Admin

Next tuesday: holiday.

- pset due thursday
- but material done TODAY (almost)
- so start/finish early, have fun on vacation
- New pset POSTED tuesday, distributed thursday

Method of Conditional Probabilities and Expectations

Derandomization.

- Theory: is P=RP?
- practice: avoid chance of error, chance of slow.

Conditional Expectation. Max-Cut

- Imagine placing one vertex at a time.
- $x_i = 0$ or 1 for left or right side
- $E[C] = (1/2)E[C|x_1 = 0] + (1/2)E[C|x_1 = 1]$
- Thus, either $E[C|x_1 = 0]$ or $E[C|X_1 = 1] \geq E[C]$
- Pick that one, continue
- More general, whole tree of element settings.
  - Let $C(a) = E[C \mid a]$.
  - For node a with children b, c, $C(b)$ or $C(c) \geq C(a)$.
- By induction, get to leaf with expected value at least $E[C]$
- But no randomness left, so that is actual cut value.
- Problem: how compute node values? Easy.

Conditional Probabilities. Set balancing. (works for wires too)

- Review set-balancing Chernoff bound
- Think of setting item at a time
- Let Q be bad event (unbalanced set)
- We know $\Pr[Q] < 1/n$.

- $\Pr[Q] = 1/2 \Pr[Q \mid x_{i0}] + 1/2 \Pr[Q \mid x_{i1}]$
- Follows that one of conditional probs. less than $\Pr[Q] < 1/n$.
- More general, whole tree of element settings.
  - Let $P(a) = \Pr[Q \mid a]$.
  - For node a with children b, c, $P(b)$ or $P(c) < P(a)$.
  - $P(r) < 1$ sufficient at root r.
  - at leaf l, $P(l) = 0$ or 1.
- One big problem: need to compute these probabilities!

Pessimistic Estimators.

- Alternative to computing probabilities
- three neceessary conditions:
  - $\hat{P}(r) < 1$
  - $\min\{\hat{P}(b), \hat{P}(c)\} < \hat{P}(a)$
  - $\hat{P}$ computable

  Imply can use $\hat{P}$ instead of actual.
- Let $Q_i = \Pr[\text{unbalanced set } i]$
- Let $\hat{P}(a) = \sum \Pr[Q_b \mid a]$ at tree node a
- Claim 3 conditions.
  - HW
- Result: deterministic $O(\sqrt{n \ln n})$ bias.
- more sophisticated pessimistic estimator for wiring.

Oblivious routing

- recall: choose random routing. Only $1/N$ chance of failure
- Choose N^3 random routines.
- whp, for every permutation, at most $2N^2$ bad routes.
- given the N^3 routes, pick one at random.
- so for any permutation, prob $2/N$ of being bad.

Fingerprinting

Basic idea: compare two things from a big universe U

- generally takes $\log U$, could be huge.
- Better: randomly map U to smaller V, compare elements of V.
- Probability(same)$= 1/|V|$
- intuition: $\log V$ bits to compare, error prob. $1/|V|$

We work with *fields*

- add, subtract, mult, divide
- 0 and 1 elements
- eg reals, rats, (not ints)
- talk about Z_p
- which field often won't matter.

Verifying matrix multiplications:

- Claim $AB = C$
- check by mul: n^3, or $n^{2.376}$ with deep math
- Freivald's $O(n^2)$.
- Good to apply at end of complex algorithm (check answer)

Freivald's technique:

- choose random $r \in \{0, 1\}^n$
- check $ABr = Cr$
- time $O(n^2)$
- if $AB = C$, fine.
- What if $AB \neq C$?
  - trouble if $(AB - C)r = 0$ but $D = AB - C \neq 0$
  - find some nonzero row $(d_1, \ldots, d_n)$
  - wlog $d_1 \neq 0$
  - trouble if $\sum d_i r_i = 0$
  - ie $r_1 = (\sum_{i>1} d_i r_i)/d_1$

- principle of deferred decisions: choose all $i \geq 2$ first
- then have exactly one error value for r_1
- prob. pick it is at most $1/2$

How improve detection prob?

- k trials makes $1/2^k$ failure.
- Or choosing $r \in [1, s]$ makes $1/s$.

• Doesn't just do matrix mul.

- check any matrix identity claim
- useful when matrices are "implicit" (e.g. AB)

• We are mapping matrices (n^2 entries) to vectors (n entries).

String matching

Checksums:

• Alice and Bob have bit strings of length n

• Think of n bit integers a, b

• take a prime number p, compare $a \bmod p$ and $b \bmod p$ with $\log p$ bits.

• trouble if $a = b \pmod{p}$. How avoid? How likely?

- $c = a - b$ is n-bit integer.
- so at most n prime factors.
- How many prime factors less than k? $\Theta(k/\ln k)$
- so take $2n^2 \log n$ limit
- number of primes about n^2
- So on random one, $1/n$ error prob.
- $O(\log n)$ bits to send.
- implement by add/sub, no mul or div!

How find prime?

- Well, a randomly chosen number is prime with prob. $1/\ln n$,
- so just try a few.
- How know its prime? Simple randomized test (later)

Pattern matching in strings

- m-bit pattern
- n-bit string
- work mod prime p of size at most t
- prob. error at particular point most $m/(t/\log t)$
- so pick big t, union bound
- implement by add/sub, no mul or div!