Midterm out today.
Collaborations.

**Shortest Paths**
classical shortest paths.

- dijkstra’s algorithm
- floyd’s algorithm. similarity to matrix multiplication

Matrices

- length 2 paths by squaring
- matrix multiplication. strassen.
- shortest paths by “funny multiplication.”
  - huge integer implementation
  - base-(n + 1) integers

Boolean matrix multiplication

- easy.
- gives objects at distance 2.
- gives $nMM(n)$ algorithm for problem
- what about recursive?
- well can get to within 2: let $T_k$ be boolean “distance less than or equal to $2^k$. Squaring gives $T_{k+1}$.
- $O(\log n)$ squares for unit length
- what about exact?

Seidel’s distance algorithm for unit lengths.

- log-size integers:
  - parities suffice:
    * square $G$ to get adjacency $A'$, distance $D'$
      - if $D_{ij}$ even then $D_{ij} = 2D'_{ij}$
      - if $D_{ij}$ odd then $D_{ij} = 2D'_{ij} - 1$
  - For neighbors $i, k$,
    * $D_{ij} - 1 \leq D_{kj} \leq D_{ij} + 1$
exists $k$, $D_{kj} = D_{ij} - 1$

- Parities
  - If $D_{ij}$ even, then $D'_{kj} \geq D'_{ij}$ for every neighbor $k$
  - If $D_{ij}$ odd, then $D'_{kj} \leq D'_{ij}$ for every neighbor $k$, and strict for at least one
- Add
  - $D_{ij}$ even iff $S_{ij} = \sum_k D'_{kj} \geq D_{ij}d(i)$
  - $D_{ij}$ odd iff $\sum_k D'_{kj} < D_{ij}d(i)$
  - How determine? find $S = AD'$

* Result: all distances in $O(M(n) \log n)$ time.

This is **deterministic distance algorithm**.

To find paths: Witness product

* example: tripartite one-hop hop case

Modify matrix alg:

* easy case: unique witness
  - multiply column $c$ by $c$.
  - read off witness identity

* reduction to easy case:
  - Suppose $r$ columns have witness
  - Suppose choose each with prob. $p$
  - Prob. exactly 1 witness: $rp(1 - p)^{r-1} \approx 1/e$
  - Try all values of $r$
  - Wait, too many.

* Approx
  - Suppose $p = 2/r$
  - Then prob. exactly 1 is $\approx 2/e^2$
  - So anything in range $1/r \ldots 1/2r$ will do.
  - So try $p$ all powers of 2.
  - suppose $2^k \leq r \leq 2^{k+1}$
  - choose each column with probability $2^{-k}$.
  - prob. exactly one witness: $r \cdot 2^{-k}(1 - 2^{-k})^{r-1} \geq (1/2)(1/e^2)$
  - so try $\log n$ distinct powers of 2, each $O(\log n)$ times

* So, can find shortest paths by doing one Matrix mul for each distance value
– $n$ matrix muls
generalize to more distances:
– distances now known
– for each node, dest, find neighbor with distance one less
– boolean matrix $R$ of “distance is $k - 1$”
– boolean witness product of $RA$

• Mod 3:
  – Recall some neighbor distance down by one
  – so compute distances mod 3.
  – suppose $D_{ij} = 1 \mod 3$
  – then look for $k$ neighbor of $i$ such that $D_{kj} = 0 \mod 3$
  – let $D_{ij}^{(s)} = 1$ iff $D_{ij} = s \mod 3$
  – than $AD^{(s)}$ has $ij = 1$ iff a neighbor $k$ of $i$ has $D_{kj}^{(s)}$
  – so, witness matrix mul!

**Parallel Algorithms**

PRAM

• $P$ processors, each with a RAM, local registers
• global memory of $M$ locations
• each processor can in one step do a RAM op or read/write to one global memory location
• synchronous parallel steps
• various conflict resolutions (CREW, EREW, CRCW)
• not realistic, but explores “degree of parallelism”

Randomization in parallel:

• load balancing
• symmetry breaking
• isolating solutions

Classes:

• NC: poly processor, polylog steps
• RNC: with randomization. polylog runtime, monte carlo
• ZNC: las vegas NC
• immune to choice of conflict resolution

Practical observations:
• very little can be done in $o(\log n)$ with poly processors
• lots can be done in $\Theta(\log n)$
• often concerned about work which is processors times time
• algorithm is “optimal” if work equals best sequential

Basic operations
• and, or
• counting ones
• parallel prefix

Addition
• Prefix sum over “kill”, “propogate”, “carry” operations
• handles $n$-bit numbers in $O(\log n)$ time
• multiplication as $n^2$ additions (better methods exist)

Sorting
Quicksort in parallel:
• $n$ processors
• each takes one item, compares to splitter
• count number of predecessors less than splitter
• determines location of item in split
• total time $O(\log n)$
• combine: $O(\log n)$ per layer with $n$ processors
• problem: $\Omega(\log^2 n)$ time bound
• problem: $n \log^2 n$ work
Perfect Matching

We focus on bipartite; book does general case.
Last time, saw detection algorithm in $\mathcal{RNC}$:

- Tutte matrix
- Symbolic determinant nonzero iff PM
- assign random values in $1, \ldots, 2m$
- Matrix Mul, Determinant in $\mathcal{NC}$

How about finding one?

- If unique, no problem
- Since only one nonzero term, ok to replace each entry by a 1.
- Remove each edge, see if still PM in parallel
- multiplies processors by $m$
- but still $\mathcal{NC}$

Idea:

- make unique minimum weight perfect matching
- find it

Isolating lemma: [MVV]

- Family of distinct sets over $x_1, \ldots, x_m$
- assign random weights in $1, \ldots, 2m$
- $\text{Pr}(\text{unique min-weight set}) \geq 1/2$
- Odd: no dependence on number of sets!
- (of course $< 2^m$)

Proof:

- Fix item $x_i$
- $Y$ is min-value sets containing $x_i$
- $N$ is min-value sets not containing $x_i$
- true min-sets are either those in $Y$ or in $N$
- how decide? Value of $x_i$
• For $x_i = -\infty$, min-sets are $Y$
• For $x_i = +\infty$, min-sets are $N$
• As increase from $-\infty$ to $\infty$, single transition value when both $X$ and $Y$ are min-weight
• If only $Y$ min-weight, then $x_i$ in every min-set
• If only $X$ min-weight, then $x_i$ in no min-set
• If both min-weight, $x_i$ is ambiguous
• Suppose no $x_i$ ambiguous. Then min-weight set unique!
• Exactly one value for $x_i$ makes it ambiguous given remainder
• So $\Pr(\text{ambiguous}) = 1/2m$
• So $\Pr(\text{any ambiguous}) < m/2m = 1/2$

Usage:
• Consider tutte matrix $A$
• Assign random value $2^{w_i}$ to $x_i$, with $w_i \in 1, \ldots, 2m$
• Weight of matching is $2 \sum w_i$
• Let $W$ be minimum sum
• Unique w/pr 1/2
• If so, determinant is odd multiple of $2^W$
• Try removing edges one at a time
• Edge in PM iff new determinant/$2^W$ is even.
• Big numbers? No problem: values have poly number of bits

$NC$ algorithm open.
For exact matching, $P$ algorithm open.