15-451 Algorithms, Fall 2010

Bounds on Randomized Algorithms

Say we have 3 possible deterministic algorithms $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ and 3 possible inputs I, J and K for a problem. Let the cost matrix for these algorithms and inputs be as follows.

	I	J	K
$\mathcal{A}$	a_i	a_j	a_k
$\mathcal{B}$	b_i	b_j	b_k
$\mathcal{C}$	c_i	c_j	c_k

1 Deterministic Bounds on the Worst-Case Cost

It is important to note that the worst-case cost of any algorithm (for e.g. $\texttt{max}(a_i, a_j, a_k)$ for algorithm $\mathcal{A}$) is definitely an upper bound.

To find a *tighter* upper bound for the worst-case cost, we simply find the maximum of each row (the worst-case cost of each algorithm) and take the minimum of these maximum values (this cost is sufficient to solve the problem for any input). Note that this upper bound is also *optimal* as we have explored all possible algorithms and the worst-case cost of any other algorithm is at least this much. (In other words, we have given an algorithm whose worst-case cost is equal to the *optimal* upper bound.)

But, in general, enumerating all possible algorithms for a problem might not be feasible and hence, an upper bound (may not be the optimal) is usually given by providing *an* algorithm and its worst-case cost is taken as *an* upper bound.

Analogously, the least time taken for any input (minimum of any column in the cost matrix) is definitely a lower bound because *every* algorithm costs at least this much.

Then, it is important to see that the *optimal* upper bound on the worst-case cost we have seen above is also a lower bound. The reason is the same – we have explored all possible algorithms and every algorithm costs at least this much in the worst-case, implying a lower bound. Again, this is also the *optimal* lower bound. As this particular algorithm's worst-case is both the *optimal* upper and lower bound, this algorithm is called *optimal*.

We can find a lower bound in a different way (need not be *optimal*). We find the minimum of each column (every algorithm costs at least this much for that input) and take the maximum of these minimum values (every algorithm costs at least this much for the worst input).

Again, in general, enumerating all possible inputs for a problem might not be feasible and hence, a lower bound is usually found by giving an *adversary* algorithm which creates a *bad-enough* input for each algorithm and take the minimum of all the corresponding costs as a lower bound (need not be the optimal).

2 Bounds for Randomized Algorithms

Let us now think about randomized algorithms. A randomized algorithm can be seen as a discrete probability distribution on the 3 deterministic algorithms. And in general, inputs can come from another discrete probability distribution.

Let p_a, p_b and p_c denote the probabilities with which the 3 algorithms are chosen and let p_i, p_j and p_k denote the probabilities with which the 3 inputs are chosen. Note that $p_a+p_b+p_c = 1$ and $p_i + p_j + p_k = 1$.

2.1 Upper Bound

Let us see how to compute the *optimal* **upper bound** on the worst-case expected cost for all possible randomized algorithms (i.e. all possible values of p_a, p_b and p_c). It is similar to the deterministic case where we find the worst-case cost of all possible algorithms and take the minimum. Given a randomized algorithm, i.e. given specific values of p_a, p_b and p_c, the worst-case expected cost would be the maximum of the expected costs for all possible input distributions. Now, given an input distribution, i.e. given specific values of p_i, p_j and p_k, the expected cost can be written as

$$p_i(p_a a_i + p_b b_i + p_c c_i) + p_j(p_a a_j + p_b b_j + p_c c_j) + p_k(p_a a_k + p_b b_k + p_c c_k) \tag{1}$$

where the expressions in the brackets denote the expected cost of that particular input for the randomized algorithm. Note that the maximum of (1) is nothing but the maximum of the expected costs for the inputs I, J and K (in other words, we need to consider only 3 of the input distributions where only one of p_i, p_j and p_k is 1; this is sufficient because every other input distribution is only going to make the expected cost smaller if not the same). Thus, given a randomized algorithm, the worst-case expected cost is

$$\texttt{max}(p_a a_i + p_b b_i + p_c c_i, p_a a_j + p_b b_j + p_c c_j, p_a a_k + p_b b_k + p_c c_k)$$

It is important to note that, the value of the above expression is definitely *an* upper bound, but not necessarily the optimal. This is again useful in the general case where we cannot enumerate all possible randomized algorithms for a problem and we still need an upper bound – we can come up with *one* randomized algorithm and take its worst-case cost as an upper bound.

Now, the optimal upper bound (of the worst-case expected cost) over all possible randomized algorithms is just the minimum of the above, i.e.

$$\texttt{min}(\texttt{max}(p_a a_i + p_b b_i + p_c c_i, p_a a_j + p_b b_j + p_c c_j, p_a a_k + p_b b_k + p_c c_k))$$

In other words, our objective now is to find the randomized algorithm whose distribution matches with the above minimum. Once we find the values of p_a, p_b and p_c which satisfies the above minimax condition, we can compute the optimal upper bound.

2.2 Lower Bound

By the minimax theorem (which will be covered in the class in future) the *optimal* lower bound is equal to the *optimal* **upper bound** we found in the previous section. Nevertheless, we describe how to compute the optimal lower bound independently.

It is similar to the deterministic case where we find the lower bounds for all possible inputs and take the maximum. Given an input distribution, i.e. given specific values of p_i, p_j and p_k, the lower bound for all randomized algorithms is the minimum of the expected costs of all randomized algorithms. Now, given a randomized algorithm, i.e. given specific values of p_a, p_b and p_c, the expected cost can be written as

$$p_a(p_i a_i + p_j a_j + p_k a_k) + p_b(p_i b_i + p_j b_j + p_k b_k) + p_c(p_i c_i + p_j c_j + p_k c_k) \tag{2}$$

(Note that this is the same expression as in (1) above.) where the expressions in the brackets denote the expected costs of each deterministic algorithm for the input distribution. The minimum of the above expression is nothing but the minimum of these expected costs of the deterministic algorithms (in other words, we need to consider only 3 of the distributions where only one of p_a, p_b and p_c is 1; this is sufficient because every other distribution on the deterministic algorithms is going to make the expected cost larger if not the same). Thus, given an input distribution, the lower bound on the expected cost for all randomized algorithms is

$$\texttt{min}(p_i a_i + p_j a_j + p_k a_k, p_i b_i + p_j b_j + p_k b_k, p_i c_i + p_j c_j + p_k c_k)$$

It is important to note that the value of the above expression is definitely *a* lower bound, but not necessarily the optimal. This is again useful in the general case where we cannot enumerate all possible input distributions for a problem and we still need to talk about a lower bound.

Now, the *optimal* lower bound for the worst-case input distribution, for all randomized algorithms is the maximum of all these lower bounds :

$$\texttt{max}(\texttt{min}(p_i a_i + p_j a_j + p_k a_k, p_i b_i + p_j b_j + p_k b_k, p_i c_i + p_j c_j + p_k c_k))$$

In other words, our objective now is to find the input distribution which matches with the above maximum. Once we find the values of p_i, p_j and p_k which satisfies the above minimax condition, we can compute the lower bound.

Last but not the least, someone asked during the recitation if we can take the maximum of the minimum expected cost of any input, given the values of p_a, p_b and p_c as the lower bound. The truth is that it is definitely a lower bound, but *need not be the optimal.* Here is a reasoning. If we rephrase what's been suggested, we are first finding the expected *best-case* cost (minimum expected cost) for a given randomized algorithm (we know p_a, p_b and p_c). If we take the maximum of all these best-case costs, what we get is *how bad the expected best-case cost of any randomized algorithm can be* which is definitely not a tight lower bound on the expected worst-case cost (the former is always smaller than or equal to the latter)!