Supplemental Chapter VIII: Hook Length Poset Computations

Cheryl A. Gann and Robert A. Proctor

July 15, 2005

Table of Contents

THIS PAGE IS INTENTIONALLY LEFT BLANK.

A. Introduction (including Definitions)

Let P be any finite partially ordered set (poset), and set $p := |P|$. In his 1970 thesis, Richard Stanley generalized the notions of integer partitions and plane partitions to the notion of P-partition for P. This is described in Section 4.5 of [St1]. Let $n \geq 0$. A P-partition of n for P is an order reversing map σ from P to the non-negative integers such that the sum of its images $\sigma(y)$ for $y \in P$ is n. The generating function $G_P(x)$ is defined to be the sum of $x^{|\sigma|}$ over all P-partitions for P. The polynomial $W_P(x)$ is defined to be the sum of $x^{\sigma(\pi)}$ over all inverse extensions π of P, where $\sigma(\pi)$ is the sum of all indices $1 \leq j \leq n-1$ such that $\pi(j) > \pi(j+1)$. The main theorem in the subject is Stanley's Theorem 4.5.8: $G_P(x) = W_P(x)/\prod_{1 \leq i \leq p}(1-x^i)$. In his thesis Stanley noted the following remarkable phenomenon for posets P which are rooted trees (with the root being the unique maximal element) and for posets P arising from Young diagrams (with the "first box" corresponding to the unique maximal element): Not only does $W_P(x)$ evenly divide this denominator, the result of this cancellation can itself be factored into the form $\prod_{1 \leq k \leq p}(1-x^{h_k})^{-1}$, where $\{h_k: 1 \leq k \leq p\}$ is a set of p positive integers. Here we say that a poset P is a <u>hook length poset</u> if there exists some such set of positive integers. (Although Stanley had begun to use this terminology by 1970, he did not include it in print in any of his principal publications on P-partitions.) In his thesis Stanley conjectured that posets arising from shifted Young diagrams also had this property; this was later proved by Emden Gansner. In 1997, Dale Peterson and Robert Proctor proved that any d-complete poset has this property. If a poset R is the direct sum of two posets P and Q, then it is not hard to see that $G_R(x) = G_P(x)G_Q(x)$.

Given the fact that much of the programming needed for performing the computations required to determine whether a given poset has the hook length property had been done to test the Littlewood-Richardson conjecture, the two authors of this supplemental chapter to Gann's project decided to quickly screen many small posets for this property. In the case of connected hook length posets, these computations reproduce and confirm the calculations performed by the second author and David Behrman between 1994 and 1996 for posets with up to 8 elements. Not only have the programs used currently been written independently of the programs used then, the posets used now are in the new standard form and are listed in the new standard order.

There are three Parts of these hook length poset computations:

I. All hook length posets with n elements for $1 \le n \le 7$.

II. Connected hook length posets with n elements for $1 \le n \le 9$.

III. Indecomposable disconnected hook length posets with n elements for $1 \le n \le 9$.

Here are the definitions and the motivation for Part III: A poset is disconnected if it has two or more connected components. A disconnected hook length poset is decomposable if it can be expressed as a direct sum of two hook length posets. There is no theoretical reason to expect every disconnected hook length poset to be expressible as a direct sum of two non-empty hook length posets. In fact, Stanley has noted that the direct sum of the three element "V" poset and the four element total order has the hook length property, even though the "V" poset does not. This raises the question of how common indecomposable disconnected hook length posets are, and whether there are any examples of the direct sum of two non-hook length posets having the hook length property.

The three sets of lists of hook length posets and their sets of hook lengths produced by Parts I - III have been posted at the Chapel Hill Poset Atlas web site. The counts of such posets by size within each class are given in Table 8.3 below.

Given the results of the hook length computations in the mid-1990's, the second author of this supplement conjectured that every connected hook length poset must have a unique maximal element. The present computations have extended the confirmation of this conjecture from $n \le 8$ to $n \le 9$.

B. Computational Strategy and Program Descriptions

Recall that the program ThrghBldUp.nb in Gann's project computed and wrote out all inverse extensions for all posets with up to 7 elements. Moreover, there are far more posets with 8 elements than there are with 7 elements, and the computation of inverse extensions for a typical 8 element poset takes significantly longer than does the same computation for a typical 7 element poset. Consequently, we compute all hook length posets only up through $n = 7$. Moreover, our computational strategies in Parts II (connecteds) and III (indecomposable disconnecteds) for $8 \le n \le 9$ will be different than our strategies in those parts for $1 \le n \le 7$. Hence we will refer to Parts IIa and IIb, and also to Parts IIIa and IIIb.

Scripts for the following computational approaches are given in Section D below, and a table which more tersely summarizes the final runs and checks is given in Section E.

The function **WPx[]** computes the polynomial $W_P(x)$ for a poset given in child form using the function InvExtsWRI[]. Recall that this function "looks up" the stored inverse extensions when $n \le 7$. The main program **GenWPx.nb** applies the function WPx[] to each poset in an input list.

The function **HookQ[]** determines whether a given poset P has the hook length property by reading in $W_P(x)$ and considering the quotient $\prod_{1 \le i \le p}(1-x^i)/W_P(x)$: If this quotient is a polynomial, then the divisions by $(1-x^n)$, $(1-x^{n-1})$, … are successively repeatedly attempted. The number of successful divisions by each attempted dividend is noted. If these divisions completely factor the quotient, then P has the hook length property. The main program **HookPsts.nb** then writes P out to the primary output list, and the sorted set of hook length exponents for P is written out to a parallel data file.

Part I of this supplemental hook length poset project is implemented simply by applying GenWPx.nb and HookPsts.nb to the lists of all posets for $1 \le n \le 7$. Part IIa (for $1 \le n \le 7$) is implemented by using Selct.nb with ConnctdQ[] to pull out the hook length posets which are connected. Using Compar.nb and the lists umaxmls*, it was re-confirmed that every connected hook length poset for $1 \le n \le 7$ has a unique maximal element. The data file creation main program **Extrct.nb** extracts the sublist Z of entries of a data list X which parallels a poset list W, corresponding to a sublist Y of posets in W. Here it is used to extract the hook length sets corresponding to the connected hook length posets.

The main program GenWPx.nb used 290 MB of the 512 MB available when computing the polynomials $W_P(x)$ for the non-d-complete connected posets for $n = 8$. Since it was clear that this program would use too much memory to do the analogous computation for $n =$

9, we rewrote it in a fashion which would use less memory. The new main program **BigWPx.nb** was efficient enough to include the connected d-complete posets for both $n = 8$ and $n = 9$.

In Part IIIa for $1 \le n \le 7$, we first used the main program **DrctSum.nb** to form the list of all hook length posets which are not of interest: these would be all connected hook length posets of the given size, together with the directs sums of any two smaller hook length posets. The formation of the standard form of the direct sum of any two posets was accomplished using the function **DirectSm[]**. The resulting list was removed from the list of all hook length posets of the given size by Complemnt.nb, and the corresponding hook length sets were extracted by Extrct.nb. Only four posets were produced by this search; each had 7 elements. These posets would not have been surprising to Stanley, since they consisted of the directs sums formed from the "V" poset together with each of the four rooted trees which have four elements. (There will always be a denominator factor of $(1-x^n)$ corresponding to the maximal element of a rooted tree with n elements. In particular, the factor $(1-x^4)$ arising here cancels the numerator $1+x^2$ from $G_P(x)$ for the "V" poset, leaving an acceptable quotient of $(1-x^2)$.)

This approach will not work in Part IIIb when $8 \le n \le 9$, since we do not have the lists of all hook length posets available for $n = 8$ and $n = 9$. Since we are seeking disconnected hook length posets R, it suffices to search for direct sums of two (not necesarily connected) posets P and Q with a total of n elements such that the product $G_P(x)G_Q(x)$ satisfies the hook length criteria for n-posets. Let $R := P \oplus Q$, $p := |P|$, and $q := |Q|$. From now on assume that $p \le q$. Note that $W_R(x) = W_P(x)W_Q(x)C(p+q,p;x)$, where $C(p+q,p;x)$ is the Gaussian coefficient polynomial. So we seek posets P and Q such that $p + q = n$ and such that $W_P(x)W_Q(x)C(p+q,p;x)$ satisfies the usual test on $W_R(x)$ for R to be a hook length poset.

Our alternative approach to finding all of the indecomposable disconnected hook length posets relied upon some mathematical reasoning. First we note:

Proposition 1. Let R be a hook length poset. Suppose that R can be expressed as the direct sum of a one element poset P and another poset Q. Then the poset Q must be a hook length poset.

Proof. Recall that $G_R(x) = G_P(x)G_Q(x)$. Suppose that Q is not hook length. Then $W_Q(x)$ must contain some factors which cannot cancel within the standard denominator of $G_Q(x)$. Note that $G_P(x) = 1/(1-x)$. For R to be hook length, the binomial $1-x$ must help with the $G_Q(x)$ cancellation challenge. This implies that $1-x$ must divide $W_Q(x)$. This would imply that the sum of the coefficients of $W_Q(x)$ is zero. But the coefficients of $W_Q(x)$ are non-negative, and at least one of them is positive. Therefore Q must be hook length. □

This proposition implies that there is no point in considering direct sums $P \oplus Q$ where $p = 1$. So for $n = 9$ we have $q \le 7$, meaning that our stored lists of $W_P(x)$ suffice. Since we want R to be an indecomposable hook length poset, we do not consider pairs of P and Q where both P and Q are known to be hook length. Our main program **IDHLP.nb** implements this approach.

It may seem that we are now ready to perform the desired searches in Part IIIb. However, the following twist can occur: The direct sum P of the three element vee poset V and the one element poset is not hook length. Let Q be the four element total order. It is know that $V \oplus Q$ is hook length. Therefore $R = P \oplus Q$ is a decomposable disconnected hook length poset. It may have seemed that we were avoiding the consideration of this R due to Proposition 1 and our consequent requirement that $p \ge 2$. However, the poset R can sneak into our list of IDHLP's as the direct sum of the two 4-posets P and Q.

So to prevent the inclusion of decomposable disconnected hook length posets, we see that it is not sufficient to merely check that not both P and Q are hook length. As in Part IIIa, we must form the list of of all direct sums with a total of n elements which can be formed from two hook length posets. In contrast to Part IIIa, we do not need to adjoin the list of connected hook length n-posets to the exclusion list, since these posets are not being considered to start with. When $n = 8$ we are now ready to go, since we have available the list of all hook length 7-posets for the poset Q: We first use the program DrctSum.nb with an empty list of 8-posets in lieu of the list of connected hook length 8-posets to form the exclusion list.

However, when $n = 9$ this list of all hook length 8-posets is not available to be fed into DrctSum.nb as posets Q for the formation of the exclusion list smhlpos9 with the one element poset P. The following mathematical reasoning shows that it suffices to replace this unavailable list hookpos8 with the list of all indecomposable disconnected hook length 8-

posets idhlpos8, which we have just found in Part IIIb for $n = 8$. We say that a poset is a/b/…/c-sized if the sizes of its connected components are a, b, …, c.

Proposition 2. Let R be an disconnected hook length poset of size 9. Suppose that R has a connected component P_1 of size 1. Suppose that $R - P_1$ is not connected. Suppose that R cannot be expressed as the direct sum of two hook length posets, each of size at least 2. Then $R - P_1$ must be an indecomposable disconnected hook length poset of size 8.

Corollary. Running the main program IDHLP.nb with the file idhlpos8 substituted for the file hookpos8 will produce exactly the set of all indecomposable disconnected hook length posets of size 9.

Proof of Corollary. The potential problem posed by the unavailability of the file hookpos8 is that direct sums of two hook length posets of sizes 1 and 8 will not be added to the list of inelgible 9-posets. Let R be a disconnected hook length 9-poset which consequently inappropriately would appear in idhlpos9 if no posets were added to smhlpos9 during the 1/8-iteration of the loop in Step IIIb.1 due to the unavailabilty of the file hookpos8. So R must have a connected component of size 1. Call it P_1. Using Proposition 1, IDHLP.nb does not even consider direct sums of 1-posets with 8-posets. In particular, posets of connected component sizes 1/8 are not considered to start with, and hence do not need to be added to smhlpos9. So $R - P_1$ is not connected. Even before the ad hoc file substitution fix, the list smhlpos8 will include all sums of two hook length posets in which each poset has at least two elements. We have confirmed that all three hypotheses of the proposition are satisfied. Applying the proposition, we learn that $R - P_1$ must be in the file idhlpos8. Therefore with the ad hoc fix, the poset R will appear in the file smhlpos9. Therefore all of the posets which would not be excluded due to the unavailability of the file hookpos8 will in fact be excluded because of this ad hoc fix. □

Proof of Proposition 2. Let R and P_1 be as in the statement of the proposition. Using the first hypothesis, consider the 8-poset $R_2 := R - P_1$. Since R is hook length and $|P_1| = 1$, Proposition 1 implies that R_2 is hook length. By the second hypothesis, the poset R_2 is disconnected. Suppose that R_2 can be expressed as a direct sum $P_2 \oplus P_3$ of two hook

length posets, with $|P_2| \le |P_3|$. Then $R = (P_1 \oplus P_2) \oplus P_3$ would decompose R into two hook length posets, each of size at least 2. This would contradict the third hypothesis. Therefore R_2 must be an indecomposable hook length poset. We conclude that R_2 must appear in the list idhlpos8 of indecomposable disconnected hook length posets of size 8. □

Both Parts IIb and IIIb supersede Parts IIa and IIb respectively. The descriptions of Parts IIa and IIb have been retained since they were relatively easy to implement, and since using the largely independent programs to compute the answers for the overlapping cases when $4 \le n \le 7$ produces valuable checks of program correctness.

References

[Beh] D. Behrman, An investigation of hook-length posets, UNC Masters Project, December, 1996.

[St1] R.P. Stanley, *Enumerative Combinatorics, Vol. 1,* Wadsworth & Brooks/Cole, Monterey, 1986.

C. Table 8.1: Program Interaction and Storage

	Program	Used By	Stored In
General Subroutines	NoSpcs[], WriteLstOfLst[]	All main programs added in the hook length supplement.	GenrlUtils.m
	PrntCnsmptn[]	GenWPx.nb, HookPsts.nb	GnerlUtils.m
Inverse	InvExtsWRI[]	GenWPx.nb	InvExIsoFncts.m
Extensions and	StdFormIso[]	DrctSumb.nb	InvExIsoFncts.m
Iso. Functions			
Hook Functions	WPx[]	GenWPx.nb	HookUtils.m
	HookQ[]	HookPsts.nb	HookUtils.m
	DirectSm[]	DirctSum.nb	HookUtils.m
	GaussPolyn[]	IDHLP.nb	HookUtils.m
	ToPolyn[]	IDHLP.nb	HookUtils.m
	InvStdTab[]	BigWPx.nb	HookUtils.m
$W_P(x)$ Main	GenWPx.nb		GenWPx.nb
Programs	BigWPx.nb		BigWPx.nb
Hook Length	HookPsts.nb		HookPsts.nb
Main Program			
Indecomposable	IDHLP.nb		IDHLP.nb
Disconnected			
Main Program			
Routine Poset	ConnctdQ[]	Selct.nb	PstPropTsts.m
Property Test			
Functions			
File Inspection	Compar.nb		Compar.nb
And Comparison			
Main Programs			
Data File	Selct.nb		Selct.nb
Creation Main	Complemnt.nb		Complemnt.nb
Programs	Extrct.nb		Extrct.nb
Direct Sum	DirctSum.nb		DirctSum.nb
Main Program			
Auxiliary	invexts*	InvExtsWRI[], GenWPx.nb	
Data Files	stdisos*	InvExtsWRI[], GenWPx.nb	
	lookups*	InvExtsWRI[], GenWPx.nb	

THIS PAGE IS INTENTIONALLY LEFT BLANK.

D. Run Scripts and Data File Descriptions

I. Hook Length Posets $(1 \le n \le 7)$

1. Compute the $W_P(x)$'s

Synopsis: Read in the list of all posets. Apply WPx[], which finds the list of inverse extensions for each poset by using InvExtsWRI[]. Write the lists of ascending coefficients of the resulting $W_P(x)$'s to a file which parallels the poset file, each as the sequence of coefficients of increasing powers.

Size Range:	$1 \le n \le 7$
Main Program:	GenWPx.nb
Function:	InvExtsWRI[], WPx[]
Input Subdirectory:	StdPsts
Data Subdirectories:	StdIsos, Lookups, InvExts
Output Subdirectory:	StdWPxs

2. Find the Hook Length Posets

Synopsis: Read in the lists of all posets and their $W_P(x)$'s. The function HookQ[] takes a $W_P(x)$ as its argument and tests it to determine if P has the hook length property. This function returns the empty list if not, and the list of hook exponents in increasing order if so. The main program writes the poset to a file and its sorted list of hook lengths to a parallel file.

Size Range:	$1 \le n \le 7$
Main Program:	HookPsts.nb
Function:	HookQ[]
Input Subdirectories:	StdPsts, StdWPxs
Output Subdirectories:	HookPos, HookLns

IIa. Connected Hook Length Posets $(1 \le n \le 7)$

1. Select Connected Hook Length Posets

Synopsis: Read in the lists of all hook posets and pull out the connected ones.

Size Range:	$1 \le n \le 7$
Main Program:	Selct.nb
Function:	ConnctdQ[]
Input Subdirectory:	HookPos
Output Subdirectory:	CoHkPos

1.1. Check Against Ancient List

Synopsis: Compare this modern list to the one found by 1996.

Size Range:	$7 \le n \le 7$
Main Program:	Compar.nb
Input Subdirectories:	Ancints, CoHkPos

1.2. Check Against DP-RP "d-Completes are Hook Length" Theorem

Synopsis: As a check: see if within the set of all connected posets, whether the set of d-complete posets is indeed a subset of the set of all hook length posets.

Size Range:	$1 \le n \le 7$
Main Program:	Compar.nb
Input Subdirectories:	Cnctdcs, CoHkPos

2. Test the Unique Maximal Element Conjecture

Synopsis: See if the list of all connected hook length posets is a subset of the list of all posets with a unique maximal element (is already known true up through $n = 8$).

Size Range:	$1 \le n \le 7$
Main Program:	Compar.nb
Input Subdirectories:	CoHkPos, UMaxmls

3. Extract Hook Length Sets for Connected Cases

Synopsis: Obtain the lists of hook length sets which correspond to the lists of connected hook length posets.

Size Range:	$1 \le n \le 7$
Main Program:	Extrct.nb
Input Subdirectories:	HookPos, CoHkPos, HookLns
Output Subdirectory:	CoHkLns

IIb. Connected Hook Length Posets ($8 \le n \le 9$)

1. Test Run for the New $W_P(x)$ Program

Synopsis: Test the new main program on all non-d-complete connected posets with 8 elements.

Size Range:	n = 8
Main Program:	BigWPx.nb
Function:	InvStdTab[]
Input Subdirectories:	Cnctndc
Data Subdirectories:	StdIsos, Lookups, InvExts
Output Subdirectory:	TestWPx/ctndcws8

2. Compute the $W_P(x)$'s

Synopsis: Use the new main program to compute $W_P(x)$ for all connected posets.

Size Range:	$8 \le n \le 9$
Main Program:	BigWPx.nb
Function:	InvStdTab[]
Input Subdirectory:	Connets
Data Subdirectories:	StdIsos, Lookups, InvExts
Output Subdirectory:	ConWPxs

3. Find the Hook Length Posets

Synopsis: Same as I.2, but just for connected posets.

Size Range:	$8 \le n \le 9$
Main Program:	HookPsts.nb
Function:	HookQ[]
Input Subdirectories:	Connets, ConWPxs
Output Subdirectories:	CoHkPos, CoHkLns

4. Test the Unique Maximal Element Conjecture

Synopsis: Identical to IIa.2.

Size Range:	$8 \le n \le 9$
Input Subdirectories:	CoHkPos, UMaxmls

IIIa. Indecomposable Disconnected Hook Length Posets ($1 \le n \le 7$)

1. Form All Direct Sums of Two Hook Length Posets

Synopsis: For each given n, find all of hook length posets in standard form which are connected or which can be obtained by forming the direct sum of two hook length posets.

Size Range:	$1 \le n \le 7$
Main Program:	DrctSum.nb
Function:	DirectSm[], StdFormIso[]
Input Subdirectories:	HookPos, CoHkPos
Output Subdirectory:	SumHLPs

2. Find Preliminary IDHLP's

Synopsis: Remove the sets of decomposable hook length posets found in Part 1 from the sets of all hook length posets and then extract the hook sets corresponding to the surviving posets of interest. Since IDHLP's will be found again in Part IIIb, these outputs will be designated as "preliminary".

Size Range:	$1 \le n \le 7$
Main Program:	Complemnt.nb
Input Subdirectories:	HookPos, SumHLPs
Output Subdirectory:	PIDHLPs
Main Program:	Extrct.nb
Input Subdirectories:	HookPos, IDHLPos, HookLns
Output Subdirectory:	PIDHLPh

IIIb. Indecomposable Disconnected Hook Length Posets ($4 \le n \le 9$)

1. Find Direct Sums of Hook Length Posets

Synopsis: In the next step we will want to ignore all posets which can be expressed as the direct sum of two hook length posets. Since we want to use the existing main program DrctSum.nb, some dummy empty files emptyps* must first be created by hand for $1 \le * \le 9$.

Size Range:	$4 \le n \le 9$
Main Program:	DrctSum.nb
Functions:	DirectSm[], StdFormIso[]
Input Subdirectories:	HookPos, EmptyPs
Output Subdirectories:	SmHLPos

2. Find IDHLP's

Synopsis: Run through all ways of producing a poset R with n elements from two smaller posets P and Q which are not both hook length posets and which have at least two elements apiece. (The one element poset does not need to be considered as a component.) The polynomial $W_R(x)$ for the direct sum poset R is obtained by multiplying $W_P(x)$ and $W_Q(x)$. This polynomial is tested for the hook length property. If it does, then it is checked whether R is a member of the list produced in Step 1 before being added to the list of IDHLP's.

Size Range:	$4 \le n \le 9$
Main Program:	IDHLP.nb
Functions:	DirectSm[], StdFormIso[]
Input Subdirectories:	StdPsts, StdWPxs, HookPos, SmHLPos
Output Subdirectories:	IDHLPos, IDHLPhl

Insert Table 8.2

F. Descriptions of Final Program Runs and Checks

Final Run Stage I.1

Main Program: GenWPx.nb

Date/Time of Run: 6/09/2005 5:00 PM
6/16/2005 1:40 PM

Starting poset size: n = 1
n = 6

Ending poset size: n = 5
n = 7

Input Subdirectory: StdPsts
Output Subdirectory: StdWPxs

Resource Consumption for the 2 Largest Values of n

Poset Size	**n = 6**	**n = 7**
RAM (MB)	51	62
Real Time (min)	0.43	1.45
CPU Time (min)	0.35	1.30
Output File Sizes (KB)		
stdwpxs*	8	70

Lengths of Files

For * = n, the number of lines in the file stdwpxs* will be the number of posets of size n.

Final Run Stage I.2

Main Program: HookPsts.nb

Date/Time of Run: 6/09/2005 6:00 PM
6/12/2005 4:35 PM

Starting poset size: n = 1
n = 7

Ending poset size: n = 6
n = 7

Input Subdirectories: StdPsts, StdWPxs
Output Subdirectories: HookPos, HookLns

Resource Consumption for the 2 Largest Values of n

Poset Size	**n = 6**	**n = 7**
RAM (MB)	0.80	0.94
Real Time (min)	0.01	0.175
CPU Time (min)	0.01	0.144
Output File Sizes (KB)		
hookpos*	2	6
hooklns*	1	3

Lengths of Files

For * = n, the number of lines in the file hookpos* and hooklns* should be the number of hook length posets of size n.

Final Run Stage IIa.1

Main Program: Selct.nb

Date/Time of Run: 6/09/2005 6:20 PM
6/12/2005 4:43 PM

Starting poset size: n = 1
n = 7

Ending poset size: n = 6
n = 7

Input Subdirectory: HookPos
Output Subdirectory: CoHkPos

Resource Consumption for the 2 Largest Values of n

The consumption of time and memory in these runs was not significant and so is not recorded here.

	n = 6	**n = 7**
Output File Sizes (KB)		
cohkpos*	1	3

Lengths of Files

For * = n, the number of lines in the file cohkpos* should be the number of connected hook length posets of size n. This was found during Behrman's work with Proctor in 1996. This is verified using Inspct.nb.

Comparison Check IIa.1.1

The file hklngth7 contains all connected hook length posets of size 7 as found by the old programs. To check the contents of cohkpos7 we compare it to hklngth7. Using Compar.nb we see that the sorted list of standard forms of posets in hklngth7 is identical to the sorted version of cohkpos7.

Final Run Stage IIa.1.2

Main Program: Compar.nb **Date/Time of Run:** 6/12/2005 5:00 PM
Starting poset size: n = 1 **Ending poset size:** n = 7

Input Subdirectories: Cnctdcs, CoHkPos

Comparison Check

For * ≤ 5, the files cnctdcs* and cohkpos* are found to be identical. For 6 ≤ * ≤ 7, we find that the posets in the file cnctdcs* form a proper subset of the posets in cohkpos*.

Final Run Stage IIa.2

Main Program: Compar.nb **Date/Time of Run:** 6/12/2005 5:10 PM
Starting poset size: n = 1 **Ending poset size:** n = 7

Input Subdirectories: CoHkPos, UMaxmls

Comparison Check

For * ≤ 4, the files cohkpos* and umaxmls* are found to be identical. For 5 ≤ * ≤ 7, we find that the posets in the file cohkpos* form a proper subset of the posets in umaxmls*.

Final Run Stage IIa.3

Main Program: Extrct.nb **Date/Time of Run:** 6/14/2005 1:50 PM
Starting poset size: n = 1 **Ending poset size:** n = 7

Input Subdirectories: HookPos, CoHkPos, HookLns
Output Subdirectory: CoHkLns

Resource Consumption for the 2 Largest Values of n

The consumption of time and memory was not measured for this run but was not significant.

	n = 6	**n = 7**
Output File Sizes (KB)		
cohklns*	1	2

Lengths of Files

For * = n, the number of lines in the file cohklns should be the number of connected hook length posets of size n. This is verified using Inspct.nb

Final Run Stage IIb.1

Main Program: BigWPx.nb
Date/Time of Run: 7/07/2005 3:25 PM
Starting poset size: n = 8
Ending poset size: n = 8

Input Subdirectory: Cnctndc
Output Subdirectory: TestWPx

Resource Consumption for this run

RAM (MB)	33
Real Time (min)	10.1
CPU Time (min)	8.5
Output File Size (KB)	
ctndcws8	690

Length of File

The number of lines in the file cnctndc8 should be the number of connected non-d-complete posets of size 8. This is verified using Inspct.nb and our 2005 counts.

Comparison Check IIb.1.1

Originally, we used GenWPx.nb for cnctndc8. This used 290 MB and took 8.1 minutes. This approach was discarded because it would not work for cnctndc9. The file olcndcw8 was created using that approach. Using Compar.nb, we find that ctndcws8 is identical to olcndcw8.

Final Run Stage IIb.2

Main Program: BigWPx.nb **Date/Time of Run:** 7/07/2005 3:49 PM
Starting poset size: n = 8 **Ending poset size:** n = 9

Input Subdirectory: Connets
Output Subdirectory: ConWPxs

Resource Consumption for this run

Poset Size	**n = 8**	**n = 9**
RAM (MB)	33	430
Real Time (min)	9.5	430
CPU Time (min)	8.7	420
Output File Sizes (KB)		
conwpxs*	690	11,400

Length of File

The number of lines in the file conwpxs* should be the number of connected posets of size *. This is verified using Inspct.nb.

Final Run Stage IIb.3

Main Program: HookPsts.nb **Date/Time of Run:** 7/08/2005 1:22 PM
Starting poset size: n = 8 **Ending poset size:** n = 9

Input Subdirectories: Connets, ConWPxs
Output Subdirectories: CoHkPos, CoHkLns

Resource Consumption for this run

Poset Size	**n = 8**	**n = 9**
RAM (MB)	19.9	72
Real Time (min)	1.07	14.7
CPU Time (min)	1.03	14.4
Output File Sizes (KB)		
cohkpos*	9	26
cohklns*	5	13

Lengths of Files

The number of lines in the file cohkpos* and cohklns* should be the number of connected hook length posets of size *. For * = 8, this is verified using Inspct.nb and the 1990's count for the number of connected hook length posets of size 8.

Final Run Stage IIb.4

Main Program: Compar.nb **Date/Time of Run:** 7/08/2005 1:50 PM
Starting poset size: n = 8 **Ending poset size:** n = 9

Input Subdirectories: CoHkPos, UMaxmls

Comparison for Conjecture

For * = 8 and 9, the posets in the file cohkpos* form a proper subset of the posets in umaxmls*, thereby re-confirming and extending the 1990's confirmation of the conjecture that every connected hook length poset has a unique maximal element.

Final Run Stage IIIa.1

Main Program: DrctSum.nb

Date/Time of Run: 6/12/2005 4:52 PM
6/14/2005 7:00 PM

Starting poset size: n = 7
n = 1

Ending poset size: n = 7
n = 6

Input Subdirectories: HookPos, CoHkPos
Output Subdirectory: SumHLPs

Resource Consumption for the 2 Largest Values of n

Poset Size	**n = 6**	**n = 7**
RAM (MB)	0.01	0.49
Real Time (min)	0.003	0.20
CPU Time (min)	0.003	0.189
Output File Sizes (KB)		
sumhlps*	2	5

Lengths of Files

For * = n, the number of lines in the file sumhlps* should be the number of connected hook length and decomposable hook length posets of size n.
These numbers are: 1, 2, 4, 10, 23, 63, 161.

Final Run Stage IIIa.2

Main Program: Complemnt.nb **Date/Time of Run:** 6/12/2005 4:55 PM
6/14/2005 4:27 PM

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

Input Subdirectories: HookPos, SumHLPs
Output Subdirectory: PIDHLPs

Resource Consumption for the 2 Largest Values of n

The time and memory was not measured for this run but the consumption was not significant.

	n = 6	n = 7
Output File Sizes (KB)		
idhlpos*	0	1

Lengths of Files

For * = n, the number of lines in the file idhlpos* is the number of indecomposable disconnected hook length posets of size n. This is found using Inspct.nb.

Main Program: Extrct.nb **Date/Time of Run:** 6/12/2005 4:58 PM
Starting poset size: n = 7 **Ending poset size:** n = 7

Input Subdirectories: HookPos, IDHLPos, HookLns
Output Subdirectory: PIDHLPh

Resource Consumption for this run

The time and memory was not measured for this run but the consumption was not significant.

The file idhlphl is empty for * ≤ 6 and is not created by this program run. This file was created by hand for the sake of completeness.*

Output File Size (KB)	
idhlphl7	1

Final Run Stage IIIb.1

Main Program: DrctSum.nb

Date/Time of Run: 7/07/2005 2:38 PM
7/08/2005 2:12 PM

Starting poset size: n = 4
n = 9

Ending poset size: n = 8
n = 9

Input Subdirectories: HookPos, EmptyPs
Output Subdirectory: SmHLPos

Resource Consumption for this run

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	0.47	3.9	35
Real Time (min)	0.183	1.57	9.9
CPU Time (min)	0.117	1.54	9.7
Output File Sizes (KB)			
smhlpos*	3	9	18

Lengths of Files

For * = n, the length of the file smhlpos* should be the number of decomposable hook length posets of size n. These numbers are: 5, 12, 32, 86, 242, 470.

Comparison Check IIIb.1.1

The file smhlpos7 is a proper subset of sumhlps7. The difference in the length of these files is verified to be the number of connected hook length posets of size 7.

Final Run Stage IIIb.2

Main Program: IDHLP.nb **Date/Time of Run:** 7/07/2005 2:50 PM
7/08/2005 2:25 PM

Starting poset size: n = 4 **Ending poset size:** n = 8
n = 9 n = 9

Input Subdirectories: StdPsts, StdWPxs, HookPos, SmHLPos
Output Subdirectories: IDHLPos, IDHLPhl

Resource Consumption for this run

Poset Size	**n = 7**	**n = 8**	**n = 9**
RAM (MB)	0.75	3.1	11.7
Real Time (min)	0.0180	0.09	0.75
CPU Time (min)	0.0104	0.08	0.72
Output File Sizes (KB)			
idhlpos*	1	1	2
idhlphl*	1	1	1

Lengths of Files

For * = n, the length of the files idhlpos* and idhlphl* should be the number of indecomposable hook length posets of size n. This is found using Inspct.nb.

Comparison Check IIIb.2.1

For * = 6, 7, the files pidhlps* and idhlpos* are identical. Using an earlier approach, we obtained a list of indecomposable hook length posets of size 8. This list was stored in the file pidhlps8. We also find that idhlpos8 is identical to pidhlps8.

Comparison Check IIIb.2.2

For * = 6, 7, the files pidhlph* and idhlphl* are identical. The file pidhlph8 was created in the earlier approach along with pidhlps8. We also have that pidhlps8 and idhlphl8 are identical.

Insert Table 8.3.

THIS PAGE IS INTENTIONALLY LEFT BLANK.