8 Algorithms 1 (fms27)

Given an array \( a \) containing \( n \) items to be sorted, a bottom-up implementation of mergesort performs, non-recursively, several passes on \( a \).

(a) Derive \( p(n) \), the number of passes performed. [1 mark]

(b) Derive \( m(n, i) \), the number of merge operations performed in pass \( i \), where passes are numbered starting from 0 and ending at \( p(n) - 1 \). [2 marks]

(c) A programmer has (correctly) read that an array \( a \) of \( n \) elements can be sorted with bottom-up mergesort using scratch workspace of size \( \lceil n/2 \rceil \) elements. The programmer decides to implement this by requiring the caller to arrange that \( a \) starts with \( n \) cells containing the values to be sorted, followed by \( \lceil n/2 \rceil \) cells to be used as workspace, and produces the following pseudocode:

```python
def bums(a, n):
    """Bottom-up-merge-sort a[:n], using a[n:] as scratch space."""
    assert len(a[n:]) >= n/2  # NB: here n/2 is not integer for odd n
    s = 1  # Size of the chunks to be merged in this pass
    for pass between 0 included and p(n) excluded:
        for pair between 0 included and m(n, pass) excluded:
            copy a[s*pair:s*(pair+1)] to a[n:n+s]
            srcA = n
            maxA = n + s
            srcB = s * (pair+1)
            maxB = max(s * (pair+2), n)
            dst = s * pair
            while (srcA < maxA) or (srcB < maxB):
                if a[srcA] < a[srcB]:
                    a[dst++] = a[srcA++]
                else:
                    a[dst++] = a[srcB++]
            s = 2 * s
```

This pseudocode contains three serious bugs. For each of them:

(i) Explain the bug clearly, focusing on the difference between programmer’s intention and the code as written; then suggest how to fix it (no pseudocode is required). [3 marks each]

(ii) In the spirit of unit testing, exhibit a simple input pair \( (a, n) \) that triggers that bug but neither of the others, contrasting intended and actual behaviour. [2 marks each]

(d) Assuming the bugs in Part (c) are corrected, is this bottom-up mergesort implementation stable? Give reasons. [2 marks]