Assiut University	Faculty of Computers
Third Year	Algorithms Analysis & Design
Midterm Exam, Dec 2005	Time:1 ½

1- The following is a recursive sorting algorithms called A-SORT(A,*p*,*r*). GETMIN(A,*p*) gets the p^{th} smallest element from the array A starting form the p^{th} element to the r^{th} element.

A-SORT (A , *p* , *r*)
1- **if** $p < r$
2- GETMIN(A , *p*)
3- $p = p + 1$
4- A-SORT (A , *p* , *r*)

The 1st call of A-SORT is A-SORT(A, 1, *length* (A))

1- Suggest algorithm steps of GETMIN(A,*p*) and analyze its running time (Lower bound constants cost is required). (9pt)
2- Write a recurrence equation for A-SORT(A,*p*,*r*) algorithms, give a definition for each part of the recurrence equation and solve it. (9pt)
3- Can the master method be used to solve the recurrence of A-SORT(A,*p*,*r*)? If not? Explain why? (3pt)
4- If your suggested algorithm for GETMIN(A,*p*) has a running time higher than $O(\lg n)$? Suggest an idea how to perform GETMIN(A,*p*) in $O(\lg n)$ running time. (assume any needed initial conditions) (9pt)

2- It is required to find the *m* smallest elements of an array of *n* integers, in sorted order. For example, given the array [3,5,6,1,8,4,7,2] and $m = 4$, the required is [1,2,3,4].

1- Algorithm A sorts the numbers using merge sort and output the 1st *m* elements in sorted order. Analyze the worst-case running time of algorithm A in terms of *n* and *m*. (3pt)
2- Algorithm B builds a min-heap from the numbers and call EXTRACT_MIN *m* times. Analyze the worst-case running time of algorithm in terms of *n* and *m*. (3pt)
3- Can the GETMIN that used in A-SORT in the 1st question be used to get the 1st *m* smallest elements? If yes? Write an algorithm C that output the smallest *m* elements using GETMIN and analyze the worst-case running time terms of *n* and *m*. (4pt)

3- Prove or disprove the following asymptotic relations:

1- $\frac{n^2}{2} - 3n = \Theta(n^2)$, (5pt)

2- $(\sqrt{2})^{\lg n} = \Theta(n)$, (5pt)

3- $f(n) + g(n) = \Theta(\max(f(n), g(n)))$. (5pt)

4- Solve the following recurrence equations:

1- $T(n) = 4T(n/2) + n^3$ (5pt)

2- $T(n) = 3T(n^{\frac{1}{3}}) + \log_3 n$ (Hint: use changing variables method) (5pt)

3- $T(n) = T(n/2) + T(n/3) + n$ (5pt)

5- Assume that there is a sorting algorithm called B-Sort that runs in $O(n\sqrt{n})$ worst-case running time. The constant factors in B-Sort make it faster than Merge-Sort for small n. Thus, it makes sense to use B-Sort within Merge-Sort when subproblems become sufficiently small. Consider a modification to Merge-Sort in which n/k sublists of length k are sorted using B-Sort and then merged using the standard Merge-Sort mechanism, where k is a value to be determined.

1- Determine the worst-case running time to sort n/k sublists of length k using B-Sort? (6pt)
2- What is the worst-case running time to merge the sublists? (6pt)
3- What is the largest asymptotic (Θ-notation) value of k as a function of n for which the modified algorithms has the same asymptotic running time as standard Marge-Sort? (9pt)
4- How k be chosen in practice? (i.e. for a specific implementation, where the constants associated with the running time can be calculated) (9pt)

With my best wishes
Dr. Tarek Hagras