CS/ECE 560 Foundations of Fine-Grained Parallelism — Spring 2012 HW #0
Review Due January 25, 2012 at 11:59pm

Use the provided latex template file to learn latex and complete this homework (see the assignments web page). ALL of the homeworks and project writeups in this course must be written using latex. Homework assignments are to be completed individually. Homeworks need to be submitted electronically via RamCT AND by email to mstrout@cs.colostate.edu by 11:59pm on the due date. Total points: 100

1. [30 points] Definitions
   Define each of the following terms IN YOUR OWN WORDS using one to two sentences. You can read other definitions online, but then write your own definition without using someone else's wording.

   - function
   - domain
   - range
   - one to one
   - onto
   - bijective
   - set theoretic notation
   - conjunction
   - disjunction
   - big O notation

2. [50 Points] Basic Performance Analysis

   We will start our exploration of application performance with a Smith-Waterman implementation provided at `http://www.cs.bc.edu/ clote/ComputationalMolecularBiology/` by Peter Clote (We did do some modifications so download the source file from the CS 560 assignments web page). Smith-Waterman is used to compare protein sequences and find where portions of the sequences match. It is a common algorithm used in Bioinformatics research. You should read some about this algorithm from various sources on the internet. The goal of this homework is to start measuring, manipulating, and reporting the performance of benchmarks.

   (a) The first step is to download the smithWaterman.c source code posted with this homework (HW0) on the assignments page. Then you should copy the file to one of the vege machines in the department (see the machines at the top of the list at `http://www.cs.colostate.edu/ info/machines`). Now compile the program with the following command:

```
gcc -O3 smithWaterman.c
```

Stay organized. Create a directory for this exercise and keep a log in a text file within the directory detailing what experiments you run and any difficulties you run into.

(b) Now copy the 30k_1.txt, 30k_2.txt, and pam250 files from the assignments page. These files were found (and slightly modified) in the plsa benchmark inputs, which are part of the BioBench suite of benchmarks (http://www.ece.umd.edu/biobench/). Run the compiled program as follows:

```
time ./a.out 30k_1.txt 30k_2.txt  pam250 -10
```

What is the output of time? What does the output of time mean? (Use a search engine to help you answer this question, but DO NOT copy text. Explain the command in your own words.)

(c) The only change to the input files is the addition of an integer so that we can more easily dynamically allocate memory for each sequence of input characters. The integer is currently 30000 to indicate that the smithWaterman.c driver should only read 30000 of the characters in the protein string. You need to make at least five other versions of the input files with each of the other versions using fewer than 30000 characters of the same input files. For example, you could make a 20k_1.txt that is a copy of 30k_1.txt but with the integer 20000 on the second line of the file. Use the time command while running with different problem sizes and report the execution time results in a latex table in your homework (see the example latex table in the provided latex template).

(d) Graph performance of different problem sizes using gnuplot. See the latex template file for a brief tutorial on gnuplot that provides you all of the commands necessary for this task. Feel free to improve the look of the resulting graph by learning more about gnuplot on your own.

(e) Graph the performance of different problem sizes when you switch lines 207 and 208 (plot another line for the permuted loop version on the graph showing execution time versus problem size). Do you get the same answer? Why? Explain the performance difference (hint: how is the allocated memory organized?)

(f) Put in calls to `gettimeofday()` to determine what loop takes the most amount of time. Present your results in a table and attempt to explain what is causing the slowest loop to be the performance bottleneck.

(g) Code can ALWAYS be improved in some way. Describe one way the given code can be improved. How would you evaluate your suggested improvement? (Many answers are acceptable here. The key is for you to THINK about possible improvements and clearly explain one as well as describing a reasonable evaluation.)

3. [20 Points] Mathematical Concepts

One of the main goals of this course is to automate parallelization and other loop transformations. Many approaches to automation involve using mathematical representations of computations. Therefore, it is time to review some important math concepts. Look terminology up on the internet if you do not recall the details.

(a) What is the recurrence equation that defines how the h[][] array is computed in the provided Smith-Waterman code?

(b) Inspect the provided Smith-Waterman code and determine the computation's algorithmic complexity. In other words, what is the Big-O?

(c) What loop(s) dominate the execution time based on the Big-O? Does that match the `gettimeofday()` time measurements? If not, hypothesize as to why that might be the case. How would you test that hypothesis?

(d) Given the matrix $A = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}$ and the vector $v = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, what is A^2 (use matrix multiplication) and what is Av (matrix vector multiplication)?

(e) What formula is equivalent to the following summation?

$$\sum_{i=0}^{X} i$$