CSE 454 - Case Studies

Design of Alta Vista

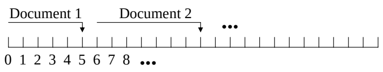
Based on a talk by Mike Burrows

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
- **Slides adapted from talk by Mike Burrows**

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

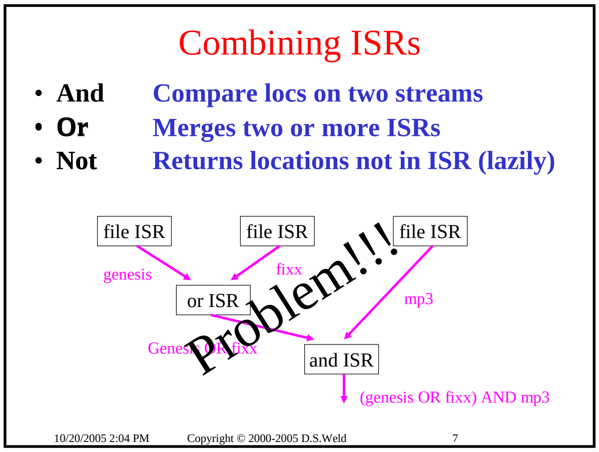
- **Words ordered lexicographically**
- **Each word followed by list of locations**
- **Common word prefixes are compressed**
- **Locations encoded as deltas**
  - Stored in as few bytes as possible
  - 2 bytes is common
  - Sneaky assembly code for operations on inverted files
    - Pack deltas into 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

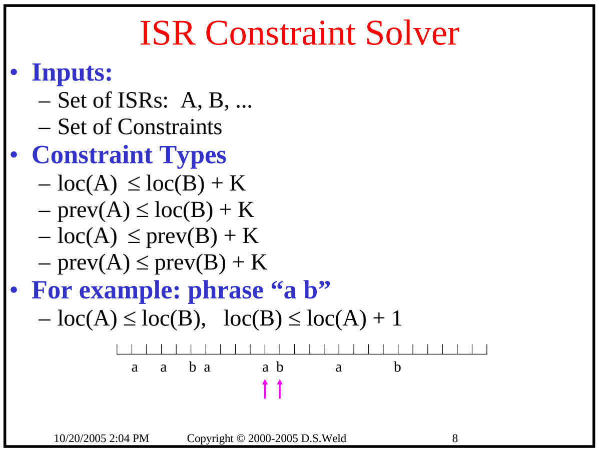
Processing Simple Queries

- **User searches for "mp3"**
- **Open ISR on "mp3"**
  - Uses hash table to avoid scanning entire file
- **Next(), next(), next()**
  - returns locations containing the word



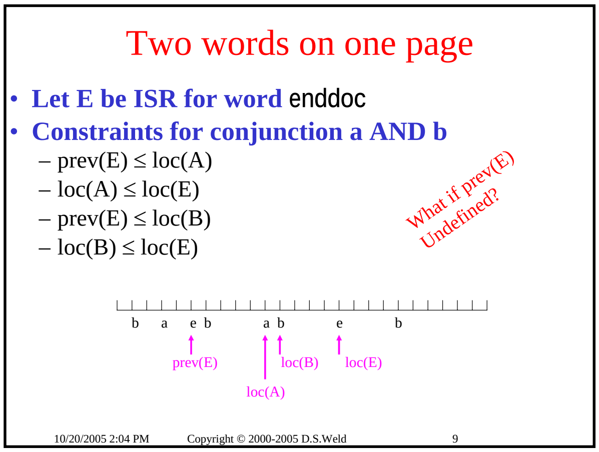
Combining ISRs

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



ISR Constraint Solver

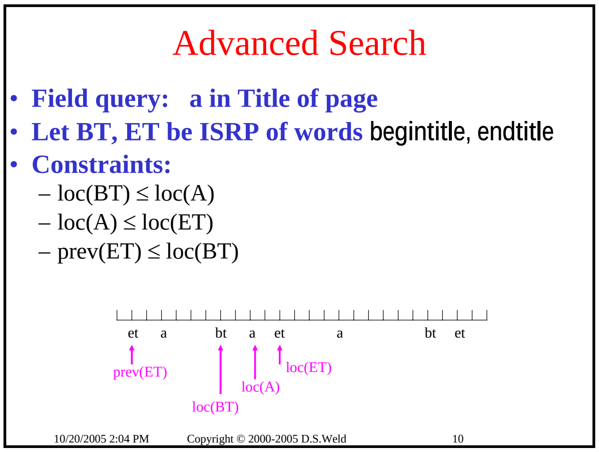
- **Inputs:**
  - Set of ISRs: A, B, ...
  - Set of Constraints
- **Constraint Types**
  - loc(A) ≤ loc(B) + K
  - prev(A) ≤ loc(B) + K
  - loc(A) ≤ prev(B) + K
  - prev(A) ≤ prev(B) + K
- **For example: phrase “a b”**
  - loc(A) ≤ loc(B), loc(B) ≤ loc(A) + 1



Two words on one page

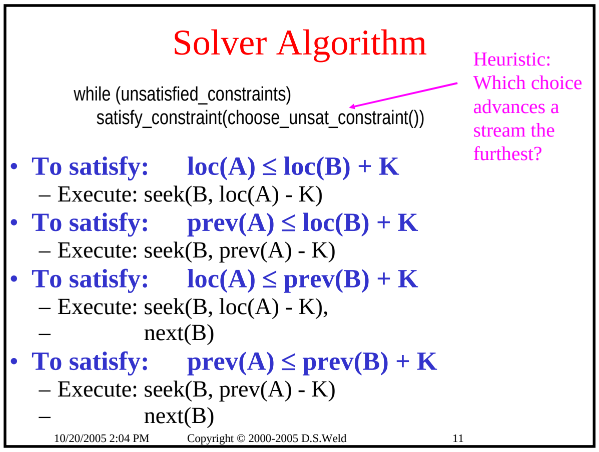
- **Let E be ISR for word** enddoc
- **Constraints for conjunction a AND b**
  - prev(E) ≤ loc(A)
  - loc(A) ≤ loc(E)
  - prev(E) ≤ loc(B)
  - loc(B) ≤ loc(E)

What if prev(E) Undefined?



Advanced Search

- **Field 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)

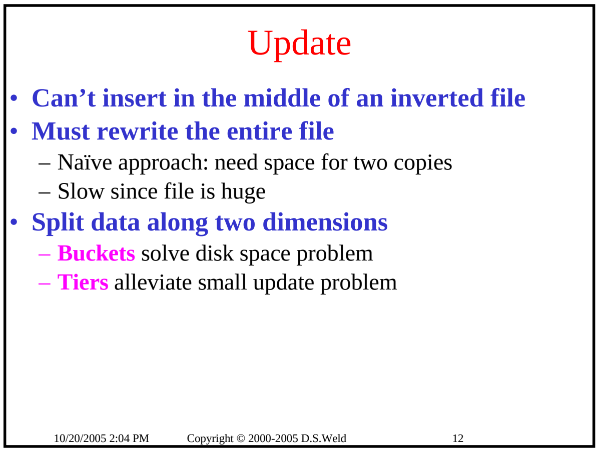


Solver Algorithm

```
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)

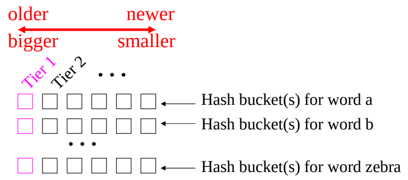


Update

- **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
- **Delete documents by adding** deleted **word**
  - Expunge deletions when merging tier 0

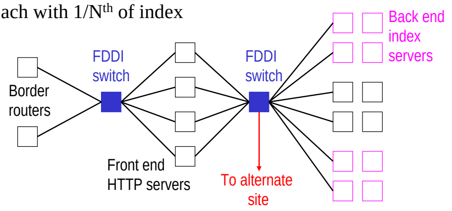

older
newer
bigger
smaller
Tier 1
Tier 2
• • •
Hash bucket(s) for word a
Hash bucket(s) for word b
• • •
Hash bucket(s) for word zebra

Scaling

- **How handle huge traffic?**
  - AltaVista Search ranked #16
  - 10,674,000 unique visitors (Dec'99)
- **Scale across N hosts**
  1. Ubiquitous index. Query one host
  2. Split N ways. Query all, merge results
  3. Ubiquitous index. Host handles subrange of locations. Query all, merge results
  4. Hybrids

AltaVista Structure

- **Front ends**
  - Alpha workstations
- **Back ends**
  - 4-10 CPU Alpha servers
    - 8GB RAM, 150GB disk
  - Organized in groups of 4-10 machines
    - Each with 1/Nth of index


Back end
index
servers
FDDI
switch
FDDI
switch
Border
routers
Front end
HTTP servers
To alternate
site