Aula Prática 2

Implementar o método de busca local *Hill Climbing*, utilizando os três critérios para seleção do vizinho.

Dica: Para gerar os vizinhos de uma solução, utilize o método troca_bit(int *s, int j), que recebe o vetor de solução *s* e a posição *j* correspondente ao bit a ser trocado.

1. **Primeiro Aprimorante**: em cada iteração assim que se encontra uma solução melhor que a corrente essa solução vira a solução

```
procedimento BuscaLocalPrimeiroAprimorante
     melhoria = 1;
     enquanto (melhoria) faça
          melhoria = 0;
          para todo s'∈ N(s) faça
               se f(s') > f(s) então // problema de max.
                    s = s'; // copiar posição a posição
                    melhoria = 1;
                    break;
               fim_se
          fim_para
     fim_enquanto
     retorne s; // no código s já é a solução global
fim BuscaLocalPrimeiroAprimorante;
```

2. **Melhor Aprimorante**: em cada iteração toda a vizinhança é pesquisada e a melhor solução da vizinhança vira a nova solução corrente.

```
procedimento BuscaLocalMelhorAprimorante
     melhoria = 1;
     enquanto (melhoria) faça
          melhoria = 0; f_max = -INF;
          para todo s'∈ N(s) faça
               se f(s') > f_max então // problema de max.
                    s_max = s'; // guardar só a posição
                    f_max = f(s');
               fim_se
          fim_para
          se f_max > f(s) então
               s = s_max; // copiar posição a posição
               melhoria = 1;
          fim_se
     fim_enquanto
     retorne s;
fim BuscaLocalMelhorAprimorante;
```

3. **Randômica** ou **Aleatória**: cada iteração consiste em escolher um vizinho qualquer e aceita-lo somente se ele for de **melhora**. Se o vizinho não for de melhora, a solução corrente permanece inalterada e outro vizinho é gerado. O procedimento é interrompido após um certo número fixo de iterações sem melhora no valor da melhor solução obtida até então. A solução final não é necessariamente um ótimo local.

```
procedimento BuscaLocalRandomica
     iter = 0; // contador de iterações sem melhora
     enquanto (iter < iterMax) faça
          iter = iter +1;
          selecione aleatoriamente s'∈ N(s);
          se f(s') > f(s) então // problema de max
               iter = 0;
               s = s';
          fim_se
     fim_enquanto
     retorne s;
fim BuscaLocalRandomica;
```