1 Grafos e Algoritmos

1.1 Algoritmos e contagem do número de operações

- Problemas de otimização e problemas computacionais em geral são resolvidos por meio de algoritmos.
- De uma forma vaga, podemos dizer que um algoritmo é um conjunto finito de instruções do tipo usado em linguagens de programação tais como: operações aritméticas, instruções condicionais, intruções de leitura e escrita.
- O tempo de execução de um algoritmo depende de vários fatores, entre eles: boa prática de programação, codificação das instruções de forma inteligente, estrutura de dados, equipamento onde está sendo executado.
- Apesar da importância destes fatores, estamos interessados em avaliar a qualidade de um algoritmo independentemente da forma em que está codificado ou da máquina onde está sendo executado.
- Uma maneira de avaliar o desempenho computacional de um algoritmo independentemente de uma implementação particular é calcular aproximadamente o número de operações (aritméticas, condicionais, etc.) que o mesmo executa.
- Esta prática é em geral satisfatória, apesar de desconsiderar que operações com números inteiros de poucos dígitos são menos trabalhosas que operações envolvendo números reais de alta precisão, ou números inteiros de muitos dígitos.
- Um cálculo mais preciso considera também os dados do problema.
- Neste curso, iremos considerar apenas os dados relativos à dimensão do problema.

1.2 Contagem do número de operações

- Procura-se então fazer uma estimativa do crescimento do número de operações em função dos parâmetros que definem o problema.
- Para maior precisão na estimativa do número de operações é utilizada a expressão “ordem de magnitude”.

1.2.1 Definição

Sejam $f, g : \mathbb{R}_+ \to \mathbb{R}_R$. Dizemos que:

1. $f(n)$ é $O(g(n))$ (f é da ordem de g) se existirem contantes c, n_0 tais que $f(n) \leq cg(n)$ para $n \geq n_0$ (complexidade no pior caso).
2. $f(n)$ é $\Omega(g(n))$ (f é da ordem de g) se existirem constantes c, n_0 tais que $f(n) \geq cg(n)$ para $n \geq n_0$ (complexidade no melhor caso).
3. $f(n)$ é $\Theta(g(n))$ se $f(n)$ é $O(g(n))$ e $f(n)$ é $\Omega(g(n))$ (complexidade exata).

1.3 Como avaliar se a complexidade de um dado algoritmo é boa ou ruim?

- De maneira geral algoritmos com complexidade computacional polinomial são considerados rápidos e eficientes enquanto que os algoritmos com complexidade exponencial são vistos como lentos e ineficientes.
- Este ponto de vista se justifica em muitas, mas não todas, situações.
- A otimização linear é um exemplo onde algoritmos exponenciais (baseados no método simplex) e algoritmos polinomiais (métodos de ponto anterior) competem em pé de igualdade.
- O estudo de complexidade de algoritmos será útil para determinarmos o grau de dificuldade de resolução de problemas em Grafos.
- Em geral as medidas de complexidade são feitas em função da dimensão do problema.
- No caso de grafos em função do número de vértices, n, e do número de arestas, m.