Admin

Discuss collaboration.
Discuss median finding.

Median finding.

change from book. List L

- idea; **random sampling**
- median of sample looks like median of whole. neighborhood.
- Algorithm
  - choose s samples *with replacement*
  - take fences before and after sample median
  - keep items between fences. sort.
- Analysis
  - claim (i) median within fences and (ii) few items between fences.
  - Without loss of generality, L contains $1, \ldots, n$.
  - Samples $s_1, \ldots, s_m$ in sorted order.
  - lemma: S_r near rn/s.
    * Chernoff: $\forall k$, number elements before k is $(1 \pm \epsilon)ks/n$, where $\epsilon = \sqrt{(6n \ln n)/ks}$.
    * Thus, when $k > n/4$, error $ks/n(1 \pm \sqrt{24 \ln n/s}) = ks/n(1 \pm \epsilon)$.
    * $S_{(1+\epsilon)ks/n} > k$
    * $S_r > rn/s(1+\epsilon)$
    * $S_r < rn/s(1-\epsilon)$.
  - Let $r_0 = \frac{s}{2}(1-\epsilon)$
  - Then w.h.p., $\frac{n}{2}(1-\epsilon)/(1+\epsilon) < S_{r_0} < n/2$
  - Let $r_1 = \frac{s}{2}(1-\epsilon)$
  - Then $S_{r_1} > n/2$
  - But $S_{r_1} - S_{r_0} = O(\epsilon n)$
- Number of elements to sort: s
- Set containing median: $O(\epsilon n) = O(n\sqrt{(\log n)/s})$.
- balance: $\tilde{O}(n^{2/3})$ in both steps.

Randomized is strictly better:

- Optimum deterministic: $\geq (2+\epsilon)n$
- Optimum randomized: $\leq (3/2)n + o(n)$

Routing

- synchronous message passing
- bidirectional links, one message per step
- queues on links
- **permutation** routing
- **oblivious algorithms** only consider self packet.
- **Theorem** Any deterministic oblivious permutation routing requires $\Omega(\sqrt{N/d})$ steps on an N node degree d machine.
  - reason: some edge has lots of paths through it.
  - **homework:** special case
- Hypercube.
  - N nodes, $n = \log_2 N$ dimensions
  - bit representation
  - natural routing: bit fixing (left to right)
  - paths of length n
  - Nn edges for N length n paths
  - lower bound n
- Routing algorithms:
  - $O(n) = O(\log N)$ randomized
  - beats $\Omega(\sqrt{N/n})$ deterministic
  - how? load balance paths.
- Random destination (not permutation!), bit correction
  - Average case, but a good start.
  - $T(e_i)$ = number of paths using e_i
  - by symmetry, all $E[T(e_i)]$ equal
  - expected path length $n/2$
  - LOE: expected total path length $Nn/2$
  - nN edges in hypercube
  - $E[T(e_i)] = 1/2$
  - Chernoff: every edge gets $\leq 3n$ (prob $1 - 1/N$)
- Naive usage:

- n phases, one per bit
- $3n$ time per phase
- $O(n^2)$ total
- From intermediate destination, route back!
- routes worst case permutation in $O(n^2)$.

- What if don't wait for next phase?
  - FIFO queuing
  - total time is length plus **delay**
  - Expected delay $\leq E[\sum T(e_l)] = n/2$.
  - Chernoff bound? no. dependence of $T(e_i)$.

- High prob. bound:
  - consider paths sharing route $(e_0, \ldots, e_k)$
  - Suppose S packets intersect route (use at least one of e_i)
  - claim delay $\leq |S|$
  - Suppose true: Let $H_{ij} = 1$ if j hits i's (fixed) route.

$$\begin{aligned} E[|S|] &= E[\sum H_{ij}] \\ &\leq E[\sum T(e_l)] \\ &\leq n/2 \end{aligned}$$

  - Now Chernoff **does** apply (H_{ij} independent for fixed i-route).
  - $|S| = O(n)$ w.p. $1 - 2^{-5n}$, so $O(n)$ delay for all 2^n paths.

- Lag argument
  - Exercise: once packets separate, don't rejoin
  - Route for i $\rho_i = (e_1, \ldots, e_k)$
  - charge each delay to a departure of a packet from ρ_i.
  - Packet waiting to follow e_j at time t has: **Lag** $t - j$
  - Delay of v_i is lag crossing e_k
  - When v_i delay rises to $l+1$, some packet from S has lag l (since crosses e_j instead of v_i).
  - Consider last time t' where a lag-l packet exists
    * some lag-l packet w crosses $e_{j'}$ at t' (others increase to lag-$(l+1)$)
    * w leaves at this point (if not, then l at $e_{j'+1}$ next time)
    * charge one delay to w.