Time Comparison of Different Sequences in Shell Sort

N	Knuth	Hibbard	Pratt
100	16.42	16.56	24.68
200	53.8	55.13	85.57
300	113.95	117.99	187.28
400	202.67	210.42	334.25
500	324.18	332.02	530.21
600	474.21	481.49	777.53
700	655.27	662.39	1073.26
800	871.02	874.08	1426.71
900	1124.18	1119.25	1837.25
1000	1400.28	1399.32	2312.64
1500	3350.85	3346.26	5475.99
2000	6287.45	6279.48	10178.68
2500	10325.99	10302.17	16596.83
3000	15512.23	15472.97	24738.16

Figure 1. Times (in ms) of completion for 1000 iterations of each N, times based on 100 tests. Pratt had no added restrictions other than the sequence could not contain a number greater than N.

Pratt Sequence w/ Different Restrictions							
N	N/3	N/5	N/7	N/10	N/15	N/20	Insertion
100	23.03	20.78	18.53	14.89	16.45	18.53	12.91
200	83.7	78.24	67.61	52.55	55.25	66.93	59.53
300	190.53	175.15	154.1	124.95	123.62	153.32	172.98
400	329.18	310.18	276.08	217.34	220.05	272.04	354.75
500	517.53	492.3	436.58	339.08	347.49	433.33	634.02
600	747.91	714.44	641.37	490.58	507.31	635.59	1023.93
700	1026.83	985.81	886.03	673.99	697.84	880.37	1558.81
800	1368.07	1313.22	1182.0	890.61	922.53	1172.86	2242.27
900	1758.63	1688.27	1515.52	1141.63	1186.82	1505.93	3106.71
1000	2194.4	2108.27	1890.67	1427.9	1483.96	1883.49	4169.4

Figure 2. Times (in ms) of completion for 1000 iteration of each N for the restrictions listed. N/3 means that only numbers up to N/3 were added to the sequence to be used in Shell Sort. Times based on 100 tests for each restriction.

Figure 1 shows that Pratt is very slow compared to the rest of the sequences applied in shell sort, while Hibbard and Knuth take virtually the same amount of time. Why is this? I think the answer lies in the creation of the sequence for Pratt. Pratt's sequence is not easily attained like Hibbard and Knuth. A lot of if-statements are required to put the Pratt sequence together, in order, in O(N) time. There are other ways of doing the same thing, but many of them take O(nlogn) time, like creating an unordered array of the sequence, then ordering it. So the method implemented in my version of ShellSortPratt is relatively

fast, just not compared to Hibbard or Knuth which does not require any if statements as creating an ordered sequence is just as simple as calculating one number, then adding it to the sequence.

The data in Figure 2 suggests there is an "optimal" restriction for Pratt sequence. This optimal restriction is around N/10. The reason behind this is possibly the fact that when the restriction is too big (i.e. N/3, N/5, etc.), there are too many numbers in the sequence for Shell Sort to be optimally efficient. It just has to sort too many different sets. Comparatively, this relatively low efficiency compared to N/10 becomes very efficient when compared to Insertion Sort (final column). However, the difference in restrictions can't be denied. So, is there an "optimal" restriction for each sequence? I would say there is. Is this "optimal" restriction always N/10. I would say no, because Pratt sequence increases exponentially it is reasonable to think that there is a possibility that a sequence that does not increase exponentially may have a different "optimal" restriction. Also, notice as the restriction continues to get smaller and smaller (N/15, N/20), we start losing the advantage of Shell Sort. I think this is because although you don't have to perform Shell Sort on so many sets, if the sequence is too small (length 2 or 3 or 5), then the efficiency starts to approach Insertion Sort, which is Shell Sort with a sequence of just [1].