Distributed Algorithms 2021

Randomized algorithms
Recap

• Deterministic algorithms in PN model
  • $init_d(...), send_d(...), receive_d(...)$

• Deterministic algorithms in LOCAL model
  • add *unique identifiers*

• Deterministic algorithms in CONGEST model
  • add *bandwidth constraints*
Randomized algorithms

- Randomized algorithms in PN model
  - $\text{init}_a(...), \text{receive}_a(...)$: probability distribution

- Randomized algorithms in LOCAL model
  - add unique identifiers

- Randomized algorithms in CONGEST model
  - add bandwidth constraints
Guarantees

• Monte Carlo
  • guaranteed running time
  • probabilistic output quality

• Las Vegas
  • probabilistic running time
  • guaranteed output quality
Guarantees

• Monte Carlo
  • \textit{guaranteed} running time
  • probabilistic output quality

• Las Vegas
  • probabilistic running time
  • \textit{guaranteed} output quality

• “With high probability” (w.h.p.)
Role of randomness

• Sometimes randomness is the only way to design fast distributed algorithms

• Example: **sinkless orientation**
  • deterministic LOCAL: $O(\log n)$ is best possible
  • randomized LOCAL: $O(\log \log n)$ w.h.p.
    is best possible
Role of randomness

• Sometimes randomness is just one of many ways to break symmetry

• Example:
  • PN model + randomness + knowledge of $n$: you can construct unique identifiers w.h.p.
Quiz
This week’s quiz

- Random permutation of \{1, \ldots, 10\} in a 10-cycle
- Expected number of local maxima?
Video
Pretty simple idea:

• nodes are active with probability 1/2
• only active nodes try to pick a random free color
• stop if successful
Simplest possible idea:

• everyone tries to pick a *random free color*
• stop if successful
Exam
Exam

• **Take-home exam**
  - googling fine, asking someone for help not published ≥ 24h before exam ends
  - submit answers in MyCourses

• **Grading:** pass/fail
  - or pass/borderline/fail if needed
  - borderline can be upgraded to pass with some extra homework
Exam

• Expected:
  • you know *exactly what is a distributed algorithm* (formally, not just waving hands)
  • you can *design* new distributed algorithms
  • you can *analyze* distributed algorithms, with the help of usual graph-theoretic concepts

• Not needed:
  • memorizing technical details
What next?
What’s coming next?

• 1st period:
  • models of distributed computing
  • how to design fast distributed algorithms?

• 2nd period:
  • how to prove impossibility results?
  • what cannot be solved at all in the PN model?
  • what cannot be solved fast in the LOCAL model?