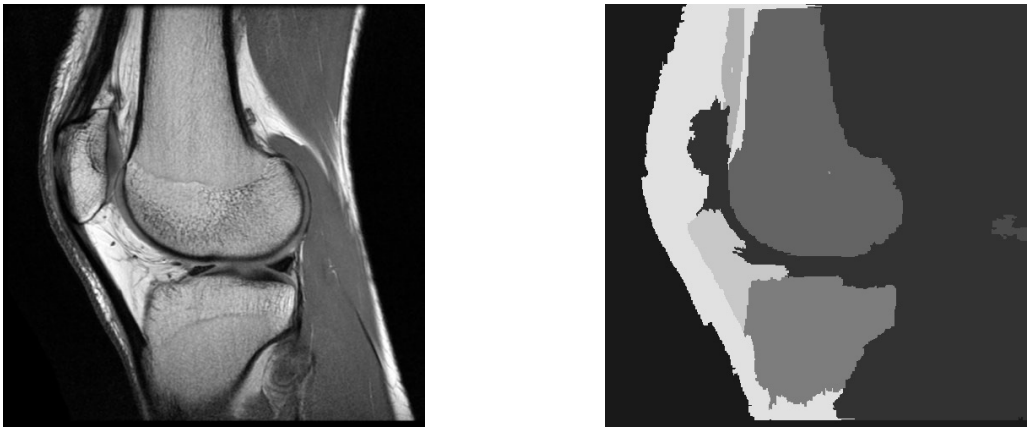
Lab 03
MC558A

Data de Entrega: 19/09

Estamos construindo um novo equipamento inteligente para capturar imagens médicas, com o diferencial fazer análises em tempo real dos exames.

Mas é preciso, primeiro, segmentar as imagens (separar ossos, músculos, cartilagens, etc) para que seja possível analisar cada grupo diferente. Assim, o operador da maquina seleciona "pontos chaves"da imagens e cabe ao equipamento segmentar e processar a imagem a partir desses pontos.

Sua função será a de desenhar um algoritmo para segmentar as imagens conforme vamos descrever abaixo, lembre-se que as imagens são grandes, então use tudo que sabe para deixar o seu código o mais rápido possível.



(a) Imagem do exame (b) Imagem Segmentada em 10 pontos

Figura 1: Exemplo de imagem segmentada

(a) Imagem original (b) Pontos para segmentação (c) Imagem Segmentada

Figura 2: Exemplo de imagem com pontos para segmentação

A segmentação funciona da seguinte forma:

Cada pixel é representado com um valor inteiro entre 0 e 255, representando uma escala de cinza onde 0 é a cor preta e 255 a cor branca.

Vamos considerar que cada pixel está relacionado somente aos seus vizinhos localizados na direita, na esquerda, acima e abaixo.

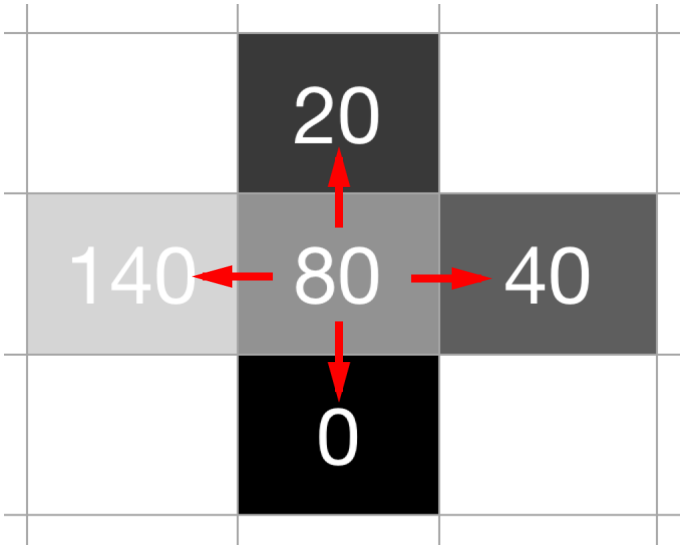


Figura 3: Representação dos píxeis da imagem

Dizemos que a diferença entre os pixeis é dado pelo módulo da subtração de suas cores. Por exemplo, na Figura 3, a diferença entre o pixel central e o pixel do topo é de 60, pois é o módulo da subtração de $80 - 20$. Para o pixel à direita, a diferença de valor é 40 $(80 - 40)$. Note que por serem valores sempre positivos, a **diferença entre dois pixeis também está entre 0 e 255.**

O objetivo da segmentação é agrupar pixeis com valores semelhantes. Ou seja, dada a imagem e os pontos iniciais, **queremos dividir a imagem em áreas onde a soma das diferenças entre os pixeis de mesmo componente seja mínima.**

Veja que podem existir diferentes segmentações de custo mínimo, aqui vamos aceitar qualquer uma delas.

Descrição da entrada :

A entrada é baseada no formato PGM , este representa uma imagem, mas também pode ser lido como texto, mais detalhes em www.netpbm.sourceforge.net/doc/pgm.html.

A primeira linha é dada pela string "P2"que informa o tipo de codificação utilizada, seguida de dois inteiros M N ($1 \leq M, N \leq 1000$), respetivamente o número de colunas e linhas da imagem.

Um inteiro no valor 255, representando o valor máximo utilizado. Uma matriz de inteiros com valores entre 0 a 255.

Ao fim da tabela, temos um inteiro K ($1 \leq K \leq 255$), representando o número de pontos para segmentação, e K-linhas com 2 valores, X Y, respetivamente a posição de coluna, linha do ponto inicial de cada segmento.

Descrição da saída :

Um inteiro representando a soma dos custos entre elementos do mesmo segmento. É garantido que esse valor cabe em um *long long int*.

Formato de entrada	**Matriz Final**	**Formato de saída**
P2	255 255 255 255 255	0
5 5	255 0 0 0 255	
255	255 0 0 0 255	
10 10 10 10 10	255 0 0 0 255	
10 90 90 90 10	255 255 255 255 255	
10 90 90 90 10		
10 90 90 90 10		
10 10 10 10 10		
2		
0 0		
2 2		

Tabela 1: Exemplo 1

Análise Exemplo 1 Temos os pontos $(0, 0)$ e $(2, 2)$ como iniciais. Temos o "custo"entre os pontos $(0, 0)$ $(0, 1)$ e $(0, 0)$ $(1, 0)$ igual a 0, pois $abs(10 - 10) = 0$. Análogo ao ponto $(2, 2)$, com todos os seus 4 vizinhos a um custo 0. Assim, a soma do custo de conectar os elementos do mesmo segmento é 0.

Note que dos elementos centrais (com cor 90) o custo deles até os pontos da borda (com custo 10) seria 80, mas como a partir do ponto (0,0) é possível conectar todos eles a um custo menor, isso justifica a borda pertencer ao primeiro segmento e os pontos do centro ao segundo.

A matriz final foi preenchida com valores distintos somente para ilustrar os diferentes segmentos, ela não faz parte da saída mas pode ser usada para ajudar o debug.

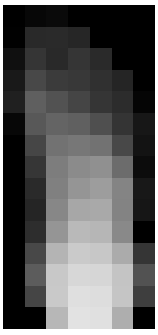
Formato de entrada	**Matriz Final**	**Formato de saída**
P2	255 255 255 255 255	15
5 5	255 0 0 0 255	
255	255 0 0 0 255	
10 15 10 10 10	255 0 0 0 255	
15 90 90 90 10	255 255 255 255 255	
10 90 90 90 10		
10 90 90 90 10		
10 10 10 10 10		
2		
0 0		
2 2		

Tabela 2: Exemplo 2

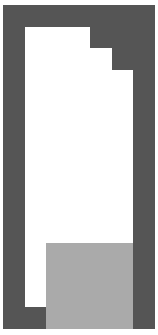
Análise Exemplo 2 Temos o "custo"entre os pontos $(0, 0)$ $(0, 1)$ e $(0, 0)$ $(1, 0)$ igual a **5**, pois $abs(10 - 15) = 5$. Portanto, temos custo 10 para unir os segmentos (0,0), (0,1), (1,0). E também temos custo 5 de (1,0) para (2,0) e de (0,1) para (0,2). Mas veja que, uma seja uma vez visitado (0,2) ou (2,0), é possível "dar a volta"e conectar todos os outros elementos de valor 10 a custo 0, assim, vamos totalizar 15 o custo para cobrir os elementos do primeiro segmento. Assim como no exemplo 1, o custo do segundo segmento permanece 0. Logo, temos custo final igual a $(15 + 0)$, que é a saída esperada.

Formato de entrada	**Matriz Final**	**Formato de saída**
P2	85 85 85 85 85 85 85	972
7 15	85 255 255 255 85 85 85	
255	85 255 255 255 255 85 85	
0 7 10 0 0 0 0	85 255 255 255 255 255 85	
1 41 42 38 0 0 0	85 255 255 255 255 255 85	
16 53 41 53 42 0 0	85 255 255 255 255 255 85	
25 72 59 56 48 37 0	85 255 255 255 255 255 85	
32 95 82 69 53 44 5	85 255 255 255 255 255 85	
7 88 100 96 79 52 23	85 255 255 255 255 255 85	
0 70 114 119 114 72 19	85 255 255 255 255 255 85	
0 54 125 139 144 108 12	85 255 255 255 255 255 85	
0 42 128 149 157 132 24	85 255 170 170 170 170 85	
0 37 135 167 169 131 22	85 255 170 170 170 170 85	
0 45 149 185 182 137 0	85 255 170 170 170 170 85	
0 72 180 202 199 171 53	85 85 170 170 170 170 85	
0 92 200 215 214 196 82		
0 69 201 223 221 197 62		
0 0 183 226 224 174 0		
3		
0 0		
4 14		
4 5		

Tabela 3: Exemplo 3



(a) Entrada



(b) Imagem resultante da matriz final

Figura 4: Figuras do Exemplo 3

Atenção!
Vamos trabalhar com imagens, seu código irá enfrentar instâncias de até 1000 x 1000 pontos, cuidado com a complexidade do algoritmo implementado!

Note que existem várias possíveis imagens finais de custo mínimo, por isso sua aplicação deve informar somente o valor do custo total.

Você pode usar a imagem final para auxiliar no debug da aplicação. No código base disponibilizado há uma função que exibe a imagem intermediária, basta salvar o output do programa no formato .PGM

Observações e Avaliação:

- Os programas que não estiverem compilando ou não passarem em algum dos testes pré-instalados no SuSy, terão nota 0.
- O arquivo fonte deve estar bem comentado! Qualquer função ou trecho de código não trivial deve conter uma breve descrição sobre o seu propósito.
- No inicio do arquivo fonte enviado ao SUSY, deverá haver uma descrição em alto nível sobre como a solução foi pensada e implementada. Também é necessário justificar a complexidade do algoritmo. É esperado uma descrição com cerca de 100 a 200 palavras.
- Um esqueleto da aplicação está disponível na pagina da disciplina, o uso é obrigatório. Entrada e saída de dados já estão disponíveis na main. É necessário somente implementar a função *solve*(), você pode implementar também outras funções auxiliares caso julgue necessário. Para compilar, basta executar o comando *make* no endereço do fonte, para testar com o exemplo, basta executar *make run*.
- É esperado que esta atividade demande cerca de 4 a 8 horas para ser finalizada, caso esteja demorando muito mais tempo, ou caso tenha qualquer dúvida, procure atendimento o quanto antes!
- Em caso de plagio, todos os alunos envolvidos serão imediatamente reprovados.