CS/ECE 560 Homework 0 Key

Michelle Mills Strout and Manaf Gharaibeh

February 8, 2012

1 Definitions

- function: In general, a function can be thought of as a unique association between an input value from a *domain A* and an output value from a *codomain B*. That is, an input value a from A can be associated with only one output value b from B via a function f. However different input values can have the same f output value.
- domain: A set of input values for which a function f can operate on.
- range: A range is sometimes defined in terms of the function image and sometimes in terms of function codomain. In the first case it is the set of output values for a given domain or subset of that domain. And in the second case it is the set of all possible output values for a function f.
- One to One: One to One functions are those that associate every value in the codomain with *at most* one value from the domain. Note that not all codomain values are necessarily paired with values from domain.
- onto: Also known as surjection, onto is the case where every output value in the codomain is associated with at least one input value from the domain.
- bijective: A bijective function is a bijection if every element in the domain is paired with exactly one element in the codomain, and every element in the codomain is paired with exactly one element in the domain.
- set theoretic notation: Set Theory is a branch of mathematics that describes and studies collections of typically related objects and the operation that can be applied to them. The set theoretic notation is a symbolic notation used to represent set theory concepts.
- conjunction: An operation to relate a number of logical values where the result is considered true if all of the logical values evaluate to true (aka AND).

Problem size	User CPU time	System CPU time	Elapsed time
5k	1.011	0.177	1.19
10k	8.547	0.705	9.35
15K	46.308	1.668	48.31
20k	108.128	2.843	111.82
25k	188.048	4.458	193.60
30k	321.693	6.361	330.33

Table 1: Execution time for different problem sizes

- disjunction: An operation to relate a number of logical values where the result is considered true if one or more of the logical values evaluate to true (aka OR).
- big O notation: In Computer Science, Big-O notation is a way to describe the complexity or the growth rate of an algorithm or a function in terms of time or space required corresponding to input size. A more formal definition can be as follows: $T(n) = O(f(n))$ where $T(n)$ represents the processing time in terms of a function of n, the number of items to be processed. This definition indicates two constants, n_0 and c, where both are greater than zero, such that for all $n > n_0$, $cf(n) \geq T(n)$. That is for larger values of n, there is a constant c such that the processing time is less than $cf(n)$. So you can think of $cf(n)$ as the worst case.

2 Basic Performance Analysis

(a) No answer is required for this part

(b) You should get something close to: 321.693u 6.361s 5:30.33 The main outputs of the *time* command are: (1) the user CPU time, (2) the system CPU time, (3) elapsed time. In the above example, 321.693 seconds of CPU time were spent on user computations. 6.361 seconds were spent on system operations like forking and swapping. The last number indicates that the elapsed wall clock time for the program to finish running was 5 minutes and 30.33 seconds.

(c) Table 1 shows the time measurements of running the code on one of the vege machines for different problem sizes.

(d) Figure 1 shows the output of the time command for different problem sizes.

(e) Figure 2 shows the output of the time command after switching lines 207 and 208 for different problem sizes.

We got faster executions times after switching the two *for* loop headers at lines 207 and 208 of the original code. This improvement is due to the the better memory management

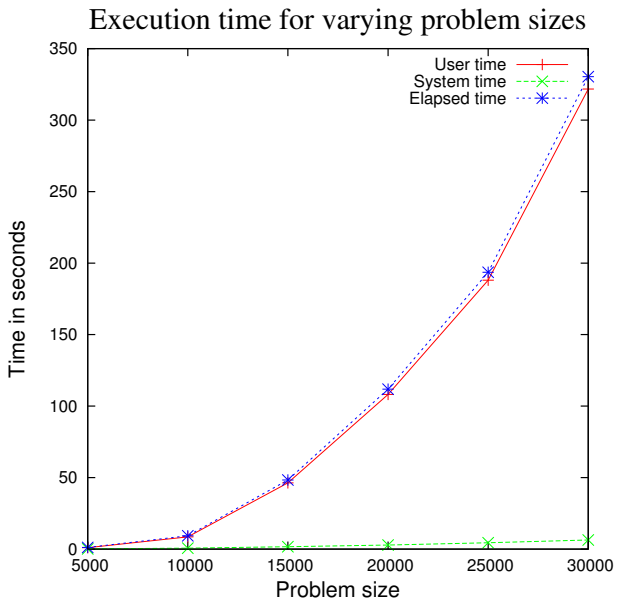

Figure 1: Execution time for original code with different problem sizes.

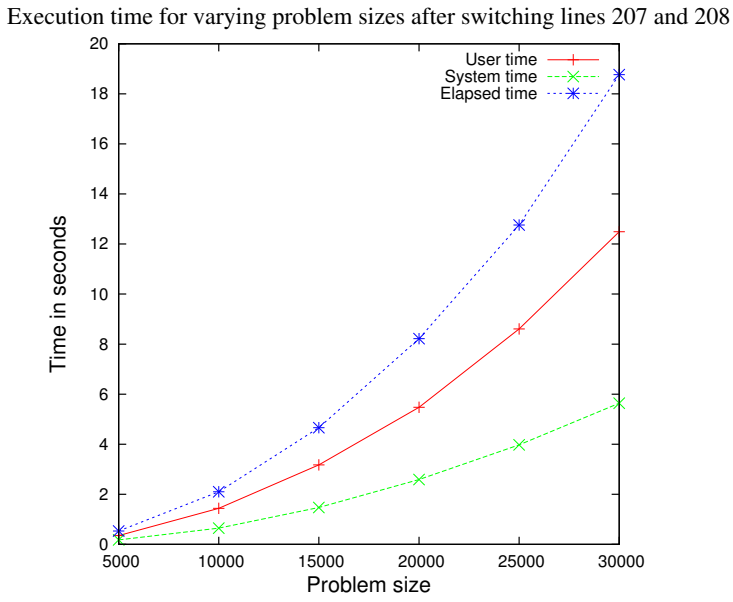

Figure 2: Execution time after switching lines 207 and 208 for different problem sizes.

Loop line number	Elapsed time (seconds)
110	0.000076
121	0.000000
130	0.000040
140	0.001043
158	0.000034
169	0.001024
190	2.886618
201	0.103727
207	14.264781
269	0.014887

Table 2: Elapsed time for all the loops in the code (with lines 207 and 208 switched)

applied by aligning the way the program is accessing memory with how the allocated memory for storing the h is organized, which reduced the cache misses.

(f) Table 2 shows the results for this part.

The loop nesting starting at line 207 is the performance bottleneck. It involves intensive computation and memory accessing for reading and writing.

(g) One Way to improve performance of the Smith-Waterman implementation here is parallelization. Particularly parallelizing the code within the performance bottleneck nested loops. One limiting factor for parallelization here is the dependencies for calculating the value of the element at $h_{i,j}$. Where the value of such an element depends on the elements to the west, north, and north-west. However we still can parallelize the calculations of elements on the anti-diagonals since they are independent of each other. We can evaluate this approach by implementing it and calculate any speedups of the parallelized code over the sequential version.

3 Mathematical Concepts

(a) The answer can be found from the nested loops starting at line 207:

$$H_{i,j} = max \begin{cases} 0 \\ H_{i-1,j-1} + similarity_{i,j} \\ H_{i-1,j} + delta \\ H_{i,j-1} + delta \end{cases}$$

(b) $O(MN) = MN$, where M and N are the lengths of protein strings to be compared.

(c) The loop nesting starting at lines 207 and 208. This matches the results of the *gettimeofday* time measurements.

(d)

$$A^2 = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

$$Av = \begin{bmatrix} 8 \\ 6 \end{bmatrix}$$

(e)

$$\sum_{i=0}^{X} i = \frac{X(X+1)}{2}$$