"Box pruning revisited" – an optimization project by Pierre Terdiman – 2017

Part 14e – a bugfix and a reboot

More than a year after version 14d, I finally come back to this project for a brief update.

So, basically, a year ago my office PC died, and it rendered half of the published timings obsolete and irrelevant. I got a replacement PC, with a more recent CPU and different features (AVX in particular). I thought I would pause the project for maybe a month (a lot of things ended up on my plate at work), but before I knew it a year had disappeared.

Oh well. That's life. Especially when you have kids.

Another thing that happened is that I found a bug in version 14d. The code computing the box index in the bipartite case was wrong (off by one error). Since I had no validity test for the bipartite case, and because the reported number of pairs was correct, I did not notice that one for a while.

Right. So here is version 14e, which is pretty much the same as version 14d except:

- The bug has been fixed.
- A validity test for the bipartite case has been added.
- The bipartite case has been refactored to avoid duplicating the loop.

The bugfix has no impact on performance. We only tracked the performance of the "complete box pruning" codepath anyway.

What does have an impact on performance though, is the new PC. I had no choice but re-measure all versions on this new machine. The results are listed in Table 1.

While I was at it, I ran the same tests on my wife's laptop at home. I should have tried that sooner: turns out her laptop is more advanced than my desktop PC – it has AVX, and it is much faster! The results for that laptop are listed in Table 2.

We only keep what we previously called "safe" versions now. We do not list the timings for the "unsafe" versions anymore. I also dropped the Delta and Speedup columns to make things easier.

New office PC – Intel i7-6850K	Timings (K-Cycles)	Overall X factor
Version2 - base	66245	1.0
Version3 – don't trust the compiler	65644	-
Version4 - sentinels	58706	-
Version5 – hardcoding axes	55560	-
Version6a – data-oriented design	46832	-
Version6b – less cache misses	39681	-
Version7 – integer cmp	36687	-
Version8 – branchless overlap test	23701	-
Version9a - SIMD	18758	-
Version9b – better SIMD	10065	-
Version9c – data alignment	10957	-
Version10 – integer SIMD	12352	-
Version11 – the last branch	11403	-
Version12 - assembly	7197	-
Version13 – asm converted back to C++	8434	-
Version14a – loop unrolling	7511	-
Version14b – Ryg unrolled assembly 1	5094	~13.00
Version14c – better unrolling	5375	-
Version14d – integer cmp 2	5452	~12.15

Table 1 – results for new office desktop PC

Home laptop – Intel i5-3210M	Timings (K-Cycles)	Overall X factor
Version2 - base	62324	1.0
Version3 – don't trust the compiler	59250	-
Version4 - sentinels	54368	-
Version5 – hardcoding axes	52196	-
Version6a – data-oriented design	43848	-
Version6b – less cache misses	37755	-
Version7 – integer cmp	36746	-
Version8 – branchless overlap test	28206	-
Version9a - SIMD	22693	-
Version9b – better SIMD	11351	-
Version9c – data alignment	11221	-
Version10 – integer SIMD	11110	-
Version11 – the last branch	10871	-
Version12 - assembly	9268	-
Version13 – asm converted back to C++	9248	-
Version14a – loop unrolling	9009	-
Version14b – Ryg unrolled assembly 1	5040	~12.36
Version14c – better unrolling	5301	-
Version14d – integer cmp 2	5011	~12.43

Table 2 – results for home laptop

We can see that the new machines are faster overall than the ones we used before, but overall the results are pretty similar to what we previously saw.

What we learnt:

A bug can remain invisible for a year, even when the code is public on GitHub. I guess nobody tried to use it.

Time passes way too quickly.

We are back on track.