Models of Computation 1

Mayank Goswami

23.06.2014

Recap-last time

- Comparison trees
- Linear decision trees.
- Algebraic decision trees.
- **Today:** Algebraic computation trees, Ben-Or's theorem.

Recap-last time

- Comparison trees
- Linear decision trees.
- Algebraic decision trees.
- **Today:** Algebraic computation trees, Ben-Or's theorem.
- **Today:** Models of computations part 1: Word-RAM, pointer machine and indexability.

Problems we considered last time

- **Sorting**: Given a sequence $\{x_1, \cdots, x_n\}$ of n distinct numbers, find the permutation π such that $x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}$.
- **Convex hull**: Given a set of points $\{p_1, \cdots, p_n\}$ in the plane, compute their convex hull.
- **Compute the maximum element in a sequence**.
- **Element uniqueness**: Are any two elements of the input set $\{x_1, \cdots, x_n\}$ equal?
- **Set intersection**: Given two sets $\{x_1, \cdots, x_n\}$ and $\{y_1, \cdots, y_n\}$, do they intersect?
- **3SUM**: Do any three elements in the input set $\{x_1, \cdots, x_n\}$ sum to zero?

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).
- Retrieving a word takes constant time.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).
- Retrieving a word takes constant time.
- Any "reasonable" operation (integer addition/division, bitwise AND, OR, XOR, left/right shifts, comparisons) on words takes constant time.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).
- Retrieving a word takes constant time.
- Any "reasonable" operation (integer addition/division, bitwise AND, OR, XOR, left/right shifts, comparisons) on words takes constant time.
- $w = \Omega(\log n)$, where n is the size of the input allows for storing an index into the data structure in a single word.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).
- Retrieving a word takes constant time.
- Any "reasonable" operation (integer addition/division, bitwise AND, OR, XOR, left/right shifts, comparisons) on words takes constant time.
- $w = \Omega(\log n)$, where n is the size of the input allows for storing an index into the data structure in a single word.
- **Query time:** Number of instructions needed to answer a query.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).
- Retrieving a word takes constant time.
- Any "reasonable" operation (integer addition/division, bitwise AND, OR, XOR, left/right shifts, comparisons) on words takes constant time.
- $w = \Omega(\log n)$, where n is the size of the input allows for storing an index into the data structure in a single word.
- **Query time:** Number of instructions needed to answer a query.
- **Space usage:** Largest address used.

Models of Computation—word RAM

In general, the closer a model is to a real computer, the harder it is to prove lower bounds in that model.The computational model most geared towards designing algorithms is the unit cost word-RAM.

- Data structure is represented as a RAM, partitioned into words of w bits each; the words have integer addresses amongst $\{0, \cdots, 2^w - 1\}$(can store pointers).
- Retrieving a word takes constant time.
- Any "reasonable" operation (integer addition/division, bitwise AND, OR, XOR, left/right shifts, comparisons) on words takes constant time.
- $w = \Omega(\log n)$, where n is the size of the input allows for storing an index into the data structure in a single word.
- **Query time:** Number of instructions needed to answer a query.
- **Space usage:** Largest address used.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.
- We say data structure uses S cells of space if only cells of addresses $\{0, \cdots, S - 1\}$ are used.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.
- We say data structure uses S cells of space if only cells of addresses $\{0, \cdots, S - 1\}$ are used.
- Given a query, the data structure probes a number of cells from memory, and at the end must answer the query.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.
- We say data structure uses S cells of space if only cells of addresses $\{0, \cdots, S - 1\}$ are used.
- Given a query, the data structure probes a number of cells from memory, and at the end must answer the query.
- **The cell probed at the next step may be any deterministic function of the query and the contents of the previously probed cells.**

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.
- We say data structure uses S cells of space if only cells of addresses $\{0, \cdots, S - 1\}$ are used.
- Given a query, the data structure probes a number of cells from memory, and at the end must answer the query.
- **The cell probed at the next step may be any deterministic function of the query and the contents of the previously probed cells.**
- Thus, all computations on read data are free of charge; arbitrary instructions allowed.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.
- We say data structure uses S cells of space if only cells of addresses $\{0, \cdots, S - 1\}$ are used.
- Given a query, the data structure probes a number of cells from memory, and at the end must answer the query.
- **The cell probed at the next step may be any deterministic function of the query and the contents of the previously probed cells.**
- Thus, all computations on read data are free of charge; arbitrary instructions allowed.
- **Query cost:** Number of cells probed.

Cell probe

We do not want lower bounds to depend on the exact set of machine instructions available.The cell-probe model is a less restrictive version of word-RAM:

- Memory of cells, each w bits long, addresses in $\{0, \cdots, 2^w - 1\}$.
- We say data structure uses S cells of space if only cells of addresses $\{0, \cdots, S - 1\}$ are used.
- Given a query, the data structure probes a number of cells from memory, and at the end must answer the query.
- **The cell probed at the next step may be any deterministic function of the query and the contents of the previously probed cells.**
- Thus, all computations on read data are free of charge; arbitrary instructions allowed.
- **Query cost:** Number of cells probed.
- Dynamic—**Update cost:** the number of cells read/written when performing an update

Need for simpler models

The cell probe model is the least restrictive of all lower bound models proposed in the literature. Thus, lower bounds proved here are very general.

Need for simpler models

The cell probe model is the least restrictive of all lower bound models proposed in the literature. Thus, lower bounds proved here are very general.Unfortunately,...

Need for simpler models

The cell probe model is the least restrictive of all lower bound models proposed in the literature. Thus, lower bounds proved here are very general.Unfortunately,...Today we will consider:

- Pointer machine model(Tarjan)

Need for simpler models

The cell probe model is the least restrictive of all lower bound models proposed in the literature. Thus, lower bounds proved here are very general.Unfortunately,...Today we will consider:

- Pointer machine model(Tarjan)
- Indexability model(Papadimitrou et. al)

It is easier to prove high lower bounds in these models.

Need for simpler models

The cell probe model is the least restrictive of all lower bound models proposed in the literature. Thus, lower bounds proved here are very general.Unfortunately,...Today we will consider:

- Pointer machine model(Tarjan)
- Indexability model(Papadimitrou et. al)

It is easier to prove high lower bounds in these models.
Finally we will consider the case when data is too large to fit into main memory. This is called the external memory model.

Problem of the day

The main problem we will be concerned is *orthogonal range reporting*:

- **Input:** A set of points $P = \{p_1, \cdots, p_n\}$ in the plane; $p_i = (x_i, y_i)$.
- **Preprocessing phase:** Data structure allowed some time(space) to preprocess P in order to answer...
- **Query:** Given a rectangle $[x_\ell, x_r] \times [y_\ell, y_r]$, report all points that lie inside this rectangle.

Problem of the day

The main problem we will be concerned is *orthogonal range reporting*:

- **Input:** A set of points $P = \{p_1, \cdots, p_n\}$ in the plane; $p_i = (x_i, y_i)$.
- **Preprocessing phase:** Data structure allowed some time(space) to preprocess P in order to answer...
- **Query:** Given a rectangle $[x_\ell, x_r] \times [y_\ell, y_r]$, report all points that lie inside this rectangle.
- We study tradeoffs between space and query time. Usually space is $O(\text{poly}(n))$, and query is $O(\text{poly}(\log n) + k)$, where k is the set of output points.
- Problem can be generalized to $\mathbb{R}^n$.

Status of 2-d Orthogonal range reporting in PMM

- Filtering search

Status of 2-d Orthogonal range reporting in PMM

- Filtering search
  - Space $O(n \log n / \log \log n)$, query $O(k + \log n)$.
  - No address manipulations are required.

Status of 2-d Orthogonal range reporting in PMM

- Filtering search
  - Space $O(n \log n / \log \log n)$, query $O(k + \log n)$.
  - No address manipulations are required.
  - This factor can be removed in many variants of the problem—grounded rectangle, fixed aspect ratio, grounded trapezoid.

Status of 2-d Orthogonal range reporting in PMM

- Filtering search
  - Space $O(n \log n / \log \log n)$, query $O(k + \log n)$.
  - No address manipulations are required.
  - This factor can be removed in many variants of the problem—grounded rectangle, fixed aspect ratio, grounded trapezoid.
- In RAM, one can actually achieve same query time with $O(n \log^{\epsilon} n)$ space, for any fixed $\epsilon > 0$.

Status of 2-d Orthogonal range reporting in PMM

- Filtering search
  - Space $O(n \log n / \log \log n)$, query $O(k + \log n)$.
  - No address manipulations are required.
  - This factor can be removed in many variants of the problem—grounded rectangle, fixed aspect ratio, grounded trapezoid.
- In RAM, one can actually achieve same query time with $O(n \log^{\epsilon} n)$ space, for any fixed $\epsilon > 0$.
- Is this space/query tradeoff the best possible?

Definition: Pointer machine model

In a nutshell: No address computation, indivisible storage.

Definition: Pointer machine model

In a nutshell: No address computation, indivisible storage.

- Memory is an unbounded collection of registers.
- Each register is a record with a fixed number of data and pointer fields.

Definition: Pointer machine model

In a nutshell: No address computation, indivisible storage.

- Memory is an unbounded collection of registers.
- Each register is a record with a fixed number of data and pointer fields.
- Thus the memory can be modeled as a directed graph with bounded outdegree.

Definition: Pointer machine model

In a nutshell: No address computation, indivisible storage.

- Memory is an unbounded collection of registers.
- Each register is a record with a fixed number of data and pointer fields.
- Thus the memory can be modeled as a directed graph with bounded outdegree.
- Increasing the storage by a multiplicative factor(duplicating nodes), we can assume outdegree is at most 2.
- Given $\{p_1, \cdots, p_n\}$, a data structure for orthogonal range reporting is a digraph $G = (V, E)$ with a source node σ.

Definition: Pointer machine model

- Each node v has a label, label(v) between 0 and n.
- If the label is i, then the node corresponds to point p_i. If the label is 0, then the node is used internally by the data structure, not to report any point.
- Many nodes can be associated with the same point.

Definition: Pointer machine model

- Each node v has a label, label(v) between 0 and n.
- If the label is i, then the node corresponds to point p_i. If the label is 0, then the node is used internally by the data structure, not to report any point.
- Many nodes can be associated with the same point.
- Given a query q, the algorithm starts traversing G at σ.

Definition: Pointer machine model

- Each node v has a label, label(v) between 0 and n.
- If the label is i, then the node corresponds to point p_i. If the label is 0, then the node is used internally by the data structure, not to report any point.
- Many nodes can be associated with the same point.
- Given a query q, the algorithm starts traversing G at σ.
  - No node can be visited if a node pointing to it has not already been visited.
  - Completion cannot occur until each label i such that $p_i \in q$ has been encountered at least once.

Definition: Pointer machine model

- Algorithm can modify data and address fields of visited nodes.
- Can add new nodes from *freelist*.
- $N(v) = \{w | (v, w) \in E\}$. Set $W = \{\sigma\}$ initially.
  - Pick any $v \in W$ and add $N(v)$ to W.

Definition: Pointer machine model

- Algorithm can modify data and address fields of visited nodes.
- Can add new nodes from *freelist*.
- $N(v) = \{w | (v, w) \in E\}$. Set $W = \{\sigma\}$ initially.
  - Pick any $v \in W$ and add $N(v)$ to W.
  - Request a new node v from the freelist and add it to W. Initialize $N(v)$ to $\emptyset$.

Definition: Pointer machine model

- Algorithm can modify data and address fields of visited nodes.
- Can add new nodes from *freelist*.
- $N(v) = \{w|(v, w) \in E\}$. Set $W = \{\sigma\}$ initially.
  - Pick any $v \in W$ and add $N(v)$ to W.
  - Request a new node v from the freelist and add it to W. Initialize $N(v)$ to $\emptyset$.
  - Pick any v,$w \in W$ and if $|N(v)| < 2$, add (v, w) to E (and v or w to V, if necessary).

Definition: Pointer machine model

- Algorithm can modify data and address fields of visited nodes.
- Can add new nodes from *freelist*.
- $N(v) = \{w|(v,w) \in E\}$. Set $W = \{\sigma\}$ initially.
  - Pick any $v \in W$ and add $N(v)$ to W.
  - Request a new node v from the freelist and add it to W. Initialize $N(v)$ to $\emptyset$.
  - Pick any v,$w \in W$ and if $|N(v)| < 2$, add (v,w) to E (and v or w to V, if necessary).
  - Pick any $v, w \in W$ and remove the edge (v,w) from E if it exists.

  At the end $W(q)$ must contain at least one node corresponding to each output point in the range q. **Query time is measured as $|W(q)|$.**

Definition: Pointer machine model

- Algorithm can modify data and address fields of visited nodes.
- Can add new nodes from *freelist*.
- $N(v) = \{w|(v,w) \in E\}$. Set $W = \{\sigma\}$ initially.
  - Pick any $v \in W$ and add $N(v)$ to W.
  - Request a new node v from the freelist and add it to W. Initialize $N(v)$ to $\emptyset$.
  - Pick any v,$w \in W$ and if $|N(v)| < 2$, add (v,w) to E (and v or w to V, if necessary).
  - Pick any $v, w \in W$ and remove the edge (v,w) from E if it exists.

  At the end $W(q)$ must contain at least one node corresponding to each output point in the range q. **Query time is measured as $|W(q)|$.**

Lower bound for ORR in PMM

Remarks: “Pick any”,

Lower bound for ORR in PMM

Remarks: "Pick any",why knowing i is not sufficient.

Lower bound for ORR in PMM

Remarks: "Pick any",why knowing i is not sufficient.This is a big difference between RAM and a pointer machine.

- Let $a, b \in \mathbb{R}^+$. $G = (V, E)$ is (a, b)-effective if for any query range q, $|W(q)| \leq a(|P \cap q| + \log^b n)$.

Lower bound for ORR in PMM

Remarks: "Pick any",why knowing i is not sufficient.This is a big difference between RAM and a pointer machine.

- Let $a, b \in \mathbb{R}^+$. $G = (V, E)$ is (a, b)-effective if for any query range q, $|W(q)| \leq a(|P \cap q| + \log^b n)$.
- Main idea to get a lower bound:
  - Nodes corresponding to output points cannot be too spread out in the graph, otherwise it would be impossible to get them in linear time.
  - Let $S = \{q_1, \cdots, q_s\}$ be a set of queries, and let $\alpha > 0$.
  - S is α-favorable if, for each $i, j (i \neq j)$,
  - $|P \cap q_i| \geq \log^\alpha n$.
  - $|P \cap q_i \cap q_j| \leq 1$.

Main theorem in proving lower bounds in PMM

Theorem

If the data structure is (a, b)-effective and the set of queries is b-favorable, then

$$|V| > |S| \frac{\log^b n}{2^{16a+4}},$$

for n large enough.

Thus the main task is to get a large set of queries, no two of which share too many points of P...

Main theorem in proving lower bounds in PMM

Theorem

If the data structure is (a, b)-effective and the set of queries is b-favorable, then

$$|V| > |S|\frac{\log^b n}{2^{16a+4}},$$

for n large enough.

Thus the main task is to get a large set of queries, no two of which share too many points of P...Proof on board.