Inlining is a very important optimization pass of today's compilers. It saves function call overhead and provides more context for other optimization passes by replacing function's call site with its body. We revisit the current "greedy" inliner in GNU Compiler Collection, which was written more then 20 years ago and propose an alternative algorithm suitable for parallel processing. We combine the current approach of using a priority queue and the approach of the early inliner of traversing the callgraph in reverse post order by running the RPO traversal multiple times with increasing limits. Our measurements suggest the presented algorithm is worth further research and that properly tuning the constants may put it on a par with the current inliner all while allowing space for future parallelization of the IPA phase.