Homework 2 Report

1 Test Environment

I tested my code on department's ix server. It has two sockets with AMD Opteron 6376 on each socket. Each CPU has 8 cores, 16 threads. So, there are 32 hardware threads in total.

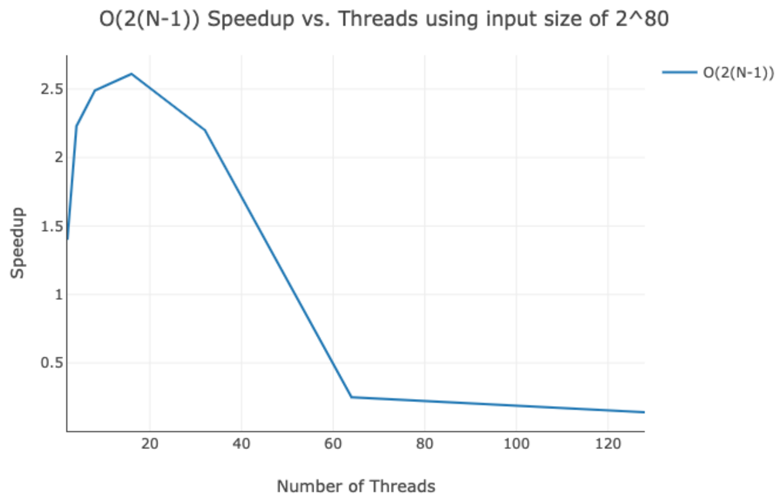
2 Test Results

	2^20	2^40	2^60	2^80
O(N-1)	6.66×10^{-3}	6.53×10^{-3}	6.58×10^{-3}	6.52×10^{-3}
O(NlogN)	4.26×10^{-2}	3.22×10^{-2}	2.32×10^{-2}	2.91×10^{-2}
O(2(N-1))	3.48×10^{-3}	3.18×10^{-3}	2.89×10^{-3}	2.90×10^{-3}

Tabelle 1: Problem Size vs. Time using 32 threads

	1	8	16	32	64	128
O(N-1)	6.65×10^{-3}	6.57×10^{-3}	6.59×10^{-3}	6.63×10^{-3}	6.69×10^{-3}	6.54×10^{-3}
O(NlogN)	4.69×10^{-2}	1.80×10^{-2}	2.08×10^{-2}	2.53×10^{-2}	3.82×10^{-2}	4.87×10^{-2}
O(2(N-1))	8.12×10^{-3}	2.64×10^{-3}	4.04×10^{-3}	5.01×10^{-3}	2.45×10^{-3}	4.38×10^{-3}

Tabelle 2: Threads vs. Time using 2^20 input size



3 Findings

Note: $O(n-1)$: base, $O(nlog(n))$: efficient serial, $O(2(n-1))$: binary tree

- By using bit-wise operation rather than pow() function to get strides, the code gets better performance when input size is large.
- When serialized, binary tree method performs slightly worse than the efficient serial method. However, when parallelized, binary tree method gains speedup.
- For input size of 2^80, the binary tree method hits peak performance when using 16 threads.