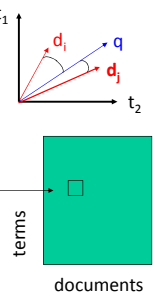
CSE 454

Search Engine Query Processing
Alta Vista Case Study

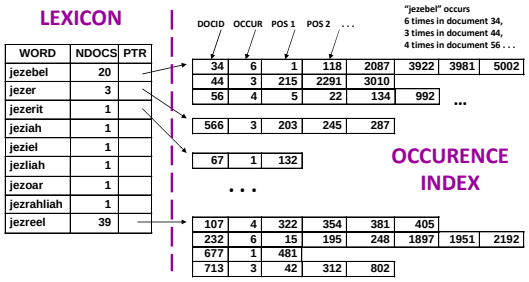
Based on a talk by Mike Burrows

Review

- **Vector Space Representation**
  - Dot Product as Similarity Metric
- **TF-IDF for Computing Weights**
  - $w_{ij} = f(i,j) * log(N/n_i)$
  - Where q = … $word_i$…
  - N = |docs| n_i = |docs with $word_i$|
- **But How Process Efficiently?**



Inverted Files for Multiple Documents



Many Variations Possible

- **Address space (flat, hierarchical)**
  - Alta Vista uses flat approach
- **Record term-position information**
- **Precalculate TF-IDF info**
- **Stored header, font & tag info**
- **Compression strategies**

How Process Complex Queries?

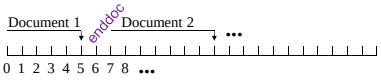
- **Phrases, term in anchor text, etc**

AltaVista: Inverted Files

- **Map each word to list of locations where it occurs**
- **Words = null-terminated byte strings**
- **Locations = 64 bit unsigned ints**
  - Layer above gives interpretation for location
    - URL
    - Index into text specifying word number
- **Preview:**
  - ***Index Stream Readers***: four primitives: loc, seek, …
  - ***Constraint Solvers:*** four constraints: $loc_A \leq loc_B+K$, …

Documents

- **A document is a region of location space**
  - Contiguous
  - No overlap
  - Densely allocated (first doc is location 1)
- **All document structure encoded with words**
  - enddoc at last location of document
  - begintitle, endtitle mark document title

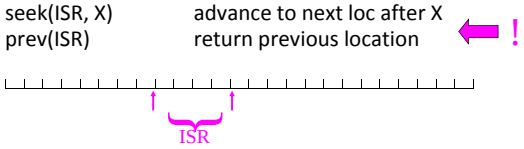


Format of Inverted Files

- **Words ordered lexicographically**
- **Each word followed by list of locations**
- **Common word prefixes are compressed**
- **Locations encoded as deltas**
  - Stored in as few bytes as possible
  - 2 bytes is common
  - Sneaky assembly code for operations on inverted files
    - Pack deltas into aligned 64 bit word
    - First byte contains continuation bits
    - Table lookup on byte => no branch instructs, no mispredicts
    - 35 parallelized instructions/ 64 bit word = 10 cycles/word
- **Index ~ 10% of text size**

Index Stream Readers (ISRs)

- **Interface for**
  - Reading result of query
  - Return ascending sequence of locations
  - Implemented using lazy evaluation
- **Methods**
  - loc(ISR) return current location
  - next(ISR) advance to next location
  - seek(ISR, X) advance to next loc after X
  - prev(ISR) return previous location

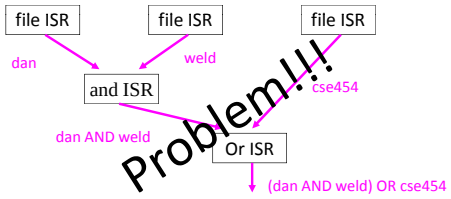


Processing Simple Queries

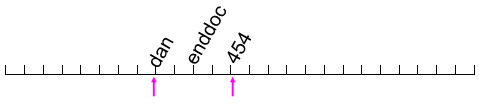
- **User searches for "weld"**
- **Open ISR on "weld"**
  - Uses hash table to avoid scanning entire file
- **Next(), next(), next()**
  - returns locations containing the word

Combining ISRs

- **And** Compare locs on two streams
- **Or** Merges two or more ISRs
- **Not** Returns locations not in ISR (lazily)

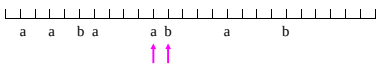


What About File Boundaries?



ISR Constraint Solver

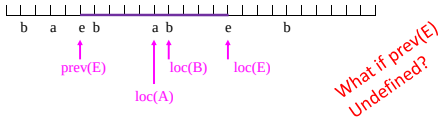
- **Inputs:**
  - Set of ISRs: A, B, ...
  - Set of Constraints
- **Constraint Types**
  - loc(A) ≤ loc(B) + K
  - prev(A) ≤ loc(B) + K
  - loc(A) ≤ prev(B) + K
  - prev(A) ≤ prev(B) + K
- **For example: phrase "a b"**
  - loc(A) ≤ loc(B), loc(B) ≤ loc(A) + 1



Two words on one page

- **Let E be ISR for word** enddoc
- **Constraints for a AND b**
  - prev(E) ≤ loc(A)
  - loc(A) ≤ loc(E)
  - prev(E) ≤ loc(B)
  - loc(B) ≤ loc(E)

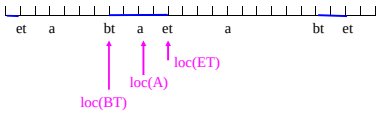
loc(A) ≤ loc(B) + K
prev(A) ≤ loc(B) + K
loc(A) ≤ prev(B) + K
prev(A) ≤ prev(B) + K



Advanced Search

- **Query: a in Title of page**
- **Let *BT*, *ET* be ISRP of words**
  - BeginTitle & EndTitle
- **Constraints:**
  - loc(BT) ≤ loc(A)
  - loc(A) ≤ loc(ET)

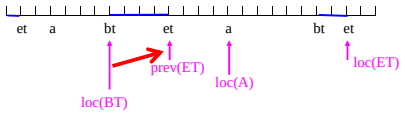
loc(A) ≤ loc(B) + K
prev(A) ≤ loc(B) + K
loc(A) ≤ prev(B) + K
prev(A) ≤ prev(B) + K



Advanced Search

- **Query: a in Title of page**
- **Let *BT*, *ET* be ISRP of words**
  - BeginTitle & EndTitle
- **Constraints:**
  - loc(BT) ≤ loc(A)
  - loc(A) ≤ loc(ET)
  - prev(ET) ≤ loc(BT) X

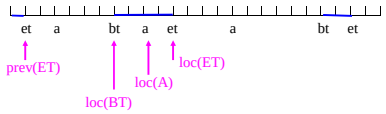
loc(A) ≤ loc(B) + K
prev(A) ≤ loc(B) + K
loc(A) ≤ prev(B) + K
prev(A) ≤ prev(B) + K



Advanced Search

- **Query: a in Title of page**
- **Let *BT*, *ET* be ISRP of words**
  - BeginTitle & EndTitle
- **Constraints:**
  - loc(BT) ≤ loc(A)
  - loc(A) ≤ loc(ET)
  - prev(ET) ≤ loc(BT)

loc(A) ≤ loc(B) + K
prev(A) ≤ loc(B) + K
loc(A) ≤ prev(B) + K
prev(A) ≤ prev(B) + K



Neat, Huh?

- **Very flexible**
- **Eg "Who points to my page?"**
- **Are we done?**

Implementing the Solver

Constraint Types

- loc(A) ≤ loc(B) + K
- prev(A) ≤ loc(B) + K
- loc(A) ≤ prev(B) + K
- prev(A) ≤ prev(B) + K

Remember: Index Stream Readers

ISR Methods

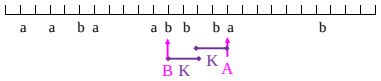
- loc(ISR) return current location
- next(ISR) advance to next location
- seek(ISR, X) advance to next loc after X
- prev(ISR) return previous location

Solver Algorithm

loc(ISR)	return cur loc
next(ISR)	adv to nxt loc return it
seek(ISR, X)	adv to nxt loc > X return it
prev(ISR)	return pre loc

while (unsatisfied_constraints)
satisfy_constraint(choose_unsat_constraint())

- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)

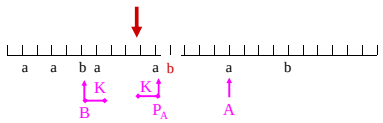


Solver Algorithm

loc(ISR)	return cur loc
next(ISR)	adv to nxt loc return it
seek(ISR, X)	adv to nxt loc > X return it
prev(ISR)	return pre loc

while (unsatisfied_constraints)
satisfy_constraint(choose_unsat_constraint())

- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)
- **To satisfy: prev(A) ≤ loc(B) + K**
  - Execute: seek(B, prev(A) - K)

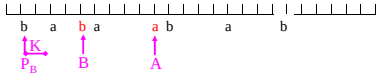


Solver Algorithm

loc(ISR)	return cur loc
next(ISR)	adv to nxt loc return it
seek(ISR, X)	adv to nxt loc > X return it
prev(ISR)	return pre loc

while (unsatisfied_constraints)
satisfy_constraint(choose_unsat_constraint())

- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)
- **To satisfy: prev(A) ≤ loc(B) + K**
  - Execute: seek(B, prev(A) - K)
- **To satisfy: loc(A) ≤ prev(B) + K**

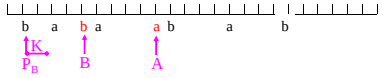


Solver Algorithm

loc(ISR)	return cur loc
next(ISR)	adv to nxt loc return it
seek(ISR, X)	adv to nxt loc > X return it
prev(ISR)	return pre loc

while (unsatisfied_constraints)
satisfy_constraint(choose_unsat_constraint())

- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)
- **To satisfy: prev(A) ≤ loc(B) + K**
  - Execute: seek(B, prev(A) - K)
- **To satisfy: loc(A) ≤ prev(B) + K**
  - Execute: seek(B, loc(A) - K),
  -

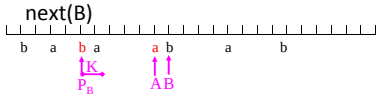


Solver Algorithm

loc(ISR)	return cur loc
next(ISR)	adv to nxt loc return it
seek(ISR, X)	adv to nxt loc > X return it
prev(ISR)	return pre loc

while (unsatisfied_constraints)
satisfy_constraint(choose_unsat_constraint())

- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)
- **To satisfy: prev(A) ≤ loc(B) + K**
  - Execute: seek(B, prev(A) - K)
- **To satisfy: loc(A) ≤ prev(B) + K**
  - Execute: seek(B, loc(A) - K),
  - next(B)



Solver Algorithm

loc(ISR)	return cur loc
next(ISR)	adv to nxt loc return it
seek(ISR, X)	adv to nxt loc > X return it
prev(ISR)	return pre loc

while (unsatisfied_constraints)
satisfy_constraint(choose_unsat_constraint())

- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)
- **To satisfy: prev(A) ≤ loc(B) + K**
  - Execute: seek(B, prev(A) - K)
- **To satisfy: loc(A) ≤ prev(B) + K**
  - Execute: seek(B, loc(A) - K),
  - next(B)
- **To satisfy: prev(A) ≤ prev(B) + K**
  - Execute: seek(B, prev(A) - K)
  - next(B)

Solver Algorithm

```
while (unsatisfied_constraints)
    satisfy_constraint(choose_unsat_constraint())
```

Heuristic??

Which choice advances a stream the furthest?

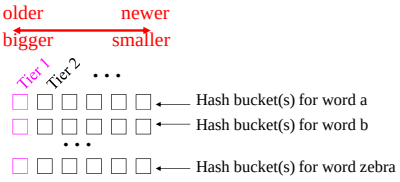
- **To satisfy: loc(A) ≤ loc(B) + K**
  - Execute: seek(B, loc(A) - K)
- **To satisfy: prev(A) ≤ loc(B) + K**
  - Execute: seek(B, prev(A) - K)
- **To satisfy: loc(A) ≤ prev(B) + K**
  - Execute: seek(B, loc(A) - K),
  - next(B)
- **To satisfy: prev(A) ≤ prev(B) + K**
  - Execute: seek(B, prev(A) - K)
  - next(B)

Update

- **Can't insert in the middle of an inverted file**
- **Must rewrite the entire file**
  - Naïve approach: need space for two copies
  - Slow since file is huge
- **Split data along two dimensions**
  - **Buckets** solve disk space problem
  - **Tiers** alleviate small update problem

Buckets & Tiers

- **Each word is hashed to a bucket**
- **Add new documents by adding a new tier**
  - Periodically merge tiers, bucket by bucket



What if Word ***Removed*** from Doc?

- **Delete documents by adding deleted word**

- **Expunge deletions when merging tier 1**