Problem A
CDS-Tree

Input file: *testdata.in*
Time limit: 1 second

Finding clusters in data streams is a popular application in data mining fields. For example, a web site may cluster customer shopping information with some criteria so that the recommendation system may suggest suitable products to each individual customer.

CDS-Tree(Cell Dimension Tree for Streams) proposed by Sun et al[1], is an effective data structure to finding clusters in data streams using a cell-based algorithm.

The basic idea of the cell-based algorithms is to:

1. Divide data space into cells,
2. Count the number of points in each cell,
3. Connect dense neighbor cells to form clusters.

The problem description:

The input data space is spanned by two data streams corresponding to a two-dimensional cell array. The cell array is a square array with **the maximum array size, 10×10**. Each cell may represent different number of data for each individual dimension. The objective of your program is to construct the CDS-Trees according to the input parameters as well as to print out the CDS-Trees by the level order.

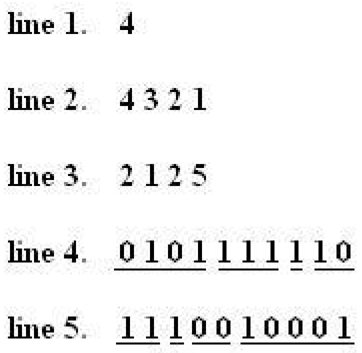
Input and Output Session

Input data file:

The input data file consists of **NS** sets of data, where $1 \leq \mathbf{NS} \leq 10$. Each set of data is defined by exact 5 consecutive lines. The first line defines the number of cells, **NC**, for both two dimension, and $1 \leq \mathbf{NC} \leq 10$. The 2nd and 3rd lines give the number of data, **ND**, for each individual cell in the dimension A_1 and the dimension A_2 respectively, where $1 \leq \mathbf{ND} \leq 10$. The **ND**s are separated by a blank space according to the dimension index in ascending order. The 4th and 5th lines are data streams for dimension A_1 and A_2 respectively. Each data stream consists of digits, 0 or 1, with maximum data length, **MAX_D_L**, where $\mathbf{NC} \leq \mathbf{MAX\_D\_L} \leq 100$. The digits of the input data stream are separated by a blank space.

For example, an input file with only one set of data is illustrated in the figure 2 and the corresponding two-dimensional cell array is depicted in the figure 1:

- The first line defines NC=4, that means the input data space will be represent in a 4×4 cell array.
- The second line assigns the **ND**s for the cell indexes 1, 2, 3, 4 of the dimension A_1 to 4, 3, 1, and 2.
- The third line assigns the **ND**s for the cell indexes 1, 2, 3, 4 of the dimension A_2 to 2, 1, 2, and 5.
- The 4th line defines the data stream of dimension A_1:
  $\underline{\mathbf{0\ 1\ 0\ 1}}\ \underline{\mathbf{1\ 1\ 1}}\ \underline{\mathtt{1}}\ \underline{\mathtt{1\ 0}}$
- The 5th line defines the data stream of dimension A_2:
  $\underline{\mathbf{1\ 1}}\ \underline{\mathtt{1}}\ \underline{\mathbf{0\ 0}}\ \underline{\mathtt{1\ 0\ 0\ 0\ 1}}$



```
line 1.  4
line 2.  4 3 2 1
line 3.  2 1 2 5
line 4.  0 1 0 1 1 1 1 1 1 0
line 5.  1 1 1 0 0 1 0 0 0 1
```

Figure 1: The input data file

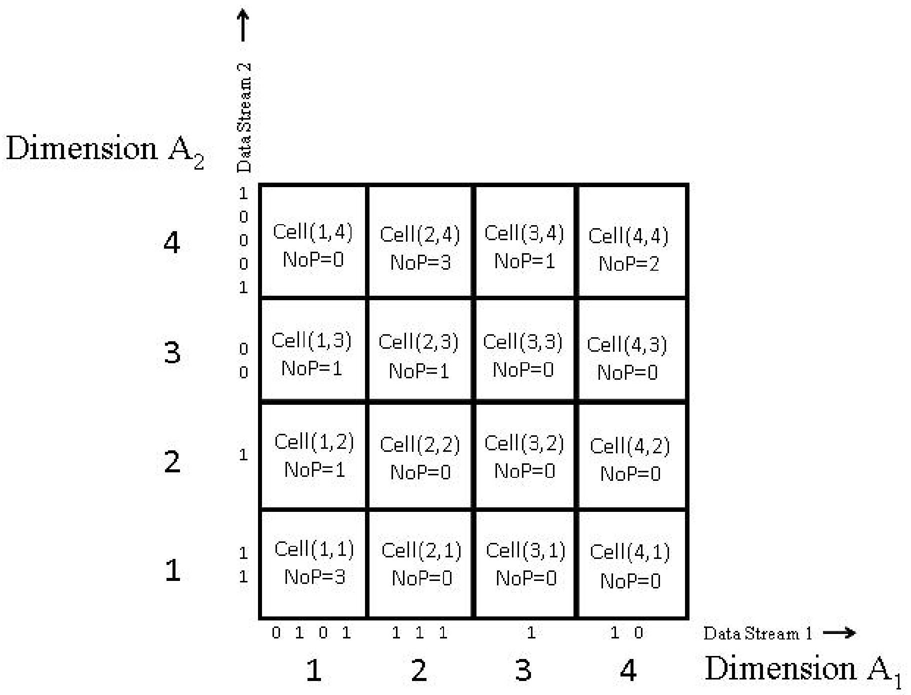


Figure 2: The corresponding cell array of the example input data in the figure 1

- The corresponding two-dimensional cell array:

– **NoP** is a summation of number of "1"s for all digit pairs represented in the same cell. The digit pairs are enumerated by the input sequence of both data streams. In this example, the sequence of the digit pairs are (0,1), (1,1), (0,1), (1,0), (1,0), (1,1), (1,0), (1,0), (1,0), and (0,1).

– The cell(1,1) is to represent the first 4 digits (**$\underline{0\ 1\ 0\ 1}$**) in dimension A_1 and the first 2 digits (**$\underline{1\ 1}$**)in dimension A_2. The first digit pair of stream 1 and stream 2 is (0,1), belongs to cell(1,1) and the number of "1"s is 1. The second digit pair is (1,1), belongs to cell(1,1) and the number of point is 2. Therefore, **NoP** of cell(1,1) is **1+2=3**.

– The cell(1,2) is to represent the first 4 digits (**$\underline{0\ 1\ 0\ 1}$**) in dimension A_1 and the 3rd digit ($\underline{1}$) in dimension A_2. The 3rd digit pair is (0,1), belongs to cell(1,2) and the number of point is 1. Therefore, **NoP** of cell(1,2) is **1**.

– The cell(2,3) is to represent the 3 digits from the 5th digit ($\underline{1\ 1\ 1}$) in dimension A_1 and the 2 digits from the 4th digit (**$\underline{00}$**) in dimension A_2. The 5th digit digit pair is (1, 0), belongs to cell(2,3) and the number of point is 1. Therefore, **NoP** of cell(2,3) is **1**.

Your program should read in data according to the cell array information in the first three lines of each data set and generate the corresponding CDS-Trees. An example of the CDS-Trees is depicted in the figure 3.

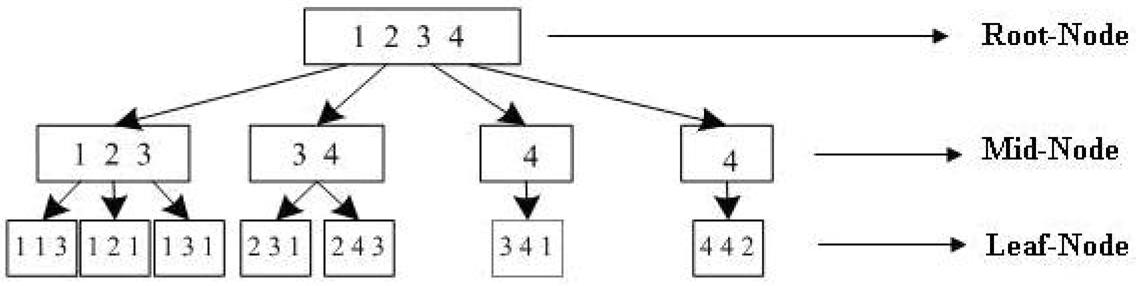


Figure 3: An example of the CDS-Trees

A CDS-Tree consists of 3 kinds of nodes:

- Root-Node:

The root of the CDS-Tree is to represent **all indexes of dimension $\mathbf{A_1}$** in ascending order.

- Leaf-Node:

  A Leaf-Node is to represent a non-zero NoP cell with three information, **$\mathbf{A_1}$_index, $\mathbf{A_2}$_index**, and **NoP**.

- Mid-Node:

  A Mid-Node is an intermediate node between the root node and leaf nodes which is to represent all **the distinct indexes of the dimension $\mathbf{A_2}$** in ascending order for all non-zero NoP cells with respected to each index of the dimension A_1.

Only those coordinates of non-zero NoPs would be inserted into CDS-Tree. Mid nodes and root node are constructed based on the inserted leaf nodes.

Finally your program should print out all CDS-Trees using the read-in order of the corresponding data sets. All printouts of the CDS-Trees are separated by a line blank. For each CDS-Tree, your program should use the level order to print out the Root-Node, all Mid-Nodes, and all Leaf-Nodes in sequence one node per line consecutively as the Sample Output. For each node your program should separate the indexes of the dimension or the NoP by one space blank.

Sample Input

```
4
4 3 1 2
2 1 2 5
0 1 0 1 1 1 1 1 1 0
1 1 1 0 0 1 0 0 0 1
```

Sample Output

```
1 2 3 4
1 2 3
3 4
4
```

```
4
1 1 3
1 2 1
1 3 1
2 3 1
2 4 3
3 4 1
4 4 2
```

Reference

[1] Huanliang Sun, Ge Yu,Yubin Bao, Faxin Zhao, Daling Wang. CDS-Tree: An Effective Index for Clustering Arbitrary Shapes in Data Streams. Proceedings of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications.