Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Loca

here

Outline

- Locality everywhere.

- Locality in Computing
- Local Coloring
- Coloring Trees

- Lower Bounds

Loca here

Locality Everywhere!

- Locality is everywhere:

  - Physics
  - Biology
  - Social Sciences

  - Mathematic

- They have differe

Locality in Physics

- An object is only directly influen iate surroundings.

- A theory using the principle of locality is said to be a "local theory".

- Relativity is a local t
  - It limits the spe el to the speed of light c

- Quantum mechanics is not a local theory.
  - A measurement made on one of a pair of separated but entangled particles causes a simultaneous effect, the collapse of the wave function, in the remote particle (i.e. an effect exceeding the speed of light).

Locality in Biology

- Phenotypes might be influenc tions and effects.

  - Shape
  - Size
  - Color
  - Nature
  - Other enviro

- In turn, this affects the genotype.
- Quantum Biology is a newly developing field for the study of non-local biological phenomena.
  - Bird navigation

Locality in Social Sciences

- Local Characteristics

  - Language
  - Behaviour
  - Culture
  - Food

- Global Phenomena

  - Cascades
  - Rumors

- How do certain events cascade?

Locality in Mathematics

- It has a proximity interpretati

- Related somehow to distance
- Concerns phenomena that are geometrically close to each other.
- Locality is influenced by distance but is not the same thing as location!

Locali

ing

Locality

- Usually it means:
  - the execution of a process de

rocesses.
  - there is no dependency between events that occur far away.
- It has a special role in computing and communication.

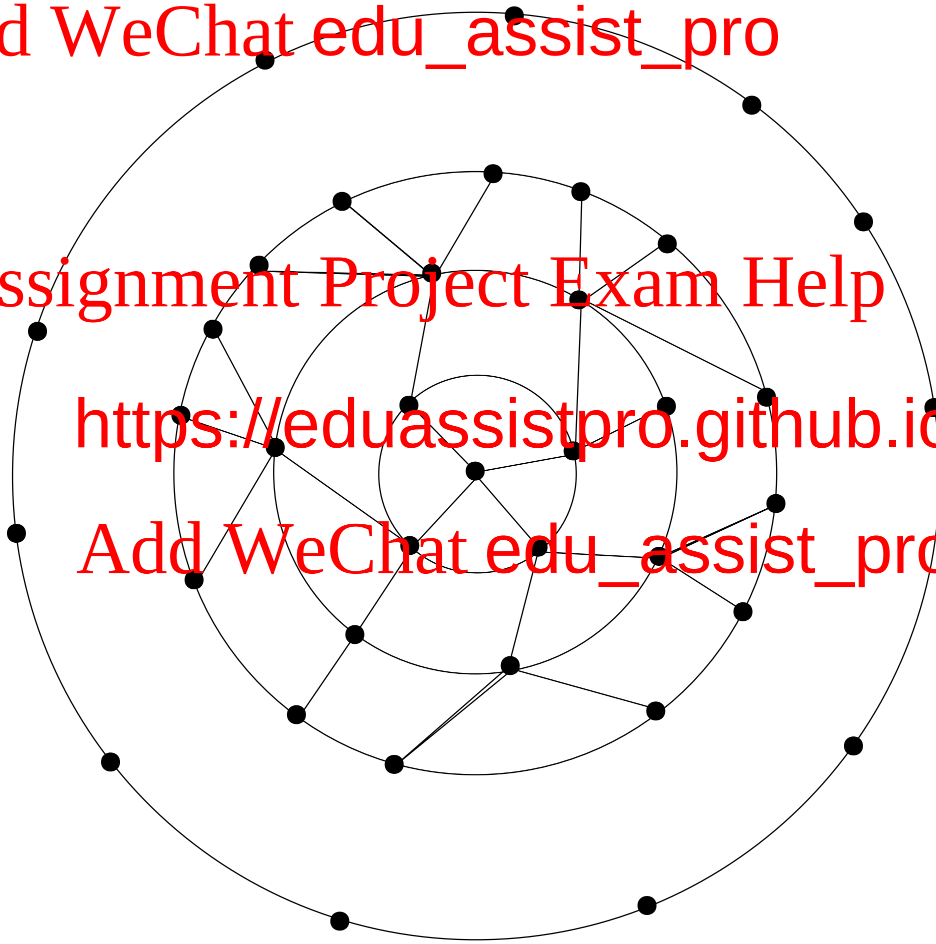
  - What can be co on on how far inform

- Can you elect a leader

  - making use only of local information?

Locality

- Decision made at node u [illegible] odes far away from u.



- How do we quantify "far away" from u?

How far is local?

- Given that locality is influence w far is far away"?
- May depend on the topology

- How do you parametrize locality?
- Best to study specific problems!

Coloring

- Global vs Local Algorithms

- On a Line
- On a Tree

Local Algorithms in DC

- An algorithm is local if messages i odes do not propagate too far from their orig

  – How can you ensure correctness of the algorithm?

  – Which problems can you solve this way?

  – How far is too far?

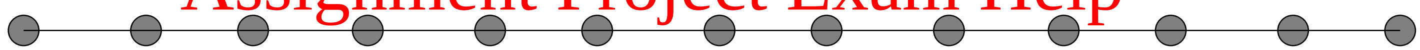
- Local approach is ation!

- Lets go back to coloring.

Coloring

- A vertex coloring is an assignme ices of a graph so that any two adjacent v ned different colors.

- How do you color a set of points on a line?



- If nodes have ident nodes with even identities blue, and with odd identities re

  - Is the algorithm correct?
  - Is this a local algorithm?
  - Is there a local colouring algorithm?

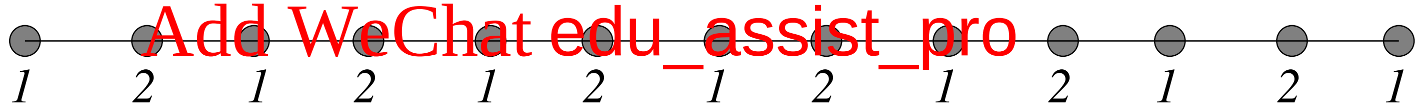
Global vs Local Coloring

- Before a node decides on its colou formation about its neighboring nodes.

- There are two ways to do this depending on how far this information collection can spread
  1. Globally
  2. Locally

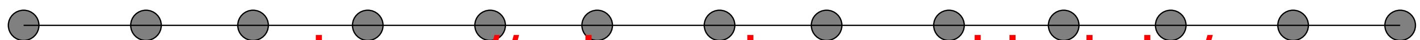
Globally/Locally

- Globally?



  – You are not constrained by # of hops.

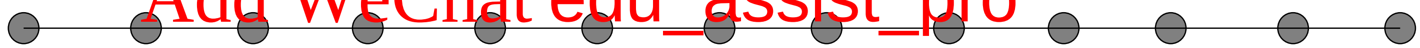
- Locally?



  – Constrained by # of hops.
  – In a distributed setting, the difficult e assignment of colors consistent throughout the graph despite the fact that propagation is limited!

Coloring with Restricted Number of Hops

- Consider nodes “independen oloring.



- If the number of hops a message can propagate is restricted you may not be able to complete the coloring!

- If a given set of nodes s you ensure consist

  – Nodes will start with their own identi

- More than that, you may have to use more than the minimum required number of colors so as to achieve a correct coloring!
- Regardless of the number of colors you use
  – can you achieve a proper coloring, and
  – at the same time restrict the number of hops?

Quantifying Locality: Network

- Consider a class $\mathcal{N}$ of net

- A typical network $G = ($ raph with n vertices.
  - Line,
  - Ring,
  - Tree,
  - etc.

- The concept should be applicable to all cla (networks).

Quantifying Locality: Distance

- Locality should depend on dist

- Let $n \to h(n)$ be an integer val
  - $h(n)$ is the number of hops allowed in a network of size n.
- Examples:

  - $n \to h(n) = 1$,
  - $n \to h(n) =$

  - $n \to h(n) = \log n$,

  - $n \to h(n) = \sqrt{n}$,
  - $n \to h(n) = n$,
  - $n \to h(n) = \log^* n$, etc

Quantifying Locality: Problems

- Consider a problem $\mathcal{P}$ (e nd a class $\mathcal{A}$ of synchronous, distributed al g $\mathcal{P}$ for $\mathcal{N}$.

  – The class $\mathcal{A}$ of distributed algorithms is h-local if during the execution of an algorithm $A \in \mathcal{A}$ on a network $G \in \mathcal{N}$ (on n vertices), a message emanating from a node will never propagate m nator.

Which Problems in DC are Local?

- Not all problems are going to be al, for a given function h.

- Which ones are h-local, fo $n \to h(n) = c$, where c a constant?
  - Leader Election
  - Spanning Tr
  - Maximum In
  - Coloring
  - Minimum Dominating Set

- For which topologies?

L

g

Coloring a Line Graph: Assumptions

- Assume you are on a line of

- To start, assume that each node v has a distinct identity id_v (for example, either their location or the network interface card would do).

  – Identity selec

  problem...besides we also know sev e this problem!

Local Coloring Algorithm

- Our main goal is to show

- **Theorem 1** *There is a color* … *hich can* 3*-color any line in* $O(\log^* n)$ *time, where*
  - $\log^* n$ *is the iterated lograithm of* n
  - *in the algorith* … .

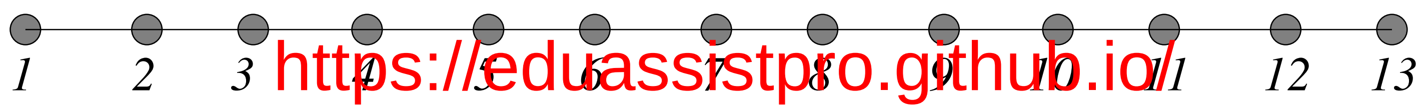
- This result is impor … e wireless) where messages should not pr

- **NB:** Note the important parameters taken into account:
  - Final number of colors in the graph.
  - Termination time of the coloring algorithm.

Assumptions for Coloring

- Let $v \to c_v$ be an arbitrary co ices.
  - Observe that $c_v := id$ g!

- For example,
  - the identity assignment below is a colouring using n colors,



  - and so is any permutation of the identi

Assumptions for Coloring

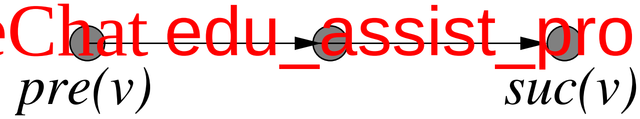
- Represent each c_v as a sequ
  - Let $|c_v|$ be the number of b v, and
  - $c_v(i)$ the i-th bit of c_v.

- **Example:**
  - $c_u = 594 = 512$ ^{9} $^6 + 2^4 + 2^1$.
  - In binary c_u
  - $c_u(i)$ is the ith bit where countin $i = 0$ from left to right: $c_u(0) = 1, c_u(2) = 0$.

- The **concatenation**
  - of two sequences s, s' of bits is the sequence ss'.
  - **Example:** if $s = 1010$ and $s' = 110$ then $ss' = 1010110$

Idea for an Algorithm on a Line

- Assume an ordering of the verti uld do).



- Starting Rule:
  - Start with any legal coloring,
    - for exampl
  - Color "leftmost vertex" with the bit 0

- Any other starting coloring would do.

[a] This is a starting condition and we will need to justify it: will do this later!

Recoloring Rule

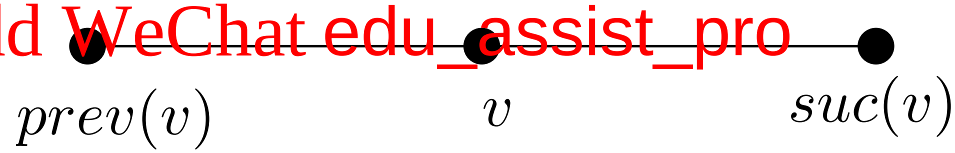
- Since nodes $u \to v$ are nei u preceding v), their current colors must be differen $\neq c_v$.
- Produce a new "legal" coloring for a vertex v from the current one, say c_v, as follows:

  - Find the first index 1 i c_v such that v's color differs from the colou
  - Set new color to "i concaten i)": $c_v \to ic_v(i)$;

- Recoloring rule guarantees that neigh **different** colors.
- **NB:** Bit representation of each new color is of length logarithmic of the length of the previous color!

Coloring Algorithm for Vertex v

- Assume an ordering of the verti uld do).



- **Coloring Algorithm:**

1. $c_v \leftarrow id_v$;
2. **Repeat:**

   (a) $\ell \leftarrow |c_v|$;

   (b) **if** v is “leftmost vertex” then set
   else set $I \leftarrow \min\{i : c_v(i) \neq c_{pre(v)}(i)\}$;

   (c) Set $c_v \leftarrow Ic_v(I)$; /* concatenation */

   (d) Inform the successor $suc(v)$ of v of this choice;

3. **Until** $|c_v| = \ell$; /*Until length does not change */

Example (1/2)

- Given two nodes $u \to v$.

- Lets show how the color of node anges from the old color c_v to a new color c_v.
  - A similar change occurs to the color of u, but this is influenced from the predecessor of u.

- Let their current co $= 631$.

- Convert to binary:

  $c_u = 512 + 64 + 16 + 2 = 2^9 + 2^6 + 2^4 + 2^1$
  $c_v = 512 + 64 + 32 + 16 + 4 + 2 + 1 = 2^9 + 2^6 + 2^5 + 2^4 + 2^2 + 2^1 + 2^0$
- $c_u = 1001010010$ and $c_v = 1001110111$

Example (2/2)

- Consider the two nodes with col

$$c_u = 1001010 \qquad c_v = 1001110111$$

- What is the smallest i such that $c_u(i) \neq c_v(i)$?

- Line up the bits

$$\uparrow$$

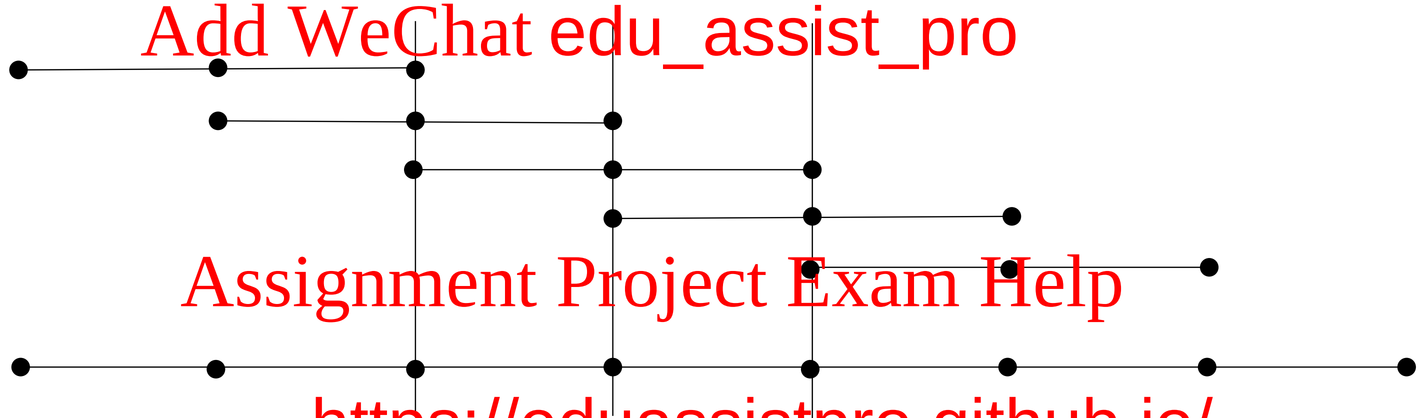
$$1001110111$$

- So $i = 4$ (counting starts from 0); in binary 4 is 100 and the new colour of v in binary representation is

$$ic_v(i) = 1001 = 9$$

Execution of Coloring Algorithm

- A node receives input from its pre . . .

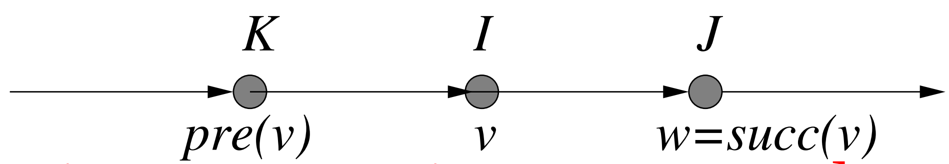


- . . .and provides input to its successor.

Correctness: Legal Coloring (1/2)

- Consider three consecutive n es u, v, w at some iteration of the algorithm with

$v), v = pre(w)$.



- Let I, J be the in p 2(b), respectively.
  - $I := \min\{i :$ $/$ $j : c_v(j) \neq c_w(j)\}$
  - v, w receive the new colours:

$$c_v \leftarrow I c_v(I)$$

and

$$c_w \leftarrow J c_w(J)$$

Correctness: Legal Coloring (2/2)

- We need to show that Ic /).

- There are two cases to consider:

1. If $I \neq J$ then rule 2(b) ensures that the new labels $Ic_v(I), Jc_w(J)$ as defined in 2(c) differ in a bit
   – because I, J do

2. If $I = J$ the s as defined in 2(c) differ in the last bit
   – Recall that $c_u(I) \neq c_v(I)$ an / $_w(J)$

   – Since $I = J$ we have that $c_u(I) \neq c_v(I)$ and $c_v(I) \neq c_w(I)$
   – The new labels for v, w will be $Ic_v(I)$ and $Ic_w(I)$ and by choice of I we have that $c_v(I) \neq c_w(I)$.

Number of Rounds

- At the start, $K_0 = K =$ x number of bits of a node in the original ID coloring.

- Let K_r denote the number of bits in the color representation after the rth iteration.

- Observe that
  - Therefore the og n bits, the third of roughly $\log \log \log$, etc.

- As a matter of fact the "sizes of the colours" shrink very rapidly!
  - The size of the colour (measured in bits) in the new step is the logarithm of the size of the colour in the previous step!

Iterated Logarithm: $\log^*$

- $\log^* n$ is not really a logarithm
  - it is rather the number of itera ction on a number n until it stops having an effect!
- **Log-Star** (in base 2) of n:
  - Is the number o l starting from
- Can be defined in any base! Here we look onl

Definition of $\log^*$

- **Iterated Definition of**

  - $\log^{(1)} n = \log n$, and
  - $\log^{(x+1)} n = \log(\log^{x} n)$, for $x \geq 1$.

  Then $\log^* n =$ first integer x such that $\log^{(x)} n \leq 2$. [a]

- **Recursive defi**

$$\log^* x = \begin{cases} 1 & \text{if} \quad \leq 2 \\ 1 + \log & \quad > 2 \end{cases}$$

[a] $\log^{(x)} n$ should not be confused with $\log^{x} n$: the logarithm to the power x.

Example

- Log-star is a very slowly growin

- Consider the number n

$$\log(2^{2^5}) = 2^5$$

$$\log(2^{2^5}) = 5$$

89

$$\log(2.32) < 2.$$

Hence, $\log^*(2^{2^5}) = 4$.

- Log-star of all the atoms in the observable universe (estimated to be 10^{80}) is 5.

The Starting Nodes: Something Wrong?

- Recall the leftmost node was giv

- It is not clear from the descriptio hy the identities of the nodes "located" at the beginning of the line are reduced to constant size.
  - By beginning we mean the first $O(\log^* n)$ nodes.

- Observe that the id $O(\log^* n)$ are indeed reduced to constant size.

- Can remedy this by adding an additional step at the end of the algorithm:
  - The first $O(\log^* n)$ nodes run a recoloring algorithm to reduce their colors to constant size.
- Note that this step takes additional time $O(\log^* n)$.

Six Coloring in $\log^* n$ Iterations

- If K_i = number of bits in the colo i iterations then
  - $K_{r+1} = \lceil \log K_r \rceil + 1$.
  - $K_{r+1} < K_r$ as long as $K_r \geq 4$.
- In the final iteration r we have that $K_r = K_{r-1} \leq 3$.
- Therefore in the fin
  - at most three ch $r - 1)$-st coloring, and
  - two choices for the value of the bit,

  which gives a total of six colors.
- It turns out,
  - we can improve on # of colors from six to three, but
  - cannot improve on the $\log^* n$.

Three Colors Suffice

- How do we reduce the number of c ... ree?

- Suppose that the algorithm we ... e has colored a line with the six colors $0, 1, 2, 3, 4, 5$ as follows

$$0\ 5\ 4\ 2\ 5\ 3\ 0\ 3\ 1\ 5\ 4\ 2\ 3\ 0\ 1\ 4\ 3\ 2\ 4\ 0\ 1\ 0\ 2\ 4\ 5$$

- How do you color it u ... $1, 2$?

Three Colors Suffice

- Start with the sequence

0 5 4 2 5 3 0 3 1 5 4 2 3 0 1 4 3 2 4 0 1 0 2 4 5

- Eliminate 5: by choosing a color from $0, 1, 2$

0 1 4 2 0 3 0 3 1 0 4 2 3 0 1 4 3 2 4 0 1 0 2 4 0

- Eliminate 4: by cho 2

0 1 0 2 0 3 0 3 1 0 1 2 3 0 1 0 3 2 1 0 1 0 2 1 0

- Eliminate 3: by choosing a color from $0, 1, 2$

0 1 0 2 0 1 0 0 1 0 1 2 1 0 1 0 1 2 1 0 1 0 2 1 0

Coloring Rings

- **Theorem 2** *There is an alg* 3-*color any ring of size* n *in* $\log^* n$ *time.*

- Same algorithm.

C

S

From Lines to Trees

- The line colouring algorithm al !

- The basic assumption is that yo e of the tree designated as the root!
- Further, other nodes must have a parent (i.e., a predecessor)!

- The main theore

6-Coloring Theorem

- **Theorem 3** *There is an alg* 6-*color any tree in* $\log^* n$ *time.*

6-Coloring Algorithm for Trees: Vertex v

- **Algorithm: 6-Color**

1. $c_v \leftarrow id_v$;
2. **Repeat:**
   (a) $\ell \leftarrow |c_v|$;
   (b) **if** v is "the root" then set I 0
   else set I $(i)\}$;
   (c) Set $c_v \leftarrow$ v
   (d) Inform all children of v of t
3. **Until** $|c_v| = \ell$;

- Why is the algorithm correct?

3-Coloring Theorem for Trees

- **Theorem 4** *There is an alg* 3*-color any tree in* $O(\log^* n)$ *time.*

- The reason is that the coloring on the descendants of a given node is independent when done on disjoint paths.

Shift-Down Algorithm

- The color reduction method is c n”.

- $\boxed{\textbf{Algorithm Shift-Down}}$
  1. Concurrently at all vertices:
  2. Recolor each non-root vertex by the color of its parent.
  3. Recolor root by a new color, different from its current one.

- Why is “shift-down” correct?
- Colors (of the original coloring) are shifted down.

Analysis of Shift-Down Algorithm

- **Lemma 1 (Analysis of Algo[illegible]wn)**
  Algorithm Shift-Down preserv[illegible]lity; also siblings are monochromatic.

- Two vertices $v = parent(w), w$ are recolored by $c_{parent(v)}$ and c_v, which are different since c was a legal colouring.
- If $v = root$, then the new colors are x and c_v, where x is some color different from c_v.
- Also, all children of some vertex v get the same new color c_v.

Final Color Reduction

- Now assume the six colors empl

- The final three reduction steps involve cancelling colors

$$3, 4, 5$$

one at a time.

- In the end, there will be three colors left 0 [illegible], 2.
  - This is done by Algorithm Six2Three

Six2Three Algorithm

- $\boxed{\textbf{Algorithm Six2Three}}$

1. **for** $x = 5, 4, 3$ **do** /* C x */
2. Perform subroutine **Shift-Down** on the current colouring;
3. **if** $c_v = x$ **then**
4. v chooses new color $c_v \in \{0, 1, 2\}$ not used by any of the neighbors.
5. **endif**
6. **endfor**

Example of Six2Three

- Recolouring method

- Example discarding color 4.

Analysis of Six2Three

- **Theorem 5 (Analysis of Alg hree)** *Algorithm Six2Three colors a t lors in time* $O(\log^* n)$.

- Each vertex colored x will find an available color from the set $\{1, 2, 3\}$,

  – since by the Shi e colors are occupied, o .

- Now note that recoloring the x s simultaneously creates no problem since they are all mutually nonadjacent.

Optimality

- Fast tree-coloring with only 2 c xponentially more expensive than coloring

  - In a tree degenerated to a line, nodes far away need to figure out whether they are an even or odd number of hops away from each other in order to get a 2-coloring.

  - To do that one h s costs time line

Assignment Project Exam Help

L https://eduassistpro.github.io/ **s**

Add WeChat edu_assist_pro

Can anything be better than $\log^* n$?

- The only thing better than unning time is $O(1)$ running time!

  - A 2-coloring is possible with $O(1)$ running time in a distributed system with GPS!

- It turns out that we can prove a lower bound of $\Omega(\log^* n)$ on the time required t ing) by three colors.

  - This implies a tight bound of $\Theta(\log$ e time required for 3-coloring the line (ring).

$\Omega(\log^* n)$ Lower Bound

- **Theorem 6** *Every deter* *uted algorithm to color a directed ring with 3 or fe* *t least* $(\log^* n)/2 - 1$ *rounds.*

- The proof uses a theorem of Frank P. Ramsey.

(22 February 1903 19 January 1930).

- We will not prove Theorem 6 here.

Generalizations and Additional Results

- Linial (1992) proves that

  - in rooted d-regular tree s r, any synchronous distributed algorithm running in time $\leq \frac{2}{3}r$ cannot color $T_{d,r}$ by fewer than $\frac{1}{2}\sqrt{d}$ colors.

  - an arbitrary graph G of order n and max degree Δ, can be colored with 5 istributively.

  - for G labeled, in time $O(\log$ olor G with $O(\Delta^2)$ colors in a distributive gorithm.

- There exists a deterministic distributed algorithm for coloring arbitrary graphs with max degree Δ;
  - can be colored with $\Delta + 1$ colors in $O(\Delta \log^* n)$ time.

Exercises[a]

1. For any graph $G = (V, E$ ... atic numbers

$$\chi_{centralized}(\quad (G), \chi_{local}(G)$$

   for centralized, distributed, and local computation.

   (a) How do they differ?

   (b) Is there a natural or

2. Define the concepts of ... l for any algorithmic computation and make a co

3. Let $n \to h(n)$ be an integer valued function, where $h(n)$ is the number of hops allowed in a network of size n to complete the computation. Formulate the various types of computation discussed above in terms of the function $h(n)$.

4. **($\star\star$)** Consider Exercise 3. If $h(n) = n$ then the number of

[a] Do not submit!

colors is 2. If $h(n) = 1$ then the number of colors is 3. For which threshold value of $h(n)$ does the number of colors jumps from 2 to 3?

5. Compute $\log^*(10^{1000})$.

6. Compute $\log^*(2^{2^{2^{16}}})$.

7. Explain in more detai at the local coloring al $\to$ three reductions) reduces to a six coloring.

8. Show in detail that on the line graph three colors suffice.

9. Prove that a $\log^*$ coloring algorithm is possible on a ring. How many colors does it require?

10. Prove in detail the correctness of the $\log^*$ tree coloring algorithm.

Sources

- L. Barenboim, and M. Elkin. Dis coloring: Fundamentals and recent de nthesis Lectures on Distributed Computing Theory 4.1 (2013): 1-171.

- N. Linial. Locality in distributed graph algorithms. SIAM Journal on Computing 21.1 (1992): 193-201.

- D. Peleg, Distrib ve Approach, SIAM, 2000.