TCSS543A – Autumn 2020
Homework 1 (30 points)

1. (**2+2+2+2+2 points**) Find and prove (e.g. using the Master Theorem or other method) tight bounds for the following recurrences (where h is a positive constant). To get full grade, you must correctly justify your answer (i.e. lucky guesses will not help):

   (a) $T(n) = \begin{cases} h & n < 32 \\ 16\,T\left(\frac{n}{32}\right) + n/\sqrt[5]{n} + 1/\sqrt{n} & n \geq 32 \end{cases}$

   (b) $T(n) = \begin{cases} h & n < 16 \\ 16\,T\left(\frac{n}{16}\right) + n^{\lg n} & n \geq 16 \end{cases}$

   (c) $T(n) = \begin{cases} h & n < 22 \\ 38\,T\left(\frac{n}{22}\right) + n\sqrt[6]{n} & n \geq 22 \end{cases}$

   (d) $T(n) = \begin{cases} h & n \leq 1 \\ \frac{6}{5}\,T\left(\frac{6n}{7}\right) + n\sqrt[4]{n} & n > 1 \end{cases}$

   (e) $T(n) = \begin{cases} h & n \leq 1 \\ \frac{3}{5}\,T\left(\frac{3n}{5}\right) + \frac{h}{n} & n > 1 \end{cases}$

2. (**10 points**) Kleinberg and Tardos: Chapter 2, Exercise 6.

   Consider the following basic problem. You're given an array A consisting of n integers $A[1]$, $A[2]$, …, $A[n]$. You'd like to output a two-dimensional n-by-n array B in which $B[i,j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$—that is, the sum $A[i] + A[i+1] + \cdots + A[j]$. (The value of array entry $B[i,j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what is output for these values.)
   Here's a simple algorithm to solve this problem.

```
for $i$ = 1, 2, …, $n$
    for $j$ = $i$ + 1, $i$ + 2, …, $n$
        add up array entries $A[i]$ through $A[j]$
        store the result in $B[i,j]$
    end for
end for
```

(a) For some function f that you should choose, give a bound of the form $O\big(f(n)\big)$ on the running time of this algorithm on an input of size *n* (i.e., a bound on the number of operations performed by the algorithm).
(b) For this same function f, show that the running time of the algorithm on an input of size *n* is also $\Omega\big(f(n)\big)$. (This shows an asymptotically tight bound of $\Theta\big(f(n)\big)$ on the running time.)
(c) Although the algorithm you analyzed in parts (a) and (b) is the most natural way to solve the problem—after all, it just iterates through the relevant entries of the array B, filling in a value for each—it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.

NB: to get full points here, you must *justify* (i.e. *prove*) your calculations and statements with the limit test and/or by explicitly computing the summations whenever necessary.

3. (**10 points**) Express the asymptotic running time (Big-Theta) of the following algorithm as a function of the input length `n`. Explain and justify your analysis:

```
void kmx(long[] a) {
    int n = a.length; // express the running time as a function of this n
    if (n <= 1) {
        return;
    }
    int t = 1;
    while (t < n - t) {
        t += t;
    }
    for (int p = t; p > 0; p >>= 1) {
        for (int j = 0; j < n - p; j++) {
            if ((j & p) == 0) {
                if (a[j + p] < a[j]) {
                    long w = a[j]; a[j] = a[j + p]; a[j + p] = w;
                }
            }
        }
        for (int q = t; q > p; q >>= 1) {
            for (int j = 0; j < n - q; j++) {
                if ((j & p) == 0) {
                    if (a[j + q] < a[j + p]) {
                        long w = a[j + p]; a[j + p] = a[j + q]; a[j + q] = w;
                    }
                }
            }
        }
    }
}
```

4. (**bonus: 2+2+1 points**) Keith and Randy, who work at the UW Tacoma Copy Center, have a series of n tasks of copying and then binding documents. They only have one copy machine and one binding machine, and at most one document can be processed by each of these machines at any time, but they can process distinct tasks simultaneously, as long as every document only enters the binding machine after it has been copied.

   Based on the number of pages of each document, Keith and Randy know that the times needed to copy and bind them are, respectively, $A_1, \dots, A_n$ and $B_1, \dots, B_n$. Our friends need to know the shortest time they can get all tasks done and which order they should be performed, and they ask you to help them.

   To that end, prove the following property:

(a) The sequence of documents processed by either machine can be made the same as that of the other machine without loss of time.

   By property (a), we can assume Keith and Randy will feed the documents to the binding machine in the same sequence they feed them to the copier. Let S denote that sequence and let X_i denote the inactive period of time for the binding machine immediately before Keith and Randy feed it the i-th document to start the binding. For instance, $X_1 = A_{S[1]}$, $X_2 = \max\left(A_{S[1]} + A_{S[2]} - B_{S[1]} - X_{S[1]}, 0\right)$, and in general $X_i = \max\left(\sum_{j=1}^{i} A_{S[j]} - \sum_{j=1}^{i-1} B_{S[j]} - \sum_{j=1}^{i-1} X_j\,, 0\right)$, from where we get the total idle time resulting from sequence S, denoted $F(S)$, to be $F(S) \coloneqq \sum_{i=1}^{n} X_i = \max\limits_{1 \le u \le n} K_u$ where $K_u \coloneqq \sum_{j=1}^{u} A_{S[j]} - \sum_{j=1}^{u-1} B_{S[j]}$. Our task is thus to find a sequence S that minimizes $F(S)$, that is, $F(S) \le F(T)$ for any other sequence T.

   Given any sequence S and an index j, let S' denote the sequence obtained from S by interchanging $S[j]$ and $S[j+1]$, that is, $S'[j] = S[j+1]$, $S'[j+1] = S[j]$, and $S'[i] = S[i]$ for all $i \notin \{j, j+1\}$. Also, let $K'_u \coloneqq \sum_{j=1}^{u} A_{S'[j]} - \sum_{j=1}^{u-1} B_{S'[j]}$.

(b) Prove that an optimal sequence is done by the following rule: document $S[j]$ precedes $S[j+1]$ iff $\max\left(K_j, K_{j+1}\right) < \max\left(K'_j, K'_{j+1}\right)$.

(c) Design an algorithm that will find the shortest time that Keith and Randy can complete their copying and binding jobs, and also suggests an ordering of tasks that attains that shortest time.