Today

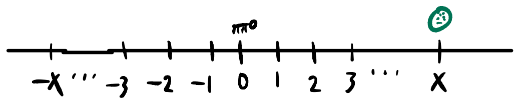
7 Proof Strategies

- Guessing
- Charging
- Potentialing (maybe)
- Doubling
- Halving
- Averaging
- (Token) Rearranging (maybe)

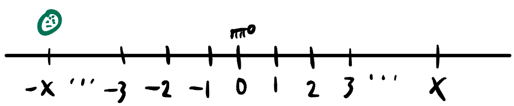
Guessing: Algorithmic Problems easier when parameters of OPT are known

Often don't know parameters but can guess them

Ant on an (Infinite) Log



or



→ Goal: Minimize distance travelled

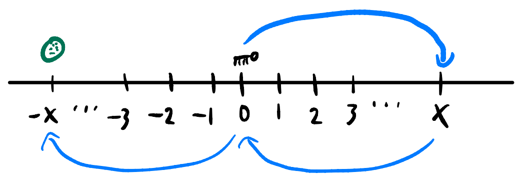
Strategy when X Known

Until @ Cookie

Go to X

Go to -X

Analysis



Ant travels $\leq 3X$

Strategy when X Not Known

Until @ cookie

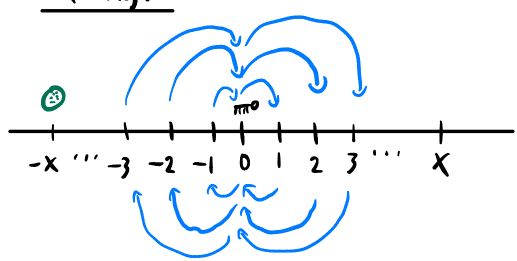
For $x' = 1, 2, 3, \ldots$ (brute force "guess" x!)

Go to x'

Go to $-x'$

Go to 0

Analysis



Ant travels $1+1+1+1+2+2+2+2+\ldots+x+x+x+x$

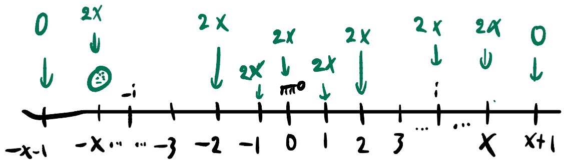
$$= 4\sum_{i=1}^{x} i = O(x^2)$$

Two Other Perspectives on Analysis

distance travelled

Charging: to upper bound $y \leq z$, create z dollars

Pay for each part of y w/ z dollars



So start w/ $2x+4x^2$ dollars

"charging"

Each time ant goes $i \rightsquigarrow i \pm 1$, reduce # dollars @ i by 1

$\leq x$ iterations and \$ at each i reduced by $\leq 2/$ iteration

so never run out of \$

$\to$ Distance travelled $\leq$ total # dollars $\leq 2x+4x^2$

Skipped

Potentialing: to upperbound y, create a "potential function"

φ s.t. initially $\varphi = 0$, $\varphi \leq B$ always and as y increases so does φ

$$\varphi_i = \begin{cases} 1 & \text{if ant has ever been to } -i \text{ \underline{and} } i \\ 0 & \text{o/w} \end{cases}$$

$$\varphi := \sum_i \varphi_i$$

$\varphi \leq x$ since if $\varphi \geq x$, ant found cookie

Each iteration φ increases by ≥ 1 and travel $\leq 4x$

So travel $\leq 4x^2$

Strategy when X Not Known'

Until @ cookie

For $x' = 1, 2, 4, 8, 16, \ldots$ (Guess x as powers of 2)

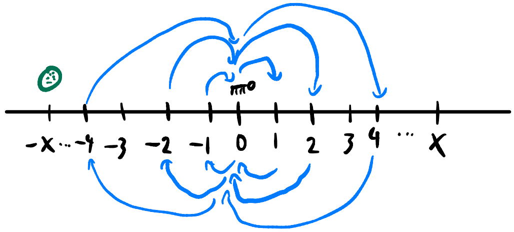
Go to x'

Go to $-x'$

Go to 0

Skipped

Analysis



Ant travels $\leq 1+1+1+1+2+2+2+2+4+4+4+4+\ldots+2x+2x+2x+2x$

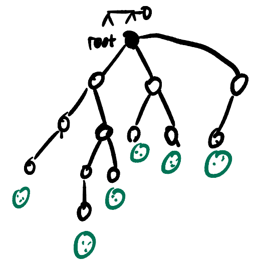
$$= 4\sum_{i=0}^{\log x} 2^i$$

$$= 4\sum_{i=0}^{\log x} 2x/2^i$$

$$= 8x\sum_{i=0}^{\log x} \frac{1}{2^i}$$

$$= O(x)$$

Ant on a Tree



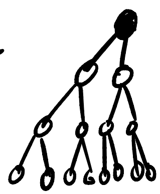
→ Goal: Minimize distance to a cookie @ leaf

Say a rooted tree $T=(V,E)$ is <u>d-nice</u> if

1) Each non-leaf has d children (d+1 regular)

2) Each $u \in V$ at depth i is a leaf iff all depth i vertices are leaves

E.g. 2-nice



level	# nodes	Subtree Size
0	1	n
1	2	$\leq n/2$
2	$2 \cdot 2$	$\leq n/4$
3	$2 \cdot 2^2$	$\leq n/8$

<u>Doubling</u>: If $y \geq 1$, $y \leq B$ and y increasing over time then # times y doubles is $\leq \log B$

<u>Halving</u>: If $y \leq B$, $y \geq 1$ and y decreases over time then # times y halves is $\leq \log B$

Strategy for d-Nice Trees

Until @ Cookie
  Go to arbitrary child

Doubling Analysis

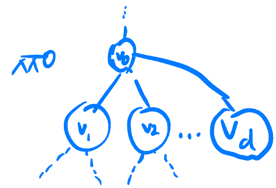
\# nodes at level 0 is 1

\# nodes at level i is d^i

But # nodes $\le n$

So $d^i \le n$ so $i \le \log_d n$

Halving Analysis



Let n_i = # nodes in v_i's subtree

So $n_0 = 1 + \sum_{i=1}^{d} n_i \ge d \cdot n_i$

So $n_i \le n_0 / d \ \forall i \in [d]$

I.e each time go to child reduce subtree size by $1/d$

So after travelling i times

$1 \le$ Subtree Size $\le n \cdot \frac{1}{d^i}$

$i \le \log_d n$

I.e a d-nice tree has depth $\le \log_d n$

Common Corollaries

d:	2	$\log n$	$2^{\sqrt{\log n}}$	$\sqrt{n}$	$n^{\varepsilon}\ \forall \varepsilon > 0$
Depth:	$\log n$	$\frac{\log n}{\log\log n}$	$\sqrt{\log n}$	2	$1/\varepsilon$

Averaging: Given $a_1, a_2, \dots, a_d \in \mathbb{R}$, $\exists a_j$ s.t. $a_j \le \frac{\sum_1^d a_i}{d}$

Strategy for (d+1)-regular Trees

non-leaf deg $= d+1$

Until @ Cookie

Let v_j be argmin n_i of curr. node v_0

v_i child of curr. node

Go to v_j

Analysis

$$n_j \le \frac{\sum_{i=1}^{d} n_i}{d} \le \frac{n_0}{d}$$

→ Each time travel, reduce subtree by $1/d$

→ So travel $\le \log_d n$ times by halving argument

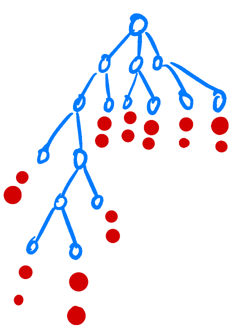
Corrolary: every $(d+1)$-regular tree has a root ⇝ leaf path of length $\le \log_d n$

Skipped

Token-Rearranging: Given objects $o_1, o_2, \dots, o_y$. To bound $y \le z$ start w/ z "tokens" and have objects pass tokens around so each object gets ≥ 1 token

Claim: # nodes in a 3-regular tree w/ ℓ leaves is $\le 2\ell$

Proof Sketch



Place 2 tokens at each leaf (so 2ℓ total)

Max level

To process level $i = D, D-1, \dots, 0$

Each node keeps 1 token

Passes remainder up to parent

Invariant after processing ith level

1) each node on level $\ge i$ has 1 token

2) each node on level $i-1$ received ≥ 1 token/child

Proof of invariant by induction

BC: Trivial

IS: Each node has ≥ 2 children so 1 token kept, ≥ 1 token sent up