CS 810 Lecture Notes

Convex Hull and Prime Testing

Dev Patel

Oct 3, 2023

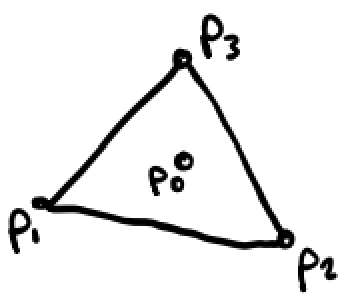
Convex Hull

Trying to grow the convex hull 1 point at a time:

INPUT: Set S of N points in 2D (assume no 3 points are collinear)

Initialization:

1. Randomly permute S: $p_1,p_2,...,p_n$
2. Let $S_3 = \{p_1,p_2,p_3\}$ and let $CH(S_3)$ be the C.H. of S_3 (in counterclockwise order)
3. Let p_0 be the centroid of S_3 (this point will always be in C.H.)



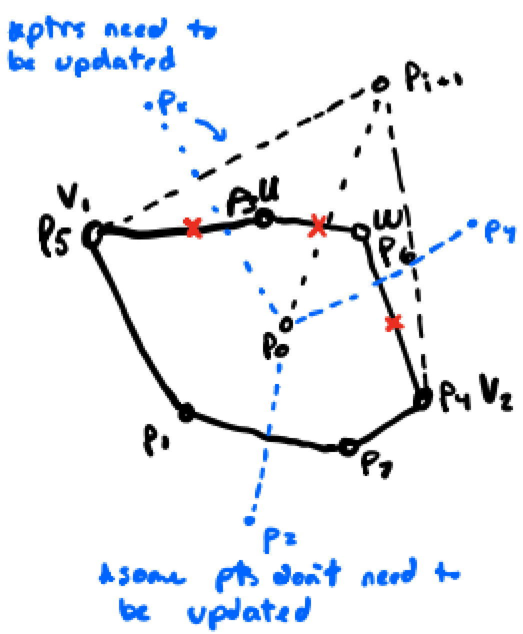
Data Structures:

1. Maintain circular list of $CH(S_i)$
    a. $S_i = \{p_1,...,p_i\}$
2. For every point in S - S_i, keep a pointer from p to edge (e_p) of S_i that $\overline{p_0 p}$ crosses

Actual Process:

- Look at next point (p_{i+1})

1. If p_{i+1} in $CH(S_i)$, do nothing
2. If p_{i+1} outside of $CH(S_i)$, let uw be the edge of $CH(S_i)$ that $\overline{p_0 p_{i+1}}$ crosses
    a. Updating $CH(s_{i+1})$:
    b. Starting from either u or w, move to the closest vertex away from p_{i+1} (consider them as v_1 and v_2)
    c. If p_{i+1} -> u -> v_1 has a ‘right’ turn, consider v_1 as u and delete old u
    d. Alternatively terminology: if p_{i+1} -> u -> v_1 is convex, replace u
    e. Recurse away from p_{i+1} in both directions until there are no more ‘right turns’
    f. Update both data structures:
        i. $S_i = S_i + p_{i+1}$
        ii. Some pointers need to be updated because edges of the convex hull have been changed
            1. This deterministic approach to managing pointers can perform very poorly in some runs
            2. It’s possible to get very lucky or unlucky on where the algorithm starts and how much work needs to be done



Time Complexity:

Take a backwards approach to calculate time complexity

- The point is that the # of updates forward and backwards is the same

Remove some random point p from S_i

- Need to update pointers and edges

For some pointer, ptr_b, to need updating:

- One of the 2 points making up $edge_b$ had to have been deleted

Therefore, P(pointer for point in S - S_i is updated) $\leq \frac{2}{i-3^*} = O(\frac{1}{i})$

* i - 3 because it can be any point except for the 3 initial non collinear points

E(# pointer updates for S_{i-1} -> S_i) $= O(\frac{n}{i})$

E(# pointer updates in total) $= O(\sum_{i=1}^{n-3}(\frac{n}{i})) = O(n\sum_{1}^{n}\frac{1}{i}) = O(n * log(n))$

Prime Testing

Figuring out if a number is prime

INPUT: Number N $\leq 10^{100}$

OUTPUT: {Yes: N is prime

{No: N is not prime

Approach 1: Brute Force (Deterministic)

Try dividing n by 2,3,...,√n (all primes)

about $\frac{\sqrt{n}}{log(n)}$ divisions, $\approx 10^{50}$

* Probabilistic algorithm can output not prime or probably prime

Approach 2: Fermat's Little Theorem (Probabilistic)

If p is prime, then $\forall a < p : a^{p-1} \equiv 1 \ (mod\ p)$

Ex1:

p = 7, a = 3

$3^1 \equiv 3 \ (mod\ 7)$

$3^2 \equiv 2 \ (mod\ 7)$

$3^4 \equiv 4 \ (mod\ 7)$

$3^6 \equiv 4 \ * 2 \equiv 8 \equiv 1 \ (mod\ 7)$ ← this says that this ‘a’ value for this p value is not a witness

Ex2:

p = 6, a = 2

$2^5 \equiv 2 \ (mod\ 6)$ ← this is a witness to p being composite

Algorithm:

For some p:

1. Pick some a’s and try to find witnesses
2. If no witnesses are found, p is probably prime
3. If a single witness is found, p is definitely composite

Limitations:

Carmichael Numbers:

These are composite numbers that $\forall a$ pass the Fermat test

Ex:

N = 3 * 11 * 17 = 561

$\forall a < 561\text{:}\ a^{560} \equiv 1 \ (mod\ 561)$

*Just because there is no ‘a’ for some p that $a^{p-1} \not\equiv 1 \ (mod\ p)$, it doesn’t prove primality

Wilson’s Theorem:

p is prime IFF (p-1)! ≡ -1 (mod p)

*limitation: computing (p-1)! is too expensive

CRT:

$x \equiv a_1 \ (\text{mod } p_1)$

$x \equiv a_2 \pmod{p_2}$

$\gcd(p_1, p_2) = 1$

Then there is a unique solution (mod p_1p_2) that satisfy these 2 congruences

Fact:

$x^2 \equiv 1 \pmod{p}$

There are exactly 2 solutions in Z_p^+ to this congruence when p is prime

*$Z_p^+ = \{1,...,p-1\}$

$x^2 - 1 \equiv 0 \pmod{p}$

$x^2 - 1 = \alpha p$

$(x+1)(x-1) = \alpha p$ $\quad\quad\quad x = 1, -1$

*p cannot divide some x that is less than p, so x= 1,-1 are the only 2 solutions

What about when p is not prime?

$p = 15$

Satisfies $x^2 \equiv 1 \pmod{p}$ by 1, -1, 4, 11

$p = qr$ s.t. q, r are prime

$x^2 \equiv 1 \pmod{q} \pm 1$

$y^2 \equiv 1 \pmod{r} \pm 1$

(1, 1), (-1, -1), (1, -1), (-1, 1) $\rightarrow$ At least 4 square roots of unity when p is not prime

IDEA: try and find square roots of unity that aren't -1, 1 quickly:

Randomly pick:

- If we find one, number is composite because this square root of unity is a witness
- If not, we might have been unlucky

$x^2 \equiv 1 \pmod{77}$

$(1, 1) \rightarrow 1$

$(-1, -1) \rightarrow p - 1 = 76$

$(1, -1) \rightarrow 43$

$(-1, 1) \rightarrow 34$

P(finding root of unity) = 4/77

Not efficient, a more deliberate algorithm will be better

More deliberate algorithm:

Given p:

1. Check if even/ odd → if even and not 2, return composite [return prime if 2]
2. If odd:
    a. Pick random 'a'
    b. If $a^{p-1} \equiv 1 \pmod{p}$ not true, then return composite
    c. Else:
        i. If $a^{\frac{p-1}{2}} \neq 1, -1$, then return composite
        ii. Else: repeat step i. with $a^{\frac{p-1}{4}} a^{\frac{p-1}{8}}$,... until:

            Once $a^{\frac{p-1}{x}} = -1$, stop the algorithm

            If a number that isn't 1 or -1 is found, p is composite.