CS 580
ALGORITHM DESIGN AND ANALYSIS

Randomized Algorithms 1

Vassilis Zikas

CONTENTION RESOLUTION

- n processes $P_1, \ldots, P_n$ competing for access to a single database
- Time is divided into discrete rounds
- Database can be accessed by at most one process at a time
- Processes *cannot* communicate with each other
- How can they "take turns" accessing the database?

CONTENTION RESOLUTION

- Simple protocol for process i:
  - Attempt to access the database with probability p (independently) in each round
    - p TBD

- Trivial to state (and implement)
- Hard (interesting) to analyze

CONTENTION RESOLUTION

- Step 1: Define relevant events!
- $A[i,t]$ = event that process i attempts to access the database in round t
  - $\Pr\left[A[i,t]\right] = p$ by definition
- Complementary event $\overline{A[i,t]}$
  - $\Pr\left[\overline{A[i,t]}\right] = 1 - p$
- $S[i,t]$ = event that process i succeeds in accessing database in round t
  - $\Pr\left[S[i,t]\right] = Pr[$ i is the only one who attempts to access the database at step $t]$
  - $=\Pr[A[i,t] \cap \left(\cap_{j\neq i} \overline{A[j,t]}\right)]$

CONTENTION RESOLUTION

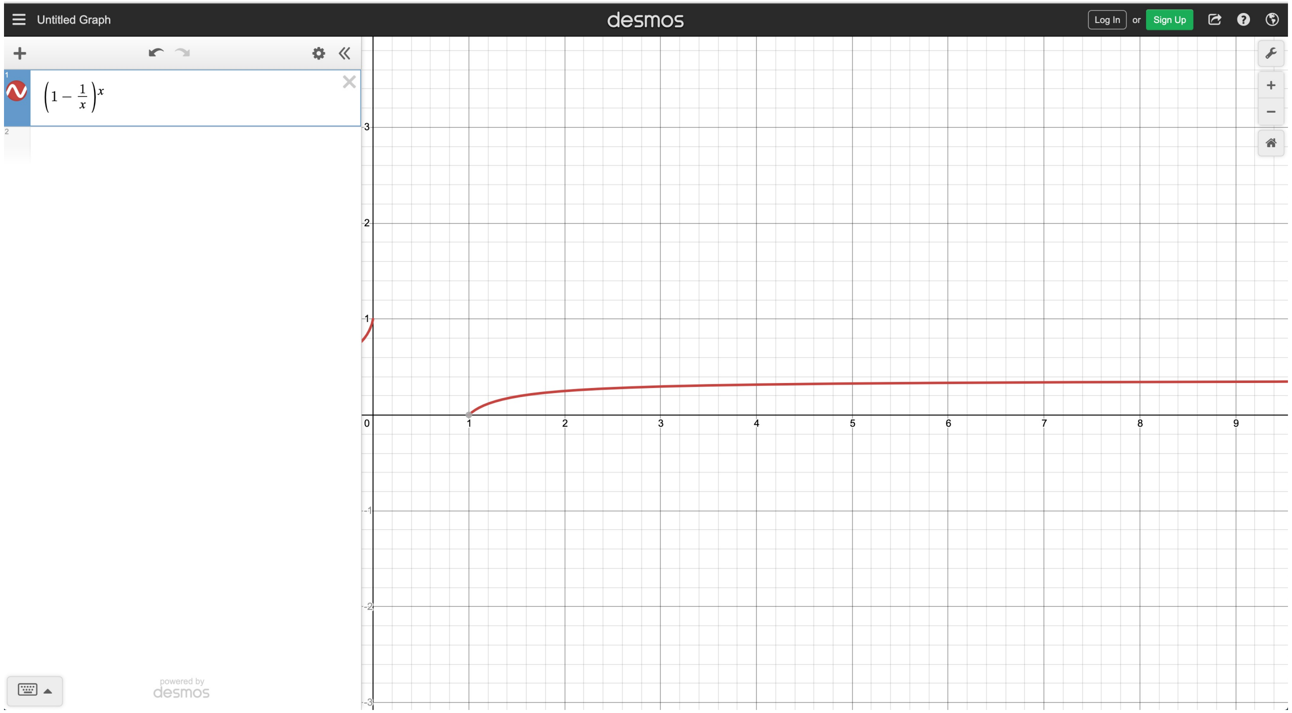
- $\Pr[S[i,t]] = \Pr\big[A[i,t]\big] \cdot \prod_{j \neq i} \Pr\big[\overline{A[j,t]}\big]$
- $\qquad\qquad\qquad = p \cdot (1-p)^{n-1}$
- Closed form!
- Now, we choose p so that the function $f(p) = p \cdot (1-p)^{n-1}$ is maximized
  - Sanity check $f(0) = f(1) = 0$

CONTENTION RESOLUTION

- $f(p) = p \cdot (1-p)^{n-1}$
- $f'(p) = (1-p)^{n-1} - p(n-1)(1-p)^{n-2}$
- $f'(p) = 0$ for $p = 1/n$ is the unique root in $(0,1)$
- Fix $p = 1/n$
- What is $\Pr[S[i,t]]$?
  - $\Pr\left[S[i,t]\right] = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}$

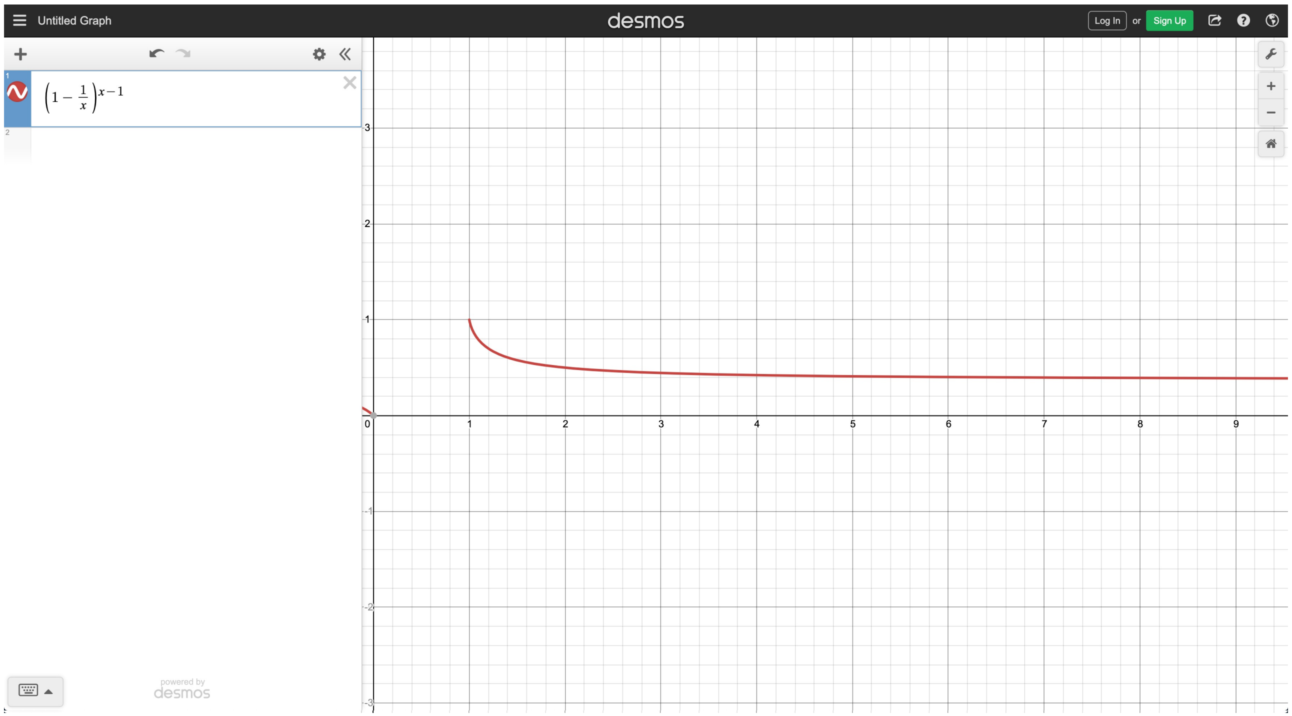
CONTENTION RESOLUTION

- What does $\left(1-\frac{1}{x}\right)^x$ behave like?
    - From $x = 2$ to ∞ it goes from ¼ to $1/e$



CONTENTION RESOLUTION

- What does $\left(1-\frac{1}{x}\right)^{x-1}$ behave like?
  - From $x = 2$ to ∞ it goes from $1/2$ to $1/e$



CONTENTION RESOLUTION

- $\Pr\left[S[i,t]\right] = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}$
- So, probability that i accesses in any given round t is small (gets smaller and smaller as n grows)
- What about probability of accessing in a window of rounds?
- $F[i,t]$ = event that i fails to access in rounds 1 through t
- $\Pr\left[F[i,t]\right] = \Pr\left[\cap_{r=1}^{t} \overline{S[i,r]}\right]$
- $\qquad\qquad\qquad = \prod_{r=1}^{t} \Pr[\overline{S[i,r]}]$
- $\qquad\qquad\qquad = \left(1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}\right)^{t}$
- Ok, let's take a step back…

CONTENTION RESOLUTION

- $\Pr\left[S[i,t]\right] = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}$ is at most $\frac{1}{2n}$ and at least $\frac{1}{en}$
- $\Pr\left[\overline{S[i,t]}\right] \leq 1 - \frac{1}{en}$
- $\Pr\left[F[i,t]\right] = \prod_{r=1}^{t} \Pr[\overline{S[i,r]}] \leq \left(1 - \frac{1}{en}\right)^{t}$
- Setting $t = en$ (or $\lceil en \rceil$ if you want to get technical) we can get something we know!
- $\Pr\left[F[i,en]\right] \leq \left(1 - \frac{1}{en}\right)^{en} \leq \frac{1}{e}$
  - The probability that i is successful in the first en rounds is at least $1 - \frac{1}{e} \approx 0.63$

CONTENTION RESOLUTION

- $\Pr\left[F[i,t]\right] = \prod_{r=1}^{t} \Pr[\overline{S[i,t]}] \le \left(1 - \frac{1}{en}\right)^t$
- Setting $t = \lceil en \rceil \cdot c\ln(n)$ we have
- $\Pr\left[F[i,t]\right] \le \left(1 - \frac{1}{en}\right)^{\lceil en \rceil \cdot c\ln(n)} \le e^{-c\ln(n)} = n^{-c}$
- Overall:
  - The probability that i fails in the first en rounds is at most a constant
  - The probability that i keeps failing much longer is tiny

CONTENTION RESOLUTION

- What about the time for everyone to get access?
- The protocol fails after round t if some process hasn't accessed the database in the first t rounds
- F_t = event that the protocol fails after round t
- $F_t = \cup_{i=1}^{n} F[i,t]$
- $Pr[F_t]$??
- Union bound!
- $\Pr[F_t] \leq \sum_{i=1}^{n} \Pr[F[i,t]]$
- $\leq n \cdot n^{-c}$, by picking $t = \lceil en \rceil c \ln(n)$
- Theorem: With probability at least $1 - 1/n$, all processes access the database in the first $t = 2\lceil en \rceil \ln(n)$ rounds

MIN-CUT

- Input: An undirected graph $G = (V, E)$
- Output: A global minimum cut

- Recall that a cut (A, B) is a partition of the vertices into two sets
- The value of a cut is the number of edges across the cut

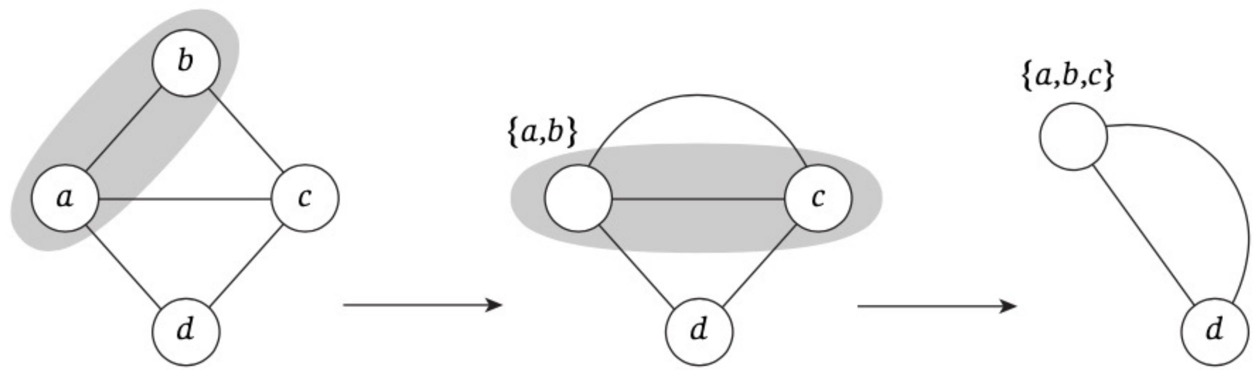
MIN-CUT

- Wait a second…
- We already know how to solve this!
- Use Max s-t flow $=$ Min s-t cut, for every pair of nodes s, t
- Oops, that was for directed graphs!
- Easy fix:
  - Replace each edge $e = (u, v)$ with two directed edges $u \to v$ and $v \to u$ with capacity 1 to get a new graph G'
  - Pick a vertex s
  - The minimum global cut separates s from something…
  - Try out all possible $t \in V$
- But, finding maximum flows was so hard…
- Can we do better?

MIN-CUT

- Contraction algorithm:
  - Pick an edge $e = (u, v)$ uniformly at random
  - Contract the edge e:
    - Replace u and v by a single super-node w
    - Preserve edges, updating the end points of u and v to w
    - Keep parallel edges, but delete self-loops

MIN-CUT



MIN-CUT

- Contraction algorithm:
    - Pick an edge $e = (u, v)$ uniformly at random
    - Contract the edge e:
        - Replace u and v by a single super-node w
        - Preserve edges, updating the end points of u and v to w
        - Keep parallel edges, but delete self-loops
    - Repeat until graph has two nodes v_1 and v_2
    - Return that cut (all nodes corresponding to super-set v_1)

MIN-CUT

- Claim: The contraction algorithms returns the global min-cut with probability at least $\frac{1}{\binom{n}{2}}$
- Proof:
  - Consider a global min-cut (A, B), and let F be the edges with one endpoint in A and the other in B. Let $k = |F|$
  - What could go wrong?
  - We could contract an edge F and put nodes in A and B in the same super-node
  - Upper bound the probability that this happens
  - In turn, need to lower bound size of E

MIN-CUT

Proof (continued):

- All vertices v have degree $\geq k$; otherwise, if v has degree $< k$, $\{v\}$ would be the minimum cut
- $|E| \geq kn/2$
- Probability that edge in F is contracted is at most $\frac{k}{|E|} \leq \frac{k}{\frac{kn}{2}} = \frac{2}{n}$
- Assume that after j iterations no edge in F has been contracted
- There are $n - j$ super-nodes
- There are at least k edges incident to every super-node; at least $k(n - j)/2$ total edges
- The probability of contracting an edge in F is at most $\frac{k}{\frac{k(n-j)}{2}} = 2/(n - j)$

MIN-CUT

Proof (continued):

- E_j = event that an edge in F is not contracted in iteration j
- $\Pr[E_1] \geq 1 - 2/n$
- $\Pr[E_2 | E_1] \geq 1 - 2/(n-1)$
- $\Pr\left[E_{j+1} \middle| E_1 \cap E_2 \cap \cdots E_j\right] \geq 1 - 2/(n-j)$
- $\Pr[no\ edge\ is\ contracted] = \Pr[E_1 \cap E_2 \ldots \cap E_{n-2}]$
- $= \Pr[E_1] \cdot \Pr[E_2 | E_1] \cdot \Pr[E_3 | E_1 \cap E_2] \cdot \ldots$
- $\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdot \ldots \cdot (1 - \frac{2}{3})$
- $= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \ \ldots \cdot \frac{2}{4} \cdot \frac{1}{3}$
- $= \frac{2}{n(n-1)} = 1/\binom{n}{2}$

MIN-CUT

- So, we fail with probability $1 - \frac{1}{\binom{n}{2}}$
- Which is basically 1…
- But less than 1…
- What if we run it again?
- The probability that we fail twice is $\left(1 - \frac{1}{\binom{n}{2}}\right)^2$
- What if we keep going?
- Repeat $\binom{n}{2}$ times
- Probability we fail all the time at most $\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}} \leq 1/e$
- Repeat $\binom{n}{2}\ln(n)$ times
- Probability we fail at most $e^{-\ln(n)} = 1/n$!

MIN-CUT

- Overall:
  - Guaranteed to run in polynomial time
  - Likely to get an optimal solution

MAX EXACT 3-SAT

- Maximization version of EXACT 3-SAT
- Given an EXACT 3-SAT formula (exactly 3 literals per clause) with n variables and k clauses, find an assignment that satisfies as many clauses as possible
  - Of course, it's still NP-complete

- Idea for a randomized algorithm: flip a coin and set each variable x_i to T with probability 1/2

MAX EXACT 3-SAT

- Claim: The expected number of clauses satisfied is $7k/8$
- Proof:
  - Random variable Z for the number of satisfied clauses
  - Consider indicator random variable Z_j for the event that clause C_j is satisfied
    - $Z_j = 1$ if C_j is satisfied and $Z_j = 0$ otherwise
  - $E\left[Z_j\right] = \Pr\left[Z_j = 1\right] = \frac{7}{8}$
    - There is only one way for C_j to not be satisfied!
  - $E[Z] = \sum_{j=1}^{k} E[Z_j] = 7k/8$

MAX EXACT 3-SAT

- Corollary: For every EXACT 3-SAT formula there exists a truth assignment that satisfies at least a 7/8 fraction of the clauses
- Proof:
  - With some probability the random variable Z takes a value at least its expectation
  - That corresponds to an outcome (i.e. truth assignment) with at least a 7/8 fraction of clauses satisfied
- This proof technique is called the probabilistic method
  - Show that something exists by showing it exists with strictly positive probability

MAX EXACT 3-SAT

- Question: Can we get a 7/8 approximation algorithm?
- Lemma 1: The probability that a random assignment satisfies at least $7k/8$ clauses is at least $\frac{1}{8k}$
- Proof:
  - Let p_j be the probability that exactly j clauses are satisfied
  - Let p be the probability that $\geq 7k/8$ clauses are satisfied
  - $E[Z] = \sum_{j=1}^{k} j \cdot p_j$
  - $= \sum_{j<\frac{7k}{8}} j \cdot p_j + \sum_{j\geq\frac{7k}{8}} j \cdot p_j$
  - $\leq \left(\frac{7k}{8} - \frac{1}{8}\right) \sum_{j<\frac{7k}{8}} p_j + k \sum_{j\geq\frac{7k}{8}} p_j$
  - $\leq \left(\frac{7\mathrm{k}}{8} - \frac{1}{8}\right) \cdot 1 + k \cdot p$
  - But, $E[Z] = 7k/8$
  - Re-arranging gives $p \geq 1/(8k)$

MAX EXACT 3-SAT

- Johnson's algorithm: Repeatedly generate random truth assignments until one of them satisfies $\geq 7k/8$ clauses
- Theorem: Johnson's algorithm is a $7/8$-approximation algorithm that has polynomial expected running time.
- Proof:
  - Lemma 1 gives that each iteration succeeds with probability at least $1/(8k)$
  - Let X be the random variable for the number of iterations until the first success
  - X follows the geometric distribution
  - $E[X] = \frac{1}{p}$, where p the probability of a success
  - So, 8k iterations in expectation

GEOMETRIC DISTRIBUTION (13.3)

- You have a coin that gives H w.p. p
- Flip coin until it comes up H
- Let X be the random variable indicating the number of flips performed
- $\Pr[X = j] = (1 - p)^{j-1} \cdot p$
- $E[X] = \sum_{j \geq 1} j \cdot \Pr[X = j] = \sum_{j \geq 1} j\,(1 - p)^{j-1} \cdot p$
- $= \frac{p}{1-p} \sum_{j \geq 1} j (1 - p)^j$
- $= \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$

MONTE CARLO VS LAS VEGAS

- Monte Carlo algorithm (e.g. the contraction algorithm for min-cut):
  - Guaranteed poly-time
  - Likely to give optimal solution
- Las Vegas algorithm (e.g. Johnson's algorithm):
  - Guaranteed to give optimal solution
  - Likely to run in poly-time

- Can always convert Las Vegas into Monte Carlo (stop algorithm at some point)

SUMMARY

- Contention Resolution (13.1)
- MIN-CUT (13.2)
- EXACT 3-SAT (13.4)

- Take a look at 13.3 for examples and exercises on linearity of expectation if you need practice!