Wrap Up!

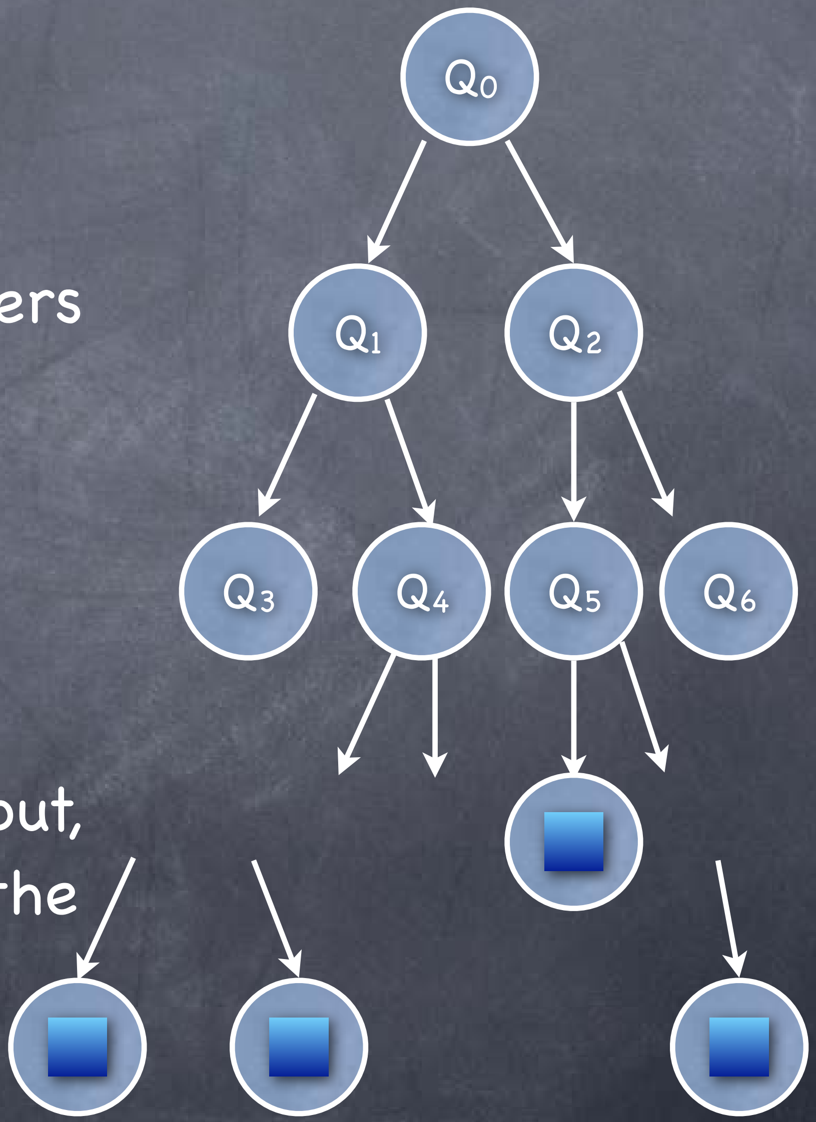
Lecture 25

Decision Trees & Branching Programs

Many Topics Not Covered!

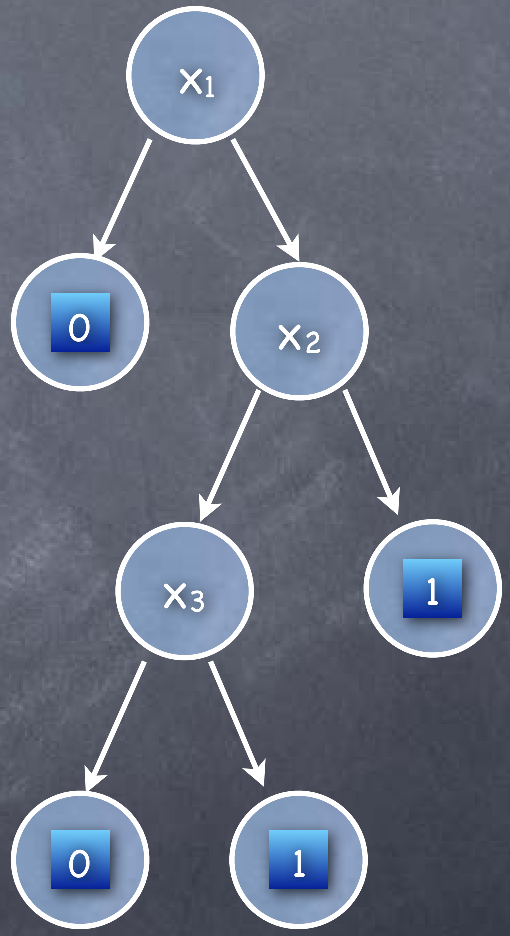
Decision Trees

- Another model of non-uniform computation
  - A full binary tree with each internal node labelled with an "elementary" boolean function of the input
    - Two children correspond to answers true and false
  - Leaves are labelled with outputs
- Evaluating a decision tree:
  - start from the root and at each node, evaluate the node's function on the input, and go to the child corresponding to the outcome
  - At the leaf produce the output



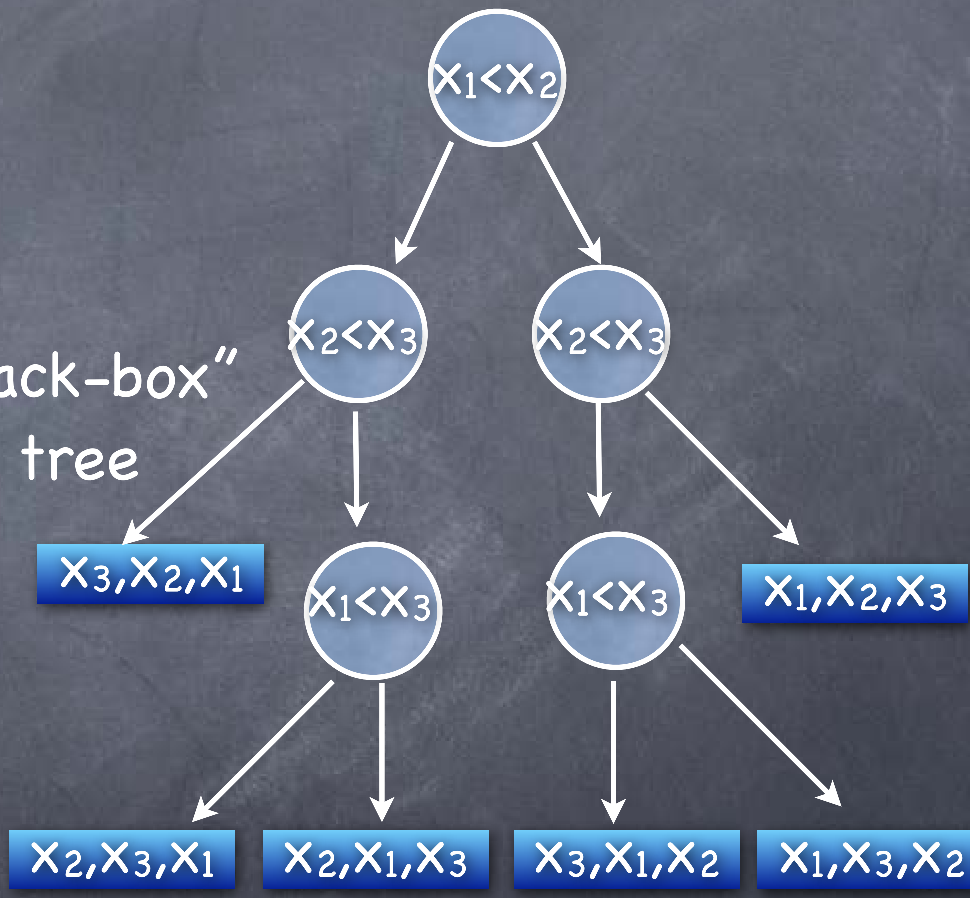
Decision Trees

- Example: $f(x_1,x_2,x_3) = x_1 \wedge (x_2 \vee x_3)$
- How about $x_1 \oplus \ldots \oplus x_n$?
- Every function $f: \{0,1\}^n \longrightarrow \{0,1\}$ has a trivial decision tree with 2^n leaves
  - At level i, use $Q_i(x_1,\ldots,x_n) = x_i$
  - For each input $(x_1,\ldots,x_n)$ a separate leaf, which is labelled with output $f(x_1,\ldots,x_n)$



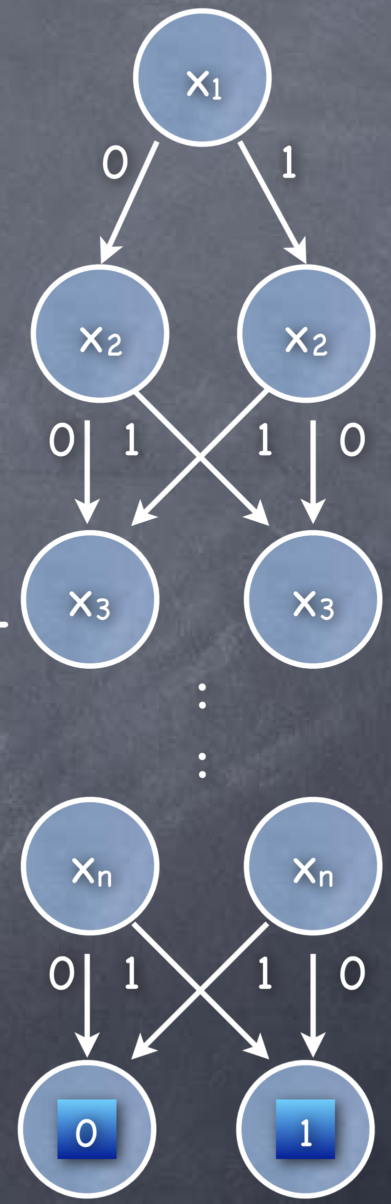
Decision Trees

- Another Example: Sorting
  - Input: $(x_1,\ldots,x_n)$, distinct
  - Output: Sorted list
- Each Q is of the form $(x_i < x_j)$
- Any sorting algorithm that uses "black-box" comparisons defines such a decision tree
  - All n! possible orderings should appear as leaves in this tree
  - #comparisons in the worst case = depth of the tree
  - If depth d, need $2^d \geq$ #leaves $\geq n!$
  - $d \geq \log n! \geq c \cdot n \log n$



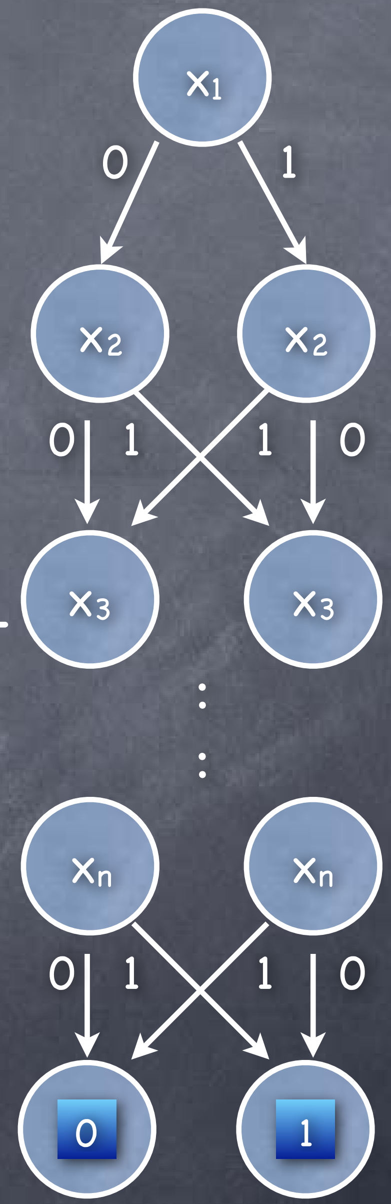
Branching Programs

- A more compact version of a decision tree: Equivalent nodes in the tree can be shared by their parents
  - Results in a DAG
- E.g., $x_1 \oplus \ldots \oplus x_n$ has a width-2 branching program with O(n) nodes
- Permutation Branching Program: Levelled DAG of width w at each level, with 0-edges mapping nodes at a level bijectively to the nodes at the next level; same for 1-edges
- Exercise: Convert a BP to a circuit of similar size
- Barrington's Theorem: A depth d boolean circuit with binary gates for f: $\{0,1\}^n \to \{0,1\}$ can be turned into a permutation branching program for f, with width 5, and length $\leq 4^d$



Branching Programs

- A more compact version of a decision tree: Equivalent nodes in the tree can be shared by their parents
  - Results in a DAG
- E.g., $x_1 \oplus \ldots \oplus x_n$ has a width-2 branching program with O(n) nodes
- Permutation Branching Program: Levelled DAG of width w at each level, with 0-edges mapping nodes at a level bijectively to the nodes at the next level; same for 1-edges
- Exercise: Convert a BP to a circuit of similar size
- Barrington's Theorem: A depth d boolean circuit with binary gates for $f: \{0,1\}^n \rightarrow \{0,1\}$ can be turned into a permutation branching program for f, with width 5, and length $\leq 4^d$



Topics covered

Recursive Def.
Generating Fun.

Bounding
big-O

Computation
Models

Induction

Counting

Trees

Numbers and
patterns therein

Graphs

Basic tools for expressing ideas
Logic, Proofs,
Sets, Relations, Functions

Topics not covered

But Could Have Been

Probability	Expectation & Variance. Conditional Probability. Entropy and Mutual Information ...
Abstract Algebra	(Discrete) Groups, Rings and Fields. Polynomials. Linear Algebra (over Finite Fields).
Codes	Error Correcting Codes. Compression.
More Graphs	Directed graphs, network flow, planar graphs, ...
More Combinatorics	Matroids, Designs, Ramsey Theory, Probabilistic Method, ...

Topics not covered

But Could Have Been

Probability	Expectation & Variance. Conditional Probability. Entropy and Mutual Infor
Abstract Algebra	(Discrete) Groups, Rings and Fiel Linear Algebra (over Finit
Codes	Error Correcting Codes. Co
More Graphs	Directed graphs, network flow, planar graphs, ...
More Combinatorics	Matroids, Designs, Extremal Combinatorics, Probabilistic Method, ...

An illustrative example from cryptography: Secret Sharing

A Game

- A “dealer” and two “players” Alice and Bob (computationally unbounded)
- Dealer has a message, say two bits m_1m_2
- She wants to “share” it among the two players so that neither player by herself/himself learns <u>anything</u> about the message, but together they can find it
- Bad idea: Give m_1 to Alice and m_2 to Bob
- Other ideas?

Sharing a bit

- To share a bit m, Dealer picks a uniformly random bit b and gives a := m⊕b to Alice and b to Bob
  - Together they can recover m as a⊕b
  - Each party by itself learns nothing about m: for each possible value of m, its share has the same probability distribution

$m = 0 \mapsto (a,b) = (0,0)$ or $(1,1)$ w/ probability 1/2 each
$m = 1 \mapsto (a,b) = (1,0)$ or $(0,1)$ w/ probability 1/2 each

  - i.e., the vector of probabilities (Pr[a=0], Pr[a=1]) is the same (namely, (0.5,0.5)) irrespective of the message. Same for (Pr[b=0], Pr[b=1])

Sharing Larger Messages

- To share a message $m \in \mathbb{Z}_n$, Dealer picks a uniformly random $b \in \mathbb{Z}_n$ and gives $a := m-b$ (in $\mathbb{Z}_n$) to Alice and b to Bob
  - Together they can recover m as $a+b$ (in $\mathbb{Z}_n$)
  - Each party by itself learns nothing about m: for each possible value of m, its share has the same probability distribution

$m \mapsto (a,b) = (m,0), (m-1,1), (m-2,2), \ldots, (m+1,n-1)$ w/ probability 1/n each

  - i.e., the vector of probabilities (Pr[a=0],...,Pr[a=n-1]) is the same (namely, (1/n,...,1/n)) irrespective of the message. Same for (Pr[b=0],...,Pr[b=n-1])

Sharing Larger Messages

- Same idea works over any finite group

  $* : G \times G \longrightarrow G$

- (Finite) Group: a (finite) set G along with a binary operation $*$, s.t.
  - Associative: $\forall a,b,c \in G\ (a * b) * c = a * (b * c)$
  - Identity Exists: $\exists\ e \in G$ s.t. $\forall a \in G,\ a * e = e * a = a$
  - Inverse Exists: $\forall a \in G,\ \exists\ a^{-1} \in G$, s.t. $a * a^{-1} = a^{-1} * a = e$
  - Optionally, Commutative: $\forall a,b \in G,\ a * b = b * a$
  - E.g.: $(\mathbb{Z}_n,+)$, $(\mathbb{Z}_n^*,\times)$, (permutations of [n], composition), (invertible square matrices, matrix multiplication), ...
- To secret share m, pick random $a,b \in G$ conditioned on $a*b=m$
  - i.e., pick random b and set $a := m * b^{-1}$
  - $\forall m \in G$, each of a,b is uniformly random over G

Makes sense as G is finite

Sharing Among N Parties

- Extends to sharing a message among N parties, so that up to N-1 parties learn nothing about the message
- To secret share m, pick random $a_1,\dots,a_N \in G$ conditioned on $a_1 * \dots * a_N = m$
  - e.g., pick random $a_2,\dots,a_N$ and set $a_1 := m * (a_2 * \dots * a_N)^{-1}$
  - For any set of N-1 parties — say all but i^{th} party — the combination of shares they obtain is distributed the same way irrespective of what the message m is.
    - Fix $m \in G$. Consider any $g_1,\dots,g_{i-1},g_{i+1},\dots,g_N \in G$
    - $\Pr[(a_1,\dots,a_{i-1},a_{i+1},\dots,a_N) = (g_1,\dots,g_{i-1},g_{i+1},\dots,g_N)]$
      $= \Pr[(a_2,\dots,a_N) = (g_2,\dots,g_N)]$ where g_i is the unique value s.t $g_1 * \dots * g_N = m$. i.e., $g_i = (g_1 * \dots * g_{i-1})^{-1} * m * (g_{i+1} * \dots * g_N)^{-1}$
    - So, $\Pr[(a_1,\dots,a_{i-1},a_{i+1},\dots,a_N) = (g_1,\dots,g_{i-1},g_{i+1},\dots,g_N)] = 1/|G|^{N-1}$

Threshold Secret-Sharing

- (N,T)-secret-sharing
  - Divide a message m into N shares $a_1,\ldots,a_N$, such that
    - any T shares are enough to reconstruct the secret
    - up to T-1 shares should have no information about the secret
- So far: (N,N) secret-sharing

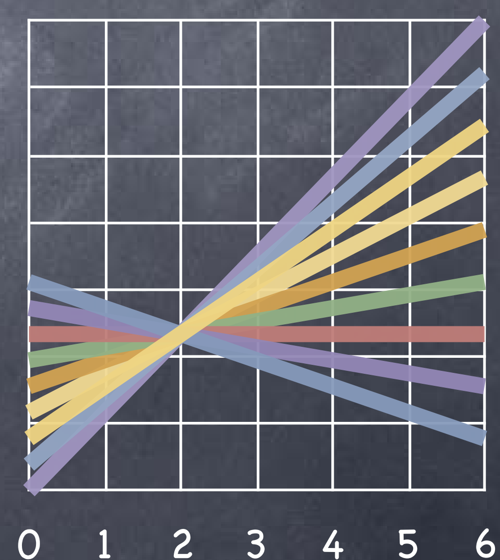
e.g., $(a_1,\ldots,a_{T-1})$ has the same distribution for every m in the message space

Threshold Secret-Sharing

- Construction: (N,2) secret-sharing (for N≥2)
- Message-space = share-space = F, a finite field (e.g. integers mod prime)
  - Share(m): pick random r. Let $a_i = r \cdot c_i + m$ (for $i=1,\dots,N < |F|$)
  - Reconstruct(a_i, a_j): $r = (a_i-a_j)/(c_i-c_j)$; $m = a_i - r \cdot c_i$
  - Each a_i by itself is uniformly distributed, irrespective of m [Why?]
  - "Geometric" interpretation
    - Sharing picks a random "line" $y = f(x)$, such that $f(0)=M$. Shares $a_i = f(c_i)$.
    - a_i is independent of m: exactly one line passing through (c_i,a_i) and $(0,m')$ for any secret m'
    - But can reconstruct the line from two points!

c_i are n distinct, non-zero field elements

Since c_i^{-1} exists, exactly one solution for $r \cdot c_i+m=d$, for every value of d



Threshold Secret-Sharing

Shamir Secret-Sharing

- (N,T) secret-sharing in a (large enough) field F
- Generalizing the geometric/algebraic view: instead of lines, use polynomials
  - Share(m): Pick a random degree T-1 polynomial f(X), such that f(0)=M. Shares are $a_i = f(c_i)$.
    - Random polynomial with f(0)=m: $z_0 + z_1X + z_2X^2 + ... + z_{T-1}X^{T-1}$ by picking z_0=M and $z_1,...,z_{T-1}$ at random.
  - Reconstruct($a_1,...,a_T$): Lagrange interpolation to find m=z_0
    - Need T points to reconstruct the polynomial. Given T-1 points, out of $|F|^{T-1}$ polynomials passing through (0,m') (for any m') there is exactly one that passes through the T-1 points

Lagrange Interpolation

- Given T distinct points on a degree T-1 polynomial (univariate, over some field of more than T elements), reconstruct the entire polynomial (i.e., find all T coefficients)
  - T variables: $z_0,\ldots,z_{T-1}$.
  - T equations: $1.z_0 + c_i.z_1 + c_i^2.z_2 + \ldots c_i^{T-1}.z_{T-1} = a_i$
  - A linear system: $W\underline{\mathbf{z}}=\underline{\mathbf{s}}$, where W is a $T\times T$ matrix with i^{th} row, $W_i= (1\ c_i\ c_i^2 \ldots c_i^{T-1})$, c_i's distinct
  - W (called the Vandermonde matrix) is invertible over any field
    - $\underline{\mathbf{z}} = W^{-1}\underline{\mathbf{a}}$

Error-Correcting Codes

- In Shamir secret sharing, field elements $z_0,\ldots,z_{T-1}$ were encoded into field elements (shares) $a_1,\ldots,a_N$
  - Any subset of T shares could be used to reconstruct all z_i (we were interested in reconstructing z_0)
- Reed-Solomon Code: Can "store" data redundantly in N disks, so that even if any N-T disks crash, can recover the data
  - Optimal rate: Can store T disks worth data in N disks and recover from N-T crashes (e.g., N=2T, can handle half the disks crashing)
    - Compare with <u>mirroring</u> disks: To handle half the disks crashing, only one disk worth of data can be stored
- What if some disks could get silently corrupted (instead of crashing)?
  - Can reconstruct the original data if < (N-T)/2 disks corrupted