Mayank Goswami and Andreas Wiese **Winter 2015/2016**

Exercises for Approximation Algorithms

www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/winter15/approx

Tutorials: Andreas Schmid

Exercise Sheet 4 Due: **6.1.2016**

Your homework must be handed in on Wednesday at the beginning of the tutorial.

You need to collect at least 50% of all points over all exercise sheets. You are allowed to work on these exercises in groups but every student has to hand in his/her own write-up.

Exercise 1 (*6 points*)

In class we saw an $\Omega(n \log n)$ lower bound on the closest pair of points problem in the linear decision tree model, and a (expected) linear time algorithm that used randomness, computing the floor function and randomness. This exercise is about an $O(n \log n)$ algorithm that does not use any of these techniques. The algorithm proceeds in a divide and conquer approach:

1. Sort all the points P using x-coordinate.
2. Split them into left and right by finding a middle line.
3. Solve the problem recursively in the left and right subsets. Let the distance of closest pair on the left be d_ℓ and on the right be d_r.
4. Find the minimum distance $d_{\ell,r}$ among the set of pairs of points where on point is to the left of the dividing line and one is to the right.
5. Output the minimum of d_ℓ, $d_{\ell,r}$ and d_r.

Giving details for all steps, prove that this algorithm runs in $O(n \log n)$ time. In particular, prove that Step 4 can be done in $O(n)$ time. [Hint: Use the "grid" property– instead of comparing all left-right distances in Step 4, note that the closest pair cannot be farther than $\min(d_\ell, d_r)$].

Exercise 2 (*14 points*)

In class we saw how to prove lower bounds in the linear decision tree model. This exercise is on two other techniques, leaf counting and adversary arguments.

a) [Counting leaves, 7 points] Prove that for any deterministic comparison-based sorting algorithm A, the average-case number of comparisons (the number of comparisons on average on a randomly chosen permutation of n distinct elements) is at least $C = \lfloor \log_2 n! \rfloor$. In other words, prove that in a decision tree, not just that there is some leaf with depth C, but in fact that the average-depth of a leaf is C.

b) [Adversary Argument, 7 points] Given an array of n elements (n is even), consider the problem of computing the minimum and the maximum (output both). Naively this can be done in $2n - 3$ comparisons (how?). Give an algorithm that does this in at most $3n/2 - 2$ comparisons. Prove that the algorithm is optimal using an adversary argument.

Exercise 3+4 (*20 points*)

In the lecture we have seen an approximation algorithm for Bin Packing by Karmakar and Karp that uses at most $OPT + O(log^2(OPT))$ many bins. As a reminder here is some of the notation used in the lecture. We group items of the same size (here we used b_1 as the number of items of largest size s_1 and b_m as the number of items of smallest size s_m). For an instance I of Bin Packing we define: $SIZE(I) := \sum_{i=1}^{m} b_i \cdot s_i$.

3 (10 points) Assume that you are given a Bin Packing instance I in which every item has size at least $\frac{1}{c}$ where c is some positive constant bigger than one. Show that in this case the algorithm by Karmakar and Karp will use at most $OPT + O(c \cdot log(SIZE(I)))$ many bins.

4 (10 points) Assume that you are given an instance I of Bin Packing. We say that an item i is big if $a_i \geq 1/SIZE(I)$. Assume you are given a packing of only the big items that uses K bins. Show that one can find a packing of all items that uses at most $\max\{K, OPT+3\}$ bins.