

Heapsort vs. Quicksort

Most groups had sound data and observed:

- Random problem instances
 - Heapsort runs perhaps 2x slower on small instances
 - It's even slower on larger instances
- Nearly-sorted instances:
 - Quicksort is worse than Heapsort on large instances.

Some groups counted comparisons:

- Heapsort uses more comparisons on random data

Most groups concluded:

- Experiments show that MH2 predictions are correct
 - At least for random data

Sorting Random Data

N	Time (us)	
	Quicksort	Heapsort
10	19	21
100	173	293
1,000	2,238	5,289
10,000	28,736	78,064
100,000	355,949	1,184,493

"HeapSort is definitely growing faster (in running time) than is QuickSort. ... This lends support to the MH_2 model."

Does it? What other explanations are there?

Sorting Random Data

N	Number of comparisons	
	Quicksort	Heapsort
10	54	56
100	987	1,206
1,000	13,116	18,708
10,000	166,926	249,856
100,000	2,050,479	3,136,104

But wait - the number of comparisons for Heapsort is also going up faster than for Quicksort. This has nothing to do with the MH_2 analysis.

How can we see if MH_2 analysis is relevant?

Sorting Random Data

N	Time (us)		Compares		Time / compare (ns)	
	Quicksort	Heapsort	Quicksort	Heapsort	Quicksort	Heapsort
10	19	21	54	56	352	375
100	173	293	987	1,206	175	243
1,000	2,238	5,289	13,116	18,708	171	283
10,000	28,736	78,064	166,926	249,856	172	312
100,000	355,949	1,184,493	2,050,479	3,136,104	174	378

Nice data!

- Why does $N = 10$ take so much longer per comparison?
- Why does Heapsort always take longer than Quicksort?
- Is Heapsort growth as predicted by MH_2 model?
 - Is N large enough to be interesting?? (Machine is a Sun Ultra 10)

... and on a 1.2 GHz Pentium III

N	Time (us)		Compares		Time / comp (ns)	
	Quicksort	Heapsort	Quicksort	Heapsort	Qsort	Hsort
1,000	906	3,502	6,150	12,641	147	277
10,000	11,339	43,478	83,398	168,716	136	258
100,000	119,440	333,330	878,000	1,430,000	136	233
1,000,000	1,000,000	3,500,000	7,230,000	13,800,000	138	254
10,000,000	9,500,000	44,000,000				
100,000,000	93,813,000	265,516,000				

Strange data !

- Heapsort time per comparison is not increasing.
- What do you think of: "Both seem to be performing at $O(n \lg n)$ " ?
- What else is surprising (or suspicious)?

... and on a 1.2 GHz Pentium III

N	Time (us)		Compares		Time / comp (ns)	
	Quicksort	Heapsort	Quicksort	Heapsort	Qsort	Hsort
1,000	906	3,502	6,150	12,641	147	277
10,000	11,339	43,478	83,398	168,716	136	258
100,000	119,440	333,330	878,000	1,430,000	136	233
1,000,000	1,000,000	3,500,000	7,230,000	13,800,000	138	254
10,000,000	9,500,000	44,000,000				
100,000,000	93,813,000	265,516,000				

Strange data !

- Heapsort time per comparison is not increasing.
- What else is surprising (or suspicious)?
 - Number of comparisons is sublinear for both methods
 - Times are sublinear - particularly last Heapsort entry
 - Some times are "round", others aren't
 - Times are huge - over 150 cycles/compare!

Experiments may be correct, but further study is certainly needed!

More data ...

N	Time (us)		Time/compare (ns)	
	Quicksort	Heapsort	Quicksort	Heapsort
1.2 GHz Pentium III; 16KB L1, 256 KB L2				
1,000	906	3,502	147	277
10,000	11,339	43,478	136	258
100,000	119,440	333,330	136	233
1,000,000	1,000,000	3,500,000	138	254
10,000,000	9,500,000	44,000,000		
100,000,000	93,813,000	265,516,000		
700 MHz Pentium III 16KB L1, 128 KB L2				
1,024	426	395	35	25
10,240	4,067	5,344	26	26
131,072	86,318	172,560	35	53
1,048,576	1,614,762	2,708,630	70	90
4,194,304	4,244,722	14,946,161	42	114
41,943,040	117,758,944	247,471,102	103	166

Fine print: For the second set of data, I made up number of comparisons
 $(N (\lg N + 2))$ for Quicksort, $1.3 N (\lg N + 2)$ for Heapsort)

Quicksort complexity, random inputs

N	266 MHz	Ultra 10	143 MHz	1.2GHz
100	131	173	103	
1,000	1,949	2,238	1,238	906
10,000	27,311	28,736	14,162	11,339
100,000	362,413	355,949	157,417	119,440
1,000,000	4,493,152		1,523,839	1,000,000

(times in microseconds)

First two groups' data looks a bit like $n \lg n$.

Third looks closer to linear.

Fourth is sublinear.

What could cause this variation??

Bonus point opportunity!

You may (if you wish) submit supplemental data and analysis of your project by Tuesday's class (firm deadline).

Goal is to overcome the objections I have found with your projects, or find and explain new phenomena.

I'm looking for evidence of deep thinking, not rote following of instructions.