Problem set for the Algoritmica 2 class (2015/16)

Roberto Grossi
Dipartimento di Informatica, Università di Pisa
grossi@di.unipi.it

October 21, 2015

Abstract

This is the problem set assigned during class. What is relevant during the resolution of the problems is the reasoning path that leads to their solutions, thus offering the opportunity to learn from mistakes. This is why they are discussed by students in groups, one class per week, under the supervision of the teacher to guide the brainstorming process behind the solutions. The *wrong* way to use this problem set: accumulate the problems and start solving them alone, a couple of weeks before the exam. The correct way: solve them each week in groups, discussing them with classmates and teacher.

1. [Randomized selection] Consider the randomized quicksort, analyzed with the indicator variables, discussed in class (also, paragraph 7.3 in the textbook CLRS - Comern, Leiserson, Rivest, Stein, *Introduction to Algorithms*, 3rd edition, MIT Press). Show how to modify the randomized quicksort so that, given an array A and an integer $1 \leq k \leq |A|$, it finds the kth largest element in A without fully sorting A. Consider the analysis with indicator variables seen in class, and adapt it to show that the selection algorithm thus obtained requires linear expected time. [hint: since the algorithm has A and k as input, define an indicator variable X_{ijk} for z_i and z_j, where $i < j$, with the knowledge that z_k is looked for. The probability will have three cases, according to the relative order of z_k with respect to z_i and z_j.]

2. [Randomized min-cut algorithm] Consider the randomized min-cut algorithm discussed in class. We have seen that its probability of success is about $2/n^2$.

   - Describe how to implement the algorithm when the graph is represented by adjacency lists, and analyze its running time.
   - A weighted graph has a weight $w(e)$ on each edge e, which is a positive real number. The min-cut in this case is meant to be min-weighted cut, where the sum of the weights in the cut edges is minimum. Describe how to extend the algorithm to weighted graphs, and show that the probability of success is still $2/n^2$. [hint: define the weighted degree of a node]

- Show that running the algorithm for $N = cn^2 \ln n$ times, for a constant $c > 0$, and taking the minimum among the N min-cuts thus produced, the probability of success can be made at least $1 - 1/n^c$ (hence, with high probability). [hint: use the fact that $(1 - 1/x)^x \approx e^{-x}$ for small x.]
- Optional. When the graph becomes smaller, the probability of hitting a bad edge is higher. To reduce this chance, what if the algorithm is stopped when the resulting graph contains half of the original vertices? Run it four times independently, starting from the same graph G, and thus obtaining four graphs G_1, G_2, G_3, G_4, each with $n/2$ vertices. Apply recursively this idea to each G_i independently. Each time that two vertices are obtained, return the edges (as before). At the end, choose the best min-cut thus found among all those generated. Show what is the time complexity and the probability of error. [hint: it is a divide-and-conquer approach whose time cost is a recurrence relation; same for the probability]

3. [External memory mergesort] In the external-memory model (hereater EM model), show how to implement the k-way merge (where $(k + 1)B \leq M$), namely, how to simultaneously merge k sorted sequences of total length N, with an I/O cost of $O(N/B)$ where B is the block transfer size. Also, try to minimize and analyze the CPU time cost. Using the above k-way merge, show how to implement the EM mergesort and analyze its I/O complexity and CPU complexity.