Roadmap

- Deterministic $\Omega(\log^\star n)$ lower bound for 3-coloring C_n
- Randomized algorithms for:
  - 3-coloring C_n
  - $(\Delta + 1)$-coloring arbitrary graph of max degree Δ

Lower Bound 3-Coloring C_n

- **Theorem** [Linial 1992] Any deterministic algorithm for computing a 3-coloring of the n-node cycle C_n with IDs in $[1,n]$ takes at least $1/2 \cdot \log^\star n - 1$ rounds.

configuration graph

- Linial's original proof:
  - C_n can be c-colored in t rounds $\Longrightarrow \chi(G_{n,t}) \leq c$
  - C_n can be c-colored in t rounds $\Longrightarrow$ C_n can be 2^{2^c}-colored in $t-1$ rounds

- We present a direct proof by Laurinharju & Suomela (2014)

Proof

Definition $\mathscr{A}$ is a k-ary c-coloring function if

- For all $1 \leq x_1 < x_2 < \ldots < x_k \leq n$, $\mathscr{A}(x_1, \ldots, x_k) \in \{1, \ldots, c\}$
- For all $1 \leq x_1 < x_2 < \ldots < x_k < x_{k+1} \leq n$, $\mathscr{A}(x_1, \ldots, x_k) \neq \mathscr{A}(x_2, \ldots, x_{k+1})$

Claim 1: t-tound algorithm $\mathscr{A}$ for 3-coloring C_n

➥ $\mathscr{A}$ is $(2t + 1)$-ary 3-coloring function

Claim 2. If $\mathscr{A}$ is a 1-ary c-coloring function then $c \geq n$.

Claim 3. If $\mathscr{A}$ is a k-ary c-coloring function, then there is a $(k-1)$-ary 2^c-colouring function $\mathscr{B}$.

Proof: The following function is a 2^c-colouring function:

$$\mathscr{B}(x_1, \ldots, x_{k-1}) = \{\mathscr{A}(x_1, \ldots, x_{k-1}, x_k) : x_k > x_{k-1}\}$$

For contradiction, let $1 \leq x_1^* < \ldots < x_k^* \leq n$ with

$$\mathscr{B}(x_1^*, \ldots, x_{k-1}^*) = \mathscr{B}(x_2^*, \ldots, x_k^*)$$

Let $d = \mathscr{A}(x_1^*, \ldots, x_k^*)$.

- ➥ $d \in \mathscr{B}(x_1^*, \ldots, x_{k-1}^*) \Longrightarrow d \in \mathscr{B}(x_2^*, \ldots, x_k^*)$
- ➥ $\exists x_{k+1}^* > x_k^* : d = \mathscr{A}(x_2^*, \ldots, x_{k+1}^*)$
- ➥ $\mathscr{A}$ is not proper. ∎

Let $\mathcal{A}$ be a t-tound algorithm for 3-coloring C_n

$\Rightarrow$ $\mathcal{A}$ is a $(2t+1)$-ary 3-coloring function *(by Claim 1)*

$\Rightarrow$ $\exists$ a $(2t)$-ary 2^3-coloring function *(by Claim 3)*

$\Rightarrow$ $\exists$ a $(2t-1)$-ary 2^{2^3}-coloring function *(by Claim 3)*

$\Rightarrow$ $\exists$ a $(2t-2)$-ary $2^{2^{2^3}}$-coloring function *(by Claim 3)*

$\vdots$

$\Rightarrow$ $\exists$ a 1-ary ${\scriptstyle 2t-1}\left\updownarrow 2^{2^{\cdot^{\cdot^{\cdot^{2^3}}}}}\right.$ -coloring function *(by Claim 3)*

$\Rightarrow$ $2^{2^{\cdot^{\cdot^{\cdot^{2^3}}}}} \geq n$ *(by Claim 2)*

$\Rightarrow$ t ≥ ½ log*n - 1. ❑

Randomized Algorithms

Elementary Randomized 3-Coloring of C_n

- MyFinalColor ← $\bot$
- Repeat
    - MyProposedColor ← color in $\{1,2,3\}$ uniformly at random
    - **Send** MyProposedColor to neighbors
    - **Receive** ProposedColors from neighbors
    - if MyProposedColor is different from the FinalColors and ProposedColors of both neighbors then MyFinalColor ← MyProposedColor
    - **Send** MyFinalColor to neighbors
    - **Receive** FinalColors from neighbors
- Until MyFinalColor $\neq \bot$

Claim This (Las Vegas) algorithm runs in $O(\log n)$ rounds w.h.p.

- A **Las Vegas algorithm** is a randomized algorithm that always gives the correct output but whose running time is a random variable.

$$\Pr[\text{running time} \leq T] \geq 1 - \epsilon$$

- A **Monte Carlo algorithm** is a randomized algorithms whose running time is deterministic, but whose output may be incorrect with a certain, typically small, probability.

$$\Pr[\text{error after time } T] \leq \epsilon$$

Definition A sequence $(\mathcal{E}_n)_{n\geq 1}$ of events holds with high probability (w.h.p.) whenever $\Pr[\mathcal{E}_n] = 1 - O(1/n^c)$ for some constant $c > 0$ (typically $c = 1$).

Elements of probability:

Recall:

« A given B holds » or
« A conditioned to B »

- $\Pr[A|B] = \Pr[A \wedge B] / \Pr[B] \Rightarrow \Pr[A \wedge B] = \Pr[A|B] \cdot \Pr[B]$

and

A and B independent
$\Leftrightarrow \Pr[A \wedge B] = \Pr[A] \cdot \Pr[B]$

- $\Pr[A] = \Pr[A|B] \cdot \Pr[B] + \Pr[A|\neg B] \cdot \Pr[\neg B]$

- Union bound: $\Pr[A \vee B] \leq \Pr[A] + \Pr[B]$

$\Pr[\exists\ s \in S : s \models \mathcal{P}] = \Pr[(s_1 \models \mathcal{P}) \vee (s_2 \models \mathcal{P}) \vee \ldots \vee (s_m \models \mathcal{P})]$

or

Claim The elementary (Las Vegas) algorithm runs in $O(\log n)$ rounds w.h.p.

Proof At every execution of the repeat loop, for every fixed node u,

$$\Pr[u \text{ terminates}] = \Pr[X \notin \{X_{-1}, X_{+1}\}] \geq \frac{1}{3}$$

Note: At first execution of the repeat loop:

$$\Pr[u \text{ terminates}] = \sum_{x \in \{1,2,3\}} \Pr[(X_{-1} \neq x) \wedge (X_{+1} \neq x)] \cdot \Pr[X = x] \geq \frac{4}{9}$$

$$\Longrightarrow \Pr[u \text{ does not terminates after } k \text{ rounds}] \leq \left(\frac{2}{3}\right)^k$$

$$\Longrightarrow \Pr[u \text{ does not terminates after } c \log_{3/2} n \text{ rounds}] \leq \frac{1}{n^c}$$

$$\Longrightarrow \Pr[\text{some } u \text{ does not terminate after } c \log_{3/2} n \text{ rounds} \leq \frac{1}{n^{c-1}}$$

$$\Longrightarrow \Pr[\text{every node } u \text{ terminates after } c \log_{3/2} n \text{ rounds} \geq 1 - \frac{1}{n^{c-1}}$$

❑

Randomized (Δ+1)-coloring

- Assume each node picks colors in $\{1,\ldots,\Delta+1\}$ u.a.r.
- For every neighbor v of u we have $\Pr[c(u)=c(v)]=1/(\Delta+1)$
- Thus $\Pr[\exists v\in N(u):c(u)=c(v)]\leq\Delta/(\Delta+1)$
- If $\Delta=O(1)$ then each node terminates with constant probability, but not if $\Delta=\omega(1)$ (i.e., depends on n)
- There is however a simple trick resolving this issue

Randomized $(\Delta + 1)$-coloring in $O(\log n)$ rounds

Algorithm (Barenboim and Elkin, 2013) for node u

while uncolored **do**

$\mathscr{C}$ = {colors previously adopted by neighbors}

pick ℓ(u) at random in {0,1,…,Δ+1} - $\mathscr{C}$

- 0 is picked w/ probability ½
- ℓ(u) ∈ {1,…,Δ+1} - $\mathscr{C}$ is picket w/ proba 1/(2(Δ+1-|$\mathscr{C}$|))

if ℓ(u) ≠ 0 **and** ℓ(u) ∉ {colors picked by neighbors}

then adopt ℓ(u) as my color

else remain uncolored

inform neighbors of status

1 round

1 round

Theorem (Barenboim and Elkin, 2013) The (Δ+1)-coloring algorithm takes, w.h.p., O(log n) rounds.

Claim For every node u, at any round, Pr[u terminates] ≥ ¼

$$\begin{aligned}
\Pr[u \text{ termine}] &= \Pr[\ell(u) \neq 0 \text{ et aucun } v \in N(u) \text{ satisfait } \ell(v) = \ell(u)] \\
&= \Pr[\forall v \in N(u), \ell(v) \neq \ell(u) \mid \ell(u) \neq 0] \cdot \Pr[\ell(u) \neq 0] \\
&= \frac{1}{2} \cdot \Pr[\forall v \in N(u), \ell(v) \neq \ell(u) \mid \ell(u) \neq 0]
\end{aligned}$$

$$\begin{aligned}
\Pr[\ell(v) = \ell(u) \mid \ell(u) \neq 0] &= \Pr[\ell(v) = \ell(u) \mid \ell(u) \neq 0 \wedge \ell(v) = 0] \ \Pr[\ell(v) = 0] \\
&\quad + \Pr[\ell(v) = \ell(u) \mid \ell(u) \neq 0 \wedge \ell(v) \neq 0] \ \Pr[\ell(v) \neq 0] \\
&= \Pr[\ell(v) = \ell(u) \mid \ell(u) \neq 0 \wedge \ell(v) \neq 0] \ \Pr[\ell(v) \neq 0] \\
&\leq \frac{1}{2} \Pr[\ell(v) = \ell(u) \mid \ell(u) \neq 0 \wedge \ell(v) \neq 0] \\
&= \frac{1}{2} \frac{1}{\Delta + 1 - |C(u)|}.
\end{aligned}$$

Error: should be ≤ because equality holds only if $\ell(v) \in C(u)$ If $\ell(v) \notin C(u)$. then proba = 0

$$\Pr[\exists v \in N(u) : \ell(v) = \ell(u) \mid \ell(u) \neq 0] \leq (\Delta - |C(u)|) \frac{1}{2(\Delta + 1 - |C(u)|)} < \frac{1}{2} \quad \blacksquare$$

O(log n) rounds w.h.p.

Pr[u does not terminate in $k\ln(n)$ rounds]

$$\leq (3/4)^{k\ln(n)} = n^{-k\ln(4/3)}$$

Pr[$\exists u$ that does not terminate in $k\ln(n)$ rounds] $\leq n^{1-k\ln(4/3)}$

Let $c > 1$, by choosing $k = \dfrac{1+c}{\ln(4/3)}$, we get:

Pr[all nodes terminates after $\dfrac{1+c}{\ln(4/3)}\ln(n)$ rounds] $\geq 1 - 1/n^c$

❏

LOCAL Model & LCL Problems

LOCAL Model

- Each process is located at a node of a network modeled as an n-node graph (n = #processes)

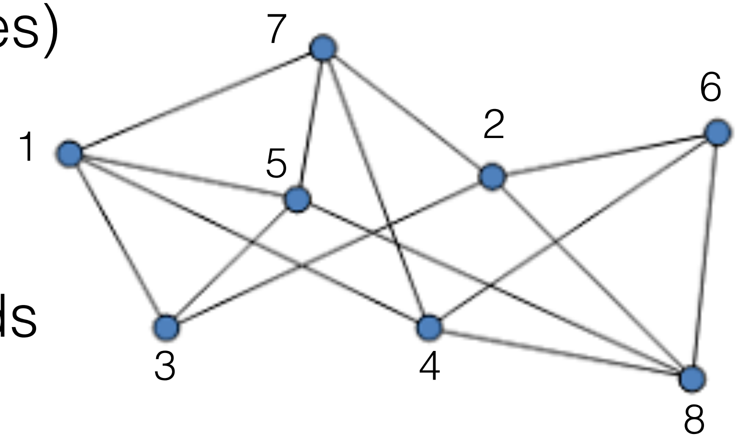
- Each process has a unique ID in $\{1, \ldots, n\}$

- Computation proceeds in synchronous rounds during which every process:

    1. Sends a message to each neighbor

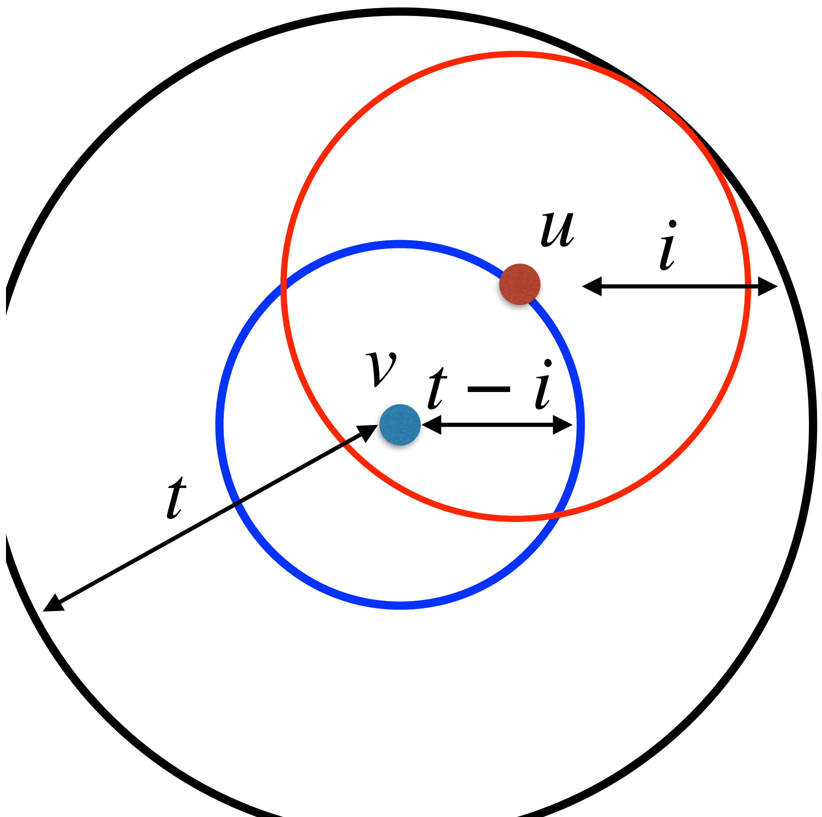
    2. Receives a message from each neighbor

    3. Performs individual computation (same algorithm for all nodes)



Complexity = #rounds

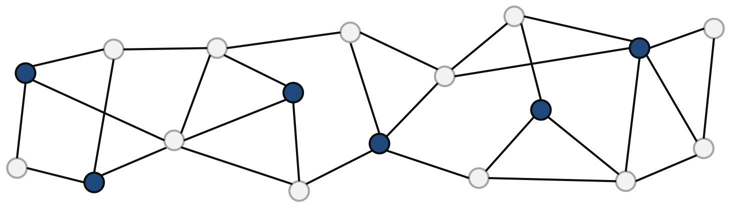
Lemma If a problem P can be solved in t rounds in the LOCAL model by an algorithm A, then there is a t-round algorithm B solving P in which every node proceeds in two phases: (1) Gather all data in the t-ball around it; (2) Simulate and compute the solution.



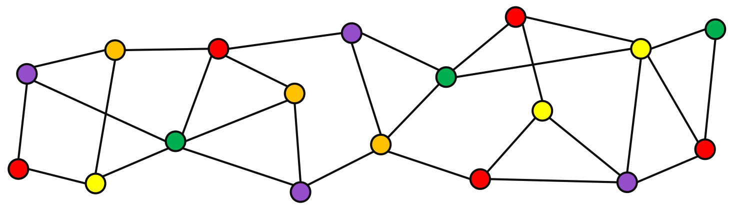
For every $i = 1, \ldots, t$
it suffices for node v to simulate the i-th round of all nodes in $B_G(v, t-i)$
$= \{u \in V(G) \mid \mathrm{dist}_G(u, v) \leq t - i\}$

Four classical problems

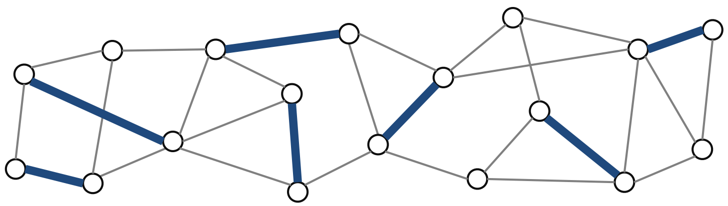
MIS



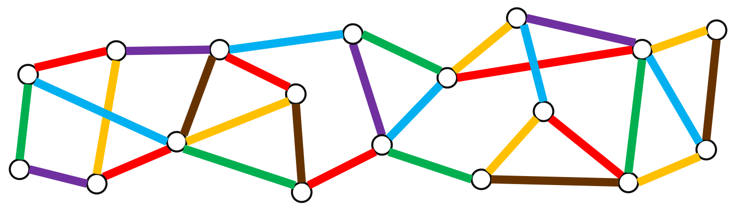
-Vertex Coloring



Maximal Matching



-Edge Coloring



End Lecture 3