Copyright ©1998, 1999 Timothy Howard Merrett
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation in a prominent place. Copyright for components of this work owned by others than T. H. Merrett must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to republish from: T. H. Merrett, School of Computer Science, McGill University, fax 514 398 3883.

The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

T. H. Merrett

©98/10
Key-to-Address Transformation Scaling

Key space (e.g., $10^9$ SINs)

Key-to-address transformation (scaling)

Address space (e.g., $10^5$ blocks)

(not to scale)
Key-to-Address Transformation
Randomizing

Key space (e.g., $10^9$ SINs)

Key-to-address transformation (randomizing)

Address space (e.g., $10^5$ blocks)
(not to scale)
Key-to-Address Transformation
Order-preserving

Key space (e.g., $10^9$ SINs)

Key-to-address transformation
(order-preserving)

Address space (e.g., $10^5$ blocks)
(not to scale)
Direct Access
(Compute block address from keys, instead of comparing them.)

Hash Functions
(Randomizing key-to-address transformations)

- Hash functions
  - Division-remainder, $h(k) = k \mod n$, $n$ prime
  - Multiplicative, $h(k) =$ middle $m$ bits of $Ak$, $n = 2^m$

- Collision resolution
  - Linear probing (cyclically downward, no pointers)
  - Separate chaining (pointer chain from each block)
  - $\mathcal{O}(N)$, worst case when all keys hash to one block or (linear probing) when file full and all blocks in an overflow chain.
Hash Algorithms

- Algorithm SS (Hash search with separate chaining)
- Algorithm SI (Hash insert with separate chaining)
- Algorithm SD (Hash delete with separate chaining)
- Algorithm LS (Hash search with linear probing)
- Algorithm LI (Hash insert with linear probing)
- Algorithm LD (Hash delete with linear probing)
Algorithm LD: Delete Record  \( r \) Found on Block  \( f \)

1. Original deletion

\[
\begin{array}{cccc}
0 & f & h(r) & n-1 \\
\end{array}
\]

2. Working downwards

\[
\begin{array}{cccc}
f & h(r) & \text{oldf} \\
\end{array}
\]

**LD1** (Remove record)
\[ \ell \leftarrow \text{loc}(r); \; \text{oldf} \leftarrow f; \; \text{mark} \; \ell \; \text{on} \; \text{oldf empty} \]

**LD2** (Move later records up)
if block  \( f \) not full (apart from deletion, if any, just made by LD1), stop.
\[ f \leftarrow \text{if } f = 0 \text{ then } n - 1 \text{ else } f - 1 \]
if  \( f \) = original home block, stop.
    
    /* else full file can thrash */

for each record,  \( r \), on block  \( f \)
    
    if  \( \text{ncycle}(f, h(r), \text{oldf}) \)
    /*  \( h(r) \) not cyclically between  \( f \),  \( \text{oldf} \)
    or =  \( \text{oldf} \); i.e.,  \( h(r) \) at or beyond  \( \text{oldf}^* \) */
        
        then copy  \( r \) to  \( \ell \) on  \( \text{oldf} \); goto LD1
    goto LD2

T. H. Merrett
Analysing Hash Functions

Parameters

- $\alpha = \frac{N}{nb}$, load factor
- $\pi = \frac{\text{total no. probes}}{N}$, probe factor
- $p_{iop} = 1 + \frac{\text{total overflows}}{N}$, optimistic probe factor

For ideal $h(k)$, given one of $n$ blocks,
prob(1 record hashes to it) = $\frac{1}{n}$
prob($k$ records hash to it) = $(\frac{1}{n})^k$
prob(exactly $k$ of $N$ hash to it)
  = $(\frac{1}{n})^k (1 - \frac{1}{n})^{N-k}$
prob(any $k$ of $N$ records hash to it)
  = $\binom{N}{k} (\frac{1}{n})^k (1 - \frac{1}{n})^{N-k}$

This binomial distribution, $B(k, N, n) = P(k, \frac{N}{n}) + O\left(\frac{1}{n}\right)$
where the Poisson distribution, $P(k, \frac{N}{n}) = P(k, \alpha b) = e^{-\alpha b (\alpha b)^k \frac{k!}{k!}}$

So the number of overflows per block,

$\Omega(\alpha, b) = \sum_{k>b} (k - b) B(k, N, n)$
$\approx \sum_{k>b} (k - b) P(k, \alpha b)$
$= \sum_{k=0}^{b} (b - k) P(k, \alpha b) - (1 - \alpha) b$

And the number of overflows per record, $\omega(\alpha, b) = \frac{\Omega(\alpha, b)}{\alpha b}$
Finally, $p_{iop} = 1 + \omega(\alpha, b)$, and this can be plotted.

T. H. Merrett
\[ 1 + w(\alpha, b) = 1 + \text{Number of Overflows per Record} \]
Direct vs. Sequential Breakeven Activity

Usage Distributions I: Direct Access

\[ u(L) dL = \text{probability(access record in } L...L+\text{d}L) \]

\[ u_\ell = \int_{(\ell-1)b}^{\ell b} u(L) dL = \text{probability(access block } \ell) \]

\[ X_\ell = \begin{cases} 
0 & \text{if block } \ell \text{ is not accessed} \\
1 & \text{if block } \ell \text{ is accessed} 
\end{cases} \]

\[ \text{prob. } (1 - u_\ell)^r \quad \text{prob. } 1 - (1 - u_\ell)^r \]

Expected no. blocks accessed ("hit rate"):

\[
\overline{X_r} = \sum_{\ell=1}^{n} \left( 0 \times (1 - u_\ell)^r + 1 \times (1 - (1 - u_\ell)^r) \right) \\
= \sum_{\ell=1}^{n} 1 - (1 - u_\ell)^r \\
= n \left( 1 - \left(1 - \frac{1}{n}\right)^r \right) \quad \text{uniform } u_\ell = \frac{1}{n} \\
\approx r \quad \text{small } r \\
\approx n \quad \text{large } r
\]
Direct Access Hit Rates for Uniform Usage
The “80-20” Distribution

\[
\frac{\sum_{\ell=1}^{0.2m} u_\ell}{\sum_{\ell=1}^{m} u_\ell} = 0.8 \\
\frac{\int_{0}^{0.2m} u(L) \, dL}{\int_{0}^{m} u(L) \, dL} = 0.8
\]

try \( \sum_{\ell=1}^{m} u_\ell = cm^\theta \)

try \( u_\ell = c(\ell^\theta - (\ell - 1)^\theta) \)

When \( \theta = \log 0.8 / \log 0.2 = 0.1386 \), 80-20 distribution.

When \( \theta = \log 0.5 / \log 0.5 = 1 \), uniform (50-50) distribution.

When \( \theta = 0 \) (continuous case), “Zipf” distribution.

For normalization, \( u(L) = \theta L^{\theta-1}/(N/\alpha)^\theta \).

T. H. Merrett

©99/10
Direct vs. Sequential Breakeven Activity

Usage Distributions II: Sequential Access

Derive “distribution of depths”

\[ u_r(L)\,dL = \text{prob.}(\max(L_1, \ldots, L_r) \text{ is in } L..L+dL) \]

via cumulative distributions:

\[
\begin{align*}
  u_r(L) &= \frac{d}{dL} U_r(L) \\
  &= \frac{d}{dL} \text{prob.}(\max(L_1, \ldots, L_r) \leq L) \\
  &= \frac{d}{dL} \text{prob.}(L_1 \leq L \text{ and} \ldots \text{and } L_1 \leq L) \\
  &= \frac{d}{dL} (\text{prob.}(L_1 \leq L))^r \\
  &= \frac{d}{dL} (U(L))^r \\
  &= \frac{d}{dL} (\int_0^L u(L)\,dL)^r \\
  &= ru(L)(U(L))^{r-1}
\end{align*}
\]
For the 80-20 family, \( u(L) = \theta L^{\theta-1}/N^\theta \),

\( u_r(L) = r\theta L^{r\theta-1}/(N/\alpha)^{r\theta} \).

Discretize,

\[
\delta d^{(r)}_\delta = \int_{\delta-1}^\delta u_r(L) dL \\
\approx (\delta^{r\theta} - (\delta - 1)^{r\theta})/n^{r\theta}
\]

Expected depth,

\[
\overline{\delta} = \sum_{\delta=1}^n \delta d^{(r)}_\delta \\
\approx (n^{r\theta} + 1 - \sum_{\delta=1}^{n-1} \delta^{r\theta})/n^{r\theta} \\
\approx \frac{nr\theta}{r\theta + 1}
\]
Direct vs. Sequential Breakeven Activity

For uniform usage, compare \( n(1 - (1 - \frac{1}{n})^r) \) with \( \frac{nr}{r+1} \)

Assume

- direct needs 1 probe, \( \rho \), per hit, but only reads the record, \( R \) bytes;
- sequential reads block after block without probing, and probe to find first block is negligible.

"Effective passes" identified for breakeven \( r \):

\[
\left[ rR + \rho n\left(1 - \left(1 - \frac{1}{n}\right)^r\right) \right] / NR = \frac{r}{r + 1}
\]

\[
a_{be} = \frac{r}{N}
\]

\[
= \frac{R}{R + \rho} + O\left(\frac{1}{N}\right)
\]

For \( \rho = 10^6 \)

<table>
<thead>
<tr>
<th>record size, ( R )</th>
<th>10</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>breakeven activity</td>
<td>0.001%</td>
<td>0.01%</td>
<td>0.1</td>
</tr>
</tbody>
</table>

T. H. Merrett
Hashing and Volatility

- Virtual hashing
- Linear hashing
- Splitting criteria
  - (Greedy): split unless this makes $\alpha < \alpha_0$.
  - (Lazy): split if $\pi > \pi_0$
- Algorithm LHI (Linear hash insert)
Algorithm LHI, Linear Hash Insert

Insert Record $r$ with Key $k$

- Initially, $j = 0, p = 0, n = \nu$ and, for any $k$, $h_{-1}(k) = -1$.

LHI1 (Hash.) $a \leftarrow$ if $p = 0$ or $h_{j-1}(k) < p$ then $h_j(k)$ else $h_{j-1}(k)$. If not already there, store $r$ in block $a$ or as an overflow to block $a$. $N \leftarrow \leftarrow$.

LHI2 (Split disallowed.) If $N/(n + 1)b < \alpha_0$ then terminate.

LHI3 (Allocate.) If $p = 0$ then $j \leftarrow \leftarrow$. Allocate block $p + 2^{j-1} \nu$. $n \leftarrow \leftarrow$.

LHI4 (Split.) Rehash block $p$, including overflows, using $h_j$.

LHI5 (Increment pointer.) $p \leftarrow$ if $p \geq 2^{j-1} \nu$ then $0$ else $p + 1$.
Algorithm LHI, Example

\[
\begin{array}{ccccccc}
 n & j & p & k & h_j(k) & h_{j-1}(k) & a \\
1 & 0 & 0 & 3 & 0 & 0 & 3 \\
 & & & 7 & 0 & 0 & 3,7 \\
 & & & 2 & 0 & 0 & 3,7 2 \\
 & & & 5 & 0 & 0 & 3,7 2,5 \\
2 & 1 & 0 & & & 0 & 2 1 3,7 5 \\
 & & & 6 & 0 & 0 & 2,6 1 3,7 5 \\
3 & 2 & 1 & & & 0 & 1 3,7 5 2 2,6 \\
 & & & 11 & 3 & 1 & 1 0 1,3,7 5,11 1 2,6 \\
 & & & 4 & 0 & 0 & 0 4 1,3,7 5,11 1 2,6 \\
4 & 0 & & & & 0 4 1 5 2 2,6 3 3,7 11 \\
 & & & 1 & 1 & 1 & 1 0 4 1,5,1 2 2,6 3 3,7 11 \\
5 & 3 & 1 & & & 0 1,5,1 2 2,6 3 3,7 11 4 4 \\
 & & & 9 & 1 & 1 & 1 0 1,5,1 9 2 2,6 3 3,7 11 4 4 \\
\end{array}
\]

T. H. Merrett

©99/10
Algorithm LHD, Linear Hash Delete

- Initially, \( j, p, \) and \( n \) are as they were left by the last call to LHI or LHD.

LHD1 (Hash.) \( a \leftarrow \) if \( p = 0 \) or \( h_{j-1}(k) < p \) then \( h_j(k) \) else \( h_{j-1}(k) \). If found, remove \( r \) from block \( a \) or from the overflows to block \( a \). \( N \leftarrow - \).

LHD2 (Merge disallowed.) If \( N/nb \geq \alpha_0 \) then terminate.

LHD3 (Decrement pointer.) \( p \leftarrow \) if \( p = 0 \) then \( 2^{j-1} \nu - 1 \) else \( p - 1 \).

LHD4 (Merge.) Rehash blocks \( p \) and \( p + 2^{j-1} \nu \) using \( h_{j-1} \).

LHD5 (Deallocate.) Deallocate block \( p + 2^{j-1} \nu \).
\( n \leftarrow - \). If \( p = 0 \) then \( j \leftarrow - \).
### Algorithm LHD, Example

<table>
<thead>
<tr>
<th>$n$</th>
<th>$j$</th>
<th>$p$</th>
<th>$k$</th>
<th>$h_j(k)$</th>
<th>$h_{j-1}(k)$</th>
<th>$a$</th>
<th>$\frac{j+2}{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>8/10 &gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>7/10 &lt;</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6/8 &lt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5/6 ≥</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4/6 &lt;</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

T. H. Merrett

©99/10
Hashing and Symmetry

- Maintain separate hash functions on each key field.
- Use an array-addressing expression to combine the segment coordinates into an address, e.g.: \(a(i, j) = wj + i\) (2D); \(a(i, j, k) = hwk + wj + i\) (3D).
Hashing and Activity

- Sort requests (keys $k$) by $h(k)$, then merge.

- What about range queries? Must preserve order

Order-Preserving Key-to-Address Transformations ($OPK2AX$): Tidy Functions
**Tidy Functions**
*(Order-preserving key-to-address transformations)*

1. Records 1, 4, 9, 16, 25, 36 loaded into file, 2 per page.

2. Ideal index would show pages end at 4, 16, 36. (But this would need 1 index entry per page.) .................

3. Approximate index is linear interpolation; needs only store 36, 0, 3(pages). _____________________________

4. 0–4, 13–16, and 25–36 will be correctly directed to their home pages (0, 1, 2, resp.), and need only 1 access.

5. 5–12 and 17–24 will be incorrectly directed, and an upwards linear probe will be needed: total, 2 accesses.

6. The number of these “overflows” \( \propto \) length of horizontal lines between the true curve and the approximate curve.

T. H. Merrett
Tidy Functions

The sum of these lengths \( \propto \) area between curves:

\[
\text{Area} = \left( \frac{l_1}{2} + \frac{l_1 + l_2}{2} + \frac{l_2}{2} \right) h = h \sum l_i
\]

The ideal curve may be seen as the cumulative distribution:
Tidy Functions

Here is a case where a) the linear probing must go downwards and b) there may be more than one extra probe (values 121–147, 181–208, 241–255).
Tidy Functions “Overflows” = Area

In more extreme cases, > 1 probe may be needed before we find any of the records that the approximation says are on the page. (Shaded areas = \( h \times \) number of extra probes \( \approx \) area between curves.)

Pointers on each page tell where to start, reducing total probes.

T. H. Merrett
Tidy Functions  Optimal partitioning

Can we improve the approximation by making a new one of two (or more) linear pieces (piece-wise linear)?

Since the overflows are proportional to the area between the curves, find a partitioning which minimizes the area.

For the previous example, the cumulative distribution ("true curve") is \( y = x^2 \) (up to a scale factor which can be removed).

Let's make two linear pieces, which meet each other and the true curve at \((\xi, \xi^2)\). The equations of these two pieces are \( y = \xi x \) and \( y = (1 + \xi)x - \xi \). The area between them and the true curve is

\[
\int_0^\xi (\xi x - x^2)\,dx + \int_\xi^1 ((1 + \xi)x - \xi - x^2)\,dx = \frac{1}{2}(\xi^2 - \xi + \frac{1}{3})
\]

Minimizing this,

\[
0 = \frac{d}{d\xi} \frac{1}{2}(\xi^2 - \xi + \frac{1}{3}) = \xi - \frac{1}{2}
\]

So the optimal approximation by two linear pieces partitions the range of search values in the middle.

T. H. Merrett  ©98/10
**Tidy Functions** Optimal partitioning
We can’t get far with calculus:
- Systems of equations, e.g., \( p = 3 \) partitions gives two simultaneous quadratic equations to solve.
- The true curve is not analytic.

So we use *dynamic programming* (X.Y. Zhao, 1995) e.g., for \( p = 2 \) (again), choose \( k \) which gives the minimum

\[
a(0,k) + a(k,n)
\]

where, for the parabolic true curve (again)

\[
a(i,j) = \int_{\sqrt{\frac{j}{n}}}^{\sqrt{\frac{i}{n}}} \left( \left( \sqrt{\frac{j}{n}} + \sqrt{\frac{i}{n}} \right) x - \sqrt{\frac{ij}{n}} - x^2 \right) dx = \frac{1}{6} \left( \sqrt{\frac{j}{n}} - \sqrt{\frac{i}{n}} \right)^3
\]

This works out to be \( k = n/4 \), or, for \( n = 6 \), \( k = 2 \).
**Tidy Functions** Dynamic programming

Find \(m(p, n)\) (minimum area) given \(a(i, j)\) (and \(m(1, j) = a(0, j)\)):

There are \(\binom{n-1}{p-1} = \sum_{k=p-2}^{n-2} \binom{k}{p-2}\)

ways of placing \(p - 1\) partition boundaries on the \(n - 1\) page boundaries. Problem is not exponential \(O(n^p)\) but cubic, if we *memoize*, because need at most \(\frac{n(n+1)}{2} - \frac{(n-p)(n-p+1)}{2}\) areas.

For \(p = 3\) partitions and \(n = 6\) pages

\[
m(3, 6) \text{ (10 subproblems) = min:}
\]
\[
m(2, 5) + a(5, 6)
\]

\[
(4 \text{ subproblems} = \min:)
\]
\[
m(1, 4) + a(4, 5)
\]
\[
m(1, 3) + a(3, 5)
\]
\[
m(1, 2) + a(2, 5)
\]
\[
m(1, 1) + a(1, 5)
\]

\[
m(2, 4)
\]

\[
(3 \text{ subproblems} = \min:)
\]
\[
m(1, 3) + a(3, 4)
\]
\[
m(1, 2) + a(2, 4)
\]
\[
m(1, 1) + a(1, 4)
\]

\[
m(2, 3)
\]

\[
(2 \text{ subproblems} = \min:)
\]
\[
m(1, 2) + a(2, 3)
\]
\[
m(1, 1) + a(1, 3)
\]

\[
m(2, 2)
\]

\[
(1 \text{ subproblem} = \min:)
\]
\[
m(1, 1) + a(1, 2)
\]

T. H. Merrett  ©98/10
**Tidy Functions** Dynamic programming

Here are all the areas, \((\sqrt{j} - \sqrt{i})^3\), (parabolic curve),

<table>
<thead>
<tr>
<th>i (\backslash) j</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2.828</td>
<td>5.196</td>
<td>8</td>
<td>11.18</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.071</td>
<td>0.39</td>
<td>1</td>
<td>1.89</td>
<td>3.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.032</td>
<td>0.2</td>
<td>0.56</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.019</td>
<td>0.13</td>
<td>0.37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>0.013</td>
<td>0.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From this we can see
- \(p = 2\): partition is at page 2
- \(p = 3\): partitions are at pages 1,3
Tidy Functions Dynamic programming
First two digits of Tbook phone numbers.

Areas for 11 pages (2x step - triangle)

<table>
<thead>
<tr>
<th>28</th>
<th>34</th>
<th>37</th>
<th>46</th>
<th>53</th>
<th>63</th>
<th>66</th>
<th>69</th>
<th>74</th>
<th>84</th>
<th>93</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>56</td>
<td>221</td>
<td>380</td>
<td>662</td>
<td>767</td>
<td>890</td>
<td>1121</td>
<td>1649</td>
<td>2188</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>114</td>
<td>231</td>
<td>453</td>
<td>540</td>
<td>645</td>
<td>846</td>
<td>1314</td>
<td>1799</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1634</td>
<td>3</td>
<td>36</td>
<td>138</td>
<td>189</td>
<td>258</td>
<td>399</td>
<td>747</td>
<td>1124</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>737</td>
<td>1937</td>
<td>9</td>
<td>33</td>
<td>66</td>
<td>117</td>
<td>228</td>
<td>516</td>
<td>839</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>4037</td>
<td>1646</td>
<td>2846</td>
<td>18</td>
<td>3</td>
<td>30</td>
<td>111</td>
<td>339</td>
<td>608</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>9246</td>
<td>4046</td>
<td>3453</td>
<td>4653</td>
<td>3</td>
<td>12</td>
<td>63</td>
<td>231</td>
<td>446</td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>9553</td>
<td>4353</td>
<td>1953</td>
<td>3153</td>
<td>4963</td>
<td>3</td>
<td>18</td>
<td>126</td>
<td>287</td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>12253</td>
<td>7053</td>
<td>4666</td>
<td>2266</td>
<td>3466</td>
<td>5266</td>
<td>9</td>
<td>39</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>449</td>
<td>20353</td>
<td>11366</td>
<td>6166</td>
<td>3766</td>
<td>3169</td>
<td>4369</td>
<td>6169</td>
<td>12</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>719</td>
<td>35369</td>
<td>16169</td>
<td>10969</td>
<td>7374</td>
<td>4974</td>
<td>4374</td>
<td>5574</td>
<td>7374</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>988</td>
<td>46069</td>
<td>24474</td>
<td>15474</td>
<td>10274</td>
<td>7484</td>
<td>5084</td>
<td>4484</td>
<td>5684</td>
<td>7494</td>
<td></td>
</tr>
</tbody>
</table>

Minimum areas (dynamic programming)

©98/10
The cubic dynamic programming problem is still too big, since \( n = \) the number of pages.

We might divide the whole problem into a number of equal-sized smaller problems, then run the optimizing algorithm on each of these.

For example, a 1 Gbyte file of 1 Kbyte pages would have \( n = 10^6 \) pages.

Dividing this into 10,000 subproblems of 100 pages each would require 10,000 \( \mathcal{O}(100^3) \)-sized optimizations.

T. H. Merrett
**Tidy Functions** Dynamic programming

All the above has been concerned with minimizing *areas*.
Interpretation: minimize probes for successful or unsuccessful searches.

Could also minimize probes for successful searches: overflows are given by number of page boundaries crossed by *vertical* lines at search values corresponding to the records actually present:

\[
\pi = 1 + \left( \sum_{i} |t(r_i) - L(r_i)| \right)/n
\]

\[
= 1 + \left( \sum_{i} |i - L(t^{-1}(i))| \right)/n
\]

where \( t() \) is the true curve, \( L() \) is the (linear) approximation, \( r_i = t^{-1}(i) \) is the search value position for the \( i^{th} \) record, \( i = 1..N \).

These vertical distances could be used in the dynamic programming instead of the areas.

T. H. Merrett
Tidying and Symmetry

- Addressing must be via axes of address space. (These are 1-D tidy functions.)

- So key and address spaces must be partitioned rectilinearly.

In $d$ dimensions, $f_i = n^\frac{1}{d}$.

The $d$ axial indexes, each of size $n^\frac{1}{d}$, might fit in RAM.

Axial distributions:

$X_1 \quad 2 \quad 1 \quad 1 \quad 1 \quad 2 \quad 1 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 1$

$X_2 \quad 2 \quad 2 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 1 \quad 2 \quad 1 \quad 2 \quad 1$

T. H. Merrett
Multipage search

1. Use axial indices to find coordinates for page(s) that can hold the data requested.

2. For each page needed (coordinates $i, j, k, ..$), use an array addressing formula to give the page address. (See “hashing and symmetry”.)

T. H. Merrett
Multidimensional paging

Algorithm MP ($N$ records)

MP1 For each axis, $i = 1..d$, find axial distributions and $V_i$. ($d$ sorts: $\mathcal{O}(dN \log N)$)

MP2 Given approximate values for $b$ (blocksize) and $\alpha$ (load factor), choose partitioning factors, $f_i, i = 1..d$. ($\mathcal{O}(1)$)

\[
    n = \prod_{i=1}^{d} f_i; \text{ heuristic: } \frac{V_i}{f_i} = \text{const}
\]

MP3 For each axis
find candidate(s) for axial partition (scan forward then back, cost $2V_i$)

MP4 Build histograms for all combinations of axial partitions by 1 pass of the data. Do $\pi$-$\alpha$ comparisons to find the best. ($\mathcal{O}(N)$)
MP1 Finding Axial Distributions

<table>
<thead>
<tr>
<th>Maker/Toy</th>
<th>Caboose</th>
<th>Calico Cat</th>
<th>Car</th>
<th>Locomotive</th>
<th>Toy Train</th>
<th>Tractor</th>
<th>Tricycle</th>
<th>Truck</th>
<th>Ukulele</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amloco Toys</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Canloco Ltd.</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Dink Inc.</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Extrafun</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Fischerman</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>General Toy Corp.</td>
<td>1 1 1</td>
<td>1 1 1 1</td>
<td>1 1</td>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mettal Toys</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1 1</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Noisy Toys</td>
<td></td>
<td>1</td>
<td></td>
<td>1 1 1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Playloco</td>
<td></td>
<td></td>
<td></td>
<td>1 1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

T. H. Merrett
MP2 Finding $f_i$:

Given $N, b, \alpha$, $n = \lfloor N/b\alpha \rfloor$.

So $f_i = cV_i$, $n = c^d \Pi V_i$, $c = (\Pi V_i / n)^{1/d}$.

But this does not give integer values for $f_i$.

Rounding, flooring, or ceiling may not give factors of $n$ ($n$ may be prime).

So we must remain flexible, by allowing changes to $b, \alpha$.

T. H. Merrett
MP3 Optimal Partition of Toy Axis:
MP3 Optimal Partition of *Maker* Axis:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>9</th>
<th>6</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
</table>

T. H. Merrett

©98/10
# Histograms for Candidate Partitionings

<table>
<thead>
<tr>
<th></th>
<th>aA</th>
<th>aB</th>
<th>aC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
</tr>
<tr>
<td>π_{opt}</td>
<td>1.13</td>
<td>1.13</td>
<td>1.16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>bA</th>
<th>bB</th>
<th>bC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
</tr>
<tr>
<td>π_{opt}</td>
<td>1.13</td>
<td>1.09</td>
<td>1.13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>cA</th>
<th>cB</th>
<th>cC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>α</td>
<td>0.67</td>
<td>0.67</td>
<td>0.67</td>
</tr>
<tr>
<td>π_{opt}</td>
<td>1.16</td>
<td>1.13</td>
<td>1.09</td>
</tr>
</tbody>
</table>

©98/10

T. H. Merrett
MP4 $\pi$-$\alpha$ analysis:

The nine cases from the example:

Pick curve with largest area: bB wins!
Dynamic Multipaging

Strategy: *preserve the access method!*

Tactic: *split the pages!*

So far

<table>
<thead>
<tr>
<th></th>
<th>lo</th>
<th>hi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetry</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Activity</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Volatility</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>
Addressing a dynamic multidimensional array

The axial indices are already there: use them also to keep track of the history of splitting.

\[ \begin{array}{cccc}
0 & 1 & 6 & 9 \\
0 & 0 & 1 & 6 & 9 \\
2 & 2 & 3 & 7 & 10 \\
4 & 4 & 5 & 8 & 11 \\
\end{array} \]

\[ \begin{array}{cccc}
0 & 1 & 2 & 6 \\
0 & 0 & 1 & 2 & 6 \\
3 & 3 & 4 & 5 & 7 \\
8 & 8 & 9 & 10 & 11 \\
\end{array} \]

\[ a(i, j) = \max(p_x(i), p_y(j)) + \text{the other one} \]
\[ a(2, 1) = \max(p_x(2), p_y(1)) + \text{the other one} \]
the left-hand example
\[ = \max(6, 2) + \text{the other one} \]
\[ = 6 + j = 6 + 1 = 7 \]
the right-hand example
\[ = \max(2, 3) + \text{the other one} \]
\[ = 3 + i = 3 + 2 = 5 \]

T. H. Merrett
• Similarly for higher dimensions:
  
  – Use the maximum of the $d$ page entries in the axial indexes to determine the first coordinate,
  
  – then the remaining coordinates address the page within the $(d-1)$-dimensional slab in the conventional way.

• What about splits in the middle of the file?
  
  – Add the new slab of pages to the outer face, as shown above,
  
  – and use the axial index to point to it out of order. (Consider swapping 3 and 8 in $p_y$ in the right-hand example, above, for a result of splitting pages 0, 1, 2, 6.)
Multipage Search and Insert

Split Criteria

1. \((\pi)\) If \(\pi > \pi_0\) then split.
2. \((\alpha)\) Split unless \(\alpha < \alpha_0\).

Direction criteria

3. \((\text{Shape})\) Increase \(f_i\) for the axis, \(i\), for which \(V_i/f_i\) is largest (in order to equalize all \(V_i/f_i\), as far as possible).
4. \((\pi)\) Split in the direction so that \(\pi\) is minimized. (N.B. Keep a log of overflows in the axial index for each row, column, ..)
5. \((\alpha)\) Increase \(f_i\) for the axis, \(i\), for which \(f_i\) is largest (to create least number, \(n/f_i\), of new pages and so decrease \(\alpha\) the least).

Shift criterion

6. \((\pi)\) If \(\pi > \pi_0\) and shifting a boundary will make \(\pi \leq \pi_0\), shift in the direction so that \(\pi\) is minimized.

Note that shape and \(\alpha\) criteria are easy to calculate. The \(\pi\) criterion in (1) must be tested by doing or simulating the shifts or splits, and so is much more expensive. However, the \(\pi\) criteria deal directly with what is usually the important consideration, namely keeping the probe factor down.

T. H. Merrett
Algorithm MPI

(A collection of alternative algorithms.)

Try to shift first:

A. 6, 1, 3, 4, 5 emphasizes $\pi$

B. 6, 1, 3, 5, 4 $\pi$, then $\alpha$

C. 6, 2, 3, 5, 4 split emphasizes $\alpha$

D. 6, 2, 3, 4, 5 split using $\alpha$, then $\pi$

No shift:

A’, B’, C’, D’ as above, but without 6

T. H. Merrett