Starred questions are suitable for students aiming for a credit or higher.

1. Apply BUBBLESORT to the input sequence \(2, 6, 4, 8, 1, 7, 5, 3\). How many passes of the outer loop ("percolations") are needed to sort the sequence. List each sequence as it stands after each of these passes. How many swaps (in the inner loop) took place overall?

2. Suppose we have real numbers \(a_1 < a_2 < \ldots < a_{N-1} < a_N\).

   (a) Describe all input sequences of these numbers which need exactly one swap to become sorted by BUBBLESORT.

   *(b)* Which input sequence of these numbers causes the maximum possible number of swaps in BUBBLESORT, and what is this maximum number? Now describe all input sequences which use exactly one less than the maximum possible number of swaps.

   *(c)* Now describe all sequences which use exactly three swaps but more than one pass of the outer loop to become sorted.

3. Apply MERGESORT to the sequence \(\cos 1, \cos 2, \cos 3, \cos 4, \cos 5, \cos 6, \cos 7, \cos 8\) where angles are in radians. What happens if angles are in degrees?

4. Consider the following reformulation of MERGESORT for sorting an input sequence \(a_1, \ldots, a_N\) where \(N = 2^k\) is a power of 2. The algorithm makes \(k\) passes then stops.

   \[\text{Pass 0: Write down } a_1, \ldots, a_N.\]
   \[\text{Pass 1: Rewrite whilst sorting each successive pair of numbers.}\]
   \[\text{Pass 2: Rewrite whilst merging each successive pair of pairs.}\]
   \[\vdots\]
   \[\text{Pass } i: \text{ Rewrite whilst merging each successive pair of lists of } 2^{i-1} \text{ elements.}\]
   \[\vdots\]
   \[\text{Pass } k: \text{ Rewrite whilst merging the final pair of lists of } 2^{k-1} \text{ elements.}\]

   (a) Apply this algorithm to the input sequence of the first question.

   *(b)* In general, total up the number of times numbers are written down and an upper bound of comparisons made in the duration of this algorithm. Deduce that MERGESORT has \(O(N \log N)\) running time.

   *(c)* How would you modify the algorithm and the calculation in (b) to deal with input sequences of any positive length?
5. (a) Verify that \(\lceil X + M \rceil = \lceil X \rceil + M\) for any real number \(X\) and integer \(M\).

*(b) Verify that if \(X\) is any positive real number and \(M\) any positive integer then
\[
\left\lfloor \frac{X}{M} \right\rfloor = \left\lfloor \frac{\lceil X \rceil}{M} \right\rfloor .
\]
Give an example to show this equation can fail if \(M\) is allowed to be real.

*(c) Fix a positive real \(X\) and a positive integer \(M\). Define \(a_0 = \lceil X \rceil\) and \(a_{n+1} = \lceil a_n / M \rceil\) for \(n \geq 0\). Prove that
\[
a_n = \left\lfloor \frac{X}{M^n} \right\rfloor
\]
for each \(n\). What is \(\lim_{n \to \infty} a_n\)?

6. Suppose we have a strictly increasing sequence \(S\) of \(N = 2^k\) real numbers: \(a_1 < a_2 < \ldots < a_{N-1} < a_N\). Put
\[
S_L = \{a_1, \ldots, a_{N/2}\} \quad \text{and} \quad S_R = \{a_{N/2+1}, \ldots, a_N\},
\]
and write
\[
M_L = M_L(S) = a_{N/2}, \quad M_R = M_R(S) = a_{N/2+1} .
\]
The following recursive algorithm takes a real number as an input and locates it in a half-closed interval associated with \(S\):

BINARYSEARCH\((S)\): Input a real number \(\lambda\).

(1) If \(S = \{X\}\) output \((-\infty, X)\) if \(\lambda < X\); output \([X, \infty)\) otherwise, and stop.

(2) If \(M_L \leq \lambda < M_R\) output \([M_L, M_R)\) and stop.

(3) If \(\lambda < M_L\) then input \(\lambda\) to BINARYSEARCH\((S_L)\).

(4) Otherwise input \(\lambda\) to BINARYSEARCH\((S_R)\).

(a) Describe what happens when BINARYSEARCH \((1, 2, \ldots, 64)\) is applied to the inputs \(\lambda = 32.4, \lambda = 12.97, \lambda = 35.1\) and \(\lambda = -1\).

(b) Explain why the running time \(T(N)\) of the algorithm, as a function of the number \(N(=2^k)\) of elements in the original ordered list, is described by
\[
T(N) = T(N/2) + O(1) .
\]

*(c) Solve the recurrence of (b) to show that \(T(N) = O(\log N)\).

*(d) How would you modify the specification of BINARYSEARCH\((S)\) where the size of \(S\) is any positive integer.

**(e) Relate the following recurrence to the running time of your modified algorithm in part (d):
\[
T(N) = T([N/2]) + O(1) .
\]
Solve this and show that you get the same conclusion as in part (c).