• Weeks 1–2: **informal introduction**
  • network = path

• **Week 3:** **graph theory**

• **Weeks 4–7:** **models of computing**
  • what can be computed (efficiently)?

• **Weeks 8–11:** **lower bounds**
  • what cannot be computed (efficiently)?

• **Week 12:** **recap**
Week 7

– Randomised algorithms
Deterministic algorithms

- $\text{init}_d(\ldots)$: state
- $\text{send}_d(\ldots)$: message vector
- $\text{receive}_d(\ldots)$: state
Randomised algorithms

- $\text{init}_d(...)$: probability distribution over states
- $\text{send}_d(...)$: message vector
- $\text{receive}_d(...)$: probability distribution over states
Randomised algorithms

• You can always toss coins when you pick the new state
Randomised algorithms

- Randomised algorithm in PN model
- Randomised algorithm in LOCAL model
- Randomised algorithm in CONGEST model
Uses of randomness

- Break symmetry
- Similar to unique identifiers
Uses of randomness

• Better than unique identifiers: worst-case inputs unlikely?

VS.

1 2 3 4 5 6 7 8

84 56 4 91 35 99 29 95
Guarantees

- **Monte Carlo**: always fast
  - running time deterministic
  - quality of output probabilistic

- **Las Vegas**: always correct
  - running time probabilistic
  - quality of output deterministic
Monte Carlo

• Example: large independent set

• Pick random values, local maxima join
Monte Carlo

• Running time always $O(1)$

• Size of the set depends on random values
Las Vegas

- Pick random values, \textit{local maxima} join,
...

\begin{center}
\begin{tikzpicture}
  \node (n1) at (0,0) {98};
  \node (n2) at (1,0) {53};
  \node (n3) at (2,0) {30};
  \node (n4) at (3,0) {16};
  \node (n5) at (4,0) {25};
  \node (n6) at (5,0) {65};
  \node (n7) at (6,0) {10};
  \node (n8) at (7,0) {4};

  \node (m1) at (0,-1) {46};
  \node (m2) at (1,-1) {29};

  \draw (n1) -- (n2);
  \draw (n2) -- (n3);
  \draw (n3) -- (n4);
  \draw (n4) -- (n5);
  \draw (n5) -- (n6);
  \draw (n6) -- (n7);
  \draw (n7) -- (n8);

  \draw (m1) -- (m2);
  \draw (m2) -- (n4);
\end{tikzpicture}
\end{center}
Las Vegas

• Pick random values, local maxima join, repeat until maximal
Las Vegas

- Output is always maximal independent set, running time probabilistic
With high probability

- Success probability $1 - \frac{1}{n^c}$
  - I can choose any constant $c$
- “algorithm $A$ stops in time $O(\log n)$ with high probability”
- “running time is $O(\log n)$ w.h.p.”
Example: Graph colouring

• Chapter 5: deterministic algorithm, $(\Delta + 1)$-colouring in $O(\Delta^2 + \log^* n)$ rounds

• Today: randomised algorithm, $(\Delta + 1)$-colouring in $O(\log n)$ rounds w.h.p.
Algorithm idea

- Colour palette: \{1, 2, \ldots, \Delta + 1\}
- Pick a random colour
- Try again if conflicts...
Algorithm idea

- Colour palette: \(\{1, 2, \ldots, \Delta + 1\}\)
- Pick a random colour
- Try again if conflicts…
Algorithm idea

- Colour palette: \(\{1, 2, \ldots, \Delta + 1\}\)
- Pick a random colour
- Try again if conflicts…
Algorithm idea

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• Pick a random colour

• Try again if conflicts…
Algorithm idea

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• Pick a random colour

• Try again if conflicts…
Algorithm idea

• Colour palette: \(\{1, 2, \ldots, \Delta + 1\}\)

• Pick a random colour

• Try again if conflicts…
Algorithm idea

- Colour palette: \{1, 2, \ldots, \Delta + 1\}
- Pick a random colour
- Try again if conflicts…
Algorithm idea

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• Pick a random colour

• Try again if conflicts…
Algorithm idea

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• Pick a random colour

• Try again if conflicts…
Algorithm idea

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• Pick a random colour

• Try again if conflicts…
Algorithm idea

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• Pick a random colour

• Try again if conflicts…

\[ \text{Diagram: } \begin{array}{c}
1 \\
3 \\
\downarrow \leftrightarrow \uparrow \\
2 \\
\downarrow \leftrightarrow \uparrow \\
4 \\
\end{array} \]
Algorithm idea

• Colour palette: \(\{1, 2, \ldots, \Delta + 1\}\)

• Pick a random colour

• Try again if conflicts…
Algorithm idea

- Colour palette: \{1, 2, \ldots, \Delta + 1\}
- Pick a random colour
- Try again if conflicts…
Algorithm idea 2

- Colour palette: \{1, 2, \ldots, \Delta + 1\}
- Pick a random \textit{free} colour
  - not used by any neighbour that has stopped
- Try again if conflicts…
Algorithm idea 3

• Colour palette: \{1, 2, \ldots, \Delta + 1\}

• **Active with probability** 1/2

• If *active*, pick a random *free* colour
  • not used by any neighbour that has stopped

• **Try again if conflicts**...
Algorithm idea 3

• Active with probability 1/2

• Intuition:
  • assume: \(d\) neighbours still running
  • roughly \(d/2\) of them active
  • at least \(d + 1\) colours in my palette
  • easy to pick a colour without conflicts (?)
• $s = 1$, $c \neq \bot$:
  • stopping state; output $c$

• $s = 1$, $c = \bot$:
  • probability $1/2$: $c \leftarrow \bot$
  • probability $1/2$: $c \leftarrow \text{random free colour}$
  • $s \leftarrow 0$

• $s = 0$:
  • if conflicts: $c \leftarrow \bot$
  • $s \leftarrow 1$
Algorithm DBRand: Randomised colouring

- **Lemma 1:** A running node succeeds with probability $1/4$
Algorithm DBRand: Randomised colouring

- **Lemma 1:** A running node succeeds with probability $1/4$

- $T(n) = 2(c + 1) \log_{4/3} n = O(\log n)$

- **Lemma 2:** For any $v$, node $v$ stops in time $T(n)$ with probability $1 - \frac{1}{n^{c+1}}$
Algorithm DBRand: Randomised colouring

- $T(n) = 2(c + 1) \log_{4/3} n = O(\log n)$

- **Lemma 2:** For any $v$, node $v$ stops in time $T(n)$ with probability $1 - \frac{1}{n^{c+1}}$

- **Theorem 3:** All nodes stop in time $T(n)$ with probability $1 - \frac{1}{n^{c}}$
Summary

• Randomness may help

• Common idea: each node makes random trials until successful

• Typical running time: $O(\log n)$ w.h.p.
  • proof technique: in each round, each node successful with some constant probability
• **Weeks 1–2:** informal introduction
  • network = path

• **Week 3:** graph theory

• **Weeks 4–7:** models of computing
  • what can be computed (efficiently)?

• **Weeks 8–11:** lower bounds
  • what cannot be computed (efficiently)?

• **Week 12:** recap