

1. Approximate $\sum_{1 \leq i \leq n} i^{-2}$.

For the next two questions start with n sorted files, each of the same length. Assume that you can merge two of the initial files in one unit of time. In general, the time to merge two files is proportional to the length of the longer file. In both of the following two questions, the goal of the algorithm is to merge all the files together to obtain one sorted file containing all of the information from the input files, but the two algorithms accomplish this in different ways.

2. The algorithm starts by merging the first two files. On the i -th step ($i \geq 2$), merge the result from the previous step with the $i + 1$ -st input file. How many units of time are used.
3. The algorithm proceeds by rounds. On the first round it merges pairs of input tapes. If the number of tapes is odd, one tape goes to the next round without being merged (at no cost in running time). Thus, the first round needs time $\lfloor n/2 \rfloor$. On the i -th round ($i \geq 2$) the algorithm merges the results from the previous round in pairs. (Again, if the number of tapes is odd, one tape advances at no cost). The algorithm continues until the round produces just one output tape. How many units of time are used? (To get a simple closed form answer in reasonable time, you may need to make some additional assumptions about n).
4. Solve $a_n = 4a_{n-1}$ with $a_1 = 1$.

5. Consider adding fractions using

$$\frac{ad + bc}{kbd} = \frac{a}{kb} + \frac{c}{kd}.$$

Thus, k is a known common factor between the two denominators. This method pays some attention to the desire to have an answer in lowest terms in that it pulls out the known factor k , but it does not go all the way (which would require the expensive check of seeing whether there were any common factors between $ad + bc$ and kbd). We want to apply this technique to the calculation

$$\frac{a_n}{k_n b_n} = \frac{a_{n-1}}{k_{n-1} b_{n-1}} + \frac{1}{(2n+1)2^{2n+1}}$$

with the boundary condition $a_1 = 1$, $b_1 = 2$, and $k_n = 2^{2n-1}$ (This question begins an analysis important to the performance of algorithms used in the term projects.)

5a. Write the recurrence for b_n .

5b. Solve the recurrence from 5a.

5c. How many binary digits are needed to write b_n ? Give both an exact answer and also an approximate answer (if the approximate answer is a lot simpler than the exact answer).

5d. How many binary digits are needed to write a_n ? (You can get a good approximation without doing calculations similar to 5a and 5b for a_n because this series approximates $\ln 3$ and $\ln 3$ is a small constant.)