UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Computação
Avenida João Naves de Ávila, 2121, Bloco 1B - Bairro Santa Mônica, Uberlândia/MG, CEP 38400-902
Telefone: +55 (34) 3239-4218 - www.facom.ufu.br - cocom@ufu.br

Bacharelado em Ciência da Computação
Bacharelado em Sistemas de Informação
Disciplina: Algoritmos e Estruturas de Dados 1 – AED1 [GBS024/GSI006]
Prof. Me. Claudiney R. Tinoco

3ª Lista de Exercícios – Revisão C: Fechamento da revisão

"Algoritmos Genéticos constituem uma técnica de busca e otimização, altamente paralela, inspirada no princípio Darwiniano de seleção natural e reprodução genética.

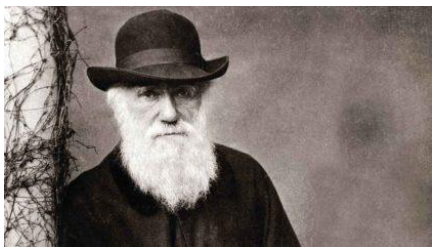

Figura 1- Charles Robert Darwin

Os princípios da natureza nos quais os GAs se inspiram são simples. De acordo com a teoria de C. Darwin, o princípio de seleção privilegia os indivíduos mais aptos com maior longevidade e, portanto, com maior probabilidade de reprodução. Indivíduos com mais descendentes têm mais chance de perpetuarem seus códigos genéticos nas próximas gerações. Tais códigos genéticos constituem a identidade de cada indivíduo e estão representados nos cromossomos...

Estes princípios são imitados na construção de algoritmos computacionais que buscam uma melhor solução para um determinado problema, através da evolução de populações de soluções codificadas através de cromossomos artificiais." [Pacheco, M. A. C.]

Considerando as informações acima, este trabalho tem como objetivo desenvolver um programa capaz de realizar uma das fases do processo evolutivo: a seleção. Dada uma população com *N* indivíduos, o algoritmo deve selecionar pares de indivíduos que participarão da fase de reprodução. Cada indivíduo possui um código genético e uma aptidão. O código genético guarda a informação que será compartilhada na fase de produção, já a aptidão representa o potencial de cada indivíduo. De acordo com a teoria de Darwin, quanto maior a aptidão do indivíduo maior é sua chance de entrar na fase de reprodução.

Os indivíduos e a população podem ser implementados com a seguinte estrutura:

```
struct individuo {
        int codigo_genetico[10]; // valores binarios aleatorios
        int aptidao_abs; // pontencial de cada individuo (0.0 <= aptidao <= 10.0)
        int aptidao_relativa; // deve ser calculada em tempo de execução...
};

struct populacao {
        struct individuo *pop; // um vetor de N individuos
        int tamanho; // tamanho N da população (N = 100)
};
typedef struct populacao populacao;
```

Uma das formas de selecionar dos pares de indivíduos é através do "método da roleta". Com podemos observar no exemplo da Figura 2, o método da roleta consiste em transformar a aptidão de cada indivíduo em uma aptidão relativa, de tal forma que essa relatividade (porcentagem) representa quão "forte" é o potencial de um indivíduo perto de todos os outros indivíduos da população. Por exemplo, o indivíduo S_2, tem 47% de chance de ser escolhido.

Indivíduo S_i		Aptidão $f(S_i)$	Aptidão Relativa
S_1	10110	2.23	0.14
S_2	11000	7.27	0.47
S_3	11110	1.05	0.07
S_4	01001	3.35	0.21
S_5	00110	1.69	0.11

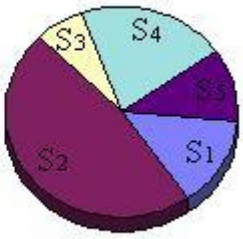

Figura 2 - Exemplo método da roleta.

Após calcular a aptidão relativa de todos os indivíduos, é feito o sorteio de um número pseudo-aleatório que representará onde a "bolinha da roleta caiu". A partir desse resultado, basta verificar qual individuo está representado nessa parte da roleta. Esse indivíduo será um dos indivíduos de um par que entrará para a fase de reprodução. Ao selecionar um par, basta imprimir esse para o usuário: par_001 = (ind-i, ind-j)

Portanto, pede-se:

- Implemente o programa descrito de seleção com o método da roleta;
- A roleta deve ser uma função que recebe com parâmetro a população e a quantidade (não pode ser um valor ímpar) de indivíduos que serão selecionados;
- Faça um teste de seleção para 100.000 indivíduos, i.e., 50.000 pares;

Obs.1: Para o sorteio de número aleatórios, utilizar as funções srand() e rand();
Obs.2: Para sortear um número binário, basta sortear um número inteiro e verificar se esse é par (1) ou ímpar (0), ou vice-versa;
Obs.3: Para sortear um número aleatório entre 0.0 e 1.0, faça:
*sorteio = ((**double**) rand()) / ((**double**) RAND_MAX);*