VII. Production Runs, Checks, Poset Counts and Resource Consumption

In this chapter we describe the exact order in which program runs were completed. We also include the descriptions of any checks made to verify the results from our programs. For each stage of the process, we indicate the main program used, the range of poset sizes tested, and the subdirectories involved. This information is summarized in Table 7.1. In the following section, we also give the consumption of time and memory for the three largest poset sizes in the longer program runs. The final part of this chapter is a table containing the number of posets fitting various criteria. Some of the entries in Table 7.2 were found only by using Inspct.nb to check the length of files created in this project. However some were obtained by looking up poset entries in the On-line Encyclopedia of Integer Sequences (OEIS) [Sl]. Others were taken by examining lists of posets from Proctor's work in the 1990s. The posets from Stembridge's website [St] were the starting point for Proctor's 1994 computations. These were verified and extended by Wilmesmeier's project [Wi]. Several of these poset lists are stored in the subdirectory Ancints. Some of the programs we created generate some of the same lists. In order to check the current work, we compare some of the files created by our programs to the files in Ancints using Compar.nb. One of the final checks is comparing the connected non-d-complete jdt posets to the connected non-d-complete L-R posets. If the files containing these lists are identical for a given n, Proctor's conjecture is verified for posets of size n. The results of all comparisons are included in section C. We begin however with some comments on timing and memory for the project.

A. Time and Memory Considerations

As mentioned in the introduction, our programs were created and run within version 5.1 of Mathematica. In section C, we include the consumption of memory and time for several cases in each of the final run descriptions. These values are rounded to three significant figures if the first digit is 1 and two otherwise. The program runs were performed on an IBM, model 6794UN9, with Windows XP. This computer has 512 MB of RAM available. The computer has an Intel Pentium 4 processor, version x86 Family 15 Model 1 Stepping 2 and a processing speed of 1.99 Ghz. On this computer, our longest run lasted approximately 7 hours. It should be noted that the Mathematica Kernel was not restarted for most of the runs. We are not sure how this affects memory and time.

Many of our programs involve reading in data from a poset list file and later writing data to one or more other files. In the development of our programs, we had to consider the way

such files would be read-in. The most elegant way to do this is to read-in all the posets in the file at once and perform the desired calculation for all the posets using a Map[] command. (This is done in all main programs except JDTLRscan.nb. For JDTLRscan.nb, a loop turned out to use less time and memory then a Map[] command, though all posets are still read-in at once before the loop begins.) As long as there are not too many posets in the file, this should also be faster than the next method. The disadvantage of this method is that it may result in more memory consumption. For our project, we considered posets with less than or equal to 9 elements and this read-in method worked. Except for one run, the most RAM used was 221 MB. In the final run of JDTLRscan.nb for n = 9 a small amount of virtual memory was needed, which likely caused the run to take more time. The memory restriction of 512 MB also prevented us from running the program JDTLRscan.nb using the input file umaxmls9. This motivated the removal of the d-complete posets before testing for the jdt and Littlewood-Richardson properties.

The amount of available memory will also cause some of the programs not to work when applied to posets of size 10 using the above method. Since the posets of size 10 were not a high priority for us, this was not a problem in the current project. To go further in the future however the following method may need to be used. Another way to get the posets contained in a file is to process the posets one at a time. As each poset is read-in individually, the desired calculation is performed and the memory consumption is analyzed. These computations are then done over several days in the background. The main disadvantage of this method is the length of time it takes. While Engine was not needed for this project, it would provide a useful environment for this second method of reading in data.

INSERT TABLE 7.1

THIS PAGE IS INTENTIONALLY LEFT BLANK.

C. Descriptions of Final Program Runs and Checks

Final Run Stage 1

Main Program: ThrghBldUp.nb **Date/Time of Run:** 5/27/2005 3:29 PM
Starting poset size: n = 1 **Ending poset size:** n = 7

Input Subdirectory: NatLabs
Output Subdirectories: NatLabs, ThrStds, InvExts, StdIsos, Lookups

Resource Consumption for the 3 Largest Values of n

Poset Size	**n = 5**	**n = 6**	**n = 7**
RAM (MB)	0.20	2.6	49.0
Real Time (min)	0.05	0.55	12.9
CPU Time (min)	0.02	0.20	6.4
Output File Sizes (KB)			
natlabs*	8	130	3,100
thrstds*	2	9	68
invexts*	11	162	3,400
stdisos*	15	220	5,200
lookups*	2	12	89

Length of Files

For * = n, the number of lines in the files natlabs* and stdisos* should be the number of naturally labeled posets of size n. Similarly the number of lines in the files thrstds*, invexts*, and lookups* should be the number of posets of size n. This was verified using Inspct.nb and the OEIS.

Final Run Stage 2

Main Program: QckBldUp.nb
Date/Time of Run: 5/29/2005 11:10 AM
Starting poset size: n = 1
Ending poset size: n = 9

Input Subdirectory: StdPsts
Output Subdirectory: StdPsts

Resource Consumption for the 3 Largest Values of n

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	1.90	18.5	220
Real Time (min)	1.72	25.0	420
CPU Time (min)	1.63	24.0	410
Output File Sizes (KB)			
stdpsts*	68	650	8,100

Length of Files

For * = n, the number of lines in the file stdpsts* should be the number of posets of size n. This was verified using Inspct.nb and OEIS.

Comparison Check 2.1

Using Compar.nb, we find that stdpsts7 and thrstds7 are identical files. This comparison took less than 30 seconds to complete.

Comparison Check 2.2

Here we compare the file stdpsts8 to canonfs8. The second file contains the child forms of posets of size 8 found in 1994. Using Compar.nb, we see that the sorted list of standard forms of elements in canonfs8 is identical to the sorted list of posets in stdpsts8. The comparison of these files took about 6.5 minutes.

Final Run Stage 3

Main Program: BldUMxmls.nb

Date/Time of Run: 5/31/2005 3:35 PM
6/08/2005 2:28 PM

Starting poset size: n = 1
n = 7

Ending poset size: n = 6
n = 9

Input Subdirectory: StdPsts
Output Subdirectory: UMaxmls

Resource Consumption for the 3 Largest Values of n

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	0.08	0.82	7.5
Real Time (min)	0.02	0.128	1.06
CPU Time (min)	0.04	0.03	0.02
Output File Sizes (KB)			
umaxmls*	11	81	770

Length of Files

For * = n, the number of lines in the file umaxmls* should be the number of posets of size n with a unique maximal element. This is also the number of posets of size n–1. Using Inspct.nb, we find that the number of lines in umaxmls* is equal to the number of posets of size n–1 obtained from OEIS.

Comparison Check 3.1

The file ancumxs7 contains the posets of size 7 with a unique maximal element found in the 1990s. The sorted list of standard forms of the posets in ancumxs7 is identical to the sorted version of umaxmls7. This verifies that umaxmls7 contains all standard posets with a unique maximal element.

Final Run Stage 4

Main Program: Selct.nb
Date/Time of Run: 6/02/2005 5:27 PM
Starting poset size: n = 8
Ending poset size: n = 8

Input Subdirectory: StdPsts
Output Subdirectory: ScrnUMs

Resource Consumption for n = 8

The time and amount of RAM used was not measured for this run.

Output File Size (KB)

scrnums8	81

Length of Files

The number of lines in the file scrnums8 should be the number of posets of size 8 with a unique maximal element. This is verified in the following comparison check.

Comparison Check 4.1

Using Compar.nb, we see that the files umaxmls8 and scrnums8 are identical.

Final Run Stage 5

Main Program: Selct.nb

Date/Time of Run: 6/03/2005 2:30 PM
6/08/2005 2:45 PM

Starting poset size: n = 1
n = 7

Ending poset size: n = 6
n = 9

Input Subdirectory: UMaxmls
Output Subdirectory: Cnctdcs

Resource Consumption for the 3 Largest Values of n

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	0.08	0.47	4.2
Real Time (min)	0.01	0.03	0.158
CPU Time (min)	0.01	0.016	0.129
Output File Sizes (KB)			
cnctdcs*	3	7	20

Length of Files

For * = n, the number of lines in the file cnctdcs* should be the number of connected posets of size n that are d-complete. For n ≤ 8, this number was known from Behrman's and Wilmesmeier's work with Proctor in the 1990's. We verify that cnctdcs* has this many lines using Inspct.nb, for * ≤ 8.

Comparison Check 5.1

The file anctdcs*, for 4 ≤ * ≤ 9 contains the connected d-complete posets of size * according to the old d-Complete program. To check the contents of the file cnctdcs*, we compare it to anctdcs*. For * ≤ 8, the sorted list of standard forms in anctdcs* is identical to the sorted version of cnctdcs*. The standard forms of the posets in anctdcs9 are a proper subset of cnctdcs9. Upon further inspection, we discovered that the file anctdcs9 is missing one d-complete poset. Proctor found the corresponding error in the 1994 routine and saw that it did not affect posets with less than 9 elements.

Final Run Stage 6

Main Program: Complemnt.nb **Date/Time of Run:** 6/03/2005 2:37 PM
6/08/2005 3:00 PM

Starting poset size: n = 1 **Ending poset size:** n = 6
n = 7 n = 9

Input Subdirectories: Umaxmls, Cnctdcs
Output Subdirectory: UMxndcs

Resource Consumption for the 3 Largest Values of n

The time and memory consumption for this run was not measured. The memory consumption is not significant and the run took less than a minute to complete.

Output File Sizes (KB)	**n = 7**	**n = 8**	**n = 9**
umxndcs*	9	75	750

Length of Files

For * = n, the number of lines in the file umxndcs* should be the number of posets of size n with a unique maximal element minus the number of connected posets that are d-complete. For $n \leq 8$, this number was known from Behrman's and Wilmesmeier's work with Proctor in the 1990's. We verify that umxndcs* has this many lines using Inspct.nb, for $* \leq 8$ and use Inspct.nb to find the number of lines in umxndcs9.

Comparison Check 6.1

The file anumnts7 contains the non-tree posets of size 7 that have a unique maximal element. To check the contents of the file umxndcs7, we compare it to anumnts7. The list of posets in umxndcs7 is a proper subset of the standard forms of the posets in anumnts7.

Final Run Stage 7

Main Program: JDTLRscan.nb **Date/Time of Run:** 6/08/2005 3:05 PM
Starting poset size: n = 5 **Ending poset size:** n = 9

Input Subdirectory: UMxndcs
Output Subdirectories: CJDTndc, CLRndcs

Resource Consumption for the 3 Largest Values of n

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	3.4	45	830
Real Time (min)	0.100	1.31	58
CPU Time (min)	0.09	1.28	26
Output File Sizes (KB)			
cjdtndc*	1	3	6
clrndcs*	1	3	6

The file cjdtndc is empty for * ≤ 4 and is not created by this program run. This file was created by hand for the sake of completeness.*

Memory/Time Remark

For n = 9, the 830 MB exceeds the 512 MB of available RAM. This resulted in virtual memory being used. Before the final run, a new hard drive was put in the computer. It should be noted that on the former hard drive the run for n = 9 took 43 minutes.

Length of Files

For * = n, the number of lines in the file cjdtndc* should be the number of connected posets of size n that have the jdt property and are not d-complete. For n ≤ 8, this number was known from Behrman's and Wilmesmeier's work with Proctor in the 1990's. We verify that cjdtndc* has this many lines using Inspct.nb, for * ≤ 8.

Comparison Check 7.1

The file anctjdt8 contains the connected jdt posets as found by the old jdt program from 1997. To check the contents of cjdtndc8 we compare it to anctjdt8. Using Compar.nb, we find that the posets in cjdtndc8 are a proper subset of the list of standard forms of posets in anctjdt8.

Comparison Check 7.2

To confirm the conjecture, we compare the files cjdtndc* and clrndcs* for 5 ≤ * ≤ 9. Using Compar.nb, we see that these files are identical.

Final Run Stage 8

Main Program: Selct.nb
Date/Time of Run: 6/09/2005 4:00 PM
Starting poset size: n = 1
Ending poset size: n = 9

Input Subdirectory: StdPsts
Output Subdirectory: Connets

Resource Consumption for the 3 Largest Values of n

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	0	0.56	46
Real Time (min)	0.112	1.01	11.6
CPU Time (min)	0.03	0.27	3.2
Output File Sizes (KB)			
connets*	56	570	7,300

Length of Files

For * = n, the number of lines in the file connets* should be the number of connected posets of size n. This was verified using Inspct.nb and OEIS.

Comparison Check 8.1

The file anctcnn8 contains the connected posets of size 8 as found in the 1990s. To check the contents of the file connets8, we compare it to anctcnn8. Using Compar.nb, we see that the sorted list of standard forms of posets in anctcnn8 is identical to the sorted version of connets8. This comparison took 4.0 minutes.

INSERT TABLE 7.2

References

[Sl] N.J.A. Sloane, The On-line Encyclopedia of Integer Sequences, http://www.research.att.com/~njas/sequences/, A001035, A006455, A000112, A000608, 1996 – 2005.

[St] J. Stembridge, Poset lists, http://www.math.lsa.umich.edu/~jrs/archive.html.

[Wi] S. Wilmesmeier, Toward Characterizing Posets with the Jeu de Taquin Property, UNC Masters Project, April, 1997.