Assignment 3. **Heaps and math.**

Given February 5, due February 12.

1. (from CLR, page 151) Give an $O(n \log(k))$ algorithm to merge $k$ sorted lists into a single sorted list of length $n$. *Hint:* Make a heap of the largest element of each of the $k$ sublists not yet merged into the final list.

2. The most famous recurrence relation in theoretical computer science is

$$T(n) = n + \frac{2}{n-1}(T(1) + \cdots + T(n-1)),$$

with the starting condition $T(1) = 1$ (or $T(1) = 0$, it does not matter).

a. Motivated by the right side of (1), define $A(n)$ to be the average of the first $n$ numbers $T(k)$:

$$A(n) = \frac{1}{n}(T(1) + \cdots + T(n)).$$

Find an expression for $T(n)$ in terms of $A(n)$ and $A(n-1)$. To do this, note that $nA(n)$ and $(n-1)A(n-1)$ have almost exactly the same terms.

b. Plug this into (1) to get a simple recurrence relation for $A(n)$.

c. Unroll this relation to get an explicit formula for $A(n)$ as a sum.

d. Use some method – integral approximation or upper and lower bounds – to find the order of magnitude of this sum.

e. Use (1) to get the order of magnitude of $T(n)$.

3. Use the method of upper and lower bounds to find the order of magnitude of

$$W(n) = \sum_{k=1}^{n} \log^2(k).$$

Do not use an integral approximation.