Aula 1

§ Complexidade de Algoritmos

Mede-se a complexidade de um algoritmo usando-se referências como **tempo** ou **memória**. Ou seja, o algoritmo é tão complexo quanto a quantidade de tempo ou memória que ele consome para obter resultados a partir dos dados de entrada.

Usar a noção de tempo de execução para comparação do grau de complexidade de diferentes programas, entre si, exige uma série de cuidados para obtenção de valores comparáveis. Por exemplo, medidas obtidas a partir da execução de programas em máquinas diferentes não proporcionam resultados comparáveis.

Para evitar esse problema, adota-se como unidade de medida a quantidade de instruções básicas executadas pelo algoritmo. Derivar o tempo a partir dessa informação, se necessário, é imediato.

Entretanto, um algoritmo pode apresentar diferentes medidas de complexidade de acordo com os dados de entrada. Por exemplo, um algoritmo para busca de uma chave em um vetor. Se esse algoritmo não abortar a busca ao encontrar a chave, a complexidade dependerá do tamanho do vetor.

♠ Algoritmo – Busca de chave

```
inicio
     i=0; achou<-F; pos <-0;
     repita
          i <- i + 1;
          se (tab[i] = ch) entao
               achou <- V;
               pos <- i;
          fim-se;
     até ( i == n ) { ou achou }
fim
```

A complexidade de um programa deve, portanto, ser uma medida relativa. No exemplo acima, o algoritmo possuiria um grau de complexidade proporcional ao tamanho do vetor.

No algoritmo exemplo, sendo *n* o número de células do vetor, e usando como referência o número de execuções da instrução de comparação entre a chave e o conteúdo de uma célula do vetor, obtém-se a seguinte medida de complexidade relativa:

Complexidade da busca = n (tempo linear)

Essa métrica possibilita, por exemplo, as seguintes análises:

♠ Suponha que, em uma máquina, num certo tempo máximo tolerável, um algoritmo de **complexidade linear** resolve problemas com entrada de tamanho x_1. Em um computador dez vezes mais rápido o mesmo algoritmo resolveria um problema com entrada de tamanho $10x_1$.

♠ Suponha um algoritmo de **complexidade quadrática**, levando tempo proporcional a n^2 para uma entrada de tamanho *n*. Num tempo máximo tolerável *t* na máquina mais lenta são resolvidos problemas com entrada de tamanho x_2, isto é, $K(x_2)^2 = t$, onde K é o tempo que a máquina gasta para executar uma instrução. Na máquina dez vezes mais rápida, esse algoritmo resolveria, no tempo *t* , um problema com entrada de tamanho $3{,}16\ x_2$.´

Desenvolvimento:

Seja y o tamanho da entrada do problema resolvido na máquina mais rápida,

$(K/10).y^2 = t \quad => \quad k\ y^2 = 10\ K\ (x_2)^2 \quad => \quad y^2 = 10\ (x_2)^2 \quad => \quad y = 3{,}16\ x_2$

Conclusão: O avanço tecnológico é mais bem aproveitado pelo programa de menor complexidade.

Exercícios:

♣ Desenvolva o mesmo cálculo acima para algoritmos de complexidade exponencial (2^n) e problemas com entrada de tamanho x_3 na máquina mais lenta.

♣ Se um computador executa uma instrução em 1 microssegundo, que tamanho de entrada de problema um algoritmo de complexidade quadrática conseguirá executar em 1 segundo?

♣ Sua empresa precisa processar problemas 6 vezes maiores que os atuais, dentro do mesmo prazo. Os custos para melhoria do poder de processamento do computador é dado pela função x(v+12), onde x é o valor atual do computador, e *v* é o aumento de poder de processamento pretendido. O algoritmo da empresa tem complexidade n^2. Uma equipe externa cobra 40x para revisar o algoritmo, e garante que diminui sua complexidade para uma escala linear. Você indicaria a contratação de tal serviço para a sua empresa, levando em conta apenas a questão de custo? Justifique.

☼ Revisão: Logaritmo:

$$\log_a a^n = n$$
$$\log_a (n.\ m) = \log_a n + \log_a m$$

Há casos de avaliação de complexidade onde se identificam mais de uma instrução básica no algoritmo para contabilidade, e eventualmente podem ser atribuídos pesos a cada tipo de instrução.

Caso o algoritmo exemplificado acima aborte a execução ao encontrar a chave desejada, a complexidade poderá ser diferente para problemas com entrada do mesmo tamanho. Neste caso, trabalha-se com valores de **complexidade no pior caso** ou **complexidade média**. A complexidade no pior caso considera a maior complexidade, dentre todas as entradas de dados possíveis de um mesmo tamanho. A complexidade média leva em conta a probabilidade de ocorrência de cada entrada de um mesmo tamanho.

♣ Calcule a complexidade no pior caso para o algoritmo exemplo.

♠ No algoritmo exemplo, considerando-se valores não repetidos no vetor, e também que todos os elementos, e somente eles, são candidatos a chave. Considerando-se também um vetor de tamanho *n*, numa dada execução do algoritmo a probabilidade de busca por um dos seus valores é 1/n. Assim tem-se como complexidade média para o algoritmo:

Tomando como operação básica para contabilização da complexidade a comparação de valores já mencionada acima, calcula-se o número médio de ocorrências dessa operação levando-se em conta a probabilidade de ocorrência de cada um dos casos: chave na 1ª posição, chave na 2ª posição, etc...

$$1/n \cdot (1+2+3+ \ldots + n) = (1+n)/2$$