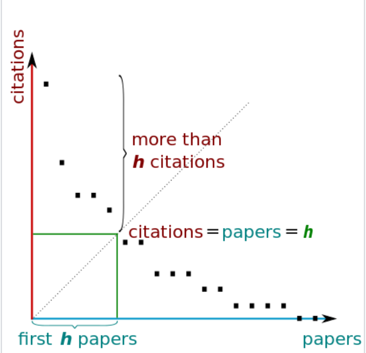
f(A)=10, f(B)=8, f(C)=5, f(D)=4, f(E)=3 → h-index=4

f(A)=25, f(B)=8, f(C)=5, f(D)=3, f(E)=3 → h-index=3

If we have the function f ordered in decreasing order from

h-index (f) = $\max\{i \in \mathbb{N} : f(i) \geq i\}$



h-index from a plot of numbers of citations for an author's numbered papers (arranged in decreasing order)

For

5 6

C=[17, 17, 12, 10, 7, 4, 2, 2, 1, 1, 1]

H-index is 5 (as there are 5 papers cited at least 5 times each).

CP 312, Winter 2023

signment 3 (6% of the final grade)

(due Friday, March 10, at 23:30)

questions in this assignment. Note that all logarithms in this assignment $f(n) \equiv \log_2 f(n)$.

' an algorithm means: describe the algorithm briefly in words, give high-level correctness, and analyze worst case running time.

1. [6 marks] **Divide and Conquer/Binary Search.**
One popular way to rank researchers is by their "h-index". A researcher's h-index is the maximum integer k such that the researcher has at least k papers that have been cited at least k times each. Suppose a researcher X has written n papers and paper i has been cited c_i times. Suppose you have these values sorted in an array C with $c_1 \geq c_2 \geq \ldots \geq c_n$. Give a divide-and-conquer (variation of binary search) algorithm to find researcher X's h-index. Describe the algorithm briefly in words, give detailed pseudocode, justify correctness, and analyze run-time. Your algorithm should run in time $O(\log n)$.

2. [8 marks] **Divide and Conquer.**
Suppose you have an unsorted array $A[1..n]$ of elements. You can only do equality tests on the elements (e.g. they are large GIFs). In particular this means that you cannot sort the array. You want to find (if it exists) a *majority* element, i.e., an element that appears more than half the time. For example in $[a, b, a]$ the majority element is a, but $[a, b, c]$ and $[a, a, b, c]$ have no majority element. Straightforward approach of checking if $A[i]$ is a majority element for all $i = 1, \ldots, n$ will have run-time in $\Theta(n^2)$ which is too slow. Consider the following approach to finding a majority element: recursively find the majority element y in the first half of the array and the majority element z in the second half of the array, and combine results of recursive calls into the answer to the problem.

   (a) Prove that if x is a majority element in A then it has to be a majority element in the first half of the array or the second half of the array (or both).

   (b) Using observation of part (a) give a divide-and-conquer algorithm to find a majority element, that runs in time $O(n \log n)$. **Detailed pseudocode is required**. Be sure to argue correctness and analyze the run time. If given array has no majority element, return FAIL.

3. [10 marks] **Greedy algorithm.**
Here is a special "matching" problem: given a set A of n numbers and a set B of n numbers, form pairs (a_1, b_1), ..., (a_n, b_n), with $\{a_1, \ldots, a_n\} = A$ and $\{b_1, \ldots, b_n\} = B$, so that the following cost function is minimized

$$\sum_{i=1}^{n} (a_i - b_i)^2 .$$

A = [3,6,2,1]
B = [8,1,3,4]

Greedy
(1-1)^2 + (3-3)^2 + (2-4)^2 + (6-8)^2
0 + 0 + 4 + 4 = 8

Optimal
(1-1)^2 + (2-3)^2 + (3-4)^2 + (6-8)^2
0 + 1 + 1 + 4 = 6

Consider the following "greedy" strategy to solve this problem: pick the pair (a, b) with the smallest difference $|a - b|$ ($a \in A, b \in B$). Then remove a from A and b from B, and repeat.

(a) Give a counterexample showing that this strategy is incorrect, i.e., it can sometimes give a non-optimal solution. **Show your work!**

(b) Give another greedy strategy that will solve this special matching problem (detailed pseudocode is required). Prove that it always returns an optimal solution. Justify correctness and analyze running time.

Submission.

1. Read `assignments_README.pdf` for the instructions.
2. Submit single pdf file `A3.pdf` to the assignment 3 dropbox on `MyLearningSpace`.

A = [3,6,2,1] = [1,2,3,6]
B = [8,1,3,4] = [1,3,4,8] just go in order once they are sorted? cant be that easy.

A = [1,5,7,7,9]
B = [1,4,6,8,9]

greedy (1-1) + (5-6) + (7-8) + (9 -9) + (7-4) = 0+ 1 + 1 + 0 + 9 = 11 (wrong)
in order (1-1) + (5-4) + (7-6) + (7-8) + (9-9) = 0 + 1 + 1 + 1+ 0 = 3

A = [5, 6, 8, 10]
B = [1, 2, 4, 5]

greedy (5-5) + (6-4) + (8 - 2) + (10 - 1) = = 121
in order (5-1) + (6-2) + (8-4) + (10-5) = 16 + 16 + 16 + 25 = 73

A = [5, 6, 8, 10]
B = [11, 12, 14 , 16]

greedy (10 -11) + (8-12) + (6-14) + (5-16) = 1 + 16 + 64 + 121 = 202
in order (5-11) + (6-12) + (8-14) + (10 - 16) = 36 + 36 + 36 + 36 = 144