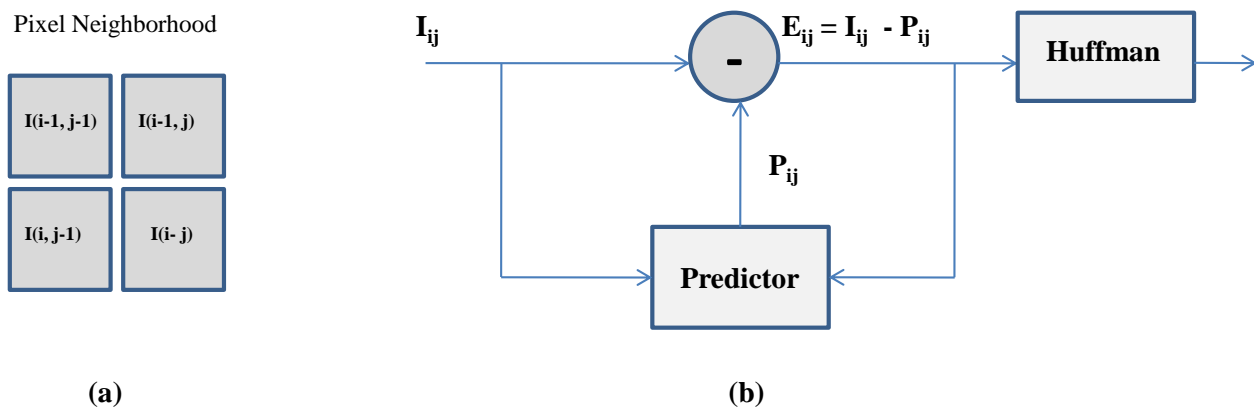


## Programming Project 1 Parallel Lossless DPCM (Differential Pulse Code Modulation) of Images

The goal of this project is to compress an image in parallel using a cluster of workstations. The image(s) to be compressed are gray scale with pixel values between 0 to 255. There are different techniques for compressing images. They are broadly classified into two classes called lossless and lossy compression techniques. As the name suggests in lossless compression techniques, no information regarding the image is lost. In other words, the reconstructed image from the compressed image is identical to the original image in every sense. Whereas in lossy compression, some image information is lost, i.e the reconstructed image from the compressed image is similar to the original image but not identical to it.

In this project we will use a lossless compression technique called Differential Pulse Code Modulation (DPCM) to compress the image. The Encoder for this technique is shown below. It mainly consists of two blocks; a Predictor and a Huffman Encoder.



In the figure above,  $I_{ij}$  is the pixel value and  $P_{ij}$  is the predicted value. For each pixel, residual error  $E_{ij}$  is calculated. This technique takes advantage of the fact that gray scale values for the neighboring pixels will be close to each other. The *Predictor Block* predicts the value of a pixel based on the values of its neighboring pixels. A typical Predictor Function is:

$$P(i, j) = a * I(i - 1, j) + b * I(i - 1, j - 1) + c * I(i, j - 1)$$

where  $P(i,j)$  is the predicted value for the pixel at coordinate  $(i,j)$ ;  $a$ ,  $b$  and  $c$  are the predictor coefficients and  $I(i,j)$  is the pixel value in the original image.

In DPCM, the difference between the predicted and actual values are computed. The image with these difference values is called the Residual Image. This Residual Image is encoded using Huffman Coding and then transmitted or stored in the database.

In this project, we first compute the Residual Image and then find the Entropy of the Residual Image in parallel. We will not be computing the Huffman codes since the Entropy gives us an approximate value for the Compression Ratio that can be obtained when Huffman Coding is used. For those who would want to obtain more information about Huffman Coding, some references are provided at the end of this text.

## Project Implementation

Project Implementation mainly consists of two parts:

1. Computing the Residual Image
2. Computing the Entropy of the Residual Image

The residual Image can be computed using the following algorithm:

```
DPCM Encode(I,a,b,c)
E(1,1) = I(1,1);
E(2:n,1) = I(2:n,1);
E(1,2:n) = I(1,2:n);
for i = 2:n
    for j = 2:n
        E(i,j) = I(i,j) - ceil{ a*I(i,j-1) + b*I(i-1,j-1) + c*I(i-1,j) };
    end for;
end for;
end;
```

Here I is the given image consisting of  $n \times n$  pixels; E is the residual image to be compressed and transmitted; a,b and c are the predictor coefficients.

To compute the Entropy, we need to find the probabilities for the occurrence of each symbol in the Residual Image. The *probability of occurrence* for a symbol is the ratio of the frequency of occurrence for that symbol to the total number of pixels. For example: Lets assume that 30 pixels are having a value of “5” in the residual image consisting of 64 pixels. Probability of occurrence for “5” is  $P(\text{“5”}) = 30/64 = 0.46$ . Once we compute the probabilities of all the symbols in the residual image, Entropy (represented by H) can be calculated using the formula:

$$H = - \sum_{i=1}^m p(s_i) \times \log_2(p(s_i))$$

where m is the total number of symbols present in the residual image.

## Data Partitioning and Communication

The Image data can be divided equally among the processors so that each processor gets equal number of rows. If the image has “N” rows and we use “p” processors then we can divide the data such that each processor gets  $N/p$  rows.

From the algorithm, we see that each processor  $P_i$  except for the first one needs the last row in  $P_{i-1}$  to proceed with its part of computing the residual image. This is the only communication needed between the processors for computing the residual image. Even this communication could be avoided if the boundary rows are replicated in processor memories.

Once processors compute the residual values for the pixels in their memories, they can start computing the probability of occurrences for all the symbols present in the residual image. To do this, each processor computes the frequency of all the symbols present in the residual image stored in its memory, then the root processor computes the entropy based on these frequencies reported by the processors.

## Data

The Image data will be available in the directory <http://web.mst.edu/~ercal/387/pr-proj-1/>. You will work on images of four different sizes with 512x512, 1024x1024, 2048x2048, and 4096x4096 pixels each. The image data will be stored row wise in the following files: data.512, data.1024, data.2048, and data.4096. You may also use matlab for viewing the images. In matlab, you can read the image data from the files into a two dimensional array and ask the system to show the image. The following matlab code can be used for this purpose:

```
fid = fopen('data.512','r');
for i = 1:512
    for j = 1:512
        x(i,j)=fscanf(fid,'%d',1);
    end;
end;
fclose(fid);
imshow(x, [0 255]);
```

## Huffman Coding

The following information is for those who are interested in more details about Huffman Coding.

The algorithm for constructing the Huffman Tree is the following:

```
Huffman Encode (Si,Pi)
begin
    create nodes labeled Si
    F = { Si | 1 <= i <= m}
    for i = 1:(m-1) do
        find the two smallest probabilities belonging to symbols Sj,Sk.
        create a new node Si' with probability (Pj + Pk)
        F = (F - {Sj,Sk}) U Si'
    end for
    label left edges 0 and right edges 1.
    for i = 1 to m do
        code S with the binary sequence by concatenating the
        bits belonging to nodes on the path from the root to Si.
    end for.
end.
```

## References

1. *Data structures, Algorithms and Applications in C++*, Sartaj Sahni
2. The following link gives you more details about Huffman Coding:  
[http://en.wikipedia.org/wiki/Huffman\\_coding](http://en.wikipedia.org/wiki/Huffman_coding)

## Project Requirements

Write a parallel program which first computes the Residual Image and then computes the entropy for it as explained above. Use  $\mathbf{a} = \mathbf{b} = \mathbf{c} = 1/3$  in your Predictor Function.

### PART I (40 pts)

Run your program on MST Cluster and obtain timing results for 4 images stored in files **data.512**, **data.1024**, **data.2048**, and **data.4096** in <http://web.mst.edu/~ercal/387/pr-proj-1/> directory using P=1, 2, 4, 8, 16, and 32 processors (a total of 24 timing results). Tabulate your findings along with the entropy value calculated. Make sure your program generates correct results (**Entropy Values: (512) 3.99; (1024) 2.62; (2048) 3.92; (4096) 1.83**), otherwise, you may lose a significant percent of the total points. Use the following formula for the calculation of pixel values in the residual image:

$$E(i,j) = I(i,j) - \text{ceil}\{ a \cdot I(i,j-1) + b \cdot I(i-1,j-1) + c \cdot I(i-1,j) \}$$

### PART II (40 pts)

Answer the following questions with respect to your timing measurements:

1. How does the speedup change when you increase the image size for a fixed P ? If you observe any anomalies in the speedup, please explain the reasons clearly.
2. How does the efficiency change when you increase image size for a fixed P ? Explain.
3. How does the efficiency change when you increase P for the same image? Explain.

### PART III (10 pts)

Electronic copy of your program. Your program must: (i) contain a program overview/summary, your id etc. in the header, (ii) execute correctly, (iii) be well documented (Must include comments before every major statement, function/subroutine calls, and every MPI call explaining what the call is for. Must include specs for every function/subroutine)

### PART IV (10 pts)

Read the file "How to submit projects?" on the class website and follow all the instructions in there. You must store the requested files in your subdirectories.

### IMPORTANT NOTE:

1) In your timings, include only the following portions in your program:

- (a) Generation of residual image
- (b) Calculating the frequency of each symbol (p(.) values) in parallel
- (c) Calculating the entropy value for the entire image

2) Your timings should not include I/O parts. Namely;

- (a) the part where the image file is read (I/O)
- (b) distribution of the image portions to the parallel processes
- (c) printf() and scanf() statements.

3) To be more accurate, take the average of 4-5 runs for the same program and report it as one timing result.

**BONUS:** The best (and correct) running time for **data.4096** and P=16 will receive bonus points as described in the syllabus.