CS 121 Homework 1: Fall 2020

Some policies: (See the course policies page at `http://madhu.seas.harvard.edu/courses/Fall2020/policy.html` for the full policies.)

- **Collaboration:** You may collaborate with other students that are currently enrolled in this course in brainstorming and thinking through approaches to solutions, but you should write the solutions on your own and may not share them with other students.
- **Owning your solution:** Always make sure that you "own" your solutions to this and other problem sets. That is, you should always first grapple with the problems on your own, and even if you participate in brainstorming sessions, make sure that what you write down to submit reflects your own understanding of the solution. This is in your interest as it ensures you have a solid understanding of the course material, which will help in the midterms and final. Getting 80% of the problem set questions right on your own will be much better to both your understanding and grade than getting 100% of the questions by gathering hints from others without true understanding.
- **Serious violations:** Sharing questions or solutions with anyone outside this course, including posting on outside websites, is a violation of the honor code policy. In particular, you may not get help from students or materials from past years of this or equivalent courses.
- **Submission Format:** The submitted PDF should be typed and in the same format as ours. Please include the text of the problems and write **Solution X:** before your solution. Please mark in Gradescope the pages where the solution to each question appears. We may deduct points if you submit in a different format.
- **Late Day Policy:** To give students some flexibility to manage your schedule, you are allowed a total of **eight** late days through the semester, but you may not take more than **two** late days on any single problem set.

By writing my name here I affirm that I am aware of these policies and abided by them while working on this problem set:

Your name: Todd Morrill

Collaborators:

No. of late days used on previous psets (not including Homework Zero): 0
No. of late days used after including this pset:

Questions

Please solve the following problems. Some of these might be harder than the others, so don't despair if they require more time to think or you can't do them all. Just do your best. If you don't have a proof for a certain statement, be upfront about it. You can always explain clearly what you are able to prove and the point at which you were stuck. Also you can always simply write **"I don't know"** and you will get 15 percent of the credit for this problem. If you are stuck on this problem set, you can use Piazza to send a private message to all staff.

Problem 1: You may use a calculator/spreadsheet for both parts of this question.

Problem 1.1 (10 points): The 1977 Apple II personal computer had a processor speed of 1.023 Mhz or about 10^6 operations per second. In 2019 the world's fastest supercomputer performed 93 "petaflops" or 9.3×10^{16} basic steps per second. For each of the following running times (as functions of the input length n), compute for each of the two computers how large an input it could handle in a week of computation, if it runs an algorithm with that running time: (a) n operations, (b) n^2 operations, (c) $10^6 n \log_2 n$ operations, (d) 2^n operations and (e) Tower(n) operations, where $\text{Tower}(0) = 1$ and $\text{Tower}(m) = 2^{\text{Tower}(m-1)}$. Your answers can be approximate, but you should get the two most significant decimal digits right.

Solution 1.1:

(a)

$$f(x) = x$$

(b)

$$f(x) = \sqrt{x}$$

(c)

$$f(x) = e^{W(x/10^6)\cdot \ln 2}$$

where W is the Lambert W function.

(d)

$$f(x) = \log_2 x$$

(e)

$$f(x) = \begin{cases} 0 & x < 2 \\ 1 & 2 \leq x < 4 \\ 2 & 4 \leq x < 16 \\ 3 & 16 \leq x < 65,536 \\ 4 & 65,536 \leq x < {}^{5}2 \end{cases}$$

After solving/showing work, please summarize your answers here:

Problem	Apple II length in a week	2019 length in a week
a	6187104000000.0	$5.6246400000000006 \times 10^{23}$
b	2487388.99	749975999615.99
c	40430.27	1124955245720374.5
d	42.49	78.9
e	4	4

Problem 1.2 (5 points): Typically the number of operations that the fastest computers can perform doubles every three years[1]. So in 2022 we may expect computers performing 186 petaflops. How would you compare the largest input that computers can handle in 2022 vs. what they could handle in 2019 if they run an algorithm that makes the following number of operations: (a) n operations, (b) n^2 operations, (c) $10^6 n \log_2 n$ operations, (d) 2^n operations and (e) Tower(n) operations. For each case express your answer as "The largest n in 2022 (roughly) grows/shrinks by an additive/multiplicative factor of X compared to 2019." for the best number X and choice of "additive" and "multiplicative" you can determine.

Solution 1.2:

Multiplicative factors for (a), (b), (c), (d), and (e) can be computed with the following function:

$$f(x_{22}, x_{19}, g) = g(x_{22})/g(x_{19})$$

where x_{22} is the 2022 super computer speed, x_{19} is the 2019 super computer speed, g is the function defined in **Solution 1.1:** for (a), (b), (c), (d), and (e), respectively.

After solving/showing work, summarize answers here:

Problem	Grows or Shrinks	Multiplicative or Additive	Rough Factor
a	grows	multiplicative	2.0
b	grows	multiplicative	1.41
c	grows	multiplicative	1.96
d	grows	multiplicative	1.01
e	grows	multiplicative	1.0

[1]For those comparing carefully with the previous problem: the Apple II was not the fastest computer of 1977.

Problem 2 (24 points): For each pair of functions F and G below, determine which[2] of the following relations hold: $F = O(G)$, $F = \Omega(G)$, $F = o(G)$, and $F = \omega(G)$.

1. $F(n) = n$, $G(n) = 100n$.
2. $F(n) = n$ and $G(n) = \sqrt{n}$.
3. $F(n) = n \log n$, $G(n) = 2^{(\log n)^2}$.
4. $F(n) = 2^n$, $G(n) = 8^n$.
5. $F(n) = \binom{n}{\lceil .1n \rceil}$ and $G(n) = 2^{.5n}$, where $\binom{n}{k}$ is the number of subsets of $[n]$ of size k, where $[n] = \{1, 2, \ldots, n\}$.

   Hint: You may use Stirling's approximation for this.

Solution 2:

Part	$F = O(G)$	$F = \Omega(G)$	$F = o(G)$	$F = \omega(G)$
1	X			
2		X		X
3	X		X	
4	X		X	
5		X		X

[2]Note: the number that hold may be 0 or more than 1.

Problem 3: The Pigeonhole Principle, as understood by pigeons, asserts that if there more pigeons than pigeonholes and all pigeons are sitting in pigeonholes, then some pigeonhole contains at least two pigeons.

Mathematicians have their own pigeonhole principle (PHP) which states that if A and B are finite sets such that $|A| > |B|$ and $f : A \to B$ is a function then f is not a one-to-one[3] function.

Problem 3.0 (0 points): Why is the math PHP called that?

Mathematicians claim that a function isn't one-to-one when it may be ambiguous which element $a \in A$ is mappped to an element $b \in B$. In other words, *many* a's may map to *one* b (i.e. many-to-one).

Problem 3.1 (1 point): Prove the mathematical PHP. (You may use any facts stated in Barak, Chapter 1.)

Solution 3.1:

Proof. If $|B| = 0$ then there is no one-to-one function and the proof is shown. Now we consider the case when $|B| > 0$. Since f is a total function, all elements $a \in A$ must be mapped into B. Since $|A| > |B|$ it must be the case that $\exists x, y \in A, b_0 \in B$ such that $(x \neq y)$ and $(f(x) = f(y) = b_0)$, which violates the definition of a one-to-one function which states that $\forall x, y \in A$ such that $(f(x) = f(y) \implies x = y)$. $\square$

Problem 3.2 (15 points): Let $n \geq 10$ and $M < 2^{\sqrt{n}}$ be positive integers. Given n integers $a_1, \ldots, a_n$ with $a_i \in \{1, \ldots, M\}$, prove that there exist two non-empty disjoint sets S and T such that $\sum_{i \in S} a_i = \sum_{j \in T} a_j$. (Hint: Use the PHP. But if you do, tell us what sets A and B and function f are you applying the PHP to.)

Solution 3.2:

Proof. The proof is by the PHP. We will show that the cardinality of set A, which is the set of all unique subsets of n digits is larger than the cardinality of the set B, which is set of unique sums resulting from the sum of n digits. $|A| = 2^n - 2$ (i.e. the power set minus the empty set and complete set of n digits) and $|B| < (n-1)M$, which is an upper bound on the number of unique sums of $n-1$ numbers, each of which can take a maximum value M. We count the sums up to $(n-1)M$ because we must leave at least one of the n integers for the other set. Since $|A| > |B|$ for sufficiently large n (see ratio below), there must exist two non-empty disjoint sets S and T such that $\sum_{i \in S} a_i = \sum_{j \in T} a_j$.

The final point to show is that the two sets $S, T \in A$ that map to one element $b \in B$ can be made to be disjoint. First, suppose the two sets S and T are disjoint, then the point is shown. Second, suppose the two sets S and T are not disjoint, then we can simply subtract the overlapping elements from two sets S and T to arrive at a new valid sum $b' \in B$ and the point is shown.

We now show that this holds for all $n \geq 10$. At 10, we have $2^{10} - 2 = 1022$ and $(10-1)2^{\sqrt{10}} \approx 80.57$. Both functions are montonically increasing after 10 and rate of growth of $|A|$ is greater than $|B|$:

$$\lim_{x \to \infty} \frac{2^n - 2}{(n-1)2^{\sqrt{n}}} = \infty$$

[3] Vocabulary note for those with different math backgrounds: "one-to-one" is a synonym of "injective"; "one-to-one correspondence" is a synonym of "bijective function", and "onto" is a synonym of "surjective".

□

Problem 3.2 (Bonus, 0 points): Same question as Problem 3.2, except now $a_i \in \{-M, \dots, +M\}$.

Problem 4:

Problem 4.1 (5 points): Show that there is a string representation of directed graphs with vertex set $[n]$ having at most $10n$ edges that uses at most $1000n \log n$ bits.

More formally: Define, for every $n, m \in \mathbb{N}$, $G_{n,m}$ to be the set of directed graphs[4] over the vertex set $[n]$ with *at most* m edges. Prove that for every sufficiently large n, there exists a one-to-one function $E : G_{n,10n} \to \{0,1\}^{1000n \log n}$.

Note: When you see a "round" constant such as 10, 100, or 1000 in a problem set, it usually means that it was chosen arbitrarily, and there is no particular significance to the actual number. In particular, it may well be that you could come up with such a scheme where E maps $G_{n,10n}$ to a string of length at most $cn \log n$ for some constant c that is significantly smaller than 1000.

Solution 4.1:

We will encode the graph G in adjacency list form to take advantage of the sparsity of the edges, where each vertex's edge list will be represented by the length of the list followed by the list of neighboring edges. We will encode natural numbers using C-style encodings to achieve a prefix-free encoding for lists of natural numbers. In particular, let $E_0 : \{0,1\}^* \to \{0,1\}^*$ be the C-style prefix-free modification of binary strings, sending 0 to 00, 1 to 11, and terminating strings with 01. Then the first element $n_0 \in \mathbb{N}$ represented in the binary string encoding the graph G will be the length of the first vertex's edge list followed by the terminating bit string 01 followed by the first adjacency list, where each element in the list is a binary representationon of the vertex number followed by the terminating string. This continues for all vertices and edges in the graph.

To decode a binary string, we simply decode two bits at a time until we reach a terminating string (i.e. 01) using the inverse of E_0 to arrive at the binary representation of the length of the adjacency list for the first vertex. We denote the length of this list as j. Then we decode the following j vertices in the adjacency list using the aforementioned scheme and then repeat the process for all remaining vertices in the graph.

This requires $n(2(\log n + 1) + 2)$ bits to represent the lengths of all the vertex adjacency lists. Then this requires $10n(2(\log n + 1) + 2)$ to represent all the edges in the adjacency lists. Taken together, this sums to $22n \log n + 44n$. Thereore, the total number of graphs $G_{n,10n}$ that can be represented by this scheme in binary strings for sufficiently large n is $|\{0,1\}|^{22n \log n + 44n} = 2^{O(22n \log n)} = 2^{O(1000n \log n)}$

Problem 4.2 (10 points): Define S_n to be the set of one-to-one and onto functions mapping $[n]$ to $[n]$. Prove that there is a one-to-one mapping from S_n to $G_{2n,n}$.

Solution 4.2:

Proof. S_n is the set of bijective functions mapping $[n]$ to $[n]$ and as such, represents a set of permutations, which we denote π_f for all functions $f \in S_n$. In particular, let $\pi_f : i \to \pi(i)$ be one particular permutation associated with function f that maps $i \in [n]$ to it's corresponding element $\pi(i) \in [n]$. To be sure, $\forall f, f' \in S_n, \exists i \in [n]$ such that $\pi_f(i) \neq \pi_{f'}(i)$ (i.e. they are unique functions). We now show that this set of permutations is one-to-one with a subset of the graphs

[4]In CS 121, unless specified otherwise, every graph is *simple*: each edge has two *distinct* vertices and no two edges are equal.

in $G_{2n,n}$. Since there are $2n$ vertices, we can make a minor modification to the mapping scheme defined above by simply adding n to $\pi(i)$ to fully index all of the $2n$ vertices. In particular, we now have $\pi_f : i \to \pi(i) + n$, where $i \in [n]$ is the first vertex in the edge and $\pi(i) + n \in [2n] \setminus [n]$ is the second edge in the vertex. This shows that there exists a one-to-one mapping from S_n to $G_{2n,n}$. ☐

Problem 4.3 (10 points): Show that the encoding length of Problem 4.1 can not be improved to $o(n \log n)$. Specifically, prove that for every sufficiently large $n \in \mathbb{N}$, there is no one-to-one function $E : G_{n,10n} \to \{0,1\}^{.0001n \log n}$.

Solution 4.3:

Proof. We begin by finding a lower bound on the number of possible configurations for the set of graphs $G_{n,10n}$ (i.e. the cardinality), which can be done with a combination.

$$\begin{aligned}\binom{n(n-1)}{10n} &= \frac{(n(n-1))!}{10n!(n^2-n-10n)!} \\ &= \frac{(n^2-n)(n^2-n-1)\cdots(n^2-n-(10n-1))}{10n(10n-1)\cdots(1)} \\ &> (\frac{n^2-11n-1}{10n})^{10n} = (\frac{n}{10} - \frac{11}{10} + \frac{1}{10n})^{10n}\end{aligned}$$

We now find the cardinality of the set of binary strings $\{0,1\}^{.0001n \log n}$.

$$|\{0,1\}|^{.0001n \log n} = 2^{.0001n \log n} = n^{.0001n}$$

Comparing the two cardinalities, we have $(\frac{n}{10} - \frac{11}{10} + \frac{1}{10n})^{10n} > n^{.0001n}$, which implies that $|G_{n,10n}| > |\{0,1\}^{.0001n \log n}|$ and therefore there is no one-to-one function $E : G_{n,10n} \to \{0,1\}^{.0001n \log n}$, which is what we needed to show. ☐