Design and analysis of algorithms

Given an algorithmic problem,

(1) Write pseudocode for an algorithm solving the problem

(2) Write/perform big-Oh runtime analysis

(3) Write a mathematical proof that algorithm is correct

Given an array A of positive integers, determine if A has a Pythagorean triple ($a^2+b^2=c^2$).

3, 4, 5

5, 12, 13

Step 1: FIND A BRUTE FORCE SOLUTION.

Solution that (typically) only uses arrays, only use linear search, nested for loops. $O(n^3)$

Step 2: Use a fancy structure/divide-and-conquer/sorting/merge, partition tricks to improve running time

→ Converts $O(n)$ to $O(\log n)$ or $O(1)$

length of A is n

$\sqrt{A[i]^2 + A[j]^2}$ find k s.t $= A[k]$

(1) for i = 0 to n-1 $O(n^3)$

(2) for j = ~~0 to~~ i+1 to n-1 $O(n^2)$

(3) for k = j+1 to n-1 $O(n)$ Linear search

(4) ~~check~~ if $(A[i]^2 + A[j]^2 == A[k]^2$ or
~~$A[k]^2 + A[i]^2 == A[j]^2$~~ or $A[k]^2 + A[j]^2 == A[i]^2)$ $O(1)$
return True

(5) return False

The algorithm is a triple-nested for-loop, where each loop runs at most n iterations. The innermost statement (Step 4) runs in $O(1)$ time. Hence, running time is $O(n^3)$.

Convert linear search to binary search

Sort the array

Use a better data structure

~~for i = 0 to n-1~~

~~for j = 0 to n-1~~

Suppose A is sorted $O(n \log n)$

(0) Sort A $O(n \log n)$

(1) for i = 0 to n-1 $O(n^2 \log n)$

{

(2) (a) for j = i+1 to n-1 $O(n \log n)$

{

(3) (i) Using binary search, find $A[k] = \sqrt{A[i]^2 + A[j]^2}$ $O(\log n)$ → If found return True

~~Using binary search, find $A[k] = \sqrt{A[i]^2 - A[j]^2}$~~

~~(assuming $A[i] \geq A[j]$)~~

~~or switch i, j~~

}

}

(2) return False

The running time is $O(n \log n) + O(n^2 \log n)$

$= O(n^2 \log n)$.

---

(0) Insert A into hash table $O(n)$

(1) for i.. $O(n^2)$

(a) for j.. $O(n)$

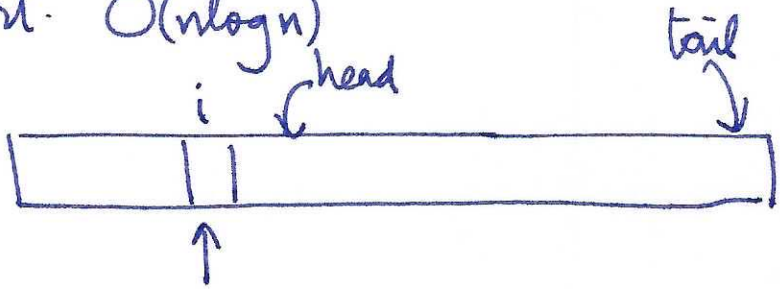
(i) Using hash table, search for ~~A[k] =~~ $\sqrt{A[i]^2 + A[j]^2}$ $O(1)$

$O(n) + O(n^2) = O(n^2)$

Get a solution with $O(1)$ / $O(\log n)$ extra memory.

Getting $O(n^2)$ without hash table.

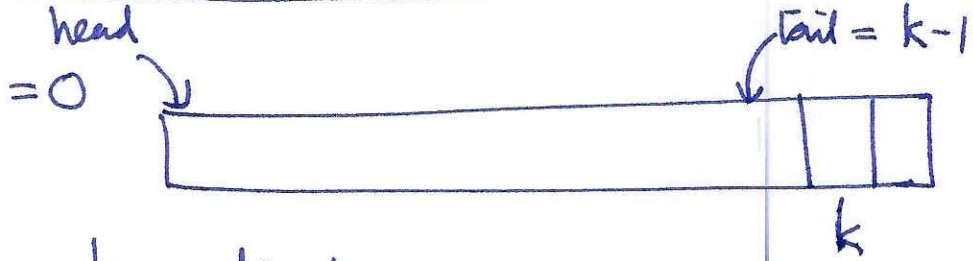
Let's sort. $O(n \log n)$



Let's find Pythagorean triples where i is the min. index.

Head init i+1 , tail init n-1

If $A[i]^2 + A[head]^2 < A[tail]^2$



k is fixed

Let's find triple where $A[i]^2 + A[j]^2 = A[k^2]$

Suppose $A[head]^2 + A[tail]^2 > A[k]^2$

Then for ANY index > head $A[index]^2 + A[tail]^2 > A[k^2]$

(sorted)

if $(A[head]^2 + A[tail]^2 > A[k]^2)$, decrement tail.

if $(A[head]^2 + A[tail]^2 < A[k]^2)$, increment head.

if (.. == $A[k^2]$), done

(a) Sort A $O(n \log n)$

(i) For $k = n-1$ to 0 : $O(n^2)$

(a) Init head $= 0$ and tail $= k-1$

(b) Use logic above to find triple $A[i]^2 + A[j]^2 = A[k]^2$ $O(n)$

$O(n^2)$ without hash table

Better than n^2? Major open problem in (3SUM problem) math / TCS → theoretical

Find i, j, k s.t. $A[i] + A[j] + A[k] = 0$