THE BINARY SORT ACCESS METHOD, US Patent 5,926,815:

Issued July 20, 1999; ASCII text to be published soon below.

BSAM adds keys over 28% faster and reads keys in sorted order over 2% faster than optimized B*-tree methods.

		Compares		Moves	
Component	Stable	Average	Maximum	Average	Maximum
heap_sort	No	n log n	n log n	n log n	n log n
quick_sort	No	n log n	n^2	n log n	n^2
radix_sort	No	n log n	n^2	n log n	n^2
BSAM	Yes	(int(1 + log2(1 + (b * (0.5 * (int((n - 1) / b)) * (int((n - 1) / b) + 1)))))) + ((((b ^ 2) * (int((n - 1) / b)) + ((mod (n - 1, b) + 1) ^ 2)) / 4) + (b * (int((n - 1) / b)))), where b is the number of blanks inserted, mod(x, y) = x - y * int(x / y), log base two as lg(x) = log(x) / log (2), and see Sedgewick and Flajolet [1996]	((b * (1 + (1.5 * (b - 1)))) * (int((n - 1) / b))) + (b *((int((n - 1) / b) + 1) * ((int((n - 1) / b) + 1) - 1) / 2)) +(mod(n - 1, b) + 1) * (1 + (1.5 * (mod(n - 1, b)))) + (int (1 +log2(1 + (b * (0.5 * (int((n - 1) / b)) * (int ((n - 1) / b) +1)))))), where b is the number of blanks inserted, mod(x, y) = x - y *int(x / y),and log base two as lg(x) = log (x) / log(2)	((((b ^ 2) * (int((n - 1) / b)) + ((mod (n - 1, b) + 1) ^2)) / 4) + (b * (int((n - 1) / b)))), where b is the number of blanks inserted, mod(x, y) = x - y * int(x / y), and see Sedgewick and Flajolet [1996].	((b * (b - 1) / 2) * (int ((n - 1) / b))) + (b * ((int((n- 1) / b) + 1) * ((int((n - 1) / b) + 1) - 1) / 2)) + (mod((n - 1), (b)) + 1) * (((mod((n - 1), (b)) + 1) - 1) / 2), where mod(x,y) = x - y * int(x / y) and b is the number of blanks inserted

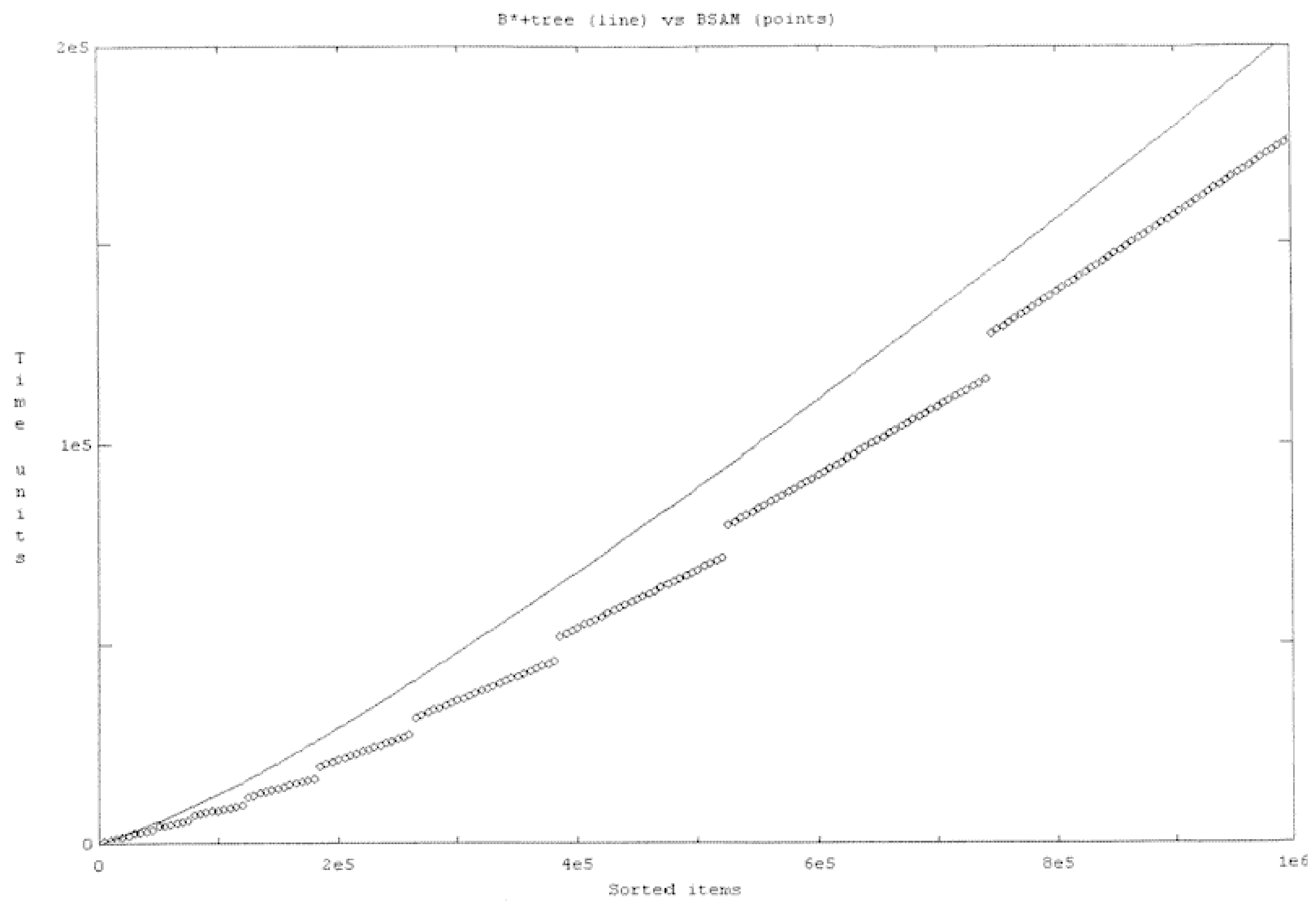
Contrast of worst performance for compares and moves of quick sort or radix sort (n^2) and BSAM.

Compares				*Moves*			
n	n^2	BSAM	Times_fewer	n	n^2	BSAM	Times_fewer
100	10^4	2 089	4.79	100	10^4	990	10.10
1 000	10^6	61 046	16.39	1 000	10^6	50 040	19.98
10 000	10^8	4 709 553	21.23	10 000	10^8	4 599 540	21.74
100 000	10^10	456 104 559	21.92	100 000	10^10	455 004 540	21.98
1 000 000	10^12	45 470 954 566	21.99	1 000 000	10^12	45 459 954 540	22.00
10 000 000	10^14	4.5456105e+12	22.00	10 000 000	10^14	4.54555005e+12	22.00
100 000 000	10^16	4.545471e+14	22.00	100 000 000	10^16	4.54546e+14	22.00
1 000 000 000	10^18	4.5454561e+16	22.00	1 000 000 000	10^18	4.545455e+16	22.00

Contrast of average performance for compares and moves of quick sort or radix sort (n lg n [+ n]) and BSAM.

Compares				*Moves*			
n	n lg n [+ n]	BSAM	Times_fewer	n	n lg n [+ n]	BSAM	Times_fewer
100	764	381	2.01	100	764	372	2.05
1 000	10 966	3 754	2.92	1 000	10 966	3 738	2.93
10 000	142 877	37 520	3.81	10 000	142 877	37 497	3.81
100 000	1 760 964	375 017	4.70	100 000	1 760 964	374 988	4.70
1 000 000	20 931 569	3 750 033	5.58	1 000 000	20 931 569	3 749 997	5.58
10 000 000	2.4253497e+8	37 500 030	6.47	10 000 000	2.4253497e+8	37 499 987	6.47
100 000 000	2.7575425e+9	3.75e+8	7.35	100 000 000	2.7575425e+9	3.75e+8	7.35
1 000 000 000	3.0897353e+10	3.75e+9	8.24	1 000 000 000	3.0897353e+10	3.75e+9	8.24

Graph of performance contrasting B*+tree and BSAM.


B*+tree (line) vs BSAM (points)
2e5
1e5
Time units
2e5
4e5
6e5
8e5
1e6
Sorted items