Exponents of 2 in the Numbers of Unlabeled Graphs and Tournaments*

Steven C. Cater and Robert W. Robinson
Computer Science Department
415 GSRC
University of Georgia
Athens, GA 30602

Abstract

Let $g(n)$ denote the number of unlabeled graphs on n nodes, and let $e(n)$ denote its 2-part, i.e., the exponent of the largest power of 2 which divides $g(n)$. It is shown that for odd $n \geq 5$,

$$e(n) = (n+1)/2 - \lfloor \log_2 n \rfloor$$

and for even $n \geq 4$

$$e(n) \geq n/2 - \lfloor \log_2 n \rfloor$$

with equality if, and only if, n is a power of 2.

Similarly, let $t(n)$ denote the number of unlabeled tournaments on n nodes and $r(n)$ its 2-part. It is shown that for all odd n,

$$r(n) = (n-1)/2$$

and for all even $n \geq 4$

$$r(n) \geq n/2$$

with equality if, and only if, $\varphi(n)/2$ is odd. Here $\varphi(n)$ is the Euler totient function.

---

*A preliminary version of this paper (without tournament results) was presented at the 22nd Southeastern International Conference on Combinatorics, Graph Theory, and Computing in Baton Rouge, LA, on February 11, 1991. The present version appears in the proceedings of that conference, *Congr. Numer.* 82 (1991) 139-155.

1 Introduction

Let $g(n)$ be the number of nonisomorphic graphs on n nodes, often referred to as unlabeled graphs. The first known publication of a formula for $g(n)$ was by Redfield [11], but his contributions went unrecognized until well after the result had been independently rediscovered several times and had become widely known. Pólya's classic paper [9] was the basis for much of the later work in graphical enumeration; it has finally become available in translation [10], accompanied by a survey of the many different kinds of research that stemmed from it over a period of 50 years. Pólya did not publish a formula for $g(n)$ since he was more interested in trees, but he included it in some of his lectures, and much later sent it to Frank Harary in a letter. His ideas were developed and expounded in Harary's influential paper [4]. This appeared just two years after Davis [1] published his own independent discovery of a formula for $g(n)$. The subject of graphical enumeration is thoroughly treated in the book of Harary and Palmer [5], where $g(n)$ is discussed in Section 4.1.

In spite of the attention accorded to $g(n)$ over the years, including asymptotic analysis and refinement by number of edges and other properties, it seems to have gone unnoticed that $g(n)$ has many factors of 2. We now show the data which led to this realization, and we prove that asymptotically there are approximately $n/2$ factors of 2 in $g(n)$. For odd n the number is determined exactly, while for even n only a lower bound is proved. The proofs are of an algebraic nature, and appear to shed no light on possible combinatorial explanations of this phenomenon.

When the question of unlabeled tournaments was raised,[1] it was natural and straightforward to perform a similar analysis on the numbers $t(n)$. The results generally parallel those for graphs, but there are slightly more factors of 2 as a function of n, and some interesting features of the numbers for even n. Tournaments were first counted by Davis [2], though the method is only a slight variation on that needed for counting graphs. The result is explained in Moon's book [8], in Section 29. It is also presented in Section 5.2 of Harary and Palmer's book [5].

Formulae for $g(n)$ and $t(n)$ require summing over partitions of n. We use $\sigma \vdash n$ to signify that σ is a partition of n. We use two different conventions for specifying a partition. The more often needed form is

$$\sigma = [s_1, s_2, \cdots, s_l]$$

where $s_1, s_2, \cdots, s_l$ are the *parts* of σ. These parts are positive integers which sum to n if $\sigma \vdash n$; their order is immaterial and repetitions are allowed. The other form is

$$\sigma = < \sigma_1, \sigma_2, \cdots, \sigma_n >$$

where σ_i denotes the *number of parts* which are equal to i. Thus the σ_i's are nonnegative integers, their order does matter, and $n = \sigma_1 + 2\sigma_2 + \cdots + n\sigma_n$ if $\sigma \vdash n$.

The contribution of a partition σ of n to $g(n)$ or $t(n)$ is denoted by $\Delta(\sigma)$. This is given by

$$\Delta(\sigma) = 2^{\Lambda(\sigma)}/\Omega(\sigma)$$

[1]The authors would like to thank the member of the audience (whose identity they do not know) at the conference presentation of this paper who asked about tournaments.

where

$$\Lambda(\sigma) = \sum_{i=1}^{l} \left\{ \left\lfloor \frac{s_i}{2} \right\rfloor + \sum_{j=i+1}^{l} (s_i, s_j) \right\}$$

and

$$\Omega(\sigma) = \prod_{i=1}^{n} \sigma_i!\, i^{\sigma_i}.$$

Here we have used both representations of σ, and (s_i, s_j) denotes the greatest common divisor.

In our analysis in Sections 3 and 4 the number of odd parts of σ will be denoted by k. It is then easy to see that the first summand in the definition of $\Lambda(\sigma)$ contributes $(n-k)/2$, so that

$$\Lambda(\sigma) = (n-k)/2 + \sum_{1 \le i < j \le l} (s_i, s_j).$$

Also, the 2-part will be denoted v_2. We frequently need the 2-part of $\Delta(\sigma)$, which can be written as

$$v_2(\Delta(\sigma)) = \Lambda(\sigma) - \sum_{i=1}^{l} v_2(s_i) - \sum_{i=1}^{n} v_2(\sigma_i!\,)$$

directly from the definition of Δ. In providing lower bounds for this expression it is often helpful to recall that

$$v_2(p!\,) = \lfloor p/2 \rfloor + \lfloor p/4 \rfloor + \lfloor p/8 \rfloor + \cdots,$$

so $v_2(p!\,) \le p-1$ for all $p \ge 1$. For odd $p \ge 3$ we then have $v_2(p!\,) = v_2((p-1)!\,) \le p-2$.

We can now express the formula for $g(n)$ quite succinctly:

$$g(n) = \sum_{\sigma \vdash n} \Delta(\sigma).$$

For $t(n)$ the sum is restricted to partitions into odd parts, which we will call *odd partitions*. Thus

$$t(n) = \sum_{\substack{\sigma \vdash n \\ \sigma\, odd}} \Delta(\sigma).$$

For simplicity in expressing the results we introduce the notation $e(n) = v_2(g(n))$ and $r(n) = v_2(t(n))$.

2 Computational Methods and Results

This work began as an attempt to investigate possible speedups for graphical enumeration algorithms when implementated on parallel architectures, especially hypercube architectures. In order to facilitate use of these algorithms by mathematicians, it was decided to use a language in which operations on large integers (larger than the word size of the machine) look the same as operations on standard integers. C++ [12] was chosen, primarily for that reason. It has the following additional advantages.

- C++ is an *object oriented* language, which means that any mathematical object (not just integers) needed can be implemented and used in a manner quite similar to the way a mathematician normally would. An example, written by us, is the `Partition` class, the C++ equivalent of the set of all partitions of the integer n, where n is a relatively small integer ($n \leq 10^6$, say). This class will be discussed below.

- C++ supports *operator overloading*, that is, the standard operators in the language can be given new meanings. For example, if one wishes to define a class `Rational`, which would represent rational numbers, one can easily redefine the plus symbol to represent infix addition of elements of type `Rational`. Similarly, the input and output operators can be redefined to allow reading and writing `Rational` values in exactly the same manner as real numbers are read and written.

- C++ can be easily and mechanically translated to C [6]. Since our first target machine, the Intel iPSC2, has only C, FORTRAN, and Lisp compilers, this was an important consideration.

- C++ is relatively easy for a non-computer scientist to read, as opposed to Lisp, another language having large integer capabilities.

- C++ is a modern langauge, having strong typing and sophisticated type checking at the time of compilation. This feature makes debugging and testing much easier.

As an example of the object oriented and operator overloading features of C++, consider the code given in Figure 1. This code declares the class `Partition`. Although it is a bit cryptic, it need never be seen by the user. This code states that the class `Partition` is being declared, that the type `Partition` can be used as if it were a type built into C++, and that objects of type `Partition` can be manipulated only by using the functions `GetSize` (to find the number which is being partitioned), `next` (to return the next partition after the current one, in lexicographic order), and `more` (to determine if there is another partition after the current one). In addition, the output operator `<<` is redefined to include output of a partition, and the array indexing operator `[ ]` is redefined to allow looking at (but not changing) the numbers of parts of each value in the current partition. A separate file containing some 40 lines of code is needed in order to complete the implementation of these declarations.

An example of the use of the `Partition` class is given in Figure 2. This complete program prompts the user to enter a partition size, creates a variable named `part` to be a partition of that size, lists all the partitions of that size, and repeats, stopping when a partition size of zero has been given.

Once the programming language had been decided, the next step was to choose the compiler. This was easy, since the GNU[2] C++ compiler [13], g++, was available

[2]GNU is a project of the Free Software Foundation, 675 Massachusetts Ave., Cambridge, MA 02139. GNU software is also available at `prep.ai.mit.edu` *via* anonymous ftp.

```
/* defs for Partition class */

#include <bool.h>

class Partition {
  friend ostream& operator<<(ostream&, Partition&);
 public:
  Partition(int);                             // initialize function
  int GetSize()  { return size; }             // return partition size
  void next();                                // return next partition
  bool more() { return !finished;}            // any more partitions?
  int operator[](int i) {return pp[i];}
 private:
  bool finished;
  int size;
  int *pp;
};
```

Figure 1: Declaration of the class `Partition`.

```
#include <stream.h>
#include "Partition.h"

main() {
  int size;

  for(;;) {
    cout << "Enter partition size: ";
    cin >> size;
    if (size == 0) exit(0);
    Partition part = Partition(size);
    while(part.more()) {
      cout << part;
      part.next();
    }
  }
}
```

Figure 2: A C++ program which uses the `Partition` class.

and cost nothing. In addition, an (almost) arbitrary length Integer class already exists for g++. Although g++ no longer supports translation to C, it was decided that the code would first be written in GNU C++, tested and debugged, and then modified for use with another C++ compiler which does support such translation.

The first step was to write sequential versions of a few of the target algorithms in C++ on a Sun 4 workstation. The first algorithm implementated was the formula for counting numbers of unlabeled graphs, given above. Using the GNU-provided arbitrary length `Integer` class and our `Partition` class, the code proved easy to write. It is given in Figure 3; note the similarity to the formula. This code was tested by enclosing it in a loop to generate counts for one through twenty.

It was anticipated that the results of this computation would provide an easy check on the soundness of the code. However, after hours of debugging, we determined that our code was probably correct, but the `Integer` class definitely had a bug in it. We submitted a bug report to Doug Lea[3], who wrote and who maintains the g++ `Integer` class [7]. Less than twenty-four hours later, he informed us that we had indeed found a new bug in his code, and supplied us with a corrected version. This level of support for a free product is nothing short of amazing!

Our code proved to be correct once the patched `Integer` class was added, and we found that we could calculate the number of unlabeled graphs of size n, for n equal one to twenty, very rapidly. As an experiment, we decided to see how far sequential calculation could take us. The results are given in Tables 1 and 2, to 21 significant digits[4]. We quit computing when the time to calculate each new value became too long; at $n = 89$, the calculation took approximately two days. Examination of this table led to the observation that the 2-part seemed to grow linearly in n. These 2-parts were then computed explicitly; the results are shown in column two of Table 3. At this point the authors' attention focused on the work described in Section 3. After the question of tournaments was raised, the programs used to generate the above data were modified to count tournaments and to calculate the 2-part of tournament numbers. The numbers of unlabeled tournaments are given in Tables 4 and 5; the 2-parts of these numbers are given in the third column of Table 3.

3 Unlabeled graph numbers

Theorem 1 *For odd $n \geq 5$, $e(n) = (n+1)/2 - \lfloor \log_2 n \rfloor$.*

Proof. It suffices to show that for odd $n \geq 5$ and $\sigma \vdash n$,

$$v_2(\Delta(\sigma)) \geq (n+1)/2 - \lfloor \log_2 n \rfloor$$

with equality if and only if $\sigma = [2^u, n - 2^u]$, where $u = \lfloor \log_2 n \rfloor$. The equality is quickly verified in the latter case, so take some other partition σ of n.

[3]Doug Lea's Internet address is `dl@g.oswego.edu`.

[4]The full integer values for all of the results are available from the authors at `cater@pollux.cs.uga.edu`, or `rwr` at the same machine.

```
/* Test of Partitions class and lambda def.  Calculates number of
   unlabeled graphs of size n. */

#include <stream.h>
#include <Integer.h>
#include <std.h>
#include "Partition.h"

int lambda(Partition);
Integer factorial(int);

main() {
  Integer nfact, sum, prod, temp;
  int n;

  cin >> n;
  sum = 0;
  cout << "Counting unlabeled graphs with " << n << " nodes.\n";
  nfact = factorial(n);
  for( Partition sigma(n); sigma.more(); sigma.next()) {
    prod = 1;
    for(int i = 1; i <= n; i++) {
      prod *= factorial(sigma[i]) * Ipow(i,sigma[i]);
    }
    sum += Ipow( 2, lambda(sigma)) * (nfact / prod);
  }
  temp = sum / nfact;
  cout << "Number of graphs = " << temp << "\n"
       << "Check = " << (temp * nfact == sum) << "\n\n" ;
}

inline Integer factorial(int n) {
  Integer prod = 1;
  for(int i = 1; i <= n; i++) {
    prod *= i;
  }
  return prod;
}
```

Figure 3: C++ code to count unlabeled graphs.

n	$g(n)$	n	$g(n)$
1	1. E 0	26	1.69487618400693135671 E 71
2	2. E 0	27	4.21260006519643885757 E 77
3	4. E 0	28	2.01929599967857139573 E 84
4	1.1 E 1	29	1.86913527224789560417 E 91
5	3.4 E 1	30	3.34494316309257669249 E 98
6	1.56 E 2	31	1.15858150442058126734 E 106
7	1.044 E 3	32	7.77510571865055903406 E 113
8	1.2346 E 4	33	1.01193386573463039354 E 122
9	2.74668 E 5	34	2.55660138713977238511 E 130
10	1.2005168 E 7	35	1.25491641977244993328 E 139
11	1.018997864 E 9	36	1.19773884587727998740 E 148
12	1.65091172592 E 11	37	2.22454014411225378689 E 157
13	5.0502031367952 E 13	38	8.04574912012549013374 E 166
14	2.9054155657235488 E 16	39	5.67076581342507135706 E 176
15	3.1426485969804308768 E 19	40	7.79384116791497795458 E 186
16	6.40010157045275578949 E 22	41	2.09010218966510082133 E 197
17	2.45935864153532932684 E 26	42	1.09432936149936537059 E 208
18	1.78757772514561170055 E 30	43	1.11928172756940473343 E 219
19	2.46378092531250045244 E 34	44	2.23756958905295273248 E 230
20	6.45490122795799841856 E 38	45	8.74749788753564969915 E 241
21	3.22202728998089834335 E 43	46	6.69076566314768312199 E 253
22	3.07084648309414430064 E 48	47	1.00174633465851788106 E 266
23	5.59946939699792080598 E 53	48	2.93715131472758447010 E 278
24	1.95704906302078447922 E 59	49	1.68721346510668416967 E 291
25	1.31331393569895519432 E 65	50	1.89963348317963058713 E 304

Table 1: Number of unlabeled graphs of order n, $1 \leq n \leq 50$.

n	$g(n)$	n	$g(n)$
51	4.19371992498551232276 E 317	71	1.34857728595610051484 E 646
52	1.81604187417696589248 E 331	72	4.42255289913777928034 E 665
53	1.54315575619395726431 E 345	73	2.86094733967208596061 E 685
54	2.57398358835840638107 E 359	74	3.65147076381690182830 E 705
55	8.43068474864503196796 E 373	75	9.19657767905459181171 E 725
56	5.42406124178183939045 E 388	76	4.57153791459731763943 E 746
57	6.85692636837120997184 E 403	77	4.48591738661032050132 E 767
58	1.70377109172429517616 E 419	78	8.69093165597980295968 E 788
59	8.32336580752713111241 E 434	79	3.32490242893041445931 E 810
60	7.99682285850609074380 E 450	80	2.51222524627085789379 E 832
61	1.51142771182020356617 E 467	81	3.74949872210439568501 E 854
62	5.62115326327816237337 E 483	82	1.10557702823513689320 E 877
63	4.11476093836466136684 E 500	83	6.44125597593382597410 E 899
64	5.92999546519608121461 E 517	84	7.41619110481278923819 E 922
65	1.68290936468815851758 E 535	85	1.68764704184555736276 E 946
66	9.40733283322775493811 E 552	86	7.59159100756593419464 E 969
67	1.03602782800452157790 E 571	87	6.75138970556373002920 E 993
68	2.24839296428712782337 E 589	88	1.18718972121256760220 E 1018
69	9.61751410767161184605 E 607	89	4.12828564639630784761 E 1042
70	8.11025469117756850102 E 626	90	2.83920560848950935498 E 1067
		91	3.86238055221184090004 E 1092

Table 2: Number of unlabeled graphs of order n, $51 \leq n \leq 91$.

n	$e(n)$	$r(n)$		n	$e(n)$	$r(n)$		n	$e(n)$	$r(n)$
1	0	0		34	16	20		67	28	33
2	1	0		35	13	17		68	32	38
3	2	1		36	16	19		69	29	34
4	0	2		37	14	18		70	30	37
5	1	2		38	15	19		71	30	35
6	2	3		39	15	19		72	35	38
7	2	3		40	17	26		73	31	36
8	1	5		41	16	20		74	32	38
9	2	4		42	17	22		75	32	37
10	4	6		43	17	21		76	34	39
11	3	5		44	19	23		77	33	38
12	4	7		45	18	22		78	34	41
13	4	6		46	19	23		79	34	39
14	5	7		47	19	23		80	36	45
15	5	7		48	20	30		81	35	40
16	4	10		49	20	24		82	36	43
17	5	8		50	21	26		83	36	41
18	8	9		51	21	25		84	38	44
19	6	9		52	23	28		85	37	42
20	8	12		53	22	26		86	38	43
21	7	10		54	23	27		87	38	43
22	8	11		55	23	27		88	41	46
23	8	11		56	25	30		89	39	44
24	9	14		57	24	28		90	40	47
25	9	12		58	25	30		91	40	45
26	10	14		59	25	29		92	—	47
27	10	13		60	27	33		93	—	46
28	15	15		61	26	30		94	—	47
29	11	14		62	27	31		95	—	47
30	12	17		63	27	31		96	—	51
31	12	15		64	26	35		97	—	48
32	11	19		65	27	32		98	—	49
33	12	16		66	31	34		99	—	49

Table 3: The 2-part of unlabeled graph and tournament numbers.

n	$t(n)$	n	$t(n)$
1	1. E 0	26	1.69484335125246268101 E 71
2	1. E 0	27	4.21255599848131447082 E 77
3	2. E 0	28	2.01928462566720826509 E 84
4	4. E 0	29	1.86912961822137124078 E 91
5	1.2 E 1	30	3.34493774260141796029 E 98
6	5.6 E 1	31	1.15858050093774074838 E 106
7	4.56 E 2	32	7.77510212704809602577 E 113
8	6.880 E 3	33	1.01193361693125457991 E 122
9	1.91536 E 5	34	2.55660105320160166874 E 130
10	9.733056 E 6	35	1.25491633284800343840 E 139
11	9.03753248 E 8	36	1.19773880195520546921 E 148
12	1.54108311168 E 11	37	2.22454010099365542315 E 157
13	4.8542114686912 E 13	38	8.04574903781750488782 E 166
14	2.8401423719122304 E 16	39	5.67076578285122471705 E 176
15	3.1021002160355166848 E 19	40	7.79384114579899128834 E 186
16	6.35304158423082651003 E 22	41	2.09010218654756435461 E 197
17	2.44912778438520759443 E 26	42	1.09432936064242347289 E 208
18	1.78339884628477797542 E 30	43	1.11928172710978640162 E 219
19	2.46056411712603767706 E 34	44	2.23756958857166158964 E 230
20	6.45022068557873570932 E 38	45	8.74749788655111907998 E 241
21	3.22073640316611753845 E 43	46	6.69076566275404571705 E 253
22	3.07016988315046833619 E 48	47	1.00174633462774035901 E 266
23	5.59879382429394075398 E 53	48	2.93715131468050239237 E 278
24	1.95692027657521876084 E 59	49	1.68721346509258584336 E 291
25	1.31326696677895002131 E 65	50	1.89963348317136324115 E 304

Table 4: Number of unlabeled tournaments of order n, $1 \leq n \leq 50$.

n	$t(n)$	n	$t(n)$
51	4.19371992497601415534 E 317	76	4.57153791459731763874 E 746
52	1.81604187417482709531 E 331	77	4.48591738661032050097 E 767
53	1.54315575619301292254 E 345	78	8.69093165597980295934 E 788
54	2.57398358835758850852 E 359	79	3.32490242893041445925 E 810
55	8.43068474864364201683 E 373	80	2.51222524627085789377 E 832
56	5.42406124178137570279 E 388	81	3.74949872210439568499 E 854
57	6.85692636837090622441 E 403	82	1.10557702823513689320 E 877
58	1.70377109172425609168 E 419	83	6.44125597593382597409 E 899
59	8.32336580752703229368 E 434	84	7.41619110481278923818 E 922
60	7.99682285850604163594 E 450	85	1.68764704184555736276 E 946
61	1.51142771182019876808 E 467	86	7.59159100756593419464 E 969
62	5.62115326327815315366 E 483	87	6.75138970556373002920 E 993
63	4.11476093836465788172 E 500	88	1.18718972121256760220 E 1018
64	5.92999546519607862231 E 517	89	4.12828564639630784761 E 1042
65	1.68290936468815813806 E 535	90	2.83920560848950935498 E 1067
66	9.40733283322775384422 E 552	91	3.86238055221184090004 E 1092
67	1.03602782800452151582 E 571	92	1.03943381142178588779 E 1118
68	2.24839296428712775396 E 589	93	5.53443756277865881384 E 1143
69	9.61751410767161169316 E 607	94	5.83089519479889119443 E 1169
70	8.11025469117756843466 E 626	95	1.21571345288728832606 E 1196
71	1.34857728595610050916 E 646	96	5.01660119875685004117 E 1222
72	4.42255289913777927077 E 665	97	4.09748551592124748579 E 1249
73	2.86094733967208595743 E 685	98	6.62522955827046632021 E 1276
74	3.65147076381690182621 E 705	99	2.12082732984630398231 E 1304
75	9.19657767905459180900 E 725		

Table 5: Number of unlabeled tournaments of order n, $51 \leq n \leq 99$.

Case 1. Every part of σ is odd.

We claim that $v_2(\Delta(\sigma)) \geq (n-1)/2$. Since $n \geq 5$, $\lfloor \log_2 n \rfloor \geq 2$, so

$$v_2(\Delta(\sigma)) \; > \; (n+1)/2 - \lfloor \log_2 n \rfloor$$

will follow. To prove the claim, suppose σ has k parts. Since n is odd, k is odd. Then

$$\Lambda(\sigma) \; \geq \; \frac{n-k}{2} + \binom{k}{2}$$

since $\sum_{i=1}^{k} \lfloor s_i/2 \rfloor \; = \; (n-k)/2$ when each s_i is odd, $\sum_{i=1}^{k} \; s_i \; = \; n$, and $(s_i, s_j) \geq 1$ for all i, j. Moreover, if $k \geq 3$ then $v_2(\Omega(\sigma)) \leq k-2$. That is because $v_2(\Omega(\sigma)) = \sum_{i=1}^{n} v_2(\sigma_i!\,)$ for odd σ. We have $\sum_{i=1}^{n} \sigma_i \; = \; k$, and $v_2(\sigma_i!\,) \leq \sigma_i - 1$ whenever $\sigma_i \geq 1$. If there are two different part values in σ, this gives $v_2(\Omega(\sigma)) \leq k-2$ at once. If there is only one part value, then

$$v_2(\Omega(\sigma)) \; = \; v_2(k!\,) \; \leq \; k-2,$$

the latter since k is odd . Thus, if $k \geq 3$ we have

$$\begin{aligned} v_2(\Delta(\sigma)) \;\; & \geq \;\; \frac{n-k}{2} + \binom{k}{2} - (k-2) \\ & = \;\; \frac{n}{2} + \frac{1}{2}(k-2)^2 \\ & > \;\; (n-1)/2. \end{aligned}$$

If $k = 1$ there is only one part, of value n, so $\sigma = [n]$. Clearly $v_2(\Delta([n])) = (n-1)/2$ since n is odd. So this is the minimum of $v_2(\Delta(\sigma))$ when all parts are odd, and this minimum is attained only when $\sigma = [n]$.

Case 2. There is exactly one odd part in σ, and at least one even part.

Let m denote the number of even parts in σ, and let $M_1, \ldots, M_m$ denote their 2-parts. We may assume that they are listed in non-decreasing order of 2-parts, so that

$$1 \leq M_1 \leq \cdots \leq M_m.$$

The greatest common divisor of the ith and jth parts when $i < j$ is then a multiple of 2^{M_i}, and thus at least 2^{M_i}. Summing over all pairs of parts gives a contribution of at least $\sum_{i=1}^{m} \; (m-i)2^{M_i}$. From the denominator, the powers give a 2-part of exactly $-(M_1 + \cdots + M_n)$, while the 2-part of the factorials must contribute at least $-(m-1)$. Adding in m as a lower bound for the sum of the greatest common divisors of the m even parts with the one odd part, and noting that the single odd part makes no contribution whatever to the 2-part of the denominator, we have

$$\begin{aligned} v_2(\Delta) \;\; & \geq \;\; \frac{n-1}{2} + m - (m-1) - \sum_{j=1}^{m} \; M_j \;\; + \; \sum_{j=1}^{m} \; (m-j)2^{M_j} \\ & = \;\; \frac{n+1}{2} + \sum_{i=0}^{m-1} \; (i2^{M_{m-i}} - M_{m-i}) \\ & \geq \;\; \frac{n+1}{2} - M_m. \end{aligned}$$

This is because $2^x > x$ for all x, so the lower bound can only be attained if $m = 1$. Obviously $u = \lfloor \log_2 n \rfloor$ is an upper bound for M_1, and only a part of size 2^u gives this value among the positive even numbers less than n. Thus for σ in Case 2 we have

$$v_2(\Delta(\sigma)) \geq \frac{n+1}{2} - \lfloor \log_2 n \rfloor,$$

with equality only if $\sigma = [2^u, n - 2^u]$.

Case 3. There are $k \geq 3$ odd parts in σ, and at least one even part.

We claim that the lower bound of Case 2 must be exceeded in this case. The greatest common divisors of the km pairs of parts where one is even and the other is odd are all greater than or equal to 1, so their sum is at least km. Combining this with the analyses given for Cases 1 and 2 yields

$$\begin{aligned}
v_2(\Delta(\sigma)) &\geq \frac{n-k}{2} + \binom{k}{2} - (k-2) + mk - (m-1) + \sum_{i=0}^{m-1} (i2^{M_{m-i}} - M_{m-i}) \\
&= \frac{n-(k-2)^2}{2} + 1 + m(k-1) + \sum_{i=0}^{m-1} (i2^{M_{m-i}} - M_{m-i}) \\
&\geq \frac{n+1}{2} + m(k-1) - M_m \\
&> \frac{n+1}{2} - \lfloor \log_2 n \rfloor,
\end{aligned}$$

as required. □

Theorem 2 *For even $n \geq 4, e(n) \geq n/2 - \lfloor \log_2 n \rfloor$ with equality if and only if n is a power of 2.*

Proof. As in the proof of Theorem 1, let k be the number of odd parts in σ and m the number of even parts. Then consider the three cases $m = 0, k = 0$, and $m \geq 1$ and $k \geq 2$.

Case 1. $m = 0$.

Since n is even, k must be even and $k \geq 2$. Our analysis of Case 1 of Theorem 1 applies except that $-(k-1)$ is our best general lower bound for the contribution of the 2-part of the factorials in the denominator, instead of $-(k-2)$, since k is even rather than odd. Thus

$$\begin{aligned}
v_2(\Delta(\sigma)) &\geq \frac{n-k}{2} + \binom{k}{2} - (k-1) \\
&= \frac{n}{2} + \frac{(k-2)^2}{2} - 1 \\
&\geq \frac{n}{2} - 1 \\
&> \frac{n}{2} - \lfloor \log_2 n \rfloor, \quad \text{since } n \geq 4.
\end{aligned}$$

Hence, the lower bound of the theorem cannot be attained in Case 1.

Case 2. $k = 0$.

We have $m \geq 1$, and can follow the analysis of Case 2 of Theorem 1 with the straightforward adjustments caused by having $k = 0$ instead of $k = 1$. This gives us

$$\begin{aligned} v_2(\Delta(\sigma)) &\geq \frac{n}{2} - (m-1) + \sum_{i=0}^{m-1} (i2^{M_{m-i}} - M_{m-i}) \\ &= \frac{n}{2} - M_m + (2^{M_{m-1}} - M_{m-1} - 1) + (2 \cdot 2^{M_{m-2}} - M_{m-2} - 1) + \cdots \\ &\geq \frac{n}{2} - \lfloor \log_2 n \rfloor. \end{aligned}$$

To attain equality we must have $M_m = \lfloor \log_2 n \rfloor$. We would also need to have contributions of zero from any applicable terms grouped in parentheses. In particular, $m \leq 2$ because if $m \geq 3$ we have $M_{m-2} \geq 1$ so $2^{M_{m-2}} > 1$ and $2^{M_{m-2}} > M_{m-2}$, hence $2 \cdot 2^{M_{m-2}} - M_{m-2} - 1 > 0$. If $m = 2, M_2 = \lfloor \log_2 n \rfloor$ from above, and so $M_1 < M_2$ since $2^{M_1} + 2^{M_2} \leq n < 2^{1+M_2}$. Then the two parts of σ must be different, and the factorials in the denominator are both $1!$, thus contributing zero to the 2-part. Our general lower bound used $-(m-1) = -1$ for this contribution, so we can add 1 in this situation when $m = 2$, which means again that the lower bound is not attained.

So we can only attain the lower bound when $m = 1$ and $M_1 = \lfloor \log_2 n \rfloor$. Since $k = 0$, this means that $\sigma = [n]$, as there is only one part in σ. In this case $v_2(\Delta([n])) = n/2 - v_2(n)$, and $M_1 = v_2(n)$. We have equality with the lower bound if, and only if, $v_2(n) = \lfloor \log_2 n \rfloor$, i.e., if, and only if, n is a power of 2.

Case 3. $m \geq 1$ *and* $k \geq 2$.

This follows Case 3 of Theorem 1 except for the replacement of $-(k-2)$ by $-(k-1)$ as noted in Case 1 of the present theorem. Thus we find

$$\begin{aligned} v_2(\Delta(\sigma)) &\geq \frac{n-k}{2} + \binom{k}{2} - (k-1) + km - (m-1) + \sum_{i=1}^{m} (i2^{M_{m-i}} - M_{m-i}) \\ &= \frac{n}{2} + \frac{(k-2)^2}{2} + (k-1)m - M_m + (2^{M_{m-1}} - M_{m-1}) + \cdots \\ &> \frac{n}{2} - \lfloor \log_2 n \rfloor. \end{aligned}$$

Hence, the lower bound cannot be attained in Case 3. $\square$

4 Unlabeled tournament numbers

Theorem 3 *For all odd n, $r(n) = (n-1)/2$. For all even $n \geq 4$, $r(n) \geq n/2$ with equality if, and only if, $\varphi(n)/2$ is odd.*

Proof. For odd n the proof of Theorem 1, Case 1, showed that

$$v_2(\Delta(\sigma)) \geq (n-1)/2$$

with equality if, and only if, $\sigma = [n]$. Thus, $r(n) = (n-1)/2$.

For even $n \geq 4$ we have an even number $k \geq 2$ of parts, since these parts are all odd. From the proof of Theorem 2, Case 1, we have

$$v_2(\Delta(\sigma)) \ \geq \ \frac{n}{2} + \frac{(k-2)^2}{2} - 1,$$

which is greater than $n/2$ if $k \geq 4$.

The case $k = 2$ must be considered in detail. Suppose first that $n/2$ is odd and both parts are equal to $n/2$. Then

$$v_2(\Delta([n/2, n/2])) \ = \ \frac{n-2}{2} + \frac{n}{2} - 1$$

since $(n/2, n/2) = n/2$ and there is a factor of 2! in the denominator. We have $n \geq 4$ and $n/2$ odd, so $n \geq 6$ and this lower bound is greater than $n/2$. Now suppose that the two odd parts are different, say a and $n - a$ with $1 \leq a < n/2$. Thus

$$\begin{aligned} v_2(\Delta([a, n-a])) \ &= \ \frac{n-2}{2} + (a, n-a) \\ &= \ \frac{n}{2} + (a, n) - 1 \\ &\geq \ \frac{n}{2}. \end{aligned}$$

Notice that the lower bound of $n/2$ is attained precisely for $\sigma = [a, n-a]$ where $a < n/2$ and $(a, n) = 1$. There are exactly $\varphi(n)/2$ such values of a, where φ is the Euler totient function. Thus $r(n) = n/2$ if, and only if, $\varphi(n)/2$ is odd. For even $n \geq 4, \varphi(n)/2$ is odd just if $n = 4$, or else $n = 2p^u$ for $u \geq 1$ and p a prime congruent to 3 (mod 4). Since there are infinitely many such primes, we have $r(n) = n/2$ infinitely often. $\square$

For even $n \geq 4$, let $S(n)$ be the contribution of the terms of minimum 2-part to $t(n)$, i.e.,

$$S(n) = \sum_{\substack{1 \leq a < n/2 \\ (a,n) = 1}} \frac{2^{n/2}}{a \cdot (n-a)}.$$

Experimentation with $S(n)$ suggested the unexpected identity

$$v_2(S(n)) = n/2 + v_2(\varphi(n)/2).$$

In other words, to compute the 2-part one could replace $1/a(n-a)$ by 1 in every term. This has been verified by Andrew Granville [3]. The identity shows that if $r(n) - n/2$ is bounded, it must be due to the effect of other terms. It is difficult to frame a conjecture about this with any degree of confidence. The data for $4 \leq n \leq 99$ reveal that

$$r(n) = n/2 + v_2(\varphi(n)/2)$$

for all the even values of n except for 40, 48, 64, 80 and 96. For these inputs the difference $r(n) - n/2 - v_2(\varphi(n)/2)$ is 3, 3, -1, 1 and -1, respectively.

5 Is There a Combinatorial Explanation?

The authors have been unable to discover any combinatorial explanation for Theorems 1, 2, and 3. To illustrate the difficulties, consider the fact that $t(5) = 12$. It follows from Theorem 3 that $v_2(t(5)) = 2$, so a combinatorial explanation for the divisibility of $t(5)$ by 4 is called for. This would presumably lead to a natural grouping, based on structural criteria, of the 12 unlabeled tournaments into 3 equivalence classes, each with 4 tournaments.

Diagrams of these tournaments are given in the Appendix of Moon's book [8], along with the abstract automorphism group and the score sequence of each. Seven have the identity automorphism group, four have C_3, and one has C_5. Two have $(1, 1, 2, 3, 3)$ as the score sequence, three have $(1, 2, 2, 2, 3)$, and the other seven score sequences correspond to unique tournaments up to isomorphism. Thus no grouping into sets of 4 can preserve the automorphism group or the score sequence.

The only structural property known to the authors which could be preserved by such a grouping is self-complementarity. This suggests an approach to the seemingly weaker problem of finding a natural grouping into pairs. That is, for a tournament or graph which is not self-complementary, pair it with its complement. For $n \equiv 2$ or 3 (mod 4) this gives a complete pairing of unlabeled graphs, since there are no self-complementary graphs on n nodes in those cases. However for $n \equiv 0$ or 1 (mod 4) there are self-complementary graphs but no known natural groupings into pairs. Self-complementary tournaments of any order exist, but again there is no known natural grouping of them into pairs.

References

[1] R. L. Davis, The number of structures of finite relations, *Proc. Amer. Math. Soc.* 4 (1953) 486–495.

[2] R. L. Davis, Structures of dominance relations, *Bull. Math. Biophys.* 16 (1954) 131–140.

[3] A. J. Granville, private communication, 1991.

[4] F. Harary, The number of linear, directed, rooted, and connected graphs, *Trans. Amer. Math. Soc.* 78 (1955) 445–463.

[5] F. Harary and E. M. Palmer, *Graphical Enumeration*, Academic, New York, 1973.

[6] B. R. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[7] D. Lea, *User's Guide to GNU C++ Library, Version 1.39*, Free Software Foundation, Cambridge, MA, 1991.

[8] J. W. Moon, *Topics on Tournaments*, Holt, New York, 1968.

[9] G. Pólya, Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen, *Acta Math* 68 (1937) 145–254.

[10] G. Pólya and R. C. Read, *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*, Springer, New York, 1987.

[11] J. H. Redfield, The theory of group-reduced distributions, *Amer. J. Math.* 49 (1927) 433–455.

[12] B. Stroustroup, *The C++ Programming Language*, Addison-Wesley, Reading, MA, 1986.

[13] D. Tiemann, *User's Guide to GNU C++, Version 1.39*, Free Software Foundation, Cambridge, MA, 1991.