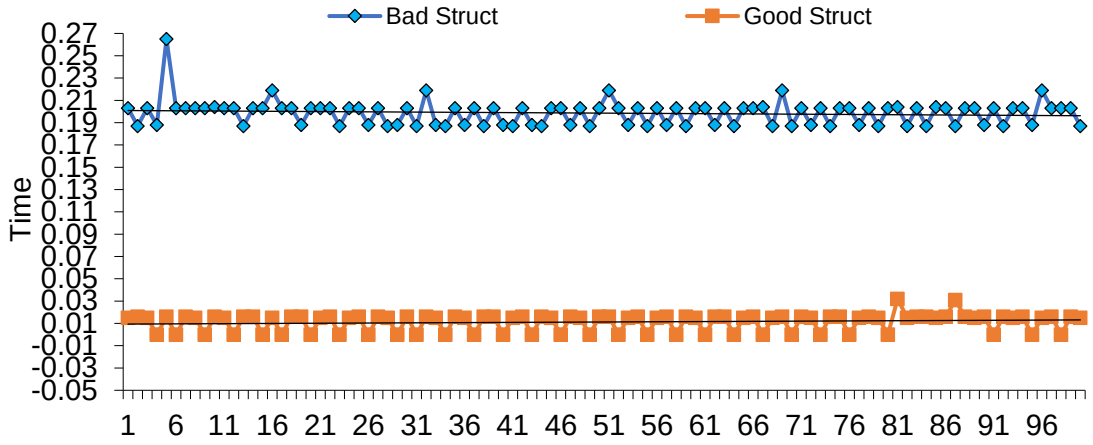
Pelo laboratório da última aula, em que usamos uma lista ligada simples com inserção de dados no fim, observamos que essa abordagem não é muito otimizada para o problema de armazenar um arquivo de imagem que usa o conceito de matriz esparsa. Assim, faça uma reflexão deste problema de desempenho (tempo) e implemente um outro tipo de lista que "resolva" o problema.

Neste laboratório você deve criar um programa que implementa a lista ligada do Lab. 04 (método "InsereFimDaListaLigada" no template abaixo) e que tenha um outro método que implementa a sua proposta para resolver o problema (método "InsereNaListaLigadaNova" no template abaixo). Por fim, vamos fazer um teste empírico para demonstrar que sua proposta é melhor em performance (tempo) do que o algoritmo implementado na última aula. Para isso, vamos executar cada algoritmo 3 vezes e medir o tempo de execução de cada execução. Depois faça um gráfico demonstrando os tempos de cada algoritmo (veja exemplo abaixo) e explique os fatores que levaram a tamanha diferença de tempo.

- O experimento de medir o tempo deve ser realizado tendo como base as imagens: img02.pgm (2.97MB), img03.pgm (4.69MB), ~~img01.pgm (6.27MB)~~. Sendo que para cada imagem, seu relatório deve ter um gráfico semelhante ao apresentado abaixo. O gráfico abaixo foi desenvolvido usando como base a imagem img00.pgm.
- Para facilitar, estou fornecendo um template do programa que implementa a parte experimental de medir o tempo.

Ao final, você deve entregar via Moodle um relatório descrevendo o problema, sua proposta de solução e uma justificativa para esta solução. Além disso, o relatório também deve apresentar os gráficos de tempo e o código fonte completo (ex., main.c). O relatório também deve apresentar um gráfico que demonstre a complexidade do algoritmo das duas implementações.

Essa tarefa deve ser feita em grupo de até três alunos.



Template

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct node{
   int Linha;
   int Coluna;
   int PixelValue;
  //.....................................................
} Tnode;

char identificador[5]=""; //P2 from file img
int MaxLin=0, MaxCol=0, MaxPixel=0; //from file img
```

```
// Insere 1 pixel da imagem no final da lista
// se a lista estiver vazia, cria
// se a lista tiver elementos, coloca no fim....
// Essa é uma lista ligada simples com inserção no fim
Tnode *InsereFimDaListaLigada(Tnode *Imagem, int Linha, int Coluna, int PixelVl ){
   //.....................................
   return Imagem;
}

// Esse método deve fazer a mesma coisa que o método "InsereFimDaListaLigada" (acima)
// contudo, ele deve implementar um tipo de lista ligada (qualquer outro) que seja mais
// otimizado para o problema que estamos trabalhando.
// Vc pode implementar qualquer tipo de lista, mas essa deve ser melhor em tempo do que aquela
// implementada no método "InsereFimDaListaLigada" (acima)
Tnode *InsereNaListaLigadaNova(Tnode *Imagem, int Linha, int Coluna, int PixelVl ){
   //.....................................................
   return Imagem;
}

// Libera a Lista da memoria.....
Tnode * LiberaMemoria(Tnode *Imagem){
   Tnode *ptr = Imagem, *tmp;
   while(ptr!=NULL){
      tmp=ptr;
      ptr = ptr->next;
      free(tmp);
      if (ptr==Imagem)
         break;
   }
   return NULL;
}

// Lê o cabeçalho da imagem
// e depois lê cada pixel da imagem, se o pixel não for a cor preta, insere ele na lista
// A lista também armazena a Linha e a Coluna onde o pixel se encontra
Tnode *CarregaImagem(Tnode *Imagem, char File[], int TipoDaLista){
   //.............
   for(/*.................*/){
      for(/*.................*/){
         /*.................*/
         if (/*.................*/){
            if (TipoDaLista==1) //Lista otimizada para o problema
               Imagem=InsereNaListaLigadaNova(Imagem, Lin, Col, Pixel);
            else //Lista com problema de desempenho (implementado no último lab)
               Imagem=InsereFimDaListaLigada(Imagem, Lin, Col, Pixel);
         }
      }
   }
   //.............
   return Imagem;
}

//Percorre a lista procurando os nós com o OldPixel e substitui por NewPixel
void ProcessaPixel(Tnode *Imagem, int OldPixel, int NewPixel){
   //.............
}
```

```
// Monta a imagem com base na lista.
// Lembre-se que a lista tem apenas os pixels não brancos
// Então vc tem que montar a imagem com base na lista
// ou seja onde "não" tiver dado, tem que gravar a cor branca.
// Imagine que a imagem deve uma matriz esparsa como abaixo,
// mas sua lista armazena apenas os elementos maiores que zero
// então para reconstruir a imagem, antes de gravar o 5 no arquivo
// vc tem que gravar o zero seis vezes. O mesmo para os outros elementos
// 0 0 0 0 0 0 5 0 8
// 0 0 1 5 0 0 8 8 0
// 1 2 2 2 5 5 0 0 0
void DescarregaNovaImagem(Tnode *Imagem, char File[]){
   //.............
}

int main(){
   Tnode *Imagem = NULL;
   char ReadFile[30]="", SaveFile[30]="";
   int OldPixel=0, NewPixel=0;
   int TipoDaLista;
   TipoDaLista=0; // 0 - Lista ligada simples com inserção no fim
   TipoDaLista=1; // 1 - sua proposta de lista.....

   printf("Informe o nome do arquivo de imagem para ler: ");
   scanf("%s",ReadFile); //"Img04.pgm"
   printf("No processamento, vc quer substituir qual pixel: ");
   scanf("%d",&OldPixel); // 255 - Branco
   printf("Substituir por qual valor: ");
   scanf("%d",&NewPixel); // 100 - um tom de cinza
   printf("Informe o nome do arquivo NOVO arquivo: ");
   scanf("%s",SaveFile); //"teste.pgm"

   int i;
   //vamos fazer um teste 3 vezes para ver o tempo
   for(i=0;i<3;i++){
      clock_t begin = clock();
      Imagem=CarregaImagem(Imagem,ReadFile, TipoDaLista);
      ProcessaPixel(Imagem, OldPixel, NewPixel);
      DescarregaNovaImagem(Imagem,SaveFile);
      Imagem=LiberaMemoria(Imagem);
      clock_t end = clock();
      double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
      printf("%lf\n",time_spent); // imprime o tempo
   }
   return 0;
}
```