Chapter 8: Symmetry Breaking (Balanced Incomplete Block Designs)

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning Overview

Licence

Outline

What we want to introduce

- BIBD - Balanced Incomplete Block Designs
- Using lex constraints to remove symmetries
- Only one of many ways to deal with symmetry in problems
- Finding all solutions to a problem
- Using timeout to limit search

Problem Definition

BIBD (Balanced Incomplete Block Design)

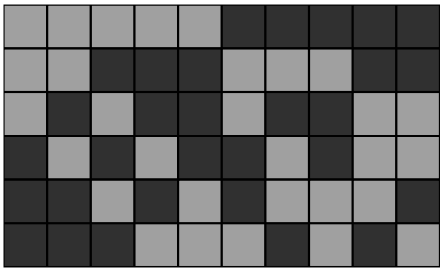
A BIBD is defined as an arrangement of v distinct objects into b blocks such that each block contains exactly k distinct objects, each object occurs in exactly r different blocks, and every two distinct objects occur together in exactly λ blocks. A BIBD is therefore specified by its parameters (v, b, r, k, λ).

Motivation: Test Planning

Consider a new release of some software with *v* new features. You want to regression test the software against combinations of the new features. Testing each subset of features is too expensive, so you want to run *b* tests, each using *k* features. Each feature should be used *r* times in the tests. Each pair of features should be tested together exactly λ times. How do you arrange the tests?

Model

Another way of defining a BIBD is in terms of its incidence matrix, which is a binary matrix with v rows, b columns, r ones per row, k ones per column, and scalar product λ between any pair of distinct rows.



A (6,10,5,3,2) BIBD

Model for (v, b, r, k, λ) BIBD

- A binary $v \times b$ matrix. Entry V_{ij} states if item i is in block j.
- Sum constraints over rows, each sum equal r
- Sum constraints over columns, each sum equal k
- Scalar product between any pair of rows, the product value is λ.

Top Level Program

```
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix),writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),⇨ Set up model
    extract_array(row,Matrix,List),⇨ Get list
    search(L,0,input_order,indomain,
           complete,[]).⇨ Search
```


Constraint Model

```
model(V,B,R,K,L,Matrix,Method):-
    dim(Matrix,[V,B]),⇨ Define Binary Matrix
    Matrix[1..V,1..B] :: 0..1,
    (for(I,1,V), param(Matrix,B,R) do
        sumlist(Matrix[I,1..B],R)
    ),⇨ Row Sum = R
    (for(J,1,B), param(Matrix,V,K) do
        sumlist(Matrix[1..V,J],K)
    ),⇨ Column Sum = K
    (for(I,1,V-1), param(Matrix,V,B,L) do
        (for(I1,I+1,V), param(Matrix,I,B,L) do
            scalar_product(Matrix[I,1..B],
                           Matrix[I1,1..B],L)
        )
    ).⇨ Scalar product between all rows
```

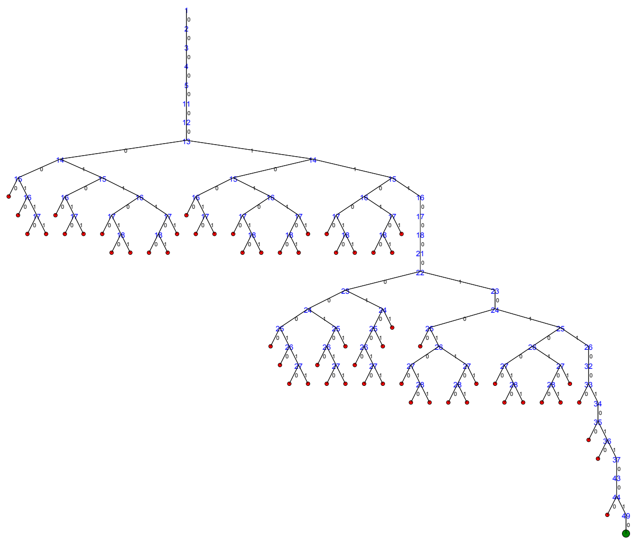
scalar_product

```
scalar_product(XVector,YVector,V):-
    collection_to_list(XVector,XList),
    collection_to_list(YVector,YList),➪ Get lists
    (foreach(X,XList),➪ Iterate over lists
     foreach(Y,YList),➪ ...in parallel
     fromto(0,A,A1,Term) do ➪ Build term
        A1 = A+X*Y➪ Construct term
    ),
    eval(Term) #= V.➪ State Constraint
```


Search Routine

- Static variable order
- First fail does not work for binary variables
- Enumerate variables by row
- Use utility predicate `extract_array/3`
- Assign with `indomain`, try value 0, then value 1
- Use simple `search` call

Basic Model - First Solution



Finding all solutions - Hack!

```
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix),writeln(Matrix),
    fail.⇨ Force Backtracking

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,
           complete,[]).
```

Finding all solutions - Proper

```
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    findall(Matrix,bibd(6,10,5,3,2,Matrix),Sols),
    writeln(Sols).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,indomain,
           complete,[]).
```


`findall` predicate

- `findall(Template,Goal,Collection)`
- Finds all solutions to `Goal` and collects them into a list `Collection`
- `Template` is used to extract arguments from `Goal` to store as solution
- Backtracks through all choices in `Goal`
- Solutions are returned in order in which they are found

Problem

- Program now only stops when it has found all solutions
- This takes too long!
- How can we limit the amount of time to wait?
- Use of the `timeout` library

Finding all solutions - Proper

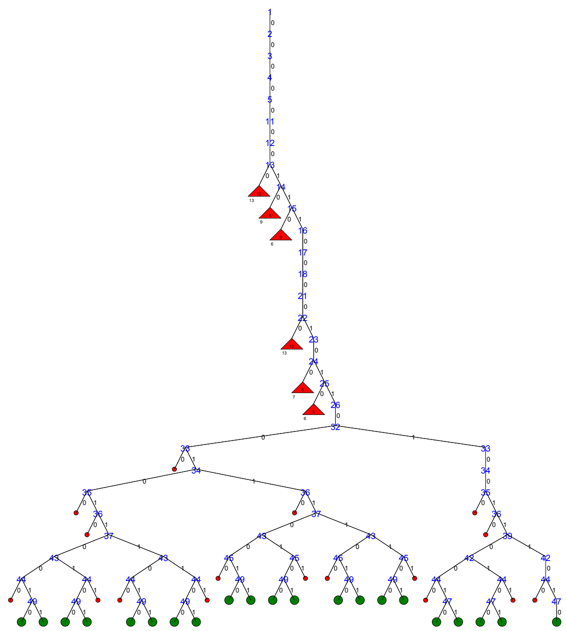
```
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).
:-lib(timeout).⇨ Load library

top:-
    findall(Matrix,timeout(bibd(6,10,5,3,2,Matrix),
                           10,⇨ seconds
                           fail),Sols),
    writeln(Sols).
```

`timeout` library

- `timeout(Goal,Limit,TimeoutGoal)`
- Runs `Goal` for `Limit` seconds
- If `Limit` is reached, `Goal` is stopped and `TimeoutGoal` is run instead
- If `Limit` is not reached, it has no impact
- Must load `:-lib(timeout).`

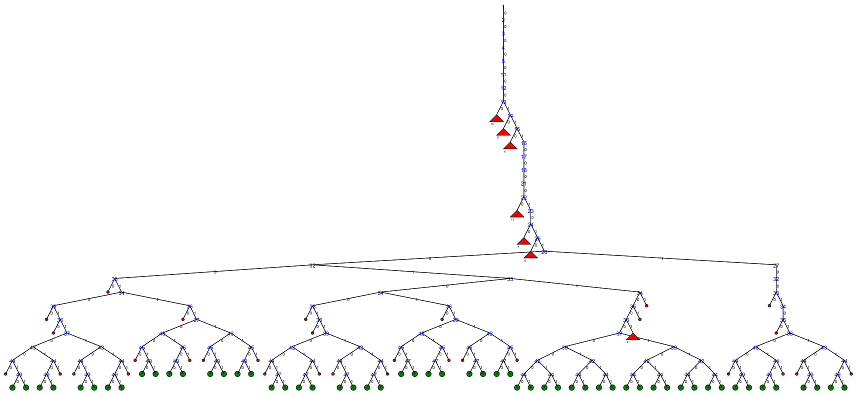
Finding all Solutions - Search Tree 200 Nodes



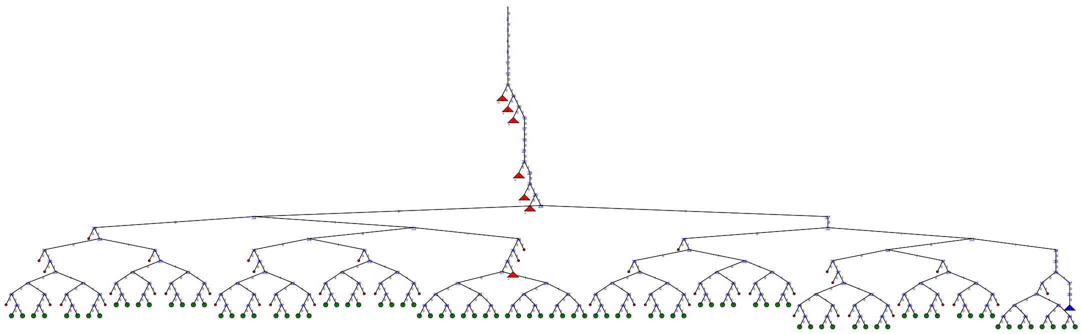
Observation

- Surprise! There are many solutions

Search Tree 300 Nodes

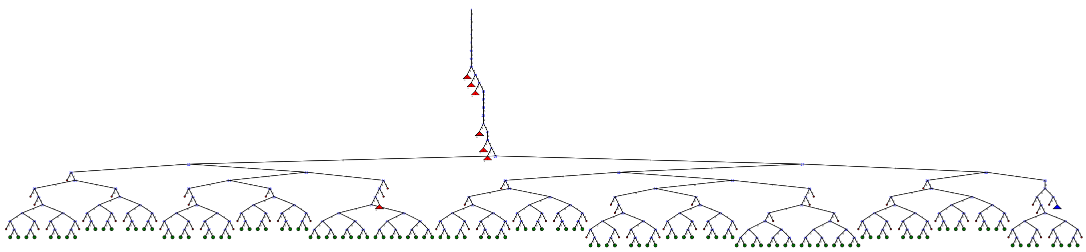


Search Tree 400 Nodes



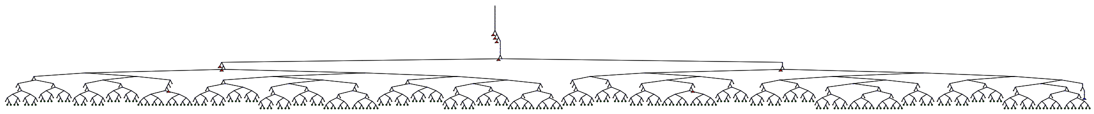
Cork Constraint Computation Centre

Search Tree 500 Nodes

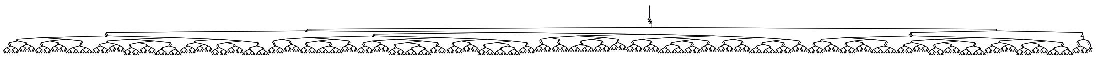


Cork Constraint Computation Centre

Search Tree 1000 Nodes



Search Tree 2000 Nodes



Problem

- There are too many solutions to collect in a reasonable time
- Most of these solutions are very similar
- If you take one solution and
  - exchange two rows
  - and/or exchange two columns
- ... you have another solution
- Can we avoid exploring them all?

Cork Constraint Computation Centre

Symmetry Breaking Techniques

- Remove all symmetries
  - Reduce the search tree as much as possible
  - May be hard to describe all symmetries
  - May be expensive to remove symmetric parts of tree
- Remove some symmetries
  - Search is not reduced as much
  - May be easier to find some symmetries to remove
  - Cost can be low

Cork Constraint Computation Centre

Symmetry Breaking Techniques

- Symmetry removal by forcing partial, initial assignment
  - Easy to understand
  - Rather weak, does not affect search
- Symmetry removal by stating constraints
  - Removing all symmetries may require exponential number of constraints
  - Can conflict with search strategies
- Symmetry removal by controling search
  - At each node, decide if it needs to be explored
  - Can be expensive to check

Solution used here: Double Lex

- Partial symmetry removal by adding lexicographical ordering constraints
- Our problem has full row and column symmetries
- Any permutation of rows adn/or columns leads to another solution
- Idea: Order rows lexicographically
- Rows must be different from each other, strict order on rows
- Columns might be identical, non strict order on columns
  - This can be improved in some cases
- Constraints only between adjacent rows(columns)

Added Constraints

```
dim(Matrix,[V,B]),
(for(I,1,V-1),
 param(Matrix,B) do
    I1 is I+1,
    lex_less(Matrix[I1,1..B],Matrix[I,1..B])
 ),➪ Row lex constraints
 (for(J,1,B-1),
  param(Matrix,V) do
     J1 is J+1,
     lex_leq(Matrix[1..V,J1],Matrix[1..V,J])
  ),➪ Column lex constraints
```


Using Two Global Constraints

- `lex_leq(List1,List2)`
  - `List1` is lexicographical smaller than or equal to `List2`
  - Achieves domain consistency
- `lex_less(List1,List2)`
  - `List1` is lexicographical smaller than `List2`
  - Achieves domain consistency

Example propagation `lex_less`

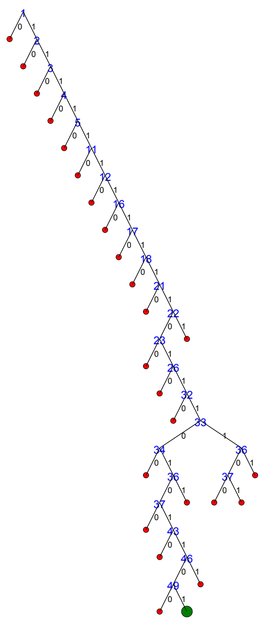
Before

[	2,	X2 $\in \{1,3,4\}$,	X3 $\in \{1,2,3\}$,	X4 $\in \{1,2\}$,	X5 $\in \{3,4\}$],
[	Y1 $\in \{0,1,2\}$,	1,	Y3 $\in \{0,1,2,3\}$,	Y4 $\in \{0,1\}$,	Y5 $\in \{0,1\}$]

After

[	2,	1,	X3 $\in \{1,2\}$,	X4 $\in \{1,2\}$,	X5 $\in \{3,4\}$],
[	2,	1,	Y3 $\in \{2,3\}$,	Y4 $\in \{0,1\}$,	Y5 $\in \{0,1\}$]

Complete Search Tree with Double Lex



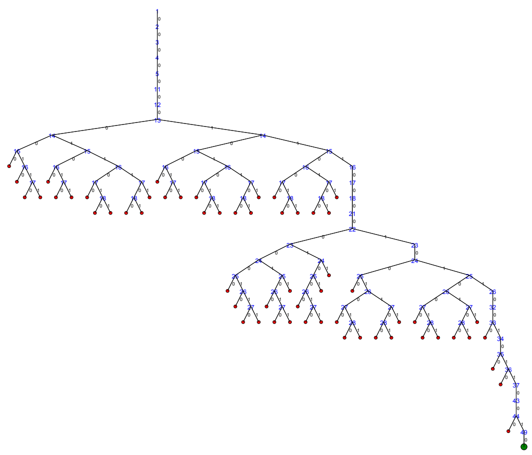
Observation

- Enormous reduction in search space
- We are solving a different problem!
- Not just good for finding all solutions, also for first solution!
- Value choice not optimal for finding first solution
- There is a lot of very shallow backtracking, can we avoid that?

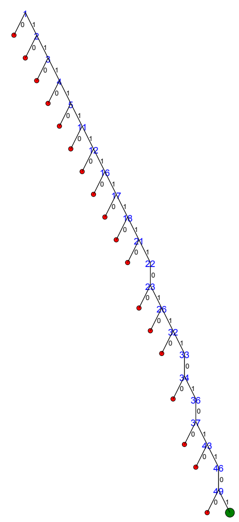
Cork Constraint Computation Centre

Effort for First Solution

Basic Model



With double Lex



Cork Constraint Computation Centre

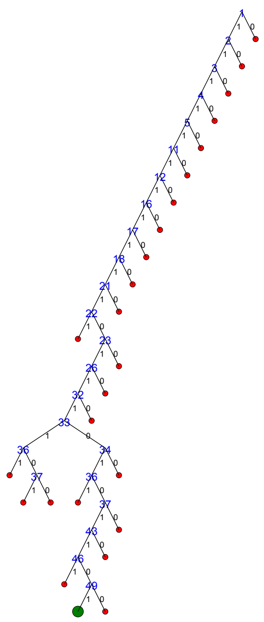
Alternative Value Order

```
:-module(bibd).
:-export(top/0).
:-lib(ic).
:-lib(ic_global).

top:-
    bibd(6,10,5,3,2,Matrix),writeln(Matrix).

bibd(V,B,R,K,L,Matrix):-
    model(V,B,R,K,L,Matrix),
    extract_array(row,Matrix,List),
    search(L,0,input_order,
           indomain_max,⇨ Start with 1
           complete,[]).
```


Assigning Value 1 First



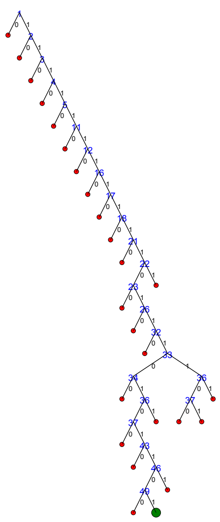
Observation

- First solution is found more quickly
- Size of tree for all solutions unchanged
- Value order does not really affect search space when exploring all choices!

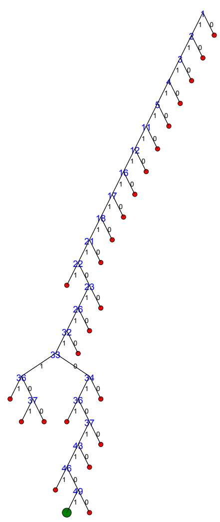
Cork Constraint Computation Centre

Effort for All Solutions

Assign 0, then 1



Assign 1, then 0



Cork Constraint Computation Centre

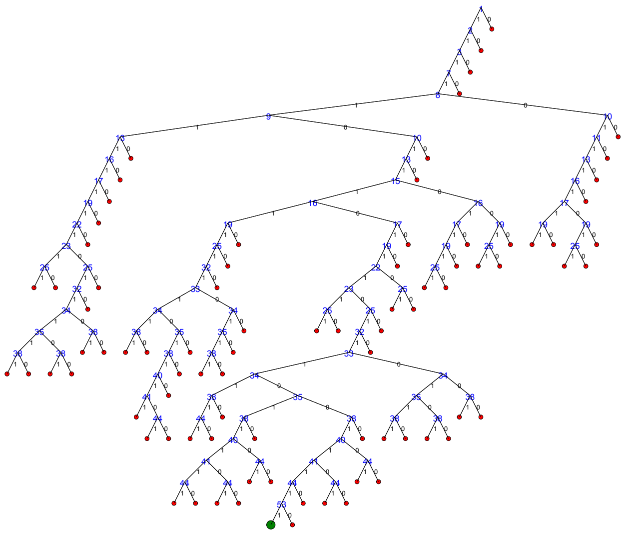
Conclusions

- Symmetry breaking can have huge impact on model
- Mainly works for pure problems
- Partial symmetry breaking with additional constraints
- Double lex for row/column symmetries
- Only one variant of many symmetry breaking techniques

Row- or Column- wise Assignment?

- We did assign matrix by row, why?
- What happens if we assign variables by column?

Variable Selection by Column

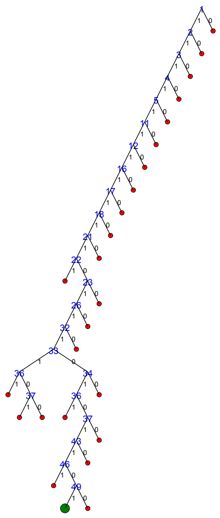


Observation

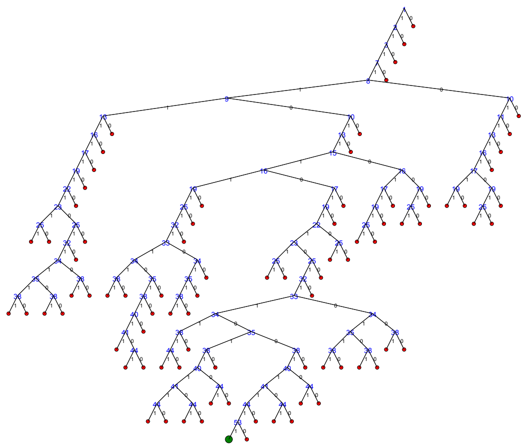
- Good, but not as good as row order
- Value choice (0/1) or (1/0) unimportant even for first solution
- Changing the variable selection does affect size of search space, even for all solutions

Effort for All Solutions

By Row



By Column



Possible Explanations

- There are fewer rows than columns
- Strict lex constraints on rows, but not on columns
    - More impact of first row
- Needs better understanding

Does this scale?

v	b	r	k	λ	asym	lex^2	STAB	lex^2 + SBNO
9	24	8	3	2	36	5,987	344	311
16	16	6	6	2	3	46	3	7
15	21	7	5	2	0	0	0	0
13	26	6	3	1	2	12,800	21	101
7	35	15	3	5	109	33,304	542	282
15	15	7	7	3	5	118	19	19
21	21	5	5	1	1	12	1	1
25	30	6	5	1	1	864	1	5
10	18	9	5	4	21	8,031	302	139
7	42	18	3	6	418	250,878	2,334	1,247
22	22	7	7	2	0	0	0	0
7	49	21	3	7	1,508	1,460,332	8,821	4,353
8	28	14	4	6	2,310	2,058,523	17,890	11,424
19	19	9	9	4	6	6,520	71	17
10	30	9	3	2	960	724,662	24,563	15,169
31	31	6	6	1	1	864	1	2
7	56	24	3	8	5,413	6,941,124	32,038	14,428
9	36	12	3	3	22,521	14,843,772	315,531	85,605
7	63	27	3	9	?	28,079,394	105,955	43,259
15	35	7	3	1	80	32,127,296	6,782	35,183
21	28	8	6	2	0	0	0	0
13	26	8	4	2	2461	3,664,243	83,337	31,323
11	22	10	5	4	4393	6,143,408	106,522	32,908
12	22	11	6	5	?	?	228,146	76,572
25	25	9	9	3	?	?	17,016	1,355
16	24	9	6	3	?	?	769,482	76,860

Scalability

- lex^2 good, but not good enough
- Still leaves too many symmetries to explore
- Better techniques in the literature
  - STAB, group theory based, Puget 2003.
  - SBNO, local search based domination check, Prestwich, 2008.

Do we need binary variables?

- The 0/1 model does very little propagation
- Consider a model with finite domain variables
- Each of b blocks consists of k variables ranging over v values
- The values in a block must be alldifferent (ordered)
- Each value can occur r times
- Scalar product more difficult
- Even better expressed with finite set variables

More Information

I. Gent, K. Petrie, and J.F. Puget.
Symmetry in constraint programming.
In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 10. Elsevier, 2006.

Jean-Francois Puget.
Symmetry breaking using stabilizers.
In Francesca Rossi, editor, *CP*, volume 2833 of *Lecture Notes in Computer Science*, pages 585–599. Springer, 2003.

More Information

S. D. Prestwich , B. Hnich , R. Rossi, and S. A. Tarim.
Symmetry Breaking by Metaheuristic Search.
SymCon 2008 - The 8th International Workshop on Symmetry and Constraint Satisfaction Problems, Sydney, Australia, September, 2008.

Exercises

1