Design and Analysis of Algorithms 2009
(Home work 3)

October 7, 2009

- All problems except problems 7 and 8 are due on Monday, October 19, before 10 a.m.
- Late home works will not be accepted.
- Please give precise arguments for all statements that you write.
- Please do not hesitate to contact me if you do not understand the problems.
- Collaboration is discouraged, but not prohibited. It is recommended that you try to solve the problems on your own. You can discuss the questions with your colleagues but you should not copy solutions. Always write down your own answers. If copying is detected that may immediately lead to a grade less than 7. (**This would be followed strictly**)
- Credits would be given to partial solutions also.
- When you write an algorithm, you should briefly discuss the main idea of your algorithm, then write a pseudo code, argue about its correctness and state and prove the running time of your algorithm.
- The answers (except for problems 7 and 8) should be typed or written clearly and a hard copy is to be submitted.

1. Give asymptotic tight bounds for $T(n)$ for the following cases. Assume that $T(n)$ is a constant if $n \leq 2$. [10 points]

   (a) $T(n) = 2T(n/2) + n^3$.

   (b) $T(n) = 7T(n/3) + n^2$.

   (c) $T(n) = 3T(n/2) + n \lg n$.

   (d) $T(n) = 2T(n-1) + 1$.

2. How many lines, as a function of n (in $\Theta(.)$ form), does the following program print? Write a recurrence and solve it. You may assume n is a power of 2. [10 points]

```
function f(n)
  if n>1,
     print("still going\n")
     f(n/2)
     f(n/2)
   end if
```

3. Write a non recursive version of the procedure *Heapify*. [10 points]

4. Give a divide and conquer algorithm to multiply two polynomials of degree $n-1$ which run in time $\Theta(n^{\lg 3})$. [10 points]

5. A Toeplitz matrix is a $n \times n$ matrix $A = [a_{ij}]$ such that $a_{ij} = a_{i-1,j-1}$ for $i = 2, 3, \ldots, n$ and $j = 2, 3, \ldots n$. Give suitable data-structure to represent a Toeplitz matrix. Show that using this representation one can add two Toeplitz matrices in $O(n)$ time. [10 points]

6. Using the same representation that you used for the previous problem, show that you can multiply a $n \times n$ Toeplitz matrix with a $n \times 1$ vector in $O(n \log(n))$ time. [10 points]

Implementations

The following problems are due on October 30 before mid night. For Problem 7 you need to send all codes with the relevant instructions for compiling and running your codes. For problem 8, you should send a pdf document containing the plots and the related explanations. Put all files in a zip or tar archive and mail it to debrup.otro@gmail.com. Put DAA09HW3 as subject of your email. [50 + 25 points]

7. In each of the following the codes should take input from a file and write the output in another file. You should write your codes in C preferably in linux).

   (a) Implement *Heapsort* using recursive version of Heapify. Implement the the procedures parent, left-child and right-child as functions.

   (b) Implement *Heapsort* using the non-recursive version of Heapify as developed in Problem 3. Implement the the procedures parent, left-child and right-child using macros.

   (c) Implement *Quicksort* with a randomized pivot.

   (d) Implement *Quicksort* using the *qsort* function provided in *stdlib.h*.

8. Run the four programs of problem 7 on n randomly generated points where $n = 100, 200, 300, \ldots, 10000$. Measure the running time of all the programs and show their variations with n in a single plot.

   To measure time you may use following code. The variable timeRequired in the following code will give the number of CPU cycles utilized by your sorting procedure. This sample code is also available in the web-page. This code should run in Pentium IV and higher Intel processors.

```
 typedef unsigned long long uint64;
 uint64 read_clock_tick()
{ unsigned int h,l;

 __asm__ ( "xor %%eax,%%eax \t \n"
           "cpuid \t \n"
          " rdtsc \t \n"
          "mov %%edx, %0 \t \n"
          "mov %%eax, %1 \t \n"
          :"=m"(h),
          "=m"(l)
          :);
   return((uint64)h<<32) + l;
}

main()
{ uint64 start,end;
  start = read_clock_tick();

  /* Your sorting procedure should be here,
   you should not include any input/output
   operations in this area*/

    end =read_clock_tick();

  timeRequired = end-start;

  printf("The time required for sorting
            = %ld CPU cycles", timeRequired);
}
```