Models of Computation 1

Mayank Goswami

23.06.2014

Last time

- Algebraic decision trees, Ben-Or's theorem.

Last time

- Algebraic decision trees, Ben-Or's theorem.
- Pointer machine model.

Last time

- Algebraic decision trees, Ben-Or's theorem.
- Pointer machine model.
- Lower bound on orthogonal range reporting in the pointer machine model.

Last time

- Algebraic decision trees, Ben-Or's theorem.
- Pointer machine model.
- Lower bound on orthogonal range reporting in the pointer machine model.
- Today:
  - Indexability model: orthogonal range reporting.

Last time

- Algebraic decision trees, Ben-Or's theorem.
- Pointer machine model.
- Lower bound on orthogonal range reporting in the pointer machine model.
- Today:
  - Indexability model: orthogonal range reporting.
  - Cell probe model: membership.

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.

Indexability model

- External memory: A disk page can hold B elements; main memory of size M. One I/O (input/output) transfers B elements(one page) to/from external memory.

Indexability model

- External memory: A disk page can hold B elements; main memory of size M. One I/O (input/output) transfers B elements(one page) to/from external memory.
- The indexability model captures the spatial issues that arise when laying out data on the disk.

Indexability model

- External memory: A disk page can hold B elements; main memory of size M. One I/O (input/output) transfers B elements(one page) to/from external memory.
- The indexability model captures the spatial issues that arise when laying out data on the disk.
- Workload $W = (I, Q)$, where I is a finite set, and Q is a set of subsets of I.
- $N = |I|$, $q = |Q|$.

Indexability model

- External memory: A disk page can hold B elements; main memory of size M. One I/O (input/output) transfers B elements(one page) to/from external memory.
- The indexability model captures the spatial issues that arise when laying out data on the disk.
- Workload $W = (I, Q)$, where I is a finite set, and Q is a set of subsets of I.
- $N = |I|$, $q = |Q|$.
- An indexing scheme (block size $B > 1$), $S = (W, \mathcal{B})$ such that W is a workload, and $\mathcal{B}$ is a family of B-sized subsets of I, such that $\mathcal{B}$ covers I.
- We will call elements of $\mathcal{B}$ as blocks. Let $K = |\mathcal{B}|$ be the number of blocks.

Indexability model

- **Main assumption:** We do not need to search. Once a query arrives, we are told for free which disk pages contain answers to that query.

Indexability model

- **Main assumption:** We do not need to search. Once a query arrives, we are told for free which disk pages contain answers to that query.
- We want to lay out the data on the disk so that we can retrieve answers fast.

Indexability model

- **Main assumption:** We do not need to search. Once a query arrives, we are told for free which disk pages contain answers to that query.
- We want to lay out the data on the disk so that we can retrieve answers fast.
- Ideal space usage is N/B. The **redundancy** of an indexing scheme is defined as the factor by which we overshoot this.
- In other words, $r = K/(N/B)$.

Indexability model

- **Main assumption:** We do not need to search. Once a query arrives, we are told for free which disk pages contain answers to that query.
- We want to lay out the data on the disk so that we can retrieve answers fast.
- Ideal space usage is N/B. The **redundancy** of an indexing scheme is defined as the factor by which we overshoot this.
- In other words, $r = K/(N/B)$.
- Ideal query time is q/B, where q is the number of output elements. The access overhead $A(Q)$ of a query Q is defined as the factor by which we overshoot this.
- In other words, if we require $A(Q).q/B$ disk accesses, then we say $A(Q)$ is the access overhead.

Indexability model

- **Main assumption:** We do not need to search. Once a query arrives, we are told for free which disk pages contain answers to that query.
- We want to lay out the data on the disk so that we can retrieve answers fast.
- Ideal space usage is N/B. The **redundancy** of an indexing scheme is defined as the factor by which we overshoot this.
- In other words, $r = K/(N/B)$.
- Ideal query time is q/B, where q is the number of output elements. The access overhead $A(Q)$ of a query Q is defined as the factor by which we overshoot this.
- In other words, if we require $A(Q).q/B$ disk accesses, then we say $A(Q)$ is the access overhead.
- The access overhead A of the indexing scheme is the maximum access overhead over all queries Q.

ORR in Indexability model

- The whole game is to get tradeoffs between redundancy and access overhead.

ORR in Indexability model

- The whole game is to get tradeoffs between redundancy and access overhead.
- Then one designs an indexing scheme matching this tradeoff. This means designing something that assumes search for free, and is only concerned with the layout.

ORR in Indexability model

- The whole game is to get tradeoffs between redundancy and access overhead.
- Then one designs an indexing scheme matching this tradeoff. This means designing something that assumes search for free, and is only concerned with the layout.
- The last step is building a data structure that also takes the search aspect into the picture.

ORR in Indexability model

- The whole game is to get tradeoffs between redundancy and access overhead.
- Then one designs an indexing scheme matching this tradeoff. This means designing something that assumes search for free, and is only concerned with the layout.
- The last step is building a data structure that also takes the search aspect into the picture.Main tool for lower bounds?

ORR in Indexability model

- The whole game is to get tradeoffs between redundancy and access overhead.
- Then one designs an indexing scheme matching this tradeoff. This means designing something that assumes search for free, and is only concerned with the layout.
- The last step is building a data structure that also takes the search aspect into the picture.Main tool for lower bounds?

Theorem

Let S be an indexing scheme, and $Q_1, \cdots, Q_M$ be queries, such that for every $1 \leq i \leq M$,

1. $|Q_i| \geq B$
2. $|Q_i \cap Q_j| \leq \frac{B}{2(\epsilon A)^2}$.

Then

$$r \geq \frac{\epsilon - 2}{2\epsilon} . \frac{1}{N} \sum_{i=1}^{M} |Q_i|$$