CS4040/CS5040 **Homework # 2**

due September 23, 2021

(CS4040 53 pts/ CS5040 58 pts)

(5 pts.) (a) Determine a good asymptotic upper bound on the recurrence $T(n) = 5T(\lfloor n/3 \rfloor) + n$ by the iteration method.

(5 pts.) (b) Draw the recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T\left(\lfloor n/2 \rfloor\right) + 9n$. Verify your bound by the substitution method.

(5 pts.) (c) Draw the recursion tree for the recurrence $T(n) = 3T\left(\lfloor n/\sqrt{3} \rfloor\right) + cn$, where c is a constant, and determine a good asymptotic upper bound on its solution. Verify your bound by the substitution method.

(5 pts.) (d) Using the substitution method, show that the solution to $T(n) = 2T(\lfloor n/2 \rfloor + 13) + 2n$ is $O(n \log n)$.

(6 pts.) (e) Use the master method to give tight asymptotic bounds for the following recurrence relations:

2 pts. (1) $T(n) = 32T(n/6) + 5n$.

2 pts. (2) $T(n) = 36T(n/6) + 5n^2 + 4n$.

2 pts. (3) $T(n) = 42T(n/6) + 25n^3 + 5n^2 + n$.

(5 pts.) (f) The recurrence $T(n) = 8T(n/2) + 2n^2$ describes the running time of algorithm 1. Algorithm 2, has a running time of $T(n) = aT(n/4) + n^2$, where a is a constant. What is the largest integer value for a such that algorithm 2 is asymptotically faster than algorithm 1.

(10 pts.) (g) Give tight bounds for the solution to the following recurrence relations. Use any rigorous method you wish. Justify your answers. Show all work. Explicitly state any assumptions you make.

1. $T(n) = 42T(n/42) + n^3$.
2. $T(n) = 36T(n/6) + n^2$.
3. $T(n) = 7T(n/2) + n^{2.333}$.
4. $T(n) = 2T(n/16) + \sqrt[4]{n}$.
5. $T(n) = T(n-2) + 3n + 1$.

(12 pts.) (h)

1. Consider the following recursive binary search algorithm. It assumes the input array A is sorted, and finds whether item x is in the array. It returns 0 if it is not found, else the index of the item.

```
bsearch(A,p,r,x) {
   if p <= r {
      q=p+r/2
      if (x == A[q]) {
         return q;
      } else if (x < A[q]) {
         return bsearch(A,p,q-1,x);
      } else {
         return bsearch(A,q+1,r,x);
      }
   } else {
      return 0;
   }
}
```

   Typically, we do not count the cost of parameter passing. In this problem we will look at how the cost of passing parameters to an algorithm might affect the runtime. There are three possible choices for how to pass parameters:

   i Pass by pointer, in which case it takes a constant amount of time or $\Theta(1)$.

   ii Pass by copying. In this case it is linear in the size of the array. If an N-element array is passed, it takes time $\Theta(N)$.

   iii Pass by copying only the subrange that might be used by the called function. In this case the time is proportional to the length of the array passed. In the above algorithm it passes the portion of the array from p to q, so would take time $\Theta(q - p + 1)$ if the sub-array A from p to q is passed.

   Give a recurrence relation for the worst-case running time of bsearch for each of the three methods outlined above. Solve each of these recurrence relations giving good bounds on the solutions of the recurrences. Hint: use N as the size of the original call to the function, and n as the size of the subproblem.

2. Now consider the merge-sort algorithm as described in the book chapter 2. Once again, give a recurrence relation for the worst-case running time of merge-sort for each of the three methods outlined above. Solve each of these recurrence relations giving good bounds on the solutions of the recurrences. Hint: use N as the size of the original call to the function, and n as the size of the subproblem.

CS5040 students must also do the following problem

(5 pts.) (i) Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(n - a) + T(a) + cn$ where $a \geq 1$ and $c > 0$ are constants.