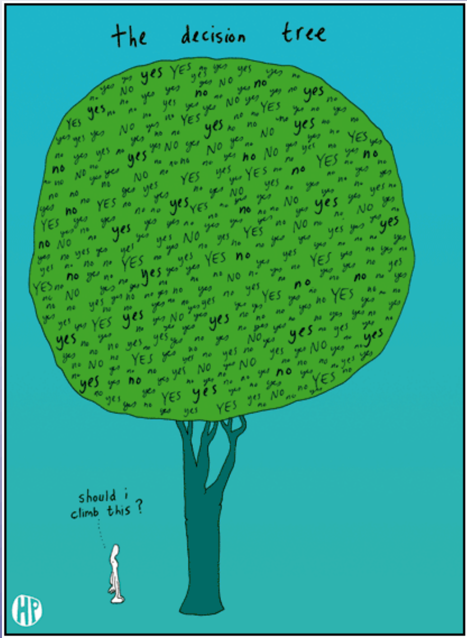
Decision Trees and Data Structures

Communication Complexity Seminar
Instructor: Ronitt Rubinfeld
Fall 2009

By: Ayelet Leigh

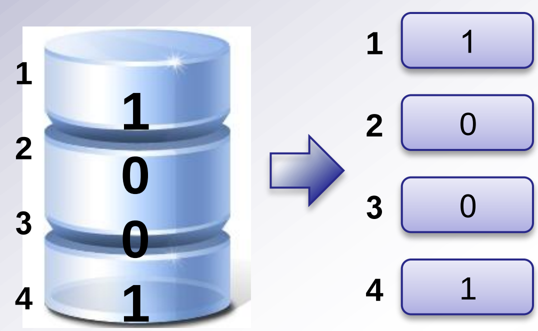
Plan for today:

- communication complexity lower bounds yield data structure lower bounds



What are we doing ?

- Implementation of a “database” - D:
  - D represents a subset S of $\{1 \ldots N\}$



- Access to D via "membership queries" - Q
  - for each i, can ask: “$is\ \ i\ \in\ S\ \ ?$”
  - denote possible queries by $Q\ =\{1,\ldots,N\}$

The Cell Probe Model

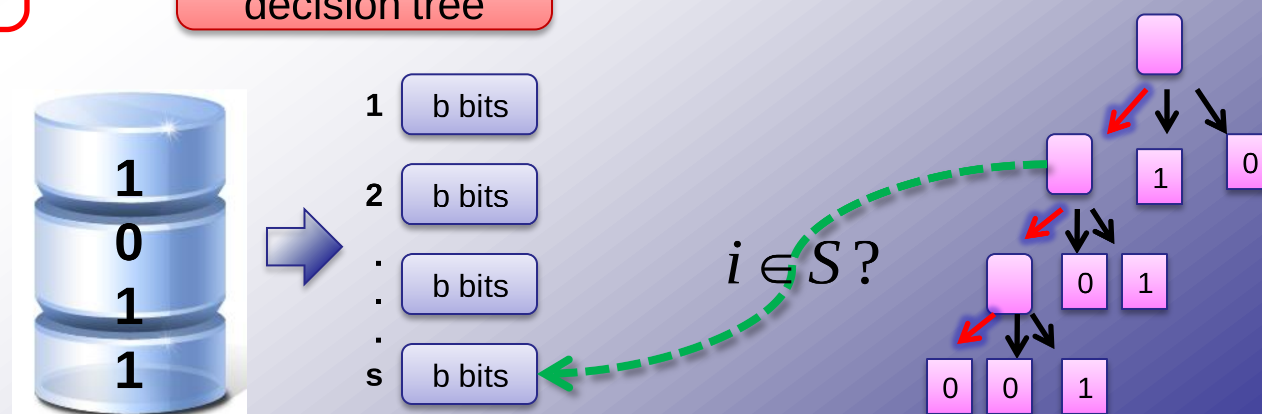
check,
investigate,
inquiry

- Database Definition:
  - Memory mapped into fixed-size cells (b bits per cell)
- Query via Decision Trees
  - Each node probes one cell
  - Continue based on bits received (2^b options)
  - Leaves contain query answers

Read value of

- $\boxed{\text{Time} = \max_{\# probs}}$

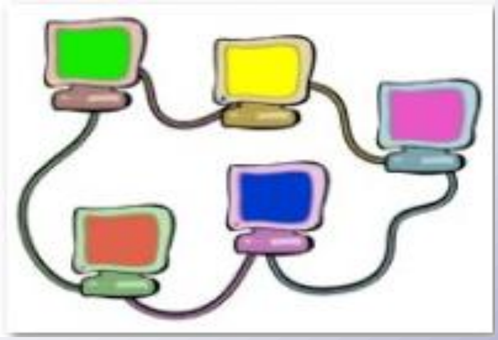
Depth of worst decision tree



What is the goal today?

- Won't see *time* lower-bound
- Won't see *space* lower-bound
- Will see *trade-off* lower-bound
  - Time-space trade-off

- Reduction from database problem to communication complexity problem
  - We know communication lower-bounds



Reduction to communication complexity problem

- Function $f(q,D)$ – answer to q on D:
  - Assumptions:
    - D is stored using s cells of b bits each
    - Any $q \epsilon Q$ can be solved using at most t probes
  - **Lemma 1:** f can be solved by t-round communication protocol, at each round:
    - Alice sends $log(s)$ bits holds q
    - Bob sends b bits holds D

Assumptions:
s cells b bits per cell
q solves with ≤ t probes

To Show
Each of ≤ t rounds:
Alice $\log s$ bits
Bob b bits

Proof

Alice

Bob

Simulates the tree corresponding q

Implements probe to cell by sending cell index

There are s cells

Log(s) bits

Reply with cell content

b bits

Updates her current node accordingly

There are only t probs

Example 1

- U – vector subspace of Z_2^n
- $\forall q \in \{0,1\}^n$ We need to answer: "$q \in U$?"
- Cell size: $b = n$ bits

- How to store subspace U ?
  - Set of $n - \dim(U)$ linear equations defining U

Reminder...

U subspace of Z_2^n

$n = 2$

Linear equations defining subspace U:

$(x\ y)\binom{1}{1} = 0 \ \Longrightarrow U\colon (1\ 1), (0\ 0)$

$\dim(\mathrm{U}) = 1 \ \ \Longrightarrow 1\ equation$

$\dim(\mathrm{U}) = 2 \ \ \Longrightarrow 0\ equations - all\ z_2^2$

$(x\ y)\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = 0 \ \Longrightarrow U\colon (0\ 0)$

$\dim(\mathrm{U}) = 0 \ \ \Longrightarrow 2\ equation$

Example 1

- U – vector subspace of Z_2^n
- $\forall q \in \{0,1\}^n$ We need answer: "$q \in U?$"
- Cell size: $b = n$ bits

- How to store subspace U ?
  - Set of $n - \dim(U)$ linear equations defining U
  - Use cell for each equation
    - Reminder: every equation in U described by n bits
  - $\forall q$ we access all cells to check if q satisfies each of them

Space: $s = n$ cells

Query Time: $t = n$

Example - can we do it faster?

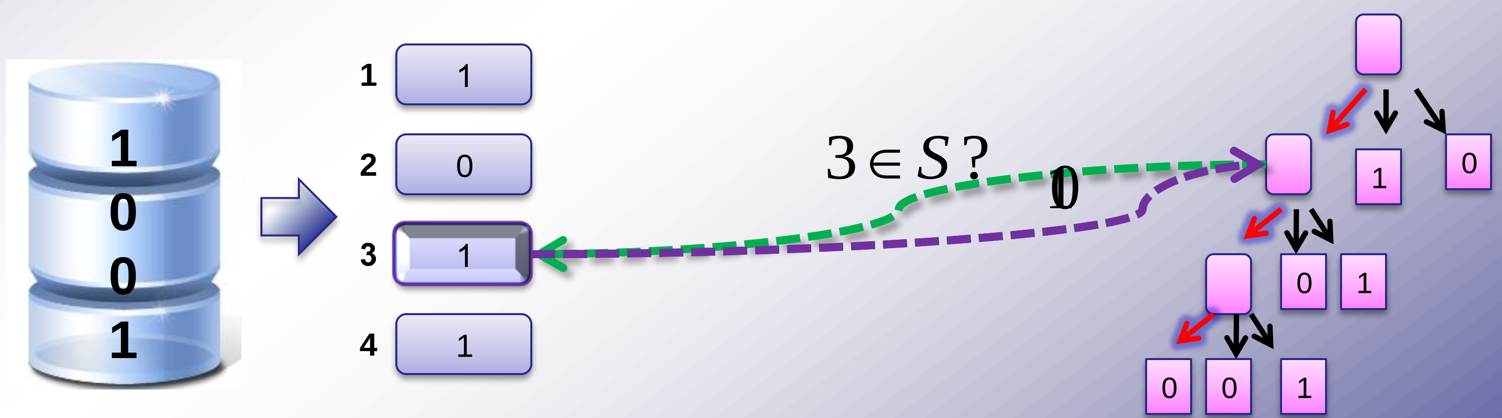
- Can we answer with $t = o(n)$ time?
- How many cells needed?
  - Reminder: $f(q, U)$ is the SPAN problem (last week)
  - We saw: Alice sends $\Omega(n)$ or Bob sends $\Omega(n^2)$
  - From lemma 1, using s-cell solution for subspace gives protocol for SPAN, satisfying:
    - Alice sends $t \log s$ bits
    - Bob sends tn bits
  - If $t = o(n)$:
    - Then Bob sends $o(n^2)$ bits
    - Therefore Alice sends $\Omega(n)$ bits

$\Rightarrow$ $s \geq 2^{\Omega(n/t)}$

Dynamic Data Structures

- Definition:
  - A data structure whose data may change during its lifetime.

- The problem:
  - Maintaining a database D under update operations:



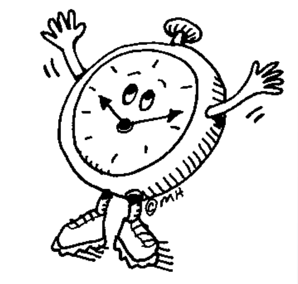
The dynamic problem

- For any k, the dynamic problem function:

$$f_k(q, \vec{u})$$

  - $\vec{u} = u_1, \ldots, u_k$ - sequence of updates
  - q - a query
- The value of f is the answer to the query, after the updates were performed.

Time lower bound



- Assumptions:
  - Every update can be performed in ≤ t probes
  - Every query can be performed in ≤ t probes
  - Cell size is b bits
- **Lemma 2:**

  f_k has a communication protocol where:
  - Alice sends $O(t\log(kt))$ bits
  - Bob sends $O(t(b+\log|Q|)+\log(kt))$ bits

Notice similarity to lemma 1

The proof will be similar too

Before we prove

- More assumptions:
  - Both Alice and Bob know the initial DB
  - Only Bob knows the updates
- Reminder - hash functions:
  - mod function reduces the number of bits used

Useful fact

W is a subset of $\{1, \dots, s\}$, $|W| \leq kt$

Random prime $p \in 1, \dots, |W|^3$ will satisfy

$$\forall w, w^{'} \in W, w \neq w^{'} mod\, p$$

Assumptions:
s cells b bits per cell
q solves with ≤ t probes
Updates made with ≤ t probes

Proof

To Show
Each of ≤ t rounds:
Alice $O(t\log(kt))$ bits
Bob $O(t(b+\log|Q|)+\log(kt))$ bits

Alice

Bob

Simulates the updates: $u_1, \dots, u_k$

Fix W - set of cells that changed during updates

$$|W| \leq kt$$

Find a prime $p \leq |W|^3$ s.t.

$$\forall w, w' \in W, w \neq w' \bmod p$$

Sends p

log(kt) bits

Assumptions:
s cells b bits per cell
q solves with ≤ t probes
Updates made with ≤ t probes

To Show
Each of ≤ t rounds:
Alice $O(t\log(kt))$ bits
Bob $O(t(b+\log|Q|)+\log(kt))$ bits

Proof – cont.

Alice → **Bob**

Simulates the tree corresponding q

Implements probe to cell m by:

Sending m mod p

O(log(kt)) bits

Finds unique $w \in W, w = m \bmod p$

Sends address and the content of w

Address: log s bits
Content: b bits

Assumptions:
s cells b bits per cell
q solves with ≤ t probes
Updates made with ≤ t probes

To Show
Each of ≤ t rounds:
Alice $O(t\log(kt))$ bits
Bob $O(t(b+\log|Q|)+\log(kt))$ bits

Proof – cont.

Alice **Bob**

If $w = m$:

Alice has the content of m that Bob sent

If $w \neq m$:

The updates didn't change cell m

Alice has the initial value of m

There are only t probs

$O(t*\log(kt))$ bits

$O(t(b+\log s)+\log(kt))$ bits

prime p (sent once)

every loop

Assumptions:
s cells b bits per cell
q solves with ≤ t probes
Updates made with ≤ t probes

To Show
Each of ≤ t rounds:
Bob $O(t(b+\log|Q|)+\log(kt))$ bits

Proof - finish

- This complexity is small if:
  - s is known and small enough.
- But not necessarily if $s>|Q|$
  - Can happen: DB must keep answers to all queries, and be able to answer after future updates.
- The use of s not useful to get time lower bound!

Assumptions:
s cells b bits per cell
q solves with ≤ t probes
Updates made with ≤ t probes

Trick for saving bits

Bob

Uses encoding for w with $\log|Q|$ bits:

Conclude what q' is from the previous probes

If all correct the q' Bob found matches Alices

Sends $b+\log|Q|$ bits

$O(t(b+\log|Q|)+\log(kt))$ bits

Alice

If $w = m$:

$q = q'$

Alice has the content of m that Bob sent

If $w \neq m$:

The updates didn't change cell m

Alice has the initial value of m

Example 2

- U $-$ vector subspace of Z_2^n
- U can be updated using $add(u)$ operation
  - $add(u)$: replaces U by $span(U \cup \{u\})$
- Single query: $\dim(U)$
- Cell size: $b = n$ bits

- $\forall q \in \{0,1\}^n$ We need to answer: "$q \in U$?"

Example 2 – lower bound

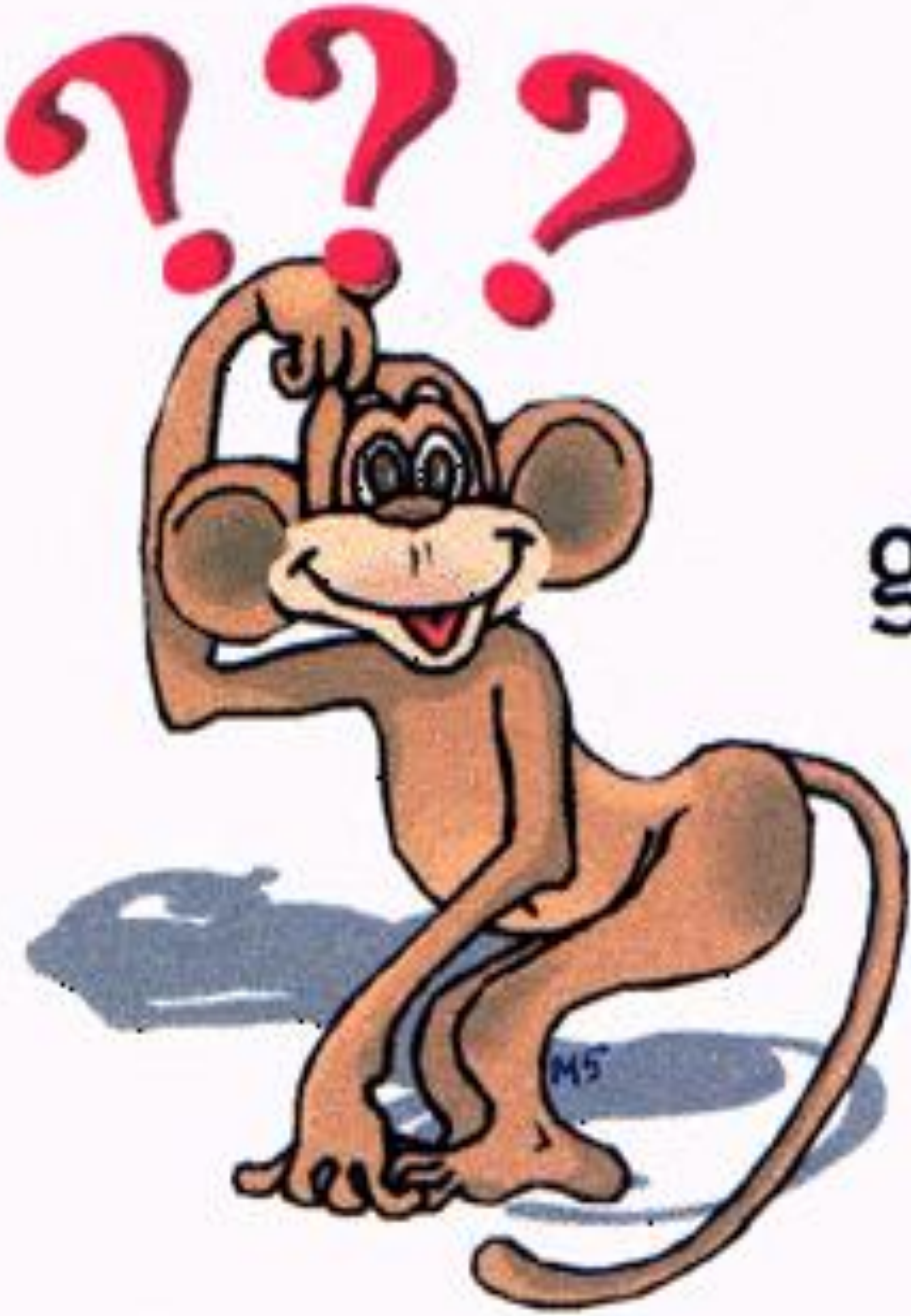
- Lower bound for the problem "$q \in U$?"
  - Reminder: $f(q,U)$ is the SPAN problem (last week)
  - We saw: Alice sends $\Omega(n)$ or Bob sends $\Omega(n^2)$
  - From lemma 2, using s-cell solution for subspace gives protocol for SPAN, satisfying:
    - Alice sends $o(t \log kt)$ bits
    - Bob sends $o(t(b + \log|Q|) + \log kt)$ bits
  - In both cases, by using lemma 2, When $|Q| = 2^n$

$$\boxed{\begin{array}{l} t \log kt = \Omega(n) \\ t(b + \log|Q|) + \log kt = \Omega(n^2) \end{array}}$$

number of probes needed :

$$t = \Omega(n/\log(n))$$

$U \subseteq Z_2^n$
$add(u) = span(U \cup \{u\})$
Query = $\dim(U)$
$b = n$ bits



Questions
are
guaranteed in
life;
Answers
aren't.


That's all Folks!