Theory of Computation

Final Examination on January 9, 2018
Fall Semester, 2017

Problem 1 (20 points) Suppose algorithm C runs in expected time $T(n)$ and always gives the right answer. How to turn it into a randomized algorithm that runs *within* time $O(T(n))$ and gives a wrong answer with probability at most, say, $1/4$.

Proof: Consider an algorithm that runs C for time $kT(n)$ and rejects the input if C does not stop within the time bound. By Markov's inequality, this new algorithm runs in time $kT(n)$ and gives the wrong answer with probability $\leq 1/k$. Pick $k = 4$. ■

Problem 2 (20 points) We showed in the class that all decision problems (decidable or otherwise) can be solved by a circuit of size 2^{n+2} where n is the input length. So the halting problem can be solved by a family of circuits that grow at most exponentially fast in the input length. What is wrong with the argument?

Proof: Such a family exists. But there is no Turing machine (our computation model) that can pick the right circuit given a halting problem of length n for all n. ■

Problem 3 (20 points) Recall that the circuit $\mathrm{CC}(X_1, X_2, \ldots)$ returns true if and only if some X_i is a clique of the input graph $G(V, E)$. Let $\mathcal{X} = \{ X_1, X_2, \ldots \}$ and $\mathcal{Y} = \{ Y_1, Y_2, \ldots \}$ be two set systems. (So $X_i, Y_j \subseteq V$ for all i and j.) Prove that $\mathrm{CC}(\{ X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y} \})$ *introduces* no false positives and no false negatives over our positive examples (graphs with a clique, i.e., a complete graph, of size k and other nodes being isolated nodes) and negative examples (graphs generated by coloring).

Proof: Suppose $\mathrm{CC}(\{ X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y} \})$ returns true. Then some $X_i \cup Y_j$ is a clique. Thus $X_i \in \mathcal{X}$ and $Y_j \in \mathcal{Y}$ are cliques, making both $\mathrm{CC}(\mathcal{X})$ and $\mathrm{CC}(\mathcal{Y})$ return true. So $\mathrm{CC}(\{ X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y} \})$ introduces no false positives. (Alternatively, you can start with a negative example that makes one of $\mathrm{CC}(\mathcal{X})$ and $\mathrm{CC}(\mathcal{Y})$ returns false, then prove $\mathrm{CC}(\{ X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y} \})$ must return false as well.)

On the other hand, suppose both $\mathrm{CC}(\mathcal{X})$ and $\mathrm{CC}(\mathcal{Y})$ accept a positive example with a clique $\mathcal{C}$ of size k. This clique $\mathcal{C}$ must contain an $X_i \in \mathcal{X}$ and a $Y_j \in \mathcal{Y}$. As this clique $\mathcal{C}$ also contains $X_i \cup Y_j$ based on our definition of a positive example, the circuit $\mathrm{CC}(\{ X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y} \})$ returns true. So this circuit introduces no false negatives. ■

Problem 4 (20 points) Calculate $(2200|999)$ and $(2017|999)$. (Answers without the steps will not receive a credit.)

Proof: We have that

$$(2200|999) = (202|999) = (101|999) = (90|101) = -(45|101) = -(11|45) = -(1|11) = -1,$$

and

$$(2017|999) = (19|999) = -(11|19) = (8|11) = -(4|11) = (2|11) = -(1|11) = -1.$$

■

Problem 5 (20 points) Suppose there are n jobs to be assigned to m machines. Let t_i be the running time for job $i \in \{1, 2, \ldots, n\}$, $A[\,i\,] = j$ be an assignment for job i on machine $j \in \{1, 2, \ldots, m\}$, and $T[\,j\,] = \sum_{A[\,i\,]=j} t_i$ be the total running time for machine j. The makespan of A is the maximum time that any machine is busy, given by

$$\text{makespan}(A) = \max_j T[\,j\,].$$

The problem LoadBalance is to compute the minimal makespan of A. It is known that the decision version of LoadBalance is NP-hard. Consider the following algorithm for LoadBalance:

```
for i ← 1 to m do
    T[i] ← 0;
end for
for i ← 1 to n do
    min ← 1;
    for j ← 2 to m do
        if T[j] < T[min] then
            min ← j;
        end if
    end for
    A[i] ← min;
    T[min] ← T[min] + t_i;
end for
return max_i{T[i]};
```

Show that this algorithm for LoadBalance is a $\frac{1}{2}$-approximation algorithm, meaning that it returns a solution that is at most 2 times the optimum.

Proof: Let OPT be the optimal makespan. Note that OPT $\geq \max_i t_i$ and OPT $\geq \frac{1}{m}\sum_{i=1}^{n} t_i$. Suppose that machine i^* has the largest total running time, and let j^* be the last job assigned to machine i^*. Since $T[\,i^*\,] - t_{j^*} \leq T[\,i\,]$ for all $i \in \{1, 2, \ldots, m\}$, $T[\,i^*\,] - t_{j^*}$ is less than or equal to the average running time over all machines. Thus,

$$T[\,i^*\,] - t_{j^*} \leq \frac{1}{m}\sum_{i=1}^{m} T[\,i\,] = \frac{1}{m}\sum_{i=1}^{n} t_i \leq \mathrm{OPT}.$$

We conclude that $T[\,i^*\,] \leq 2 \times \mathrm{OPT}$. ■