15-251: Great Theoretical Ideas In Computer Science

Recitation 5

It's Time to Learn

- The running time of an algorithm A is a function $T_A : \mathbb{N} \to \mathbb{N}$ defined by $T_A(n) = \max_{I \in S}\{\text{number of steps } A \text{ takes on } I\}$, where S is the set of instances I of size n.
- For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = O(g(n))$ if there exist constants $c, n_0 > 0$ such that $\forall n \geq n_0$, we have $f(n) \leq cg(n)$.
- For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = \Omega(g(n))$ if there exist constants $c, n_0 > 0$ such that $\forall n \geq n_0$, we have $f(n) \geq cg(n)$.
- For both of the above, your choice of c and n_0 cannot depend on n.
- For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
- In cake cutting, each player $i \in N = \{1, 2, \dots, n\}$ has a non-negative valuation function V_i over pieces of cake. The following properties about V_i hold:
  - **Additive.** For $X \cap Y = \emptyset, V_i(X) + V_i(Y) = V_i(X \cup Y)$
  - **Normalized.** $V_i([0, 1]) = 1$
  - **Divisible.** $\forall\ \lambda \in [0, 1]$, you can always cut some $I' \subseteq I$ such that $V_i(I') = \lambda V_i(I)$
- We say an allocation $A_1, A_2, \dots, A_n$ is:
  - **Proportional** if $\forall\ i \in N, V_i(A_i) \geq \frac{1}{n}$
  - **Envy-free** if $\forall\ i, j \in N, V_i(A_i) \geq V_i(A_j)$. In other words, every player values their piece of cake at least as much as they value everyone else's.

Bits and Pieces

Determine which of the following problems can be computed in worst-case polynomial-time, i.e. $O(n^k)$ time for some constant k, where n denotes the number of bits in the binary representation of the input. If you think the problem can be solved in polynomial time, give an algorithm in pseudo-code, explain briefly why it gives the correct answer, and argue carefully why the running time is polynomial. If you think the problem cannot be solved in polynomial time, then provide a proof.

(a) Give an input positive integer N, output $N!$.

(b) Given as input a positive integer N, output True if $N = M!$ for some positive integer M.

(c) Given as input a positive integer N, output True iff $N = M^2$ for some positive integer M.

Odd-Paz

State and prove a divide-and-conquer procedure for proportional cake cutting between any number of players. (The Even-Paz algorithm as described in lecture is an excellent starting point.)

(Extra) $\mathcal{O}$, I Think I Understand Asymptotics Now

Let f, g, h be functions from $\mathbb{N}$ to $\mathbb{N}$. Prove or disprove the following:

(a) If $f \in \mathcal{O}(g)$ and $g \in \mathcal{O}(h)$, then $f \in \mathcal{O}(h)$

(b) If $f \in \mathcal{O}(g)$, then $g \in \mathcal{O}(f)$

(c) $f \in \mathcal{O}(g)$ or $f \in \Omega(g)$

(Extra) Where is the Median?

You are given two sorted arrays of integers with equal length. You want to determine the median of all the elements (if there are two elements in the middle, take the smallest one to be the median). Obtain a running time bound under the comparison model in terms of n, which we define to be the total number of elements in the arrays. In the comparison model, comparing two elements takes 1 step, and all other operations are free.

(Bonus) Your Guesses are Two High!

Suppose I am thinking of a number between 1 and n, and will tell you if your guess is too high, too low, or correct. However, I only allow you to guess too high once, or you lose. How quickly can you guess my number? One possible solution is to just guess incrementally from 1 to my number, which takes $\mathcal{O}(n)$ time. Can you do better?