Análise de Algoritmos

Analisar um algoritmo significa prever os recursos de que o algoritmo necessitará. Ocasionalmente, recursos como memória, largura de banda de comunicação ou hardware de computador são a principal preocupação, mas com frequência é o tempo de computação que desejamos medir. Em geral, pela análise de vários algoritmos candidatos para um problema, pode-se identificar facilmente um algoritmo mais eficiente. Essa análise pode indicar mais de um candidato viável, mas vários algoritmos de qualidade inferior em geral são descartados no processo.

Análise da ordenação por inserção

O tempo despendido pelo procedimento INSERTION-SORT depende da entrada: a ordenação de mil números demora mais que a ordenação de três números. Além disso, INSERTION-SORT pode demorar períodos diferentes para ordenar duas sequências de entrada do mesmo tamanho, dependendo do quanto elas já estejam ordenadas. Em geral, o tempo de duração de um algoritmo cresce com o tamanho da entrada; assim, é tradicional descrever o tempo de execução de um programa como uma função do tamanho de sua entrada. Para isso, precisamos definir os termos "tempo de execução" e "tamanho da entrada" com mais cuidado.

A melhor noção de tamanho da entrada depende do problema que está sendo estudado. No caso de muitos problemas, como a ordenação ou o cálculo de transformações discretas de Fourier, a medida mais natural é o número de itens na entrada - por exemplo, o tamanho do arranjo n para ordenação.

O tempo de execução de um algoritmo em uma determinada entrada é o número de operações primitivas ou "etapas" executadas. É conveniente definir a noção de etapa (ou passo) de forma que ela seja tão independente da máquina quanto possível.

O tempo de execução do algoritmo é a soma dos tempos de execução para cada instrução executada; uma instrução que demanda c passos par ser executada e é executada n vezes, contribuirá com c*n para o tempo de execução total.

No pior caso, o tempo de execução de INSERTION-SORT é $an^2 + bn + c$ para constantes a, b e c que, mais uma vez, dependem dos custos de instrução c; portanto, ele é uma função quadrática de n.

Análise do pior caso e do caso médio

Tempo de execução do pior caso é o tempo de execução mais longo para

qualquer entrada de tamanho n.

- O tempo de execução do pior caso de um algoritmo é um limite superior sobre o tempo de execução para qualquer entrada. Conhecê-lo nos dá uma garantia de que o algoritmo nunca irá demorar mais tempo. Não precisamos fazer nenhuma suposição baseada em fatos sobre o tempo de execução, e temos a esperança

de que ele nunca seja muito pior.

- Para alguns algoritmos,o pior caso ocorre com bastante frequência. Por exemplo, na pesquisa de um banco de dados em busca de um determinado fragmento de informação, o pior caso do algoritmo de pesquisa ocorrerá frequentemente quando a informação não estiver presente no banco de dados. Em algumas aplicações de pesquisa, a busca de informações ausentes pode ser frequente.

- Muitas vezes, o "caso médio" é quase tão ruim quanto o pior caso. Suponha que sejam escolhidos aleatoriamente n números e que se aplique a eles a ordenação por inserção. Quanto tempo irá demorar para se descobrir o lugar no subarranjo A [1..j- 1] em que se deve inserir o elemento A[j]? Em média, metade dos elementos em A[l ..j- 1] são menores que A[j], e metade dos elementos são maiores. Assim, em média, verificamos que metade do subarranjo A[1..j- l], e então 9 =j/2. Se desenvolvermos o tempo de execução do caso médio resultante, ele será uma função quadrática do tamanho da entrada, exatamente como o tempo de execução do pior caso.

Ordem de Crescimento

Consideramos apenas o termo inicial de uma fórmula (por exemplo, an^2), pois os termos de mais baixa ordem são relativamente insignificantes para grandes valores de n. Também ignoramos o coeficiente constante do termo inicial, tendo em vista que fatores constantes são menos significativos que a taxa de crescimento na determinação da eficiência computacional para grandes entradas. Portanto, escrevemos que a ordenação por inserção, por exemplo, tem um tempo de execução do pior caso igual a $\theta(n^2)$ (lido como "theta de n ao quadrado").

Em geral, consideramos um algoritmo mais eficiente que outro se o tempo de execução do seu pior caso apresentauma ordem de crescimento mais baixa. Essa avaliação pode ser incorreta para entradas pequenas; porém, para entradas suficientemente grandes, um algoritmo $\theta(n^2)$, por exemplo, será executado mais rapidamente no pior caso que um algoritmo $\theta(n^3)$.

Exercícios

Exercício 1.

Expresse a função $n^3/1000 - 100n^2 - 100n + 3$ em termos da notação Θ:

R: Usando a simplificação do capitulo 2.2 nos ensina, a função será $\Theta(n^3)$

Exercício 2.

Como podemos modificar praticamente qualquer algoritmo para ter um bom tempo de execução no melhor caso?

R: Sendo o tempo de execução de um algoritmo a soma de todos os custos de instruções, para melhorar um algoritmo devemos usar instruções que tenham o menor tempo de execução possível, e a menor quantidade delas. Outra forma de melhorar qualquer algoritmo nesse sentido é utilizar a melhor opção de entrada, no caso de um algoritmo relacionado a busca a melhor entrada seria um arranjo de dados já ordenado.