Algoritmos e Estruturas de Dados

1º Trabalho

Universidade de Aveiro

Tiago Melo (113362), Miguel Mota (119934)

Conteúdo

Capítulo 1

ImageCreateChessboard

Descrição do algoritmo

O algoritmo cria uma imagem em preto e branco com um padrão de tabuleiro de xadrez, com base nos parâmetros fornecidos (largura, altura, tamanho dos quadrados e cor inicial). O algoritmo começa por verificar as condições de entrada, aloca memória para a imagem e gera cada linha do tabuleiro alternando as cores dos quadrados. Para cada linha, calcula quantos blocos são necessários e ajusta o tamanho do último bloco, caso a linha não termine exatamente no final de um bloco. A função retorna a imagem gerada, representada por uma estrutura que armazena os dados da imagem em formato de compressão de execução (RLE).

Análise do espaço de memória ocupado pelas imagens criadas pela função ImageCreateChessboard()

Dimensions (width, height, square edge, first value)	**Memmory allocated(bytes)**	**Execution Time(**µs)	**Number of runs**
8, 8, 2, 1	192	29	32
20, 50, 5, 1	1200	74	200
256, 256, 128, 1	4096	2260	512
256, 512, 128,1	8192	7792	1024
120, 80, 4, 0	10240	357	2400
256, 512, 4, 1	135168	6847	32768
200, 200, 100, 1	3200	1258	400
400, 400, 200, 1	6400	7657	800
200, 20, 1, 0	16160	13	4000
400, 400, 1, 1	643200	8002	160000

Fórmula para o número de runs

$$runsLinha = \left\lceil \frac{n}{s} \right\rceil,\ totalRuns = m \times runsLinha$$

Explicação: Sendo **n** a largura da imagem, ou seja, o número de colunas e **s** o square_edge que representa a dimensão de cada bloco de pixels, o número total de runs por linha é o ceiling da divisão de **n** por **s**. O número total de runs numa imagem é o resultado da multiplicação do número de runs por linhas pelo número de linhas.

Fórmula para o espaço de memória ocupado

$$memoriaLinhas = m \cdot \left(\left\lceil \frac{n}{s} \right\rceil + 2 \right) \cdot 4$$

$$memoriaTotal = m \cdot 8 + m \cdot \left(\left\lceil \frac{n}{s} \right\rceil + 2 \right) \cdot 4$$

Explicação: Sendo **n** a largura da imagem, ou seja, o número de colunas da imagem e **m** a altura da imagem, ou seja, o número de linhas da imagem, o espaço total ocupado por uma linha da imagem é a quantidade de espaço ocupado por cada elemento (sizeof(uint32)) vezes o número de runs da linha mais 2 para o EOR (marcador de fim de linha) e a cor inicial. A quantidade de espaço total ocupada pela imagem em função do número de linhas, colunas e tamanho da aresta é a quantidade de espaço ocupado pelo array de arrays de ponteiros mais m vezes a memória ocupada por uma linha.

Qual é o padrão de xadrez que dá origem à imagem com menos runs? Quantas são?

O padrão de xadrez que dá origem a imagem com menos runs é aquele em que cada linha é composta por apenas 2 runs o que resulta num número total de $h \times 2$, sendo h a altura da imagem, ou seja, o número de linhas.

Qual é o padrão de xadrez que dá origem à imagem com mais runs? Quantas são?

O padrão que dá origem a imagem com mais runs e aquele em que cada pixel é uma run, ou seja, cada pixel é alternado entre WHITE e BLACK. Isto resulta num número total de runs de $p \times h$,sendo p o número de pixels em cada linha da imagem e h a altura da imagem, ou seja, o número de linhas da imagem.

Capítulo 2

ImageAND

Descrição do algoritmo

O algoritmo vai realizar a operação AND (&&) entre cada pixel na mesma posição de duas imagens fornecidas nos campos de entrada da função, desde que estas tenham as mesmas dimensões, e alocar o resultado nessa mesma posição de uma nova imagem.

Análise computacional do número de operações efetuadas envolvendo os valores dos pixels das imagens na função ImageAND()

Altura(a)	Largura(l)	N(a*l)	Número de Operações	Execution Time(µs)
2	2	4	12	500
3	3	9	27	624
4	4	16	48	399
10	10	100	300	890
12	10	120	360	497
10	12	120	360	420
15	10	150	450	373
20	10	200	600	584
10	50	500	1500	387
10	100	1000	3000	829
50	25	1250	3750	360

Análise formal do algoritmo

Com uma observação rápida do código conseguimos perceber que vão ser realizadas 3 operações envolvendo os valores dos pixels das diferentes imagens, duas de entrada e uma de saída. É possível também perceber que a quantidade de operações realizadas vai estar diretamente ligada ao tamanho da imagem de saída. Assim, sendo N o tamanho da imagem final, ou seja, o número total de pixels que equivale à multiplicação da altura(a) pela largura(l) da imagem, podemos dizer que o número de operações vai ser dado pela equação:

$$OP(N) = 3N$$

Sobre os resultados obtidos

Com ambas as análises realizadas é possível afirmar que os dados obtidos com a análise computacional do algoritmo confirmam o que era expectável com a análise formal. Podemos concluir também que o número de operações está unicamente associado ao tamanho das imagens e não ao conteúdo das mesmas e que o tempo de execução do algoritmo não está diretamente ligado ao número de operações entre pixels.

É possível ainda observar que este algoritmo é determinista e que:

$OP(n) \in O(n)$, uma vez que $5N \geq 3N$

$OP(n) \in \Omega(n)$, uma vez que $N \leq 3N$

$OP(n) \in \Theta(n)$, uma vez que $N \leq 3N \leq 5N$