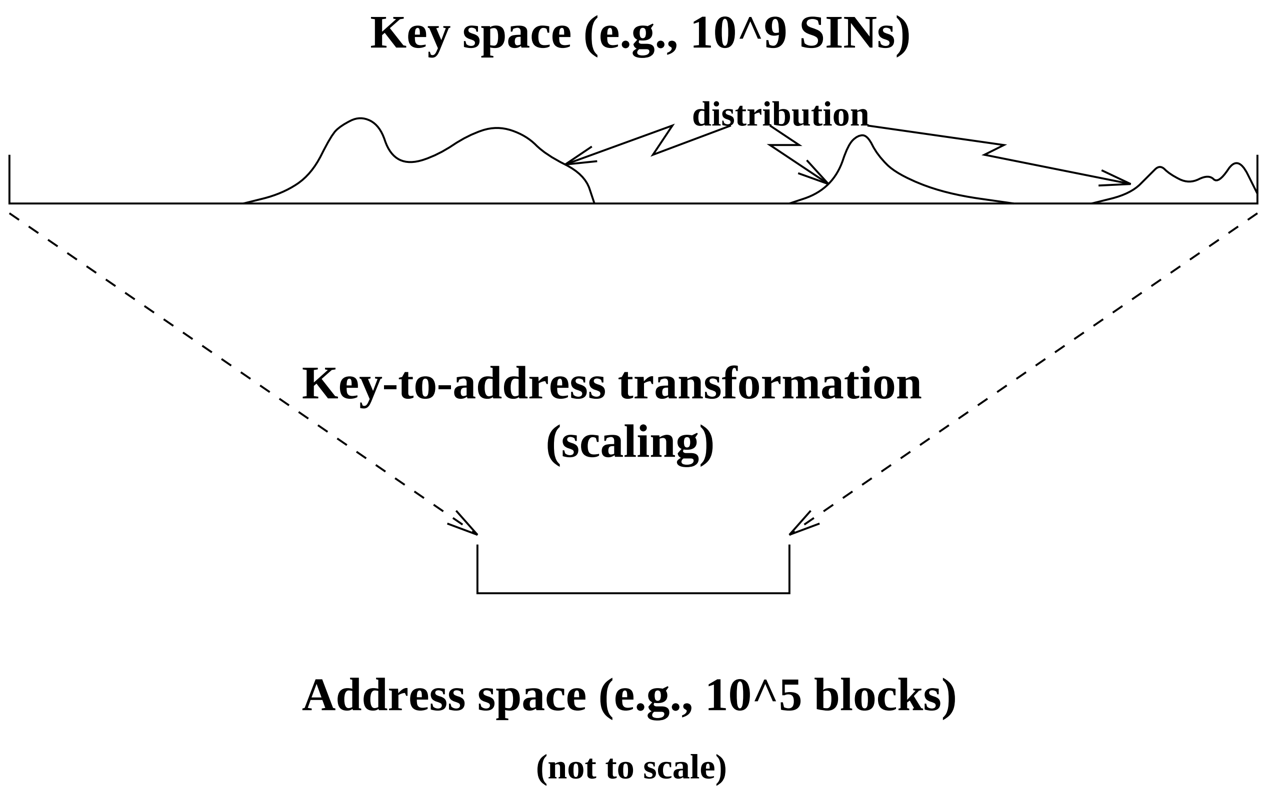
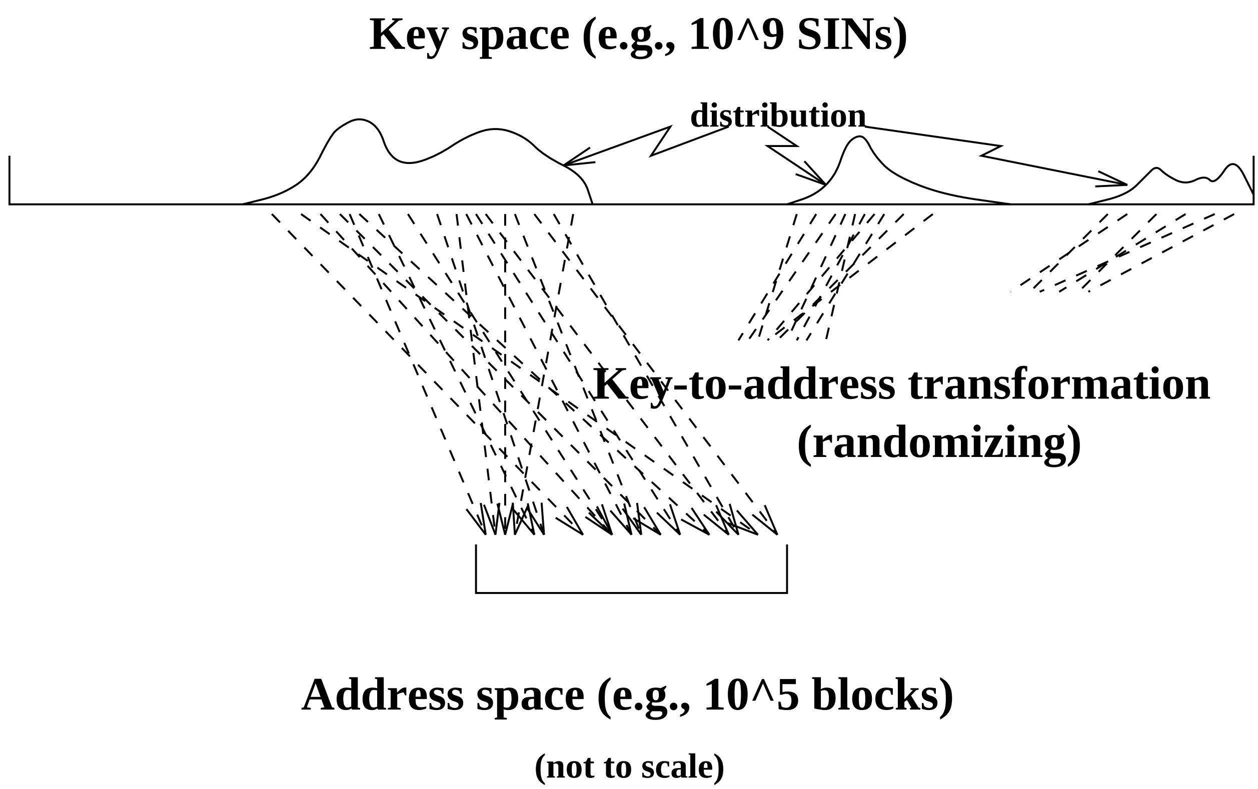
The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding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

Key-to-Address Transformation Scaling

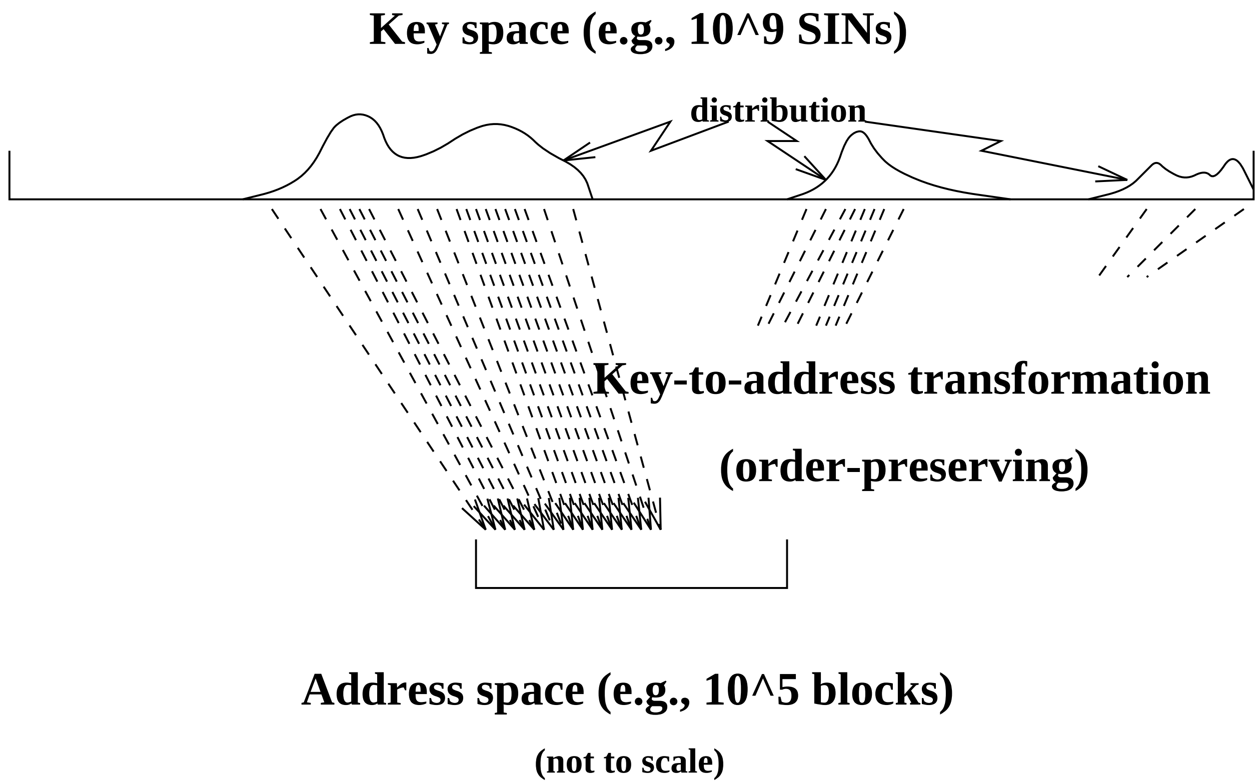


Key-to-Address Transformation Randomizing



T. H. Merrett

Key-to-Address Transformation
Order-preserving



T. H. Merrett ©99/10

Direct Access

(Compute block address from keys,
instead of comparing them.)

Hash Functions

(Randomizing key-to-address transformations)

- Hash functions
  - Division-remainder, $h(k) = k \bmod n, n$ 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)

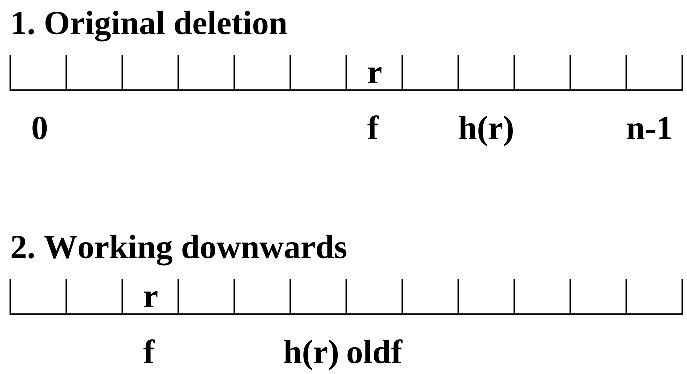
- Algorithm SD (Hash delete with separate chaining)

- 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



LD1 (Remove record)
ℓ <− loc(r); *oldf* <− f; mark ℓ on *oldf* empty

LD2 (Move later records up)
if block f not full (apart from deletion, if any,
    just made by LD1), stop.
f <− if $f = 0$ then $n - 1$ else $f - 1$
if f = original home block, stop.
    /* else full file can thrash */
for each record, r, on block f
  if ncycle($f, h(r)$, *oldf*)
  /*$h(r)$ not cyclically between f, *oldf*
  or = *oldf*; i.e., $h(r)$ at or beyond *oldf**/
    then copy r to ℓ on *oldf*; goto LD1
goto LD2

Analysing Hash Functions

Parameters

- $\alpha = \frac{N}{nb}$, load factor
- $\pi = \frac{\text{total no. probes}}{N}$, probe factor
- $pi_{\text{op}} = 1 + \frac{\text{total overflows}}{N}$, 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 preselected records of N hash to it)

$= \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{N-k}$

prob(any k of N records hash to it)

$= \left(\begin{array}{c} N \\ k \end{array} \right) \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{N-k}$

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

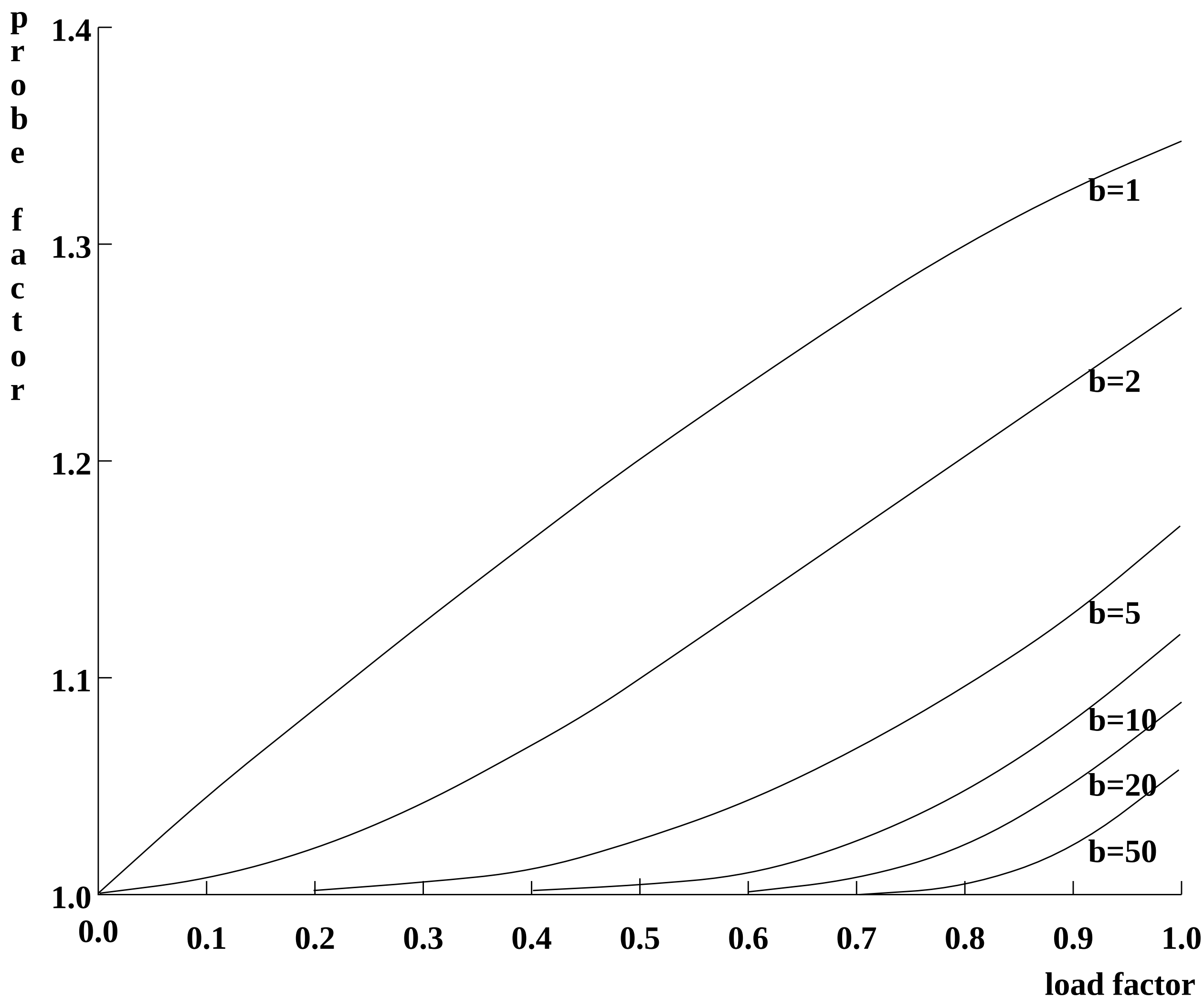
So the number of overflows per block,

$$
\begin{aligned}
\Omega(\alpha, b) &= \Sigma_{k>b}(k-b)B(k, N, n) \\
&\approx \Sigma_{k>b}(k-b)P(k, \alpha b) \\
&= \Sigma_{k=0}^{b}(b-k)P(k, \alpha b) - (1-\alpha)b
\end{aligned}
$$

And the number of overflows per record, $\omega(\alpha, b) = \frac{\Omega(\alpha,b)}{\alpha b}$

Finally, $pi_{\text{op}} = 1 + \omega(\alpha, b)$, and this can be plotted.

$1 + w(\alpha, b)$
$= 1 +$ Number of Overflows per Record



T. H. Merrett

Direct vs. Sequential Breakeven Activity

Usage Distributions I: Direct Access

$u(L)\mathrm{d}L =$ probability(access record in $L ... L+\mathrm{d}L$)

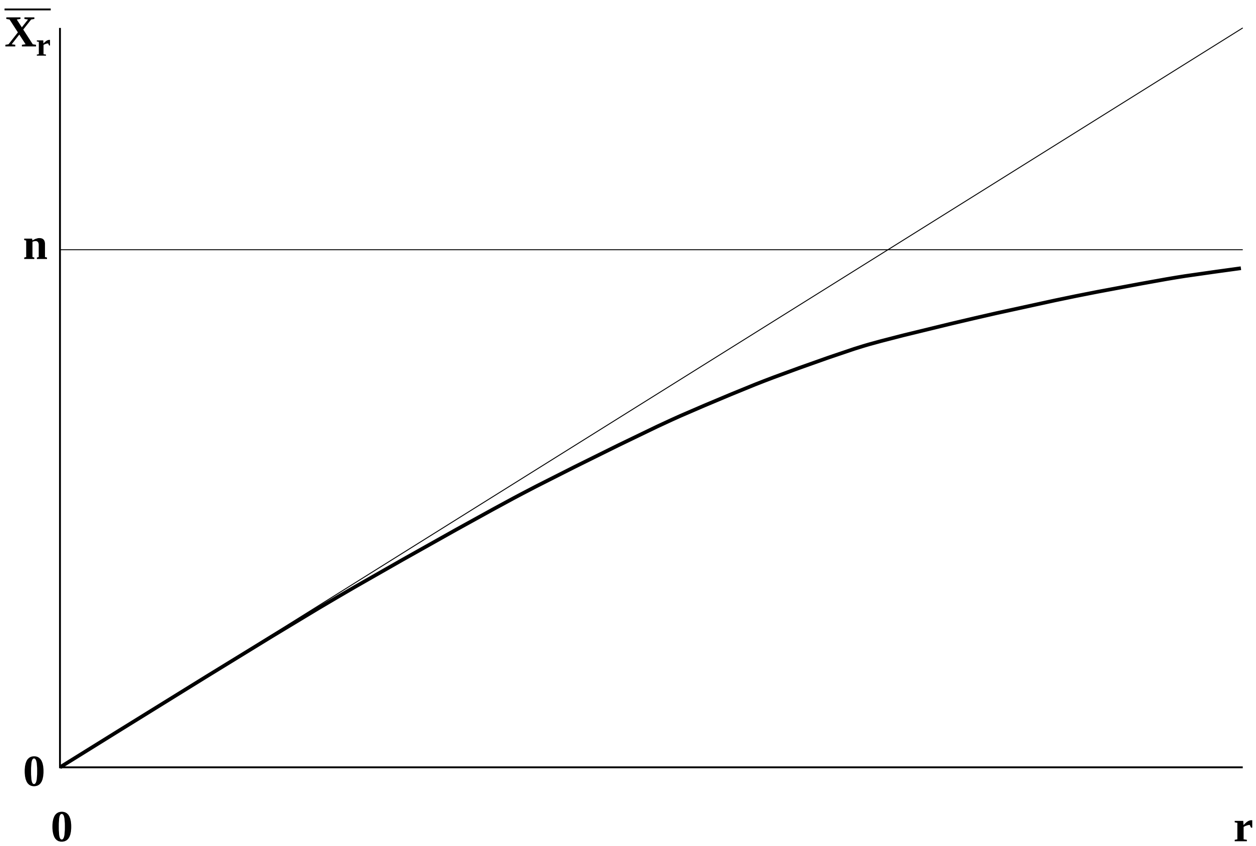
$u_\ell = \int_{(\ell-1)b}^{\ell b} u(L)\mathrm{d}L =$ probability(access block ℓ)

$$X_\ell = \begin{cases} 0 & \text{if block } \ell \text{ is not accessed} \quad \text{prob. } (1-u_\ell)^r \\ 1 & \text{if block } \ell \text{ is accessed} \quad \text{prob. } 1-(1-u_\ell)^r \end{cases}$$

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

$$\begin{aligned} \overline{X_r} &= \sum_{\ell=1}^{n} 0 \times (1-u_\ell)^r + 1 \times (1-(1-u_\ell)^r) \\ &= \sum_{\ell=1}^{n} 1-(1-u_\ell)^r \\ &= n(1-(1-\frac{1}{n})^r) \quad \text{uniform } u_\ell = \frac{1}{n} \\ &\approx r \quad \text{small } r \\ &\approx n \quad \text{large } r \end{aligned}$$

Direct Access Hit Rates for Uniform Usage



The “80-20” Distribution

$\frac{\sum_{\ell=1}^{0.2m} u_\ell}{\sum_{\ell=1}^{m} u_\ell} = 0.8$ $\frac{\int_0^{0.2m} u(L)\mathrm{d}L}{\int_0^{m} u(L)\mathrm{d}L} = 0.8$

try $\sum_{\ell=1}^{m} u_\ell = cm^\theta$ try $\int_0^{m} u(L)\mathrm{d}L = cm^\theta$

try $u_\ell = c(\ell^\theta - (\ell - 1)^\theta)$ try $u(L) = c'L^{\theta-1}$

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

Direct vs. Sequential Breakeven Activity

Usage Distributions II: Sequential Access

Derive "distribution of depths"

$u_r(L)\mathrm{d}L =$ prob.($\mathbf{max}(L_1, ..., L_r)$ is in $L..L+\mathrm{d}L$)

via cumulative distributions:

$$\begin{aligned}
u_r(L) &= \frac{\mathrm{d}}{\mathrm{d}L}U_r(L) \\
&= \frac{\mathrm{d}}{\mathrm{d}L}\text{prob.}(\max(L_1, ..., L_r) \leq L) \\
&= \frac{\mathrm{d}}{\mathrm{d}L}\text{prob.}(L_1 \leq L \textbf{ and...and } L_1 \leq L) \\
&= \frac{\mathrm{d}}{\mathrm{d}L}(\text{prob.}(L_1 \leq L))^r \\
&= \frac{\mathrm{d}}{\mathrm{d}L}(U(L))^r \\
&= \frac{\mathrm{d}}{\mathrm{d}L}(\int_0^L u(L)\mathrm{d}L)^r \\
&= ru(L)(U(L))^{r-1}
\end{aligned}$$

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,

$$
\begin{aligned}
d_{\delta}^{(r)} &= \int_{\delta-1)b}^{\delta b} u_r(L)\mathrm{d}L \\
&\approx (\delta^{r\theta} - (\delta-1)^{r\theta})/n^{r\theta}
\end{aligned}
$$

Expected depth,

$$
\begin{aligned}
\overline{\delta} &= \sum_{\delta=1}^{n} \delta d_{\delta}^{(r)} \\
&\approx (n^{r\theta+1} - \sum_{\delta=1}^{n-1} \delta^{r\theta})/n^{r\theta} \\
&\approx \frac{nr\theta}{r\theta+1}
\end{aligned}
$$

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, ρ, 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:

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

$$a_{\text{be}} = \frac{r}{N}$$

$$= \frac{R}{R+\rho}+\mathcal{O}(\frac{1}{N})$$

For $\rho = 10^6$

record size, R	10	100	1000
breakeven activity	0.001%	0.01%	0.1

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. $N{+}{+}$.

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

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

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

n	j	p	k	$h_j(k)$	$h_{j-1}(k)$	a		
1	0	0	3	0		0	$_0$ 3	1/4
			7	0		0	$_0$ 3,7	2/4
			2	0		0	$_0$ 3,7 2	3/4
			5	0		0	$_0$ 3,7 2,5	4/4
2	1	0					$_0$ 2 $_1$ 3,7 5	
			6	0		0	$_0$ 2,6 $_1$ 3,7 5	5/6
3	2	1					$_0$ $_1$ 3,7 5 $_2$ 2,6	
			11	3	1	1	$_0$ $_1$ 3,7 5,11 $_1$ 2,6	6/8
			4	0	0	0	$_0$ 4 $_1$ 3,7 5,11 $_1$ 2,6	7/8
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	8/1
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	9/1

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. $N - -$.

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}.

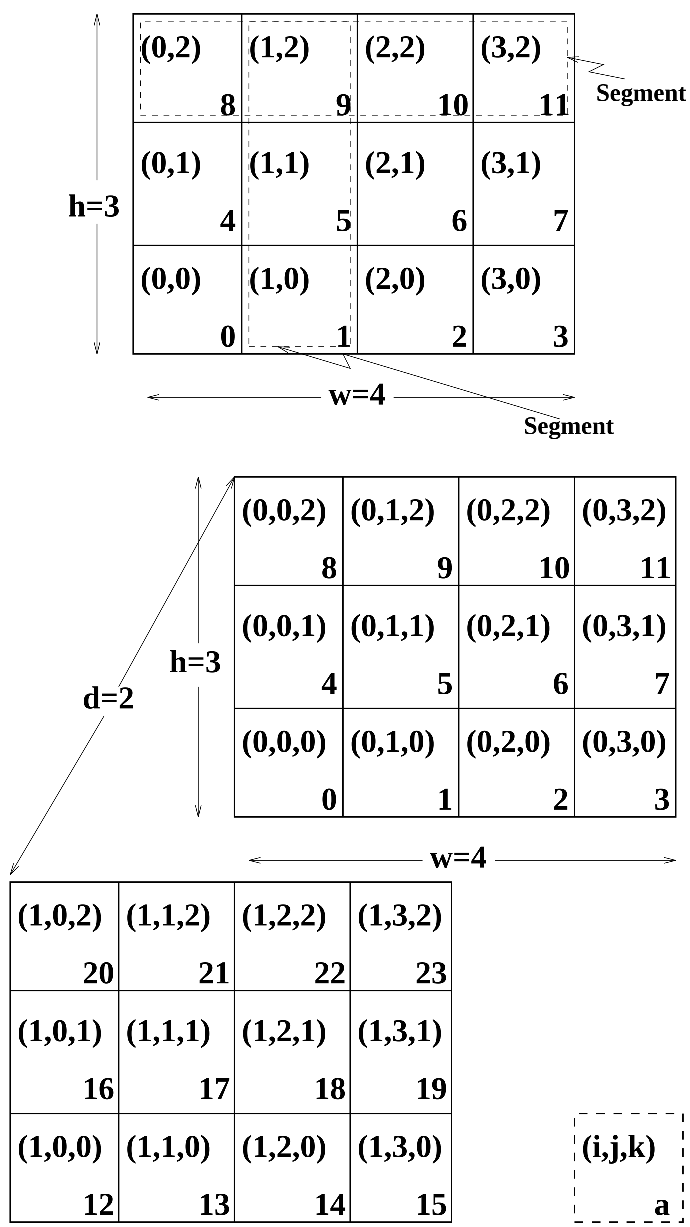
LHD5 (Deallocate.) Deallocate block $p + 2^{j-1}\nu$. $n - -$. If $p = 0$ then $j - -$.

Algorithm LHD, Example

n	j	p	k	$h_j(k)$	$h_{j-1}(k)$	a		
5	3	1	7	7	3	1	0 [ ] 1 [5,1 9] 2 [2,6] 3 [3,11] 4 [4]	8/10 $\geq$
			6	6	2	2	0 [ ] 1 [5,1 9] 2 [2] 3 [3,11] 4 [4]	7/10 $<$
		0					0 [4] 1 [5,1 9] 2 [2] 3 [3,11]	
4	2							
			2	2	0	2	0 [4] 1 [5,1 9] 2 [ ] 3 [3,11]	6/8 $<$
		1					0 [4] 1 [5,1 9, 3,11] 2 [ ]	
3								
			1	1	1	1	0 [4] 1 [5,9 3,11] 2 [ ]	5/6 $\geq$
			11	3	1	1	0 [4] 1 [5,9 3] 2 [ ]	4/6 $<$
		0					0 [4] 1 [5,9 3]	
2	1							

Hashing and Symmetry

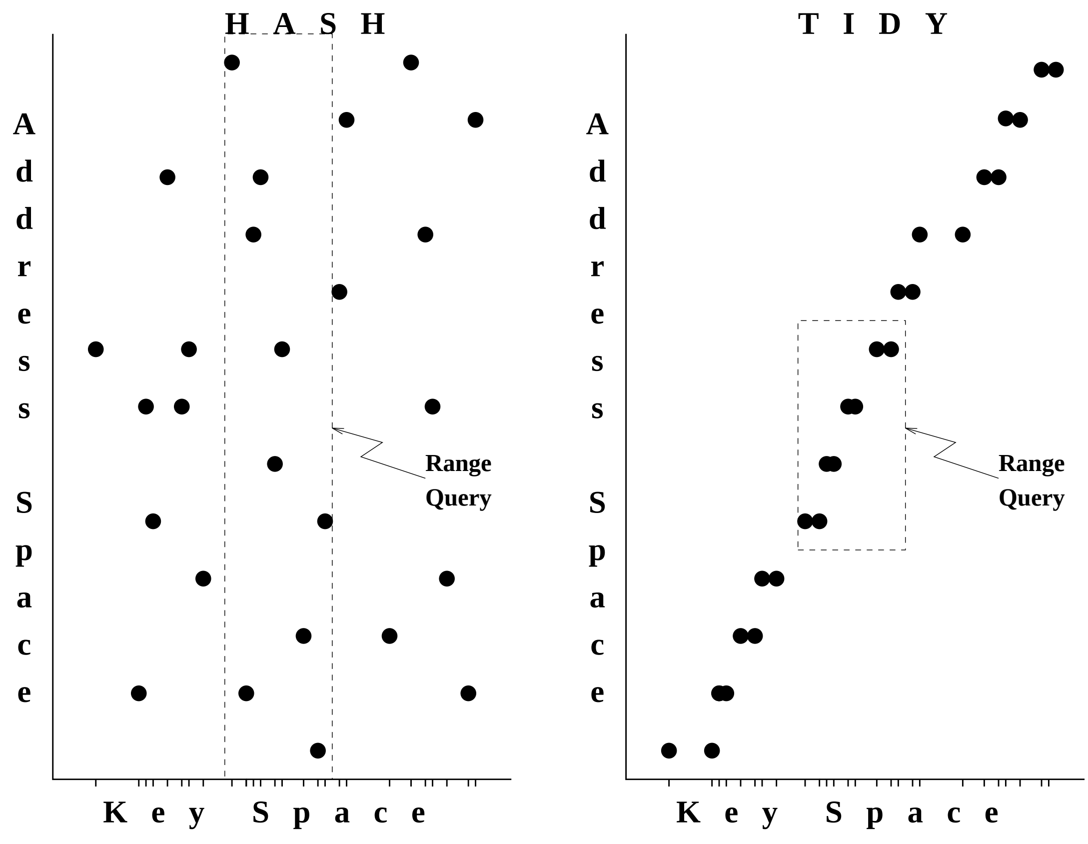
- Maintain separate hash functions on each key field.
- Use an array-addressing expression to combine the segment coordinates into an address, e.g.: $a(i,j) = wj + i$ (2D); $a(i,j,k) = hwk + wj + i$ (3D).



Hashing and Activity

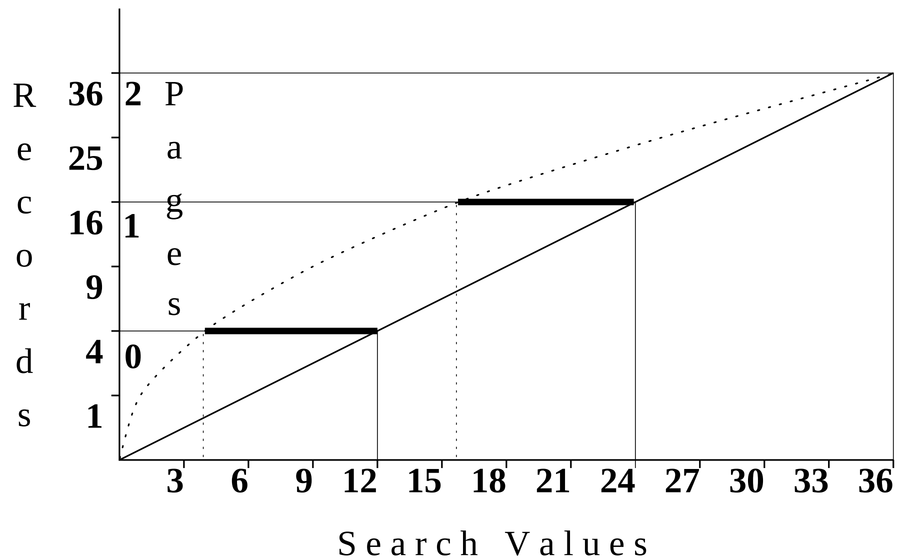
- Sort requests (keys k) by $h(k)$, then merge.
- What about *range* queries?
  Must *preserve order*

Order-Preserving Key-to-Address Transformations (*OPK2AX*): *Tidy Functions*



Tidy Functions

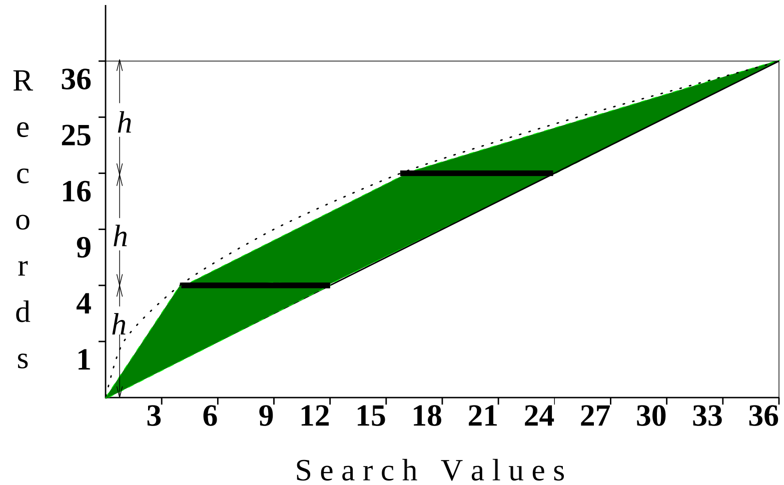
(Order-preserving key-to-address transformations)



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.

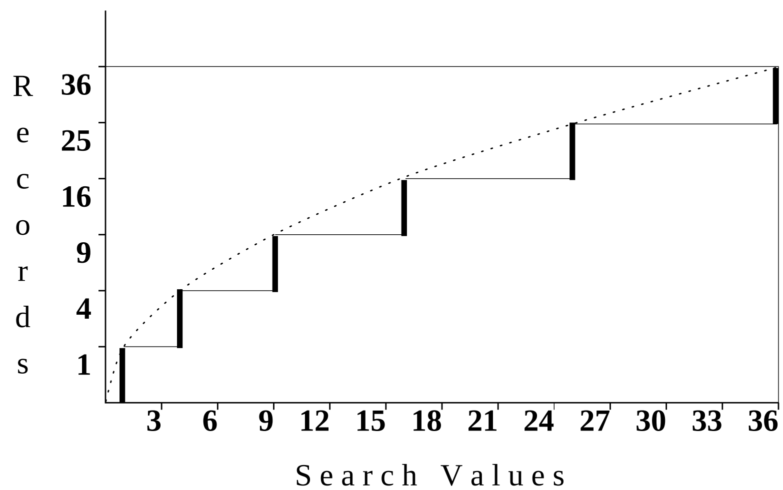
Tidy Functions

The sum of these lengths $\propto$ area between curves:



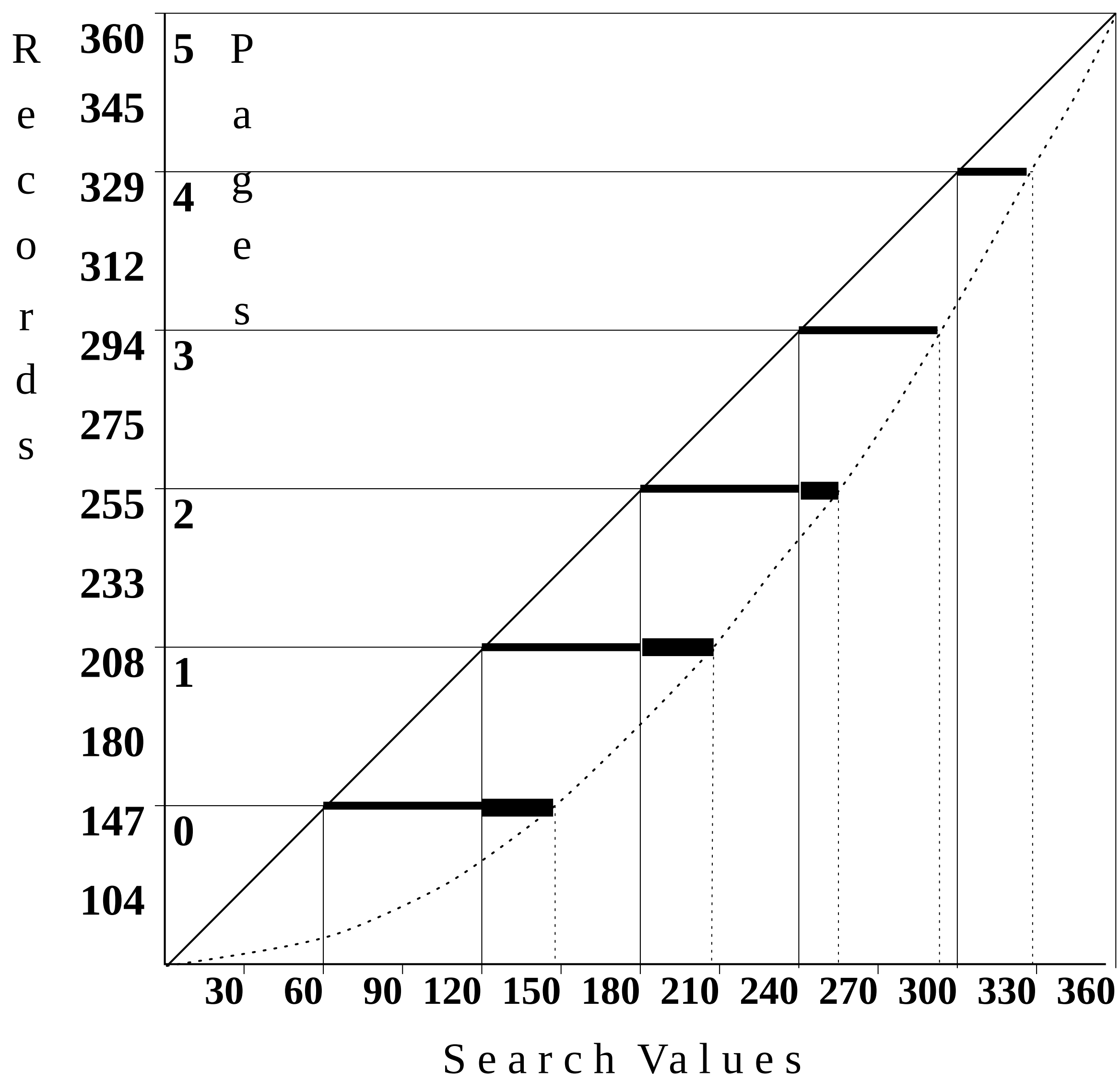
Area $= (l_1/2 + (l_1 + l_2)/2 + l_2/2)h = h\Sigma l_i$

The ideal curve may be seen as the cumulative distribution:



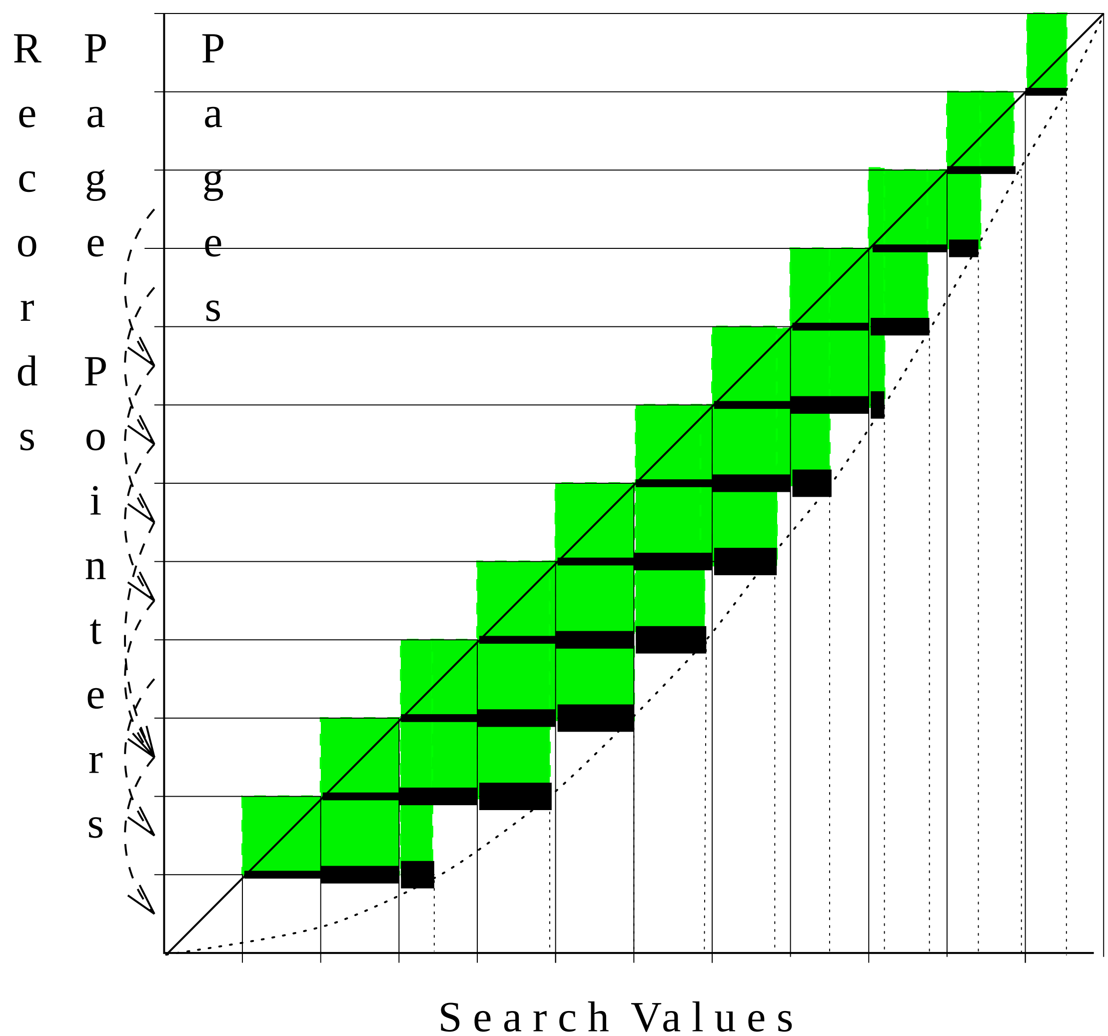
Tidy Functions

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.

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 (ξ, ξ^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

$$\int_0^\xi (\xi x - x^2)\mathrm{d}x + \int_\xi^1 ((1+\xi)x - \xi - x^2)\mathrm{d}x = \frac{1}{2}(\xi^2 - \xi + \frac{1}{3})$$

Minimizing this,

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

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

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)

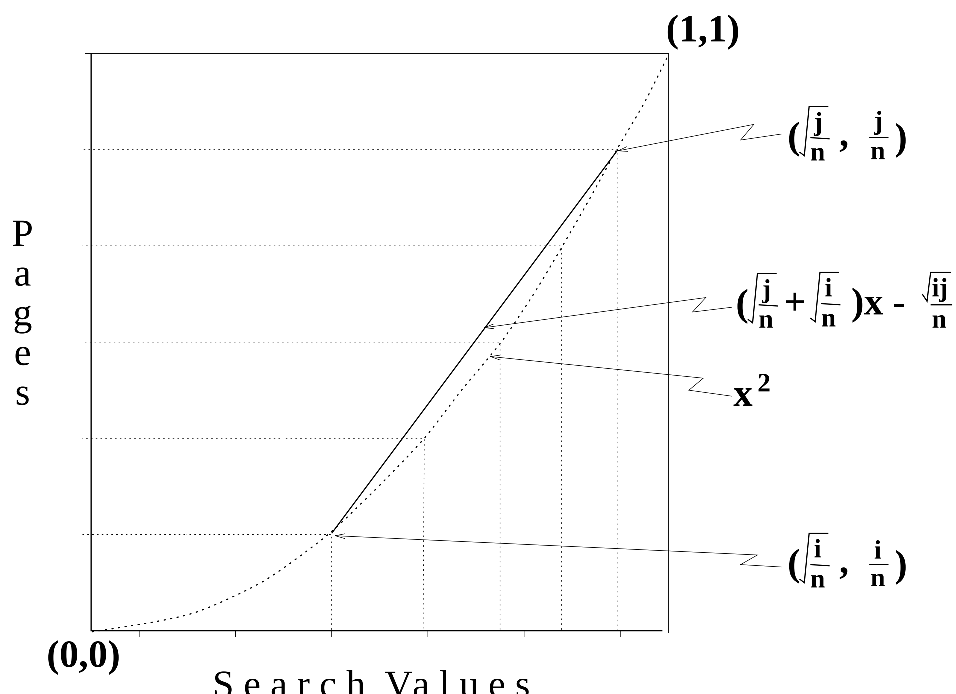
e.g., for $p = 2$ (again), choose k which gives the minimum

$$a(0,k) + a(k,n)$$

where, for the parabolic true curve (again)

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

This works out to be $k = n/4$, or, for $n = 6$, $k = 2$.



Tidy Functions Dynamic programming

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

There are
$$\left(\begin{array}{c} n-1 \\ p-1 \end{array}\right) = \sum_{k=p-2}^{n-2} \left(\begin{array}{c} k \\ p-2 \end{array}\right)$$
ways of placing $p-1$ partition boundaries on the $n-1$ page boundaries. Problem is not exponential ($\mathcal{O}(n^p)$) 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

$m(3,6)$ (10 subproblems) = min:

- $m(2,5)$ $+a(5,6)$
  - (4 subproblems) = min:
    - $m(1,4) + a(4,5)$
    - $m(1,3) + a(3,5)$
    - $m(1,2) + a(2,5)$
    - $m(1,1) + a(1,5)$
- $m(2,4)$ $+a(4,6)$
  - (3 subproblems) = min:
    - $m(1,3) + a(3,4)$
    - $m(1,2) + a(2,4)$
    - $m(1,1) + a(1,4)$
- $m(2,3)$ $+a(3,6)$
  - (2 subproblems) = min:
    - $m(1,2) + a(2,3)$
    - $m(1,1) + a(1,3)$
- $m(2,2)$ $+a(2,6)$
  - (1 subproblem) = min:
    - $m(1,1) + a(1,2)$

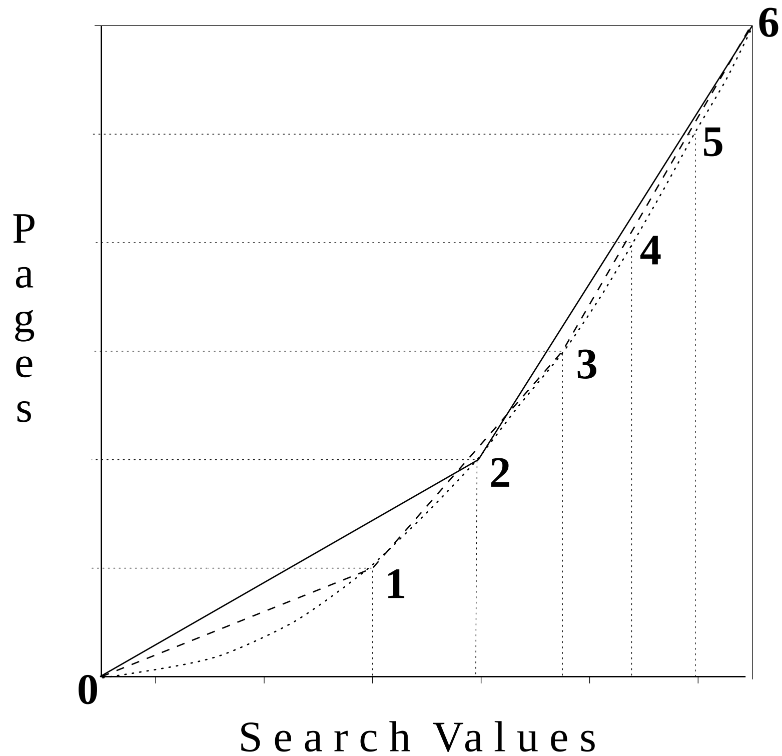
Tidy Functions Dynamic programming

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

$i\backslash j$	0	1	2	3	4	5	6
0		1	2.828	5.196	8	11.18	14.7
1			0.071	0.39	1	1.89	3.04
2				0.032	0.2	0.56	1.1
3					0.019	0.13	0.37
4						0.013	0.09
5							0.009
6							

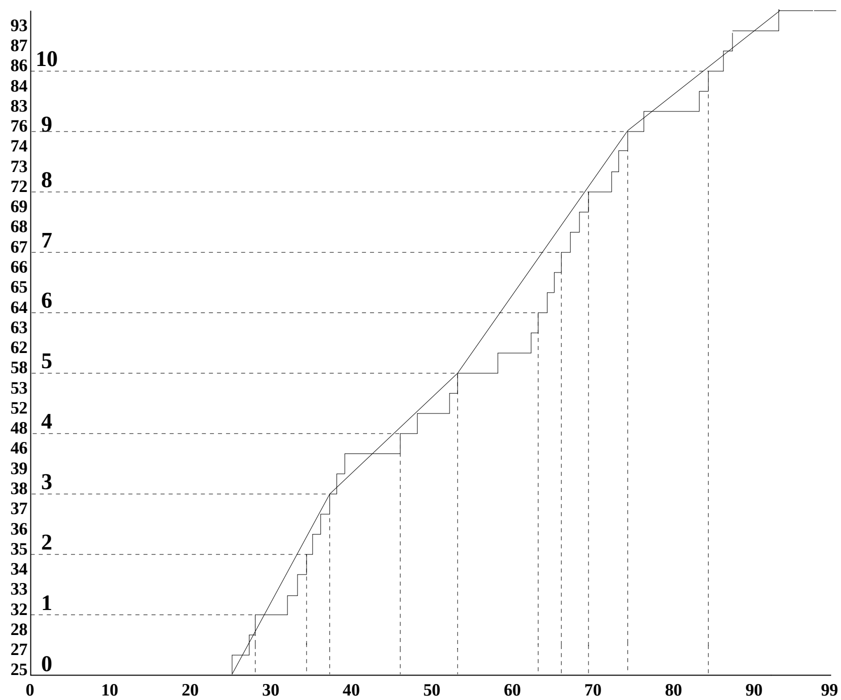
From this we can see

- $p = 2$: 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 (2× | *step* − *triangle* |)

28	34	37	46	53	63	66	69	74	84	93
1	23	56	221	380	662	767	890	1121	1649	2188
	12	3	114	231	453	540	645	846	1314	1799
13_{28}		3	54	129	291	360	447	618	1026	1457
4_{28}	16_{34}		3	36	138	189	258	399	747	1124
59_{37}	7_{37}	19_{37}		9	33	66	117	228	516	839
92_{37}	40_{37}	16_{46}	28_{46}		18	3	30	111	339	608
194_{37}	92_{46}	40_{46}	34_{53}	46_{53}		3	12	63	231	446
245_{37}	95_{53}	43_{53}	19_{53}	31_{53}	49_{63}		3	18	126	287
314_{37}	122_{53}	70_{53}	46_{66}	22_{66}	34_{66}	52_{66}		9	39	146
449_{46}	203_{53}	113_{66}	61_{66}	37_{66}	31_{69}	43_{69}	61_{69}		12	41
719_{53}	353_{69}	161_{69}	109_{69}	73_{74}	49_{74}	43_{74}	55_{74}	73_{74}		1
988_{53}	460_{69}	244_{74}	154_{74}	102_{74}	74_{84}	50_{84}	44_{84}	56_{84}	74_{94}	

Minimum areas (dynamic programming)

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.

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:

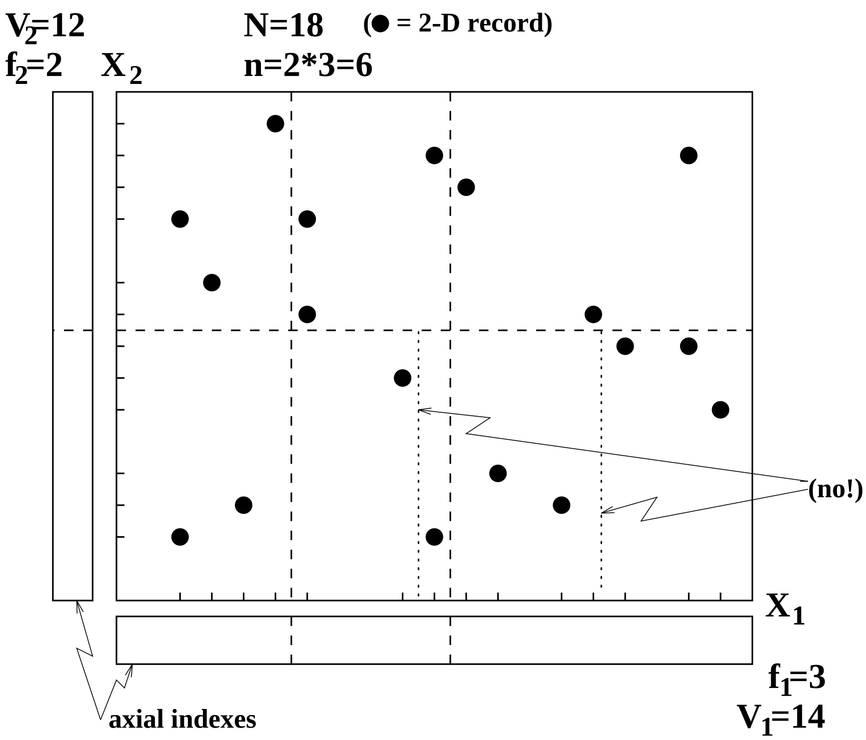
$$\begin{aligned}\pi &= 1 + (\sum_i \lfloor t(r_i) - L(r_i) \rfloor)/n \\ &= 1 + (\sum_i \lfloor i - L(t^{-1}(i)) \rfloor)/n\end{aligned}$$

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.

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 2 1 1 1 2 1 2 1 1 1 1 1 2 1

X_2 2 2 1 1 1 2 2 1 2 1 2 1

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”.)

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 α (load factor), choose partitioning factors, $f_i, i = 1..d$. ($\mathcal{O}(1)$)

$$n = \Pi_1^d f_i; \text{heuristic} : \frac{V_i}{f_i} = \text{const}$$

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 π-α comparisons to find the best. ($\mathcal{O}(N)$)

MP1 Finding Axial Distributions

Maker\Toy	Caboose	Calico Cat	Car	Locomotive	Toy Train	Tractor	Tricycle	Truck	Ukulele	
Amloco Toys				1						1
Canloco Ltd.				1						1
Dink Inc.			1	1						2
Extrafun		1	1	1						3
Fischerman	1		1	1						4
General Toy Corp.	1	1	1	1	1	1	1	1	1	9
Mettal Toys	1		1	1		1	1	1		6
Noisy Toys		1	1	1	1	1				5
Playloco				1						1
	3	3	6	9	4	2	2	2	1	

MP2 Finding f_i:

Given $N, b, \alpha,\ n = \lceil N/b\alpha \rceil$.

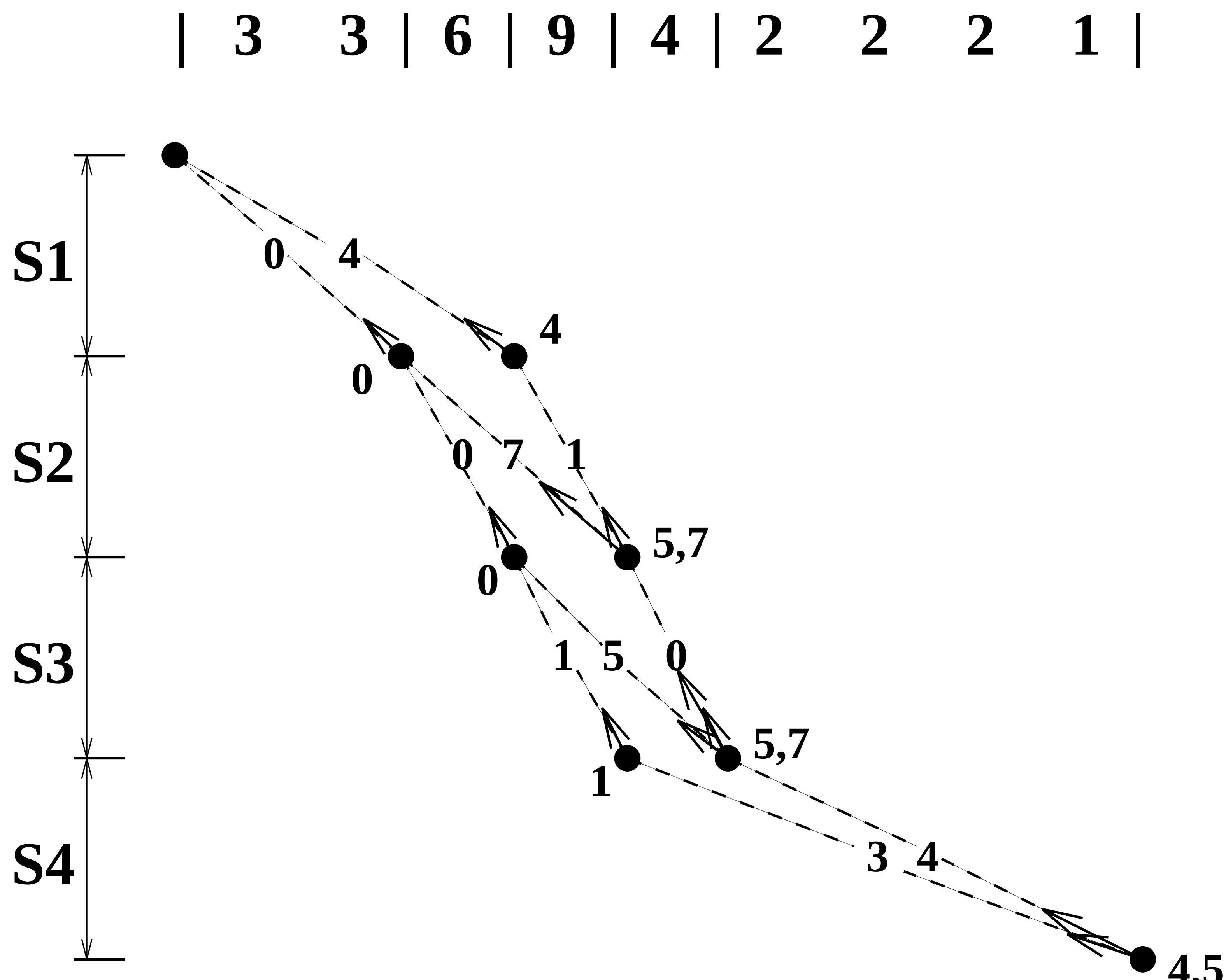
So $f_i = cV_i, n = c^d \Pi V_i, c = (\Pi V_i/n)^{1/d}$.

But this does not give integer values for f_i.

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

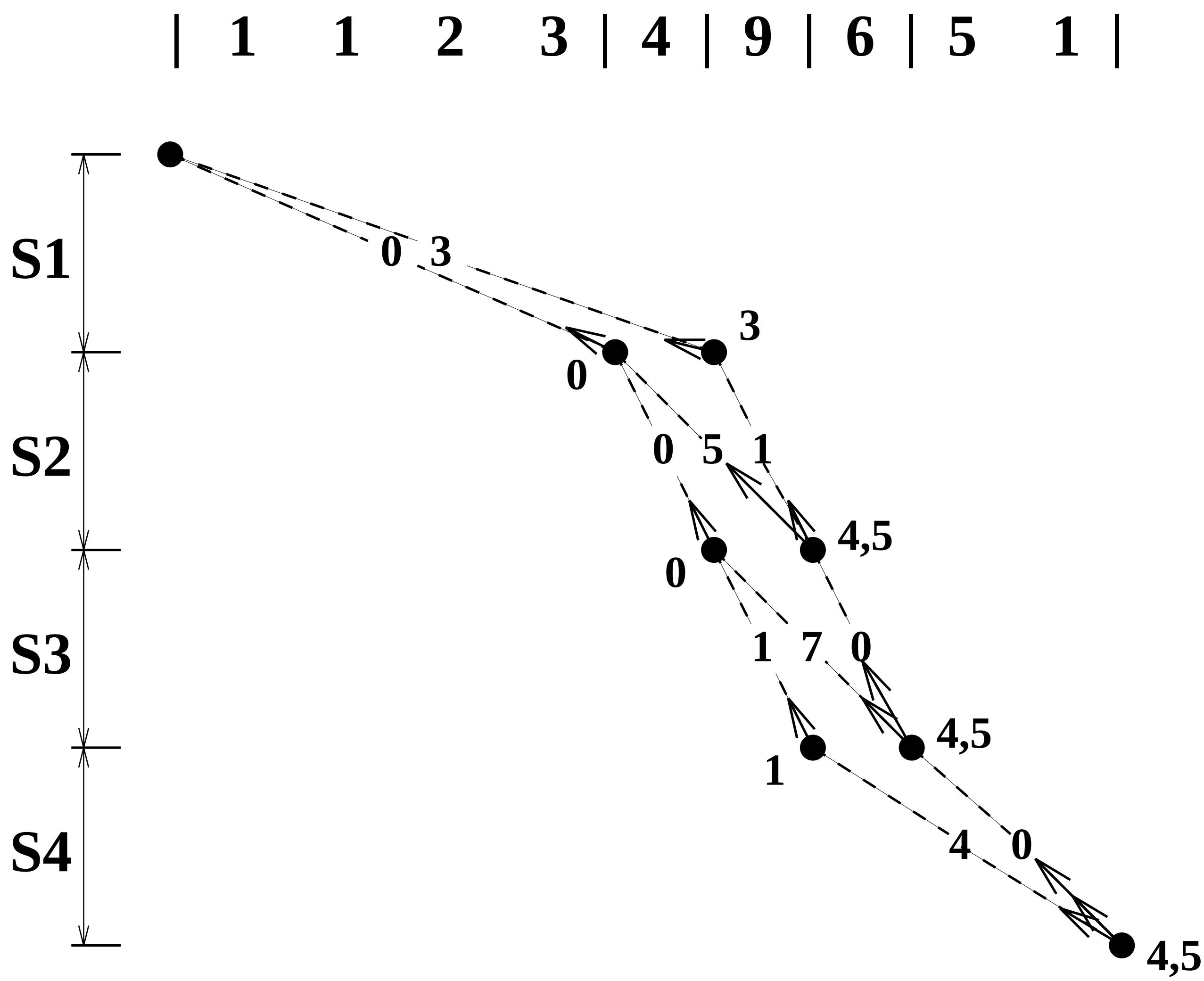
So we must remain flexible, by allowing changes to b, α.

MP3 Optimal Partition of *Toy* Axis:



| 3 3 | 6 | 9 | 4 | 2 2 2 1 |
S1
S2
S3
S4
0
4
4
0
0
7
1
5,7
0
1
5
0
5,7
1
3
4
4,5

MP3 Optimal Partition of *Maker* Axis:



T. H. Merrett ©98/10

Histograms for Candidate Partitionings

aA

2	3	5	1
2	1	1	5
1	1	1	3
1	1	2	2

α	0.67	1.0
π_{opt}	1.13	1.25

aB

2	3	6	0
2	1	2	4
1	1	2	2
1	1	3	1

α	0.67	1.0
π_{opt}	1.13	1.25

aC

5	5	1	0
3	1	1	4
2	1	1	2
2	2	1	1

α	0.67	1.0
π_{opt}	1.16	1.28

bA

1	2	4	0
3	2	2	6
1	1	1	3
1	1	2	2

α	0.67	1.0
π_{opt}	1.13	1.25

bB

1	2	4	0
3	2	4	4
1	1	2	2
1	1	3	1

α	0.67	1.0
π_{opt}	1.09	1.25

bC

3	4	0	0
5	2	2	4
2	1	1	2
2	2	1	1

α	0.67	1.0
π_{opt}	1.13	1.25

cA

1	2	4	0
1	1	1	1
2	1	1	5
2	2	3	5

α	0.67	1.0
π_{opt}	1.16	1.28

cB

1	2	4	0
1	1	2	0
2	1	2	4
2	2	5	3

α	0.67	1.0
π_{opt}	1.13	1.25

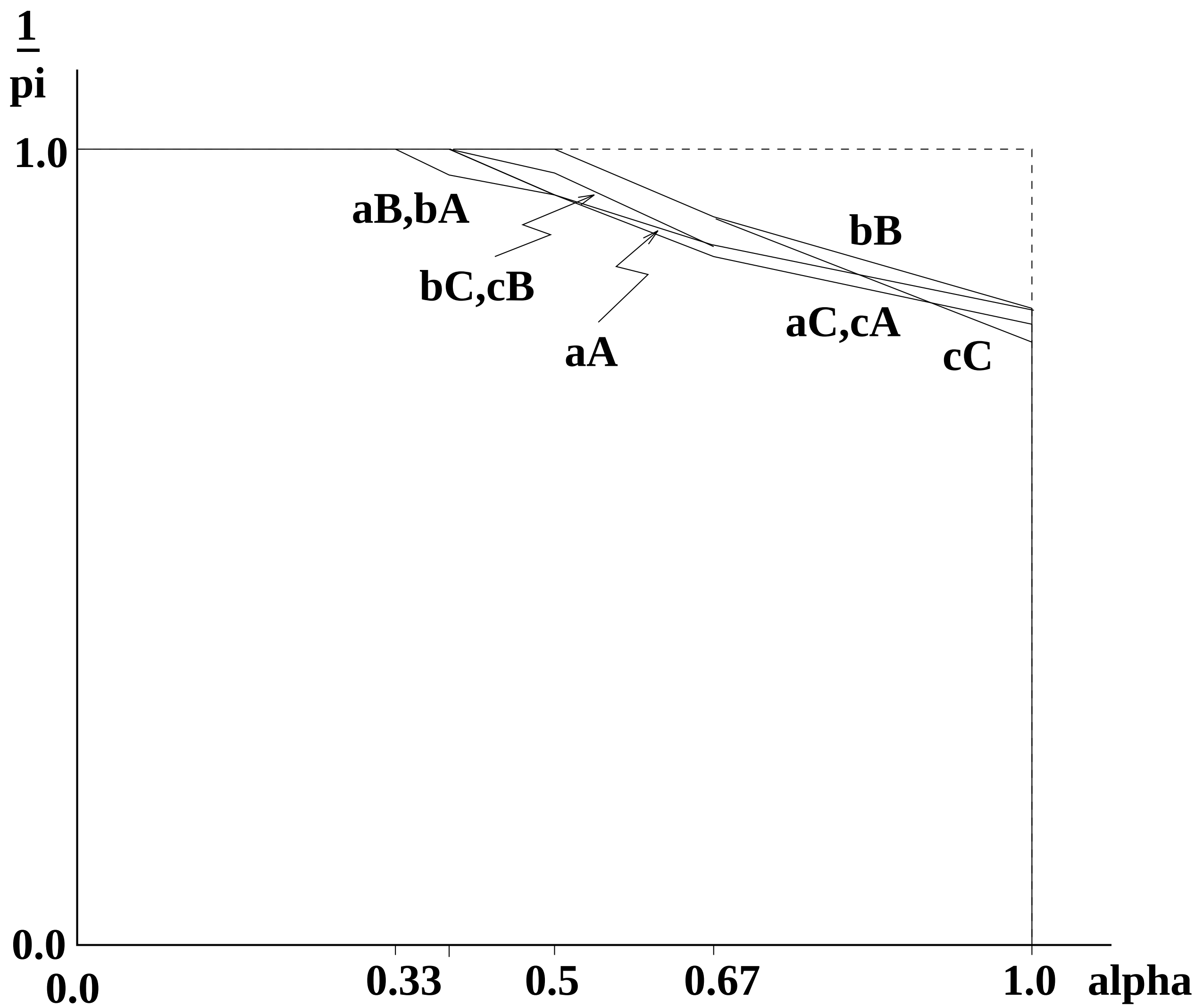
cC

3	4	0	0
2	1	1	0
3	1	1	4
4	3	2	3

α	0.67	1.0
π_{opt}	1.09	1.31

MP4 π-α analysis:

The nine cases from the example:



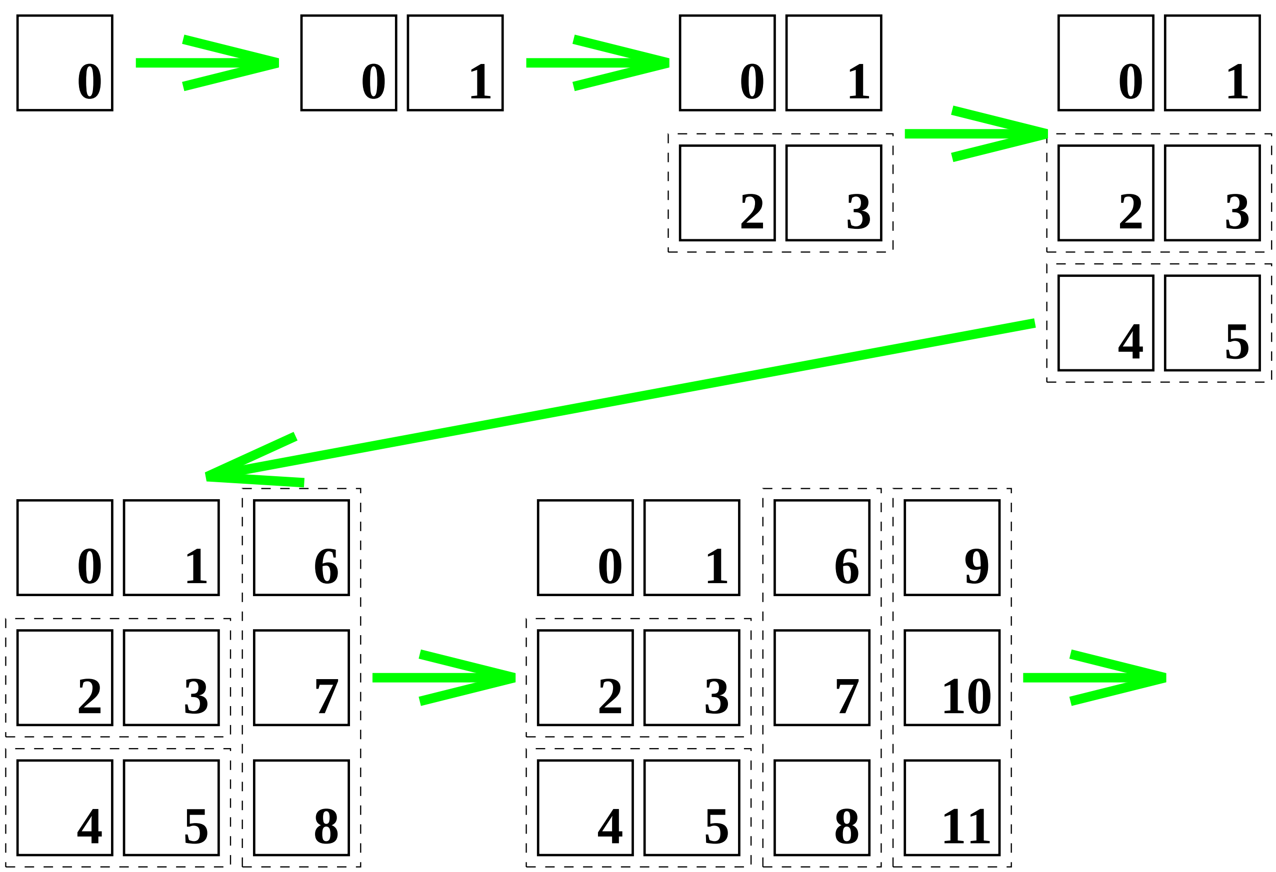
Pick curve with largest area: bB wins!

So far	lo	hi
Symmetry	√	√
Activity	√	√
Volatility	√	×

Dynamic Multipaging

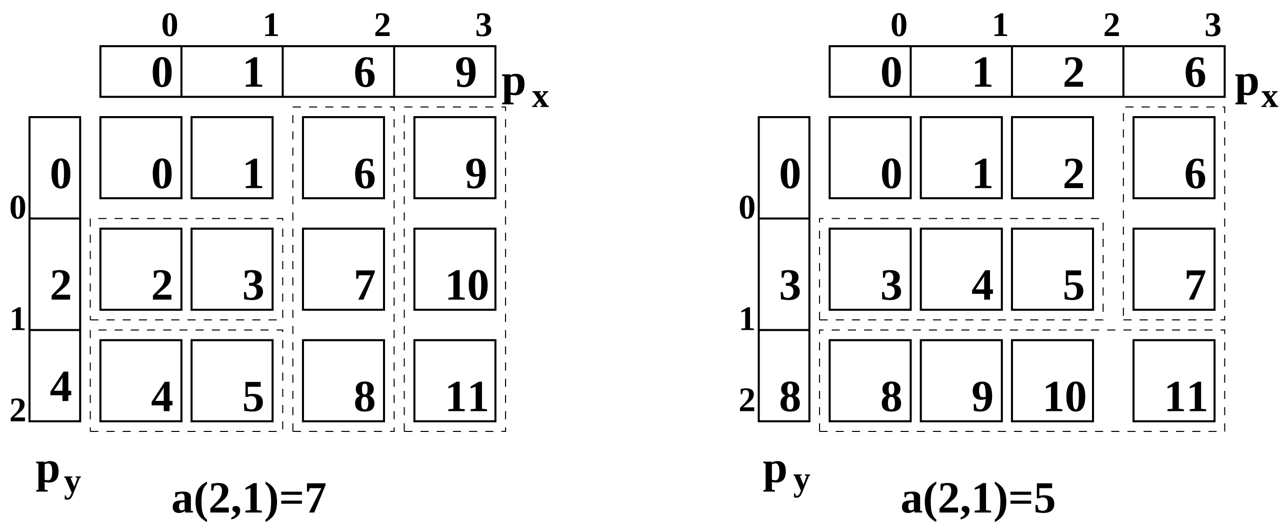
Strategy: *preserve the access method!*

Tactic: *split the pages!*:



Addressing a dynamic multidimensional array

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



$$
\begin{aligned}
a(i,j) &= \max(p_x(i), p_y(j)) + \text{ the other one}\\
a(2,1) &= \max(p_x(2), p_y(1)) + \text{ the other one}
\end{aligned}
$$

the left-hand example

$$
\begin{aligned}
&= \max(6,2) + \text{ the other one}\\
&= 6 + j = 6 + 1 = 7
\end{aligned}
$$

the right-hand example

$$
\begin{aligned}
&= \max(2,3) + \text{ the other one}\\
&= 3 + i = 3 + 2 = 5
\end{aligned}
$$

- 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. (π) If $\pi > \pi_0$ then split.

2. (α) Split unless $\alpha < \alpha_0$.

Direction criteria

3. (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. (π) Split in the direction so that π is minimized. (N.B. Keep a log of overflows in the axial index for each row, column, ..)

5. (α) 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 α the least).

Shift criterion

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

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

Algorithm MPI

(A collection of alternative algorithms.)

Try to shift first:

A. 6, 1, 3, 4, 5	emphasizes π
B. 6, 1, 3, 5, 4	π, then α
C. 6, 2, 3, 5, 4	split emphasizes α
D. 6, 2, 3, 4, 5	split using α, then π

No shift:

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