RAIRO. RECHERCHE OPÉRATIONNELLE

A. GUÉNOCHE

Counting and selecting at random bipartite graphs with fixed degrees

RAIRO. Recherche opérationnelle, tome 24, n° 1 (1990), p. 1-14

<http://www.numdam.org/item?id=RO_1990__24_1_1_0>

Recherche opérationnelle/Operations Research
(vol. 24, n° 1, 1990, p. 1 à 14)

COUNTING AND SELECTING AT RANDOM BIPARTITE GRAPHS WITH FIXED DEGREES (*)

by A. GUÉNOCHE (1)

Abstract. – *We first recall the bijective mapping between bipartite simple graphs with specified degrees and* 0/1 *arrays with fixed sums of rows and columns, and we count these graphs, using an enumerative procedure. Tables that contain the numbers of such graphs with at most* 10 *edges are given. Then we put a total order on this set, and provide an algorithm to build the graph that have a given rank. If this rank is selected at random, uniformely or not, we get a random bipartite simple graph.*

Keywords : Combinatorics; enumeration; random graphs.

Résumé. – *Après avoir rappelé la correspondance bijective entre les graphes simples bipartis de degrés fixés et les tableaux en* 0/1 *de totaux marginaux fixés, nous dénombrons ces graphes, à l'aide d'une procédure d'énumération. Ensuite on munit cet ensemble d'un ordre total afin de présenter un algorithme qui construit le graphe de rang donné pour cet ordre. Si ce rang est tiré au hasard (uniformément), on obtient un graphe aléatoire (équiprobable). Nous donnons en annexe les tables de dénombrement des graphes bipartis de degrés fixés jusqu'à* 10 *arêtes, pour toute séquence de degrés.*

Mots clés : Algorithmes Combinatoires; énumération; graphes aléatoires.

INTRODUCTION

Frequently computer programs that permit graph editing are used for theorical research and/or teaching graph theory. These programs are more efficient if they present random generation functions for graphs with given properties. After many authors, we propose an algorithm of this family to build bipartite graphs with a specified degree sequence. To build a simple graph with given degrees the "natural" algorithm, that selects at random edges until saturation of the degrees, may cancel if one unsaturated vertex is left, or if it there remains some vertices already linked with preceding edges.

(*) Received November 1987.
(1) GRTC-CNRS, 31, Ch.-J. Aiguier, 13402 Marseille Cedex 9.

Recherche opérationnelle/Operations Research, 0399-0559/90/01 1 14/$ 3.40

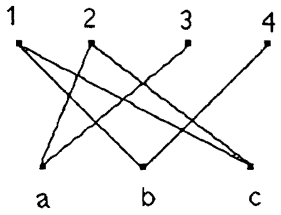
If it fails, the procedure is restarted until success. We will call this algorithm "with rejection".

N. C. Wormald [4] proposes such an algorithm to build a bipartite simple graph with $n+m$ vertices having specified degrees. The two degree sequences are partitions $A=(a_1, \ldots, a_n)$ and $B=(b_1, \ldots, b_m)$ of the number of edges $N=\sum_{i=1,\ldots,n} a_i = \sum_{i=1,\ldots,m} b_i$. He notices that a 0/1 array with n rows and m columns, that presents A as the sums of the rows and B as the sums of the columns, is a code for these graphs. So he proposes to select at random a composition of N with type $(a_1, \ldots, a_n)$, that is a word $(x_1, \ldots, x_N)$ of length N over the alphabet $[n]=\{1, 2, \ldots, n\}$, such that the letter i occurs a_i times. Therefore, the b_1 first letters of this word are the row indices of the first column, the b_2 following letters the index of the second column and so on. This procedure may fail if the same letter i appears many times in one of the b_j sequences of consecutive letters.

Exemple 1

$A=(2\,2\,1\,1)$, $B=(2\,2\,2)$, $N=6$. If the selected random word is $(2,3,1,4,1,2)$ we will obtain the following array and graph:

	a	b	c
1 :	0	1	1
2 :	1	0	1
3 :	1	0	0
4 :	0	1	0



But if the random word is $(1,1,2,3,2,4)$ we do not get a simple graph since the same vertex in the first part is linked twice to the first vertex of the second part.

N. C. Wormald proves that his iterative procedure succeeds and he gives an upper bound for the number of trials.

In this text we propose an algorithm that selects at first shot, without rejection, a 0/1 array with specified sums of rows and columns, that is a bipartite simple graph with fixed degrees. It is based on the number of such arrays that we study in the first part. We give a recurrent function and a procedure to count them. In the second part we put a total order on this set and we build the array that have rank R for this order. If R is uniformely selected, we obtain a random graph with a uniform probability.

1. NUMBER OF 0/1 ARRAYS WITH FIXED SUMS OF ROWS AND COLUMNS

Let $T(a_1, \ldots, a_{Nr}/b_1, \ldots, b_{Nc})$, also noted $T(A/B)$, be the number of 0/1 arrays with Nr rows and Nc columns such that sums of rows are $a_1 \leqq a_2 \leqq \ldots a_{Nr}$, and sums of columns are $b_1 \leqq b_2 \leqq \ldots b_{Nc}$. The orders of the rows and the colums are arbitrary. We choose the one that corresponds to the classical notation of partitions of N in lexicographical order.

Since T (A/B) is a binary array (resp. a simple graph) we must have $a_{Nr} \leqq Nc$ and also $b_{Nc} \leqq Nr$ (resp. the maximum degree of a vertex in a part cannot exceed the cardinality of the other part). So any couple of partitions of N (A and B) cannot lead to a 0/1 array. In this case we put $T(A/B) = 0$.

LEMMA

$$T(a_1, \ldots, a_{Nr}/b_1, \ldots, b_{Nc-1}, Nr) = T(a_1 - 1, \ldots, a_{Nr} - 1/b_1, \ldots, b_{Nc-1}).$$

If $b_{Nc} = Nr$ the last column contains only 1's. We do not change the number of arrays taking off this column and subtracting 1 to all the parts of A.

More generally, we are going to count these arrays using a recursive formula based on the number of ways anabling to remove as much as the number of ones in the last column, substracting one unity to some parts of A.

THEOREM

$$T(A/B) = \sum_{\mathscr{C}(b_{Nc}, Nr)} T(A'/B')$$

with $\mathscr{C}(K, L) = \{$*subsets with* K *elements in a set with* L *elements*$\}$, $B' = (b_1, \ldots, b_{Nc-1})$ *and* A' *is a partition of* $N - b_{Nc}$ *obtained from* A, *removing one unity to the parts selected in an element of* $\mathscr{C}(b_{Nc}, Nr)$.

In this formula, the last column is considered as the characteristic vector of a set with b_{Nc} elements in the set of the Nr rows. The preceding lemma is the situation where $\mathscr{C}(K, L)$ is reduced to a single element since $K = L$.

1.1. Algorithm to compute $T(A/B)$

The principle of the algorithm is to remove the greatest part $BM = b_{Nc}$ from B – we obtain a partition $B' = (b_1, \ldots, b_{Nc-1})$ – and to subtract one unity to BM parts of A. To each way to choose these BM parts among the Nr parts of A, corresponds a partition A' of N-BM.

We can enumerate characteristic vectors $C=(c(i), i=1, \ldots, Nr)$ that indicate the BM elements in a set with Nr elements, and each partition A' is equal to A-C. After that we have to reorder the parts different from 0.

Exemple 2: $A=(1\,1\,1\,2)$, $B=(1\,2\,2)$

$C=1100$,	$A'=(1\,2)$,	$B'=(1\,2)$,	$T(A'/B')=1$
$C=1010$,	$A'=(1\,2)$		$=1$
$C=1001$,	$A'=(1\,1\,1)$		$=3$
$C=0110$,	$A'=(1\,2)$		$=1$
$C=0101$,	$A'=(1\,1\,1)$		$=3$
$C=0011$,	$A'=(1\,1\,1)$		$=3$
			$T(A/B)=12$

This method provides a great lot of identical vectors A', and the procedure is too long. We get the partition (1 2) when we reduce two parts equal to 1. In the same way we get the partition (1 1 1) when we reduce one part equal to 1 and the part equal to 2. Both happen 3 times. To obtain only once each partition A', we consider the notation $A=[d_1, d_2, \ldots, d_p]$ in which $p=a_{Nr}$ and d_k is the number of parts equal to k. It is sufficient now to enumerate the compositions $C=(c_1, c_2, \ldots, c_p)$ of BM into p parts bounded by $(d_1, d_2, \ldots, d_p)$ and to weight $T(A'/B')$ with $q_c=\prod_{i=1,\ldots,p} (d_i \setminus c_i)$ where $(d_i \setminus c_i)$ is the binomial coefficient.

Taking back the previous exemple : $A=[3\,1]$, $B=(1\,2\,2)$,

$C=(2\,0)$,	$A'=[1\,1]$,	$q_c=(3 \setminus 2)$,	$B'=(1\,2)$
$C=(1\,1)$,	$A'=[]$,	$q_c=(3 \setminus 1)$,	$B'=(1\,2)$

Enumeration of compositions of M in P parts bounded with $D(1), \ldots, D(P)$

The enumeration algorithm for compositions of M in P parts given by Nijenhuis et Wilf [3], based on their bijective mapping with subsets with M elements in a set with $M+P-1$ elements, cannot be easily extended to the bounded parts. So we elaborate the following algorithm that gives, while Flag = 0, the next composition. Initially Flag := 1 and the first composition is computed.

```
IF Flag=0 GOTO a/ ELSE E=M
FOR I:=1 TO P DO
  IF E>D(I) THEN C(I):=D(I) ELSE C(I):=E
  E:=E-C(I)
IF E>0 THEN Flag:=1
Flag:=0; END
```

a/ We seek to the last part that can be reduced, and we give the greatest possible values to the following parts.

```
EE:=C(P)
FOR I:=P-1 TO 1 AND C(I)>0 DO
  EE:=EE+1; C(I):=C(I)-1 : E:=EE
  FOR J:=I+1 TO P DO
    IF E>D(J) THEN C(J):=D(J) ELSE C(J):=E
    E:=E-C(J)
  IF E=0 THEN END ELSE EE:=EE+C(I)
Flag:=1; END
```

1.2. Tabulation

To realise efficiently the algorithm that builds the given rank array, we must have tables that give for any couple of partitions of $K \leqq N$, the number of arrays that present these partitions as the sums of rows and columns. If we store these partitions themselves, we will occupy an excessive amount of memory. So we define a characteristic function for the partitions of N. Starting with a rank R, this function computes a partition A according to lexicographical order.

Assuming C. Berge's [1] notations, let

$\mathscr{P}_n = \{\text{partitions of } n\}$,

$\mathscr{P}_{n,h} = \{\text{partitions of } n \text{ such that the smallest part is equal to } h\}$,

P_n and $P_{n,h}$ their numbers of elements. We have the obvious relation:

$$P_{n,h} = P_{n-1,h-1} - P_{n-h,h-1} \quad (h>1)$$

that permits to compute the values stored into an array P with two indices n and h, starting with $P_{n,1} = P_{n-1}$, $P_{n,n} = 1$ and $\Sigma_h P_{n,h} = P_n$. The rank R of a partition $(A(I), I=1, NP)$ of N in NP parts is given by the following procedure.

```
R:=1; NN:=N; K:=1
FOR I:=1 TO NP DO
  P:=A(I); IF P>K DO
    FOR J:=K TO P-1 DO
      R:=R+P(NN, J)
    K:=P; NN:=NN-P
```

Reciprocally, to the rank R corresponds a partition of N, $(A(I), I=1, NP)$ that we obtain counting the number of partitions that have a rank smaller than R. Thus we add the number of partitions of which the smallest part is $1, 2, \ldots, I+1$, until we reach or overpass R. Its first part is equal to I and we seek now for the partition of $N-I$ of which the smallest part is at least I and having the rank $R - \sum_{J=1,\ldots,I} P(N, J)$.

```
NP:=0; I:=1; NN:=N; S:=R
WHILE NN>0 DO
  NP:=NP+1; R0:=0; I:=I-1
  WHILE R0<S DO
    I:=I+1; R:=R0; R0:=R0+P(NN, I)
  S:=S-R; A(NP):=I; NN:=NN-I
```

The number of arrays $T(A/B)$ is a weighted sum of the quantities $T(A'/B')$ that we must compute, with the same principle of decomposition. Rather than to program a recursive function to compute many times the same thing, we store into an array F the following values:

$F(.,1):=NN$, quantity of which A and B are partitions;

$F(.,2):=RA$, rank of A in the set of all the partitions of NN;

$F(.,3):=RB$, rank of B in the set of all the partitions of NN;

$F(.,4):=$ weight of this couple of partitions.

The main step of the procedure to compute the number of arrays, is a decomposition of the partitions with ranks RA et RB if $NN>6$. Each row in F generates others partitions of a smaller integer, since we remove the greatest part. These partitions are added into F. If $NN\leqq 6$, we do not perform this decomposition step, because we have provided the numbers of arrays that present any couple of partitions as sums of rows and columns.

The decomposition step is made with the following operations that we perform until F is completely examined.

- We build A, partition of NN with rank RA;
- We build B, partition of NN with rank RB;
- If the greatest part of A is greater than the greatest part of B, we exchange A and B.

Let AM and BM be the greatest parts of A and B, and Q their weight.

- we remove from B its greatest part and compute the rank R of this partition of $NN-BM$;
- we enumerate the compositions of BM bounded with $(D(I), I=1, \ldots, AM)$, where $D(I)$ is the number of parts in A equal to I;
- for each composition $(C(I), I=1, \ldots, AM)$
  - we compute the partition $P(I), I=1, \ldots, NP)$ obtained after subtracting from A, one unity to $C(I)$ parts equal to I;
  - we compute the rank S of this partition;
  - we compute the number CO of ways to do these subtractions;
  - we extend F with a new row containing $NN-BM$, R, S, CO, Q.

When the decomposition step is finished, we just have to add the products of the weights of the couple of partitions of integers $NN<6$ by the number of arrays that have these partitions as the sums of their rows and columns.

Exemple 3. – $A=(2\,2\,3\,3)$, $B=(2\,2\,3\,3)$

C	NN	RA	A D	RB	B	CO	Q
	10	33	(2 2 3 3)=[0 2 2]	33	(2 2 3 3)		1
[0 2 1]	7	6	(1 1 2 3)=[2 1 1]	12	(2 2 3)	2	2
[0 1 2]	7	8	(1 2 2 2)=[1 3]	12	(2 2 3)	2	2
[2 1 0]	4	3	(1 3)	4	(2 2)	impossible	
[2 0 1]	4	4	(2 2)	4	(2 2)	1	2
[1 1 1]	4	2	(1 1 2)	4	(2 2)	2	4
[1 2]	4	2	(1 1 2)	4	(2 2)	3	6
[0 3]	4	1	(1 1 1 1)	4	(2 2)	1	2

Finaly we get:

$$T(A/B)= \begin{array}{ll} 2 & .T(2\,2/2\,2) = 2 \\ +(6+4) & .T(1\,1\,2/2\,2) = 20 \\ +\ 2 & .T(1\,1\,1\,1/2\,2) = 12 \end{array}$$

and $T(A/B)=34$.

2. CONSTRUCTION OF THE ARRAY WITH RANK R

2.1. Ordering 0/1 arrays with fixed sums of rows and columns

First we define a lexicographical order for boolean vectors with 1 before 0. For vectors with 4 components of which 2 are equal to 1, we get:

$$(1\,1\,0\,0)<(1\,0\,1\,0)<(1\,0\,0\,1)<(0\,1\,1\,0)<(0\,1\,0\,1)<(0\,0\,1\,1).$$

Let T be a 0/1 array with Nr rows $R_1, \ldots, R_{Nr}$. Each row is a vector with Nc components and T' another array with the same sums of rows and columns:

$$T=(R_1, \ldots, R_{Nr}), \qquad T'=(R'_1, \ldots, R'_{Nr})$$

We place T before T' if and only if $\exists k \mid \forall i<k\ R_i=R'_i$ et $R_k<R'_k$.

For this order, the arrays associated with the partitions (2 2 1 1) and (2 2 2) are successively:

```
1 1 0  1 1 0  1 1 0  1 1 0
1 1 0  1 0 1  1 0 1  0 1 1
0 0 1  0 1 0  0 0 1  1 0 0
0 0 1  0 0 1  0 1 0  0 0 1, etc.
```

2.2. R ranked array

To build the array with rank R, we compute its rows according to increasing index, and apply the enumeration method proposed in [2]. Just as for the partition of given rank, we count the number of arrays of which the first row is as small as possible, until we reach or overpass R.

We enumerate boolean vectors with Nc components of which a_1 are equal to 1, in the previous order. To each vector $X=(x_1, x_2, \ldots, x_{Nc})$ corresponds arrays that have X as first row. These arrays must have $(a_2, \ldots, a_{Nr})$ as sums of the other rows, and $((b_i - x_i), i=1, \ldots, Nc)$ as sums of columns. They are counted by the function defined above. Thus we just add these quantities until we reach or overpass R.

Then the first row is X and we seek now for a 0/1 array having $(a_2, \ldots, a_{Nr})$ as sums of rows, $((b_i - x_i), i=1, \ldots, Nc)$ as sums of columns and R' as rank, with:

$R' = R -$ the number of arrays having a first row before X.

Exemple 4:

We build the 10-th array corresponding to partitions (2 2 1 1) and (2 2 2).

There are $T(2\,1\,1/1\,1\,2)=5$ of which the first row is (1 1 0)
$T(2\,1\,1/1\,2\,1)=5$ (1 0 1)

So the first row is (1 0 1) and we seek for the 5-th corresponding to the partitions (2 1 1) and (1 2 1).

There are $T(1\,1/0\,1\,1)=2$ of which the first row is (1 1 0)
$T(1\,1/0\,2\,0)=1$ (1 0 1)
$T(1\,1/1\,1\,0)=2$ (0 1 1)

The second row is (0 1 1) and we seek the 2-nd corresponding to the partitions (1 1) et (1 1 0).

There are $T(1/0\,1\,0)=1$ of which the first row is (1 0 0)
$T(1/1\,0\,0)=1$ (0 1 0)

the third row is (0 1 0) and the last one is the remaining sums of the columns. The 10-th array is:

1 0 1
0 1 1
0 1 0
1 0 0

2.3. Algorithm

To apply this algorithm we must enumerate the subsets with a_i elements in a set with Nc elements. We borrow this classical algorithm to Nijenhuis et Wilf [3]. But all these subsets are not suitable, since some columns can be saturated, and the general algorithms is:

Let $(DA(I),\ I=1,\ Nr)$ and $(DB(I),\ I=1,\ Nc)$ be the specified sums of rows and columns, and R the rank of the searched narray.

```
FOR 1:=1 TO Nr-1 DO
  RC:=0
  LET RA:=Rank of the partition (DA(L+1), . . . , DA(Nr))
FOR ANY (X(I), I=1, Nc) SUCH THAT ΣX(I)=DA(L) DO
  FOR I:=1 TO Nc DO
    P(I):=DB(I)-X(I); IF P(I)<0 GOTO a/
  LET RB:=Rank of the partition (P(I), I=1, Nc))
  NT:=T(RA/RB); RC:=RC+NT; IF RC≧R GOTO b/
  a/END
b/ R:=R-RC+NT
FOR I:=1 TO Nc DO
  DB (I):=DB(I)-X(I)
```

2.4. Complexity and conclusions

The advantage announced in the introduction – succeed at the first shot – is counter balanced by a high complexity. Even if we admit that all the necessary countings are tabulated, we must, in the worst case where the wanted array is the last one, enumerate for each row all the subsets with cardinality a_i in a set with Nc elements. The worst case will be reached when $a_i = Nc/2$. Then, there are $(Nc \setminus Nc/2)$ subsets. Thus we have a complexity $O(Nr.2^{Nc-1})$.

Therefore, the performance of this algorithm can be compared with N. C. Wormald's only in his difficult cases, that is when the searched array is practically full of ones. In this case, my algorithm is faster than his, but we can find very quickly, with the Wormald method, the complement of the searched array. More over, his algorithm requires only a few instructions.

It remains that we can compute tables for the number of bipartite graphs with specified degrees, as we show in the appendix, and select them at random following any distribution law, since it is the rank that is selected at random.

REFERENCES

[1] Cl. Berge, *Principes de Combinatoire*, Dunod, Paris, 1968.

[2] A. Guénoche, *Énumération de classes de permutations*, RAIRO Recherche Opérationnelle, Vol. 13, No. 4, 1979, pp. 379-390.

[3] H. Nijenhuis & H. Wilf, *Combinatorial Algorithms*, Acad. Press, New York, 1978.

[4] N. C. Wormald, *Generating random regular graphs*, Journal of Algorithms, Vol. 5, No. 2, 1984, pp. 247-280.

Appendix: *Tables*

The degree sequences of vertices of each part is a partition of an integer N equal to the number of egdes. The number of bipartite graphs with a couple of partitions is given in the following tables whose entries are the partitions of $3 \leqq N \leqq 10$. They are labelled in decreasing number of parts order, then in lexicographical order. For each table, the column order is the same as the row one, and for instance, the number of bipartite graphs with degree sequences (1 1 1 1 2) and (1 2 3) can be read in the table of partitions of 6; These sequences have ranks 2 and 6, so there are 22 graphs with these degrees.

Partitions of 3

```
1 : 111 : 6
2 : 12  : 3 1
3 : 3   : 1 0 0
```

Partitions of 4

```
1 : 1111 : 24
2 : 112  : 12 5
3 : 13   :  4 1 0
4 : 22   :  6 2 0 1
5 : 4    :  1 0 0 0 0
```

Partitions of 5

```
1 : 11111 : 120
2 : 1112  :  60 27
3 : 113   :  20  7 1
4 : 122   :  30 12 2 5
5 : 14    :   5  1 0 0 0
6 : 23    :  10  3 0 1 0 0
7 : 5     :   1  0 0 0 0 0 0
```

Partitions of 6

```
 1 : 111111 : 720
 2 : 11112  : 360 168
 3 : 1113   : 120  48 10
 4 : 1122   : 180  78 18 34
 5 : 114    :  30   9  1  2  0
 6 : 123    :  60  22  3  8  0 1
 7 : 222    :  90  36  6 15 0 3 6
 8 : 15     :   6   1  0  0 0 0 0 0
 9 : 24     :  15   4  0  1 0 0 0 0 0
10 : 33     :  20   6  0  2 0 0 1 0 0 0
11 : 6      :   1   0  0  0 0 0 0 0 0 0 0
```

Partitions of 7

```
 1 : 1111111 : 5040
 2 : 111112  : 2520 1200
 3 : 11113   :  840  360  88
 4 : 11122   : 1260  570 150 258
 5 : 1114    :  210   75  13  24  1
 6 : 1123    :  420  170  34  68  3 13
 7 : 1222    :  630  270  60 117  6 27 51
 8 : 115     :   42   11   1   2  0  0  0  0
 9 : 124     :  105   35   4  11  0  1  3  0  0
10 : 133     :  140   50   6  18  0  2  7  0  0  0
11 : 223     :  210   80  12  31  0  5 12  0  0  1  2
12 : 16      :    7    1   0   0  0  0  0  0  0  0  0  0
13 : 25      :   21    5   0   1  0  0  0  0  0  0  0  0  0
14 : 34      :   35   10   0   3  0  0  1  0  0  0  0  0  0  0
15 : 7       :    1    0   0   0  0  0  0  0  0  0  0  0  0  0  0
```

Partitions of 8

```
 1 : 11111111 : 40320
 2 : 1111112  : 20160 9720
 3 : 111113   :  6720 3000  800
 4 : 111122   : 10080 4680 1320 2172
 5 : 11114    :  1680  660  140  246 17
 6 : 11123    :  3360 1440  340  612 46 141
 7 : 11222    :  5040 2250  570 1008 84 258 453
 8 : 1115     :   336  108   16   30  1   3   6  0
 9 : 1124     :   840  315   55  114  4  18  39  0  1
10 : 1133     :  1120  440   80  172  6  30  68  0  2  4
11 : 1223     :  1680  690  140  284 12  58 117  0  5 12 24
12 : 2222     :  2520 1080  240  468 24 108 204  0 12 28 48 90
13 : 116      :    56   13    1    2  0   0   0  0  0  0  0  0  0
14 : 125      :   168   51    5   14  0   1   3  0  0  0  0  0  0  0
15 : 134      :   280   95   10   32  0   3  11  0  0  0  1  4  0  0  0
16 : 224      :   420  150   20   53  0   7  18  0  0  1  2  6  0  0  0  0
17 : 233      :   560  210   30   80  0  12  31  0  0  2  5 12  0  0  0  0  1
18 : 17       :     8    1    0    0  0   0   0  0  0  0  0  0  0  0  0  0  0  0
19 : 26       :    28    6    0    1  0   0   0  0  0  0  0  0  0  0  0  0  0  0
20 : 35       :    56   15    0    4  0   0   1  0  0  0  0  0  0  0  0  0  0  0  0
21 : 44       :    70   20    0    6  0   0   2  0  0  0  0  1  0  0  0  0  0  0  0  0
22 : 8        :     1    0    0    0  0   0   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Partitions of 9

1 : 111111111	:	362880														
2 : 11111112	:	181440	88200													
3 : 1111113	:	60480	27720	7800												
4 : 1111122	:	90720	42840	12600	20220											
5 : 111114	:	15120	6300	1500	2550	225										
6 : 111123	:	30240	13440	3480	5940	570	1516									
7 : 111222	:	45360	20790	5670	9540	990	2610	4383								
8 : 11115	:	3024	1092	204	366	21	58	108	1							
9 : 11124	:	7560	3045	645	1200	75	241	459	4	25						
10 : 11133	:	10080	4200	920	1740	110	372	720	6	42	72					
11 : 11223	:	15120	6510	1530	2800	200	656	1203	12	85	152	281				
12 : 12222	:	22680	10080	2520	4500	360	1140	2016	24	168	300	516	906			
13 : 1116	:	504	147	19	36	1	3	6	0	0	0	0	0	0		
14 : 1125	:	1512	525	81	172	5	23	51	0	1	2	5	12	0	0	
15 : 1134	:	2520	945	155	350	10	54	129	0	3	6	19	48	0	0	0
16 : 1224	:	3780	1470	270	565	20	101	213	0	7	17	36	78	0	0	1 2
17 : 1233	:	5040	2030	390	820	30	158	333	0	12	30	65	136	0	0	2 5 13
18 : 2223	:	7560	3150	660	1320	60	282	555	0	27	64	120	234	0	0	7 12 27
19 : 117	:	72	15	1	2	0	0	0	0	0	0	0	0	0	0	0 0 0
20 : 126	:	252	70	6	17	0	1	3	0	0	0	0	0	0	0	0 0 0
21 : 135	:	504	161	15	50	0	4	15	0	0	0	1	4	0	0	0 0 0
22 : 144	:	630	210	20	70	0	6	24	0	0	0	2	9	0	0	0 0 0
23 : 225	:	756	252	30	81	0	9	24	0	0	1	2	6	0	0	0 0 0
24 : 234	:	1260	455	60	165	0	22	60	0	0	3	8	22	0	0	0 0 1
25 : 333	:	1680	630	90	240	0	36	93	0	0	6	15	36	0	0	0 0 3
26 : 18	:	9	1	0	0	0	0	0	0	0	0	0	0	0	0	0 0 0
27 : 27	:	36	7	0	1	0	0	0	0	0	0	0	0	0	0	0 0 0
28 : 36	:	84	21	0	5	0	0	1	0	0	0	0	0	0	0	0 0 0
29 : 45	:	126	35	0	10	0	0	3	0	0	0	0	1	0	0	0 0 0
30 : 9	:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0 0 0

18 : 2223	:	51														
19 : 117	:	0	0													
20 : 126	:	0	0	0												
21 : 135	:	0	0	0	0											
22 : 144	:	1	0	0	0	0										
23 : 225	:	0	0	0	0	0	0									
24 : 234	:	3	0	0	0	0	0	0								
25 : 333	:	6	0	0	0	0	0	0	1							
26 : 18	:	0	0	0	0	0	0	0	0	0						
27 : 27	:	0	0	0	0	0	0	0	0	0	0					
28 : 36	:	0	0	0	0	0	0	0	0	0	0	0				
29 : 45	:	0	0	0	0	0	0	0	0	0	0	0	0			
30 : 9	:	0	0	0	0	0	0	0	0	0	0	0	0	0		

Partitions of 10

1 : 1111111111	:	3628800										
2 : 111111112	:	1814400	887040									
3 : 11111113	:	604800	282240	82320								
4 : 11111122	:	907200	433440	131040	207000							
5 : 1111114	:	151200	65520	16800	27900	2850						
6 : 1111123	:	302400	137760	37800	62520	6900	17000					
7 : 1111222	:	453600	211680	60480	98820	11610	28260	46152				
8 : 111115	:	30240	11760	2520	4380	330	860	1530	26			
9 : 111124	:	75600	31920	7560	13290	1095	3050	5436	95	416		
10 : 111133	:	100800	43680	10640	18840	1580	4520	8100	140	650	1032	
11 : 111223	:	151200	67200	17220	29820	2730	7610	13212	260	1196	1920	3358
12 : 112222	:	226800	103320	27720	47160	4680	12720	21564	480	2172	3480	5844
13 : 11116	:	5040	1680	280	510	25	70	132	1	4	6	12
14 : 11125	:	15120	5712	1092	2082	111	368	720	5	32	54	112
15 : 11134	:	25200	10080	2030	3990	215	775	1566	10	76	132	294
16 : 11224	:	37800	15540	3360	6330	390	1340	2547	20	149	270	523
17 : 11233	:	50400	21280	4760	8980	570	2000	3792	30	236	436	844
18 : 12223	:	75600	32760	7770	14220	1020	3390	6180	60	450	820	1479
19 : 22222	:	113400	50400	12600	22500	1800	5700	10080	120	840	1500	2580
20 : 1117	:	720	192	22	42	1	3	6	0	0	0	0
21 : 1126	:	2520	812	112	242	6	28	63	0	1	2	5
22 : 1135	:	5040	1792	266	622	15	85	210	0	4	8	26
23 : 1144	:	6300	2310	350	840	20	120	306	0	6	12	42
24 : 1225	:	7560	2772	462	990	30	156	339	0	9	22	48
25 : 1234	:	12600	4900	875	1900	60	335	735	0	22	56	128
26 : 1333	:	16800	6720	1260	2700	90	510	1092	0	36	96	210
27 : 2224	:	18900	7560	1470	3015	120	585	1194	0	48	115	228
28 : 2233	:	25200	10360	2100	4280	180	880	1776	0	78	188	370
29 : 118	:	90	17	1	2	0	0	0	0	0	0	0
30 : 127	:	360	92	7	20	0	1	3	0	0	0	0
31 : 136	:	840	252	21	72	0	5	19	0	0	0	1
32 : 145	:	1260	406	35	130	0	10	42	0	0	0	3
33 : 226	:	1260	392	42	115	0	11	30	0	0	1	2
34 : 235	:	2520	868	105	296	0	35	99	0	0	4	11
35 : 244	:	3150	1120	140	400	0	50	144	0	0	6	18
36 : 334	:	4200	1540	210	570	0	80	213	0	0	12	31
37 : 19	:	10	1	0	0	0	0	0	0	0	0	0
38 : 28	:	45	8	0	1	0	0	0	0	0	0	0
39 : 37	:	120	28	0	6	0	0	1	0	0	0	0
40 : 46	:	210	56	0	15	0	0	4	0	0	0	0
41 : 55	:	252	70	0	20	0	0	6	0	0	0	0
42 : (10)	:	1	0	0	0	0	0	0	0	0	0	0

```
12 : 112222 : 9876
13 : 11116  :   24 0
14 : 11125  :  228 0 1
15 : 11134  :  612 0 3   9
16 : 11224  : 1008 0 7  27  54
17 : 11233  : 1604 0 12 48  99 186
18 : 12223  : 2688 0 27 112 198 358  645
19 : 22222  : 4530 0 60 240 390 680 1170 2040
20 : 1117   :    0 0  0   0   0   0    0    0 0
21 : 1126   :   12 0  0   0   0   0    0    0 0 0
22 : 1135   :   68 0  0   0   1   2    7   20 0 0 0
23 : 1144   :  114 0  0   0   2   4   16   45 0 0 0 0
24 : 1225   :  108 0  0   1   2   5   12   30 0 0 0 0 0
25 : 1234   :  284 0  0   3   8  21   49  110 0 0 0 0 0 1
26 : 1333   :  444 0  0   6  15  42   87  180 0 0 0 0 0 3 10
27 : 2224   :  468 0  0  10  18  42   87  180 0 0 0 1 0 3  6  6
28 : 2233   :  740 0  0  18  34  80  156  310 0 0 0 2 0 8 18 15 34
29 : 118    :    0 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0
30 : 127    :    0 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0
31 : 136    :    4 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0
32 : 145    :   14 0  0   0   0   0    1    5 0 0 0 0 0 0  0  0  0 0 0 0 0
33 : 226    :    6 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0 0 0
34 : 235    :   32 0  0   0   0   1    3   10 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0
35 : 244    :   53 0  0   0   0   2    7   20 0 0 0 0 0 0  0  0  1 0 0 0 0 0 0 0
36 : 334    :   80 0  0   0   0   5   12   30 0 0 0 0 0 0  0  1  0 2 0 0 0 0 0 0 0 0
37 : 19     :    0 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0 0 0 0 0
38 : 28     :    0 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0 0 0 0 0
39 : 37     :    0 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0 0 0 0 0
40 : 46     :    1 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0 0 0 0 0
41 : 55     :    2 0  0   0   0   0    0    1 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0 0 0 0 0
42 : (10)   :    0 0  0   0   0   0    0    0 0 0 0 0 0 0  0  0  0 0 0 0 0 0 0 0 0 0 0

38 : 28     : 0
39 : 37     : 0 0
40 : 46     : 0 0 0
41 : 55     : 0 0 0 0
42 : (10)   : 0 0 0 0 0
```