EE/CSCI 451
Spring 2017
Programming Homework 2a (Optional)
Assigned: February 3, 2017
Due: February 10, 2017, before 11:59 pm, submit via blackboard
Total Points: 35

Instructions

These are some additional problems to illustrate the benefits of parallelization using pthreads for two classic techniques widely used in big data analysis. You can solve Problem 1 of this assignment instead Problem 1 of PHW 2. Similarly you can solve Problem 2 of this assignment instead Problem 2 of PHW 2. No other replacements are allowed.

1 Problem 1: Random Sample Consensus (RANSAC) [10 points]

Random Sample Consesus (RANSAC) [1] is a classic technique in big data analysis. It is an iterative method to estimate the parameters of a model which fits a "significant" number of observations in a given dataset. The observations which fit into the model are known as *inliers* and the observations which do not fit are known as *outliers*. Note that this approach is different from other commonly used parameter estimation techniques such as least squares which try to estimate a model which best fits all the data points.

In this problem, we will implement a simple pthreads based RANSAC algorithm to fit a 2D line ($y = mx + c$, m and c are our model parameters) to fit a set of data points of the form (x_i, y_i). Let there be $n = 800,000$ points in our dataset ("dataset_ransac.txt"). The algorithm is as follows:

1. **Hypothesize:**
    - Randomly select a *Minimal Sample Set* i.e. a set of smallest size which can be used to define a model. Since we have two model parameters: m and c, we will randomly select 2 points from the dataset.

- Create a model using the Minimal Sample Set. Let the points in the Minimal Sample Set be (x_1, y_1) and (x_2, y_2). We will create a model using the following equations:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{1}$$

$$c = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1} \tag{2}$$

2. **Test:** This phase of the algorithm determines how the model generalizes for the entire dataset.

   - Create p threads. Let us denote the threads as Thread 0, Thread 1, ... Thread $p-1$. Also create a global array called `inlier` which will store all the inliers of the datasets. Initialize it to NULL.
   - Thread i will iterate through data points (x_s, y_s) to (x_t, y_t) where $s = 100000 * i$ and $t = 100000 * i + 99999$. It will try to fit each point (x_k, y_k) to the model created in Step 1 and calculate the error (ϵ) using the following equations:

$$\epsilon = (y_k - mx_k - c)^2 \tag{3}$$

   - If $\epsilon \leq 0.0001$, the point (x_k, y_k) is added to the array `inlier`. Note that since `inlier` is a shared variable, you need to access it using a lock.

3. The algorithm runs for multiple iterations to maximize the probability of finding a set which contains most of the inliers. However, we will just run for a single iteration.

A serial version of the program is provided with this assignment (ransac.c) for your reference. Name the program you write as p2_1a.c. Report the time it takes to execute steps 1 and 2 for values of $p = 2, 4, 8, 16, 32$. Also output the values of the parameters y and m and the number of inliers.

2 Problem 2: Hough Transform [15 points]

Hough Transform [2] is a technique which uses a voting procedure to estimate the parameters of a model to fit the observations of a given dataset. The motivation behind Hough Transform is that each observation indicates its contribution to a globally consistent solution. A global variable called accumulator is used to accumulates the "votes" of all the observations which is then used to determine the parameters of the model.

In this problem, we will implement a simple pthreads based Hough Transform algorithm to fit a 2D line represented in polar co-ordinates ($r = x\cos\theta + y\sin\theta$, r and θ are our model parameters) to fit a set of data points of the form (x_i, y_i). Let there be $n = 400000$ points in our dataset ("dataset_hough.txt").

We will use the following algorithm to find the best line which fits our dataset:

1. Initialize a global `accumulator` array of size 11×3. Each row of the array will be corresponding to a single value of $\theta \in \{0, 15, \ldots, 165\}$. Initialize the array using the following equations:

$$\texttt{accumulator}[i][0] = 15i \tag{4}$$
$$\texttt{accumulator}[i][1] = 0(//runningsum) \tag{5}$$
$$\texttt{accumulator}[i][2] = 0(//runningsumofsquares) \tag{6}$$
$$\tag{7}$$

2. Create p threads. Let us denote the threads as Thread 0, Thread 1, ... Thread $p-1$.

3. Thread i will iterate through data points (x_s, y_s) to (x_t, y_t) where $s = 100000 * i$ and $t = 100000 * i + 99999$. For each point (x_k, y_k), it will calculate $r_{\theta k}$ for different values of $\theta \in \{0, 15, \ldots, 165\}$ and update the `accumulator` array using the following equations:

$$r_{\theta k} = x_k \cos\theta + y_k \sin\theta \tag{8}$$
$$\texttt{accumulator}[\theta/15][1] = \texttt{accumulator}[\theta/15][1] + r_{\theta k} \tag{9}$$
$$\texttt{accumulator}[\theta/15][2] = \texttt{accumulator}[\theta/15][2] + r_{\theta k}^2 \tag{10}$$
$$\tag{11}$$

   Note that `accumulator` is a shared variable and you need to use locks or conditional variables to access it.

4. Once all the threads terminate, the main program should find a θ for which the mean squared error ϵ of all the corresponding r is the minimum. ϵ can be calculated as follows:

$$\epsilon = \frac{\texttt{accumulator}[i][2]}{n} - (\frac{\texttt{accumulator}[i][1]}{n})^2 \tag{12}$$

A serial version of the program is provided with this assignment (hough.c) for your reference. Since this program has math functions such as cos and sin, in order to compile it use the following command: `gcc -o hough hough.c -lm`. Name the program you write as p2_2a.c. Report the time it takes to execute steps 2, 3 and 4 for values of $p = 2, 4, 8, 16, 32$. Also output the values of the parameters θ and r (average of all the r corresponding to θ) which have the minimum least square error.

3 Submission

You may discuss the algorithms. However, the programs have to be written individually. Submit the code (p2_1a.c and p2_1a.c) and the report via **Blackboard**. Your program should be written in C or C++. Make sure your program is runnable. Write clearly how to compile and run your code in the report as well. It is recommended to include a Makefile [3] along with your submission. If your program has error when we compile or run your program, you will lose at least 50% of credits.

References

[1] "RANSAC for Dummies,"
http://old.vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf

[2] "Hough Transform,"
https://en.wikipedia.org/wiki/Hough_transform

[3] "Using make and writing Makefiles,"
http://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html