Home Work 2. Dec 20. 2020

2. Werewolves

A game with n people (citizens and werewolves), citizens > werewolves.

Goal: Identify one citizen

Operation: Pick two people, ask one the partner's character. citizens tell the truth, werewolves don't have to.

(a) Give a way to identify a citizen using O(n) queries.

Solution: Iteratively pick the person and another people. Ask his identification, if and only if at least half tell the citizen, he is a citizen,

Proof: ① He's a citizen ⇒ Half say citizen.
At least half are citizens. They tell the truth.

② Half say citizen ⇒ He's a citizen.
More than half must be citizen, must be truth (werewolves are less than half).

(b) Show how to find a citizen in O(n log n).

Solution: ① Split the group equally into A and B.

② use the same algorithm recursively call A and B.

③ x = citizen(A), y = citizen(B), check x or y using pair(a), return one who is a citizen. If only one player, return that.

Proof: ① Base Case: one citizen true.

② Induction: k<n true,

one group (citizen > wolves), the algorithm returns a citizen, by pair(a), at least

③ Running Time

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n)$$

(c) Can you give a linear-time algorithm?

Solution: ① Split them into pairs

② If any one of two says the other is a wolf, discard them. If not, discard any one and keep the other.

③ If n = odd, use part (a) to ~~test~~ if the odd is a citizen. If yes, return. If not, discard.

Proof: n_1 = TT n_2 = FF n_3 = TF

citizens > wolves: $n_1 > n_2$

By algorithm, n_3 is discarded (T tell that F is wolves).

n_1 and n_2 remain one of them by next turn.

n_2 is possible to be discarded.

$T(n) = T(\frac{n}{2}) + O(n) = O(n)$

T is must always more than F.

TT FF TF FF TT

↓

TT FF TT

(or discarded)

↓

T F T

↓

TF ①✓

Induction:

Base Case: one people true.

① TF is discarded ② doesn't affect number of citizen.

TT is retained and FF may be retained. $n_{d-c} \geq n_{d-w}$

So that $n_c > n_w$

$n_c + n_{d-c} > n_w + n_{d-w}$

3. Agent Meetup

Manhattan's road system form a chakerboard, all roads are either N-S or E-W. The distance between two intersections is Manhattan distance $|x_i - x_j| + |y_i - y_j|$. You have to arrange a meeting between n agents,

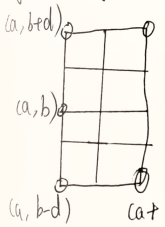
Goal: find the two agents that are closest to each other.

Note: x_i and y_i are integers.

(a) Let (a,b) be an arbitrary intersection. Suppose all agents i for which $x_i \geq a$ are Manhattan distance strictly greater than d apart from each other, where $d > 0$. Give an upper bound

$a \le x_i \le a+d$ and $b-d \le y_i \le b+d$.

i.e. how many agents can fit in this rectangle. without two of these agents being Manhattan distance d or less?



Solution: the answer is 8.

At most one agent is in the small square ($d/2$ edge)

totally 8 square for rectangle.

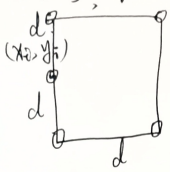
(b) Design a divide and conquer algorithm to find the minimum Manhattan distance between any two agents.

Define L and R be the left and right half of the agents.

The closest pair of agents are
1. both L recursively
2. both R recursively
3. one L and one R

For case 3: we assume shortest in 1 and 2 is $d+1$.

In case 3, we only need to consider at most d.



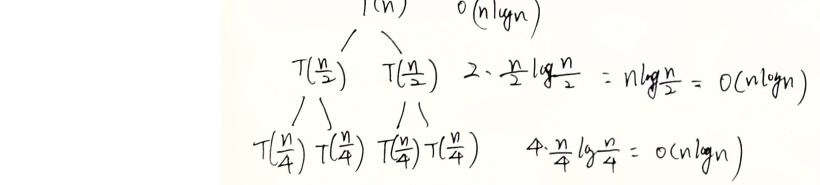
Just consider this rectangle.

If agent out of rectangle, its distance is more than d (at least d), which is the same in case 1 and 2.

return $\min(\text{case } 1, 2, 3)$.

$T(n) = 2T(\frac{n}{2}) + O(n \log n)$: $2T(\frac{n}{2})$: Left and Right

$O(n \log n)$: Sorting X and Y, Locate agents in that rectangular.



Total: $O(n \log n \cdot \log n) = O(n \log^2 n)$

4. Practice with Polynomial Multiplication with FFT

(a) Suppose want to multiply $x+1$ and $2x+1$ using FFT.

$FFT(1,1,0,0) = \begin{pmatrix} 2 \\ 1+i \\ 0 \\ 1-i \end{pmatrix}$ why choose 4 degree?

$2n-1 = 3$, the product has 3 degrees

$FFT(1,2,0,0) = \begin{pmatrix} 3 \\ 1+2i \\ -1 \\ 1-2i \end{pmatrix}$ The closest 2 power is 4.

Product: $\begin{pmatrix} 6 \\ -1+3i \\ 0 \\ -1-3i \end{pmatrix}$ inverse FFT:

$IFFT(6, -1+3i, 0, -1-3i) = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 0 \end{pmatrix}$

The product is $2x^2+3x+1$.

(b) Repeat for $1+x+2x^2$ and $2+3x$

the result has degree of 4.

$FFT(1,1,2,0) = \begin{pmatrix} 4 \\ -1+i \\ 2 \\ 1-i \end{pmatrix}$ ~~FFT~~ $\begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$ $FFT(2,3,0,0) = \begin{pmatrix} 5 \\ 2+3i \\ -1 \\ 2-3i \end{pmatrix}$

$IFFT(20, -5-i, -2, -5+i) = \begin{pmatrix} 2 \\ 5 \\ 7 \\ 6 \end{pmatrix}$

The product is $6x^3+7x^2+5x+2$.

5. Modular Fourier Transform

(a) There exists $w \in \{0,1,2,3,4\}$ such that w are 4^{th} roots of unity (mod 5), ie. $z^4 = 1$. Show that $\{1,2,3,4\}$ are 4^{th} roots of unity (mod 5). Also show that $1+w+w^2+w^3 = 0 \pmod 5$ for $w=2$.

Solution:

w	w^4	mod 5
1	1	1
2	16	1
3	81	1
4	256	1

$1+w+w^2+w^3$

$= 1+2+4+8 = 15 = 0 \pmod 5$

(b) Using Matrix Form of FT, produce the transform of $(0,1,w,2)$ degrees. modulo 5. Multiply this vector by $M_4(w)$ for $w=2$

$M_4(2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \pmod 5$ $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 2 \\ 2 \end{pmatrix}$

(c) Write down the matrix necessary to perform inverse FT.

$FT^{-1} = \frac{1}{4} M_4(\omega^{-1})$

$M_4(2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$ $M_4(2)$ inverse $4 \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix}$

$\frac{1}{4}$ replace $\rightarrow$ multiplicative inverse of 4 (mod 5) 4

2 replace $\rightarrow$ multiplicative inverse of 2 (mod 5) 3

$4 \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \end{pmatrix}$

(d) Show how to compute $(2x^2+3)\cdot(-x+3)$ using FT (mod 5).

$a = \begin{pmatrix} 3 \\ 0 \\ 2 \\ 0 \end{pmatrix}$ $b = \begin{pmatrix} 3 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 0 \\ 0 \end{pmatrix} \pmod 5$

$M_4(2)\, a = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ 0 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \pmod 5$

$M_4(2)\, b = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 4 \\ 0 \end{pmatrix} \pmod 5$

product: $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ IFT: $4 \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 1 \\ 3 \end{pmatrix}$

6. Sparse FFT

Sometimes we want to apply FFT to a sparse vector (most 0). We don't need to do $O(n \log n)$.

Modify so that it runs in $O(n \log k)$. k is number of P_i that are non-zero.

Solution: We can multiply matrix and vector in $O(nk)$ time. Ignoring the zero element.

run FFT for the first $\log k$ levels, at the $\log k$-level we do multiplication rather than recursion.

Running Time: levels $0 \sim \log k$: $O(n)$

k_i: non-zero numbers in ith subproblem.

Takes $O(\frac{n}{k} \cdot k_i)$ for ith subproblem. $\sum_{i=1}^{n} k_i = k$. each level totally costs $O(n)$.

So ~~totally~~ costs $O(n \log k)$.

7. Polynomial from roots.

Give n distinct roots at $r_1 \cdots r_n$, compute the coefficient. $T(n)$ should be $O(n \log^c n)$ for $c > 0$.

$$p(x) = \prod_{i=1}^{k} (x - r_i)$$

Solution: Multiply $(x-r_1), (x-r_2), \cdots (x-r_{n/2})$ recursively to obtain $p(x)$, and $(x - r_{n/2+1}), (x - r_{n/2+2}), \cdots (x - r_n)$ to obtain $Q(x)$. Use FFT to multiply $p(x)\,Q(x)$

Base case: one term $x - r_1$, just return.

$$T(n) = \underbrace{2T(\tfrac{n}{2})}_{\text{front and last}} + \underbrace{O(n \log n)}_{\text{FFT.}} \Rightarrow T(n) = O(n \log^2 n).$$