $\Omega(f(n))$, ele também será $\Omega(g(n))$ para toda função $g(n)$ tal que seja menor que $f(n)$

4.1.3. Θ

É o limite justo, logo, se $g(n)$ é $O(f(n))$ e $\Omega(f(n))$ se e somente se $g(n)$ é $\Theta(f(n))$