Can we create a table of integers?

```
const int ROWS = 2;
const int COLS = 5;
int main() {

  int array[ROWS * COLS] =   // initialize
    { 10,  20,  30,  40,  50,
      60,  70,  80,  90, 100  };

  array[(1*COLS) + 2] = 85;  // access the element at
                             // row = 1 and column = 2
}
```

A more convenient way

```
const int ROWS = 2;
const int COLS = 5;
int main() {

  int array[ROWS * COLS] =   // initialize
    { 10,  20,  30,  40,  50,
      60,  70,  80,  90, 100  };

  array[(1*COLS) + 2] = 85;  // access the element at
                             // row = 1 and column = 2
  // 2 dimensional array
  int array2d[ROWS][COLS] =
    { { 10,  20,  30,  40,  50 },
      { 60,  70,  80,  90, 100 }  };

  array2d[1][2] = 85;
}
```

Examples of two-dimensional arrays

```
const int HEIGHT = 3;
const int WIDTH = 4;
int main() {

  int a1[128][256]; // 2 dimensional array

  double a2[2][5] = // initialized with values
    { { 0.1,   3.1,  4.2,   1.1,   3.9 },
      { 7.7,  1.23,  5.3,  6.81,  12.5 } };

  int a3[HEIGHT][WIDTH] = // another example
    {  {   1,    4,    8,   16 },
       {  32,   64,  128,  265 },
       { 512, 1024, 2048, 4096 }  };

  int a4[220][340] = {0}; // all are zeroes
}
```

Using two-dimensional arrays

```
#include <iostream>
int main() {
  int table[15][25];
  // assign values:
  for(int i = 0; i < 15; i++) {
    for(int j = 0; j < 25; j++) {
      table[i][j] = (i + j) % 2;
    }
  }
  // print the table!
  for(int i = 0; i < 15; i++) {
    for(int j = 0; j < 25; j++) {
      std::cout << table[i][j] << ' ';
    }
    std::cout << std::endl;
  }
}
```

Multidimensional arrays

```
int main() {

  int arr[128][256][512] = {0}; // 3 dimensional array

  // assign some values:
  for(int i = 0; i < 128; i++) {
    for(int j = 0; j < 256; j++) {
      for(int k = 0; k < 512; k++) {

        arr[i][j][k] = 1024 + i;

      }
    }
  }
  // we had to use 3 nested loops to go through each element

}
```

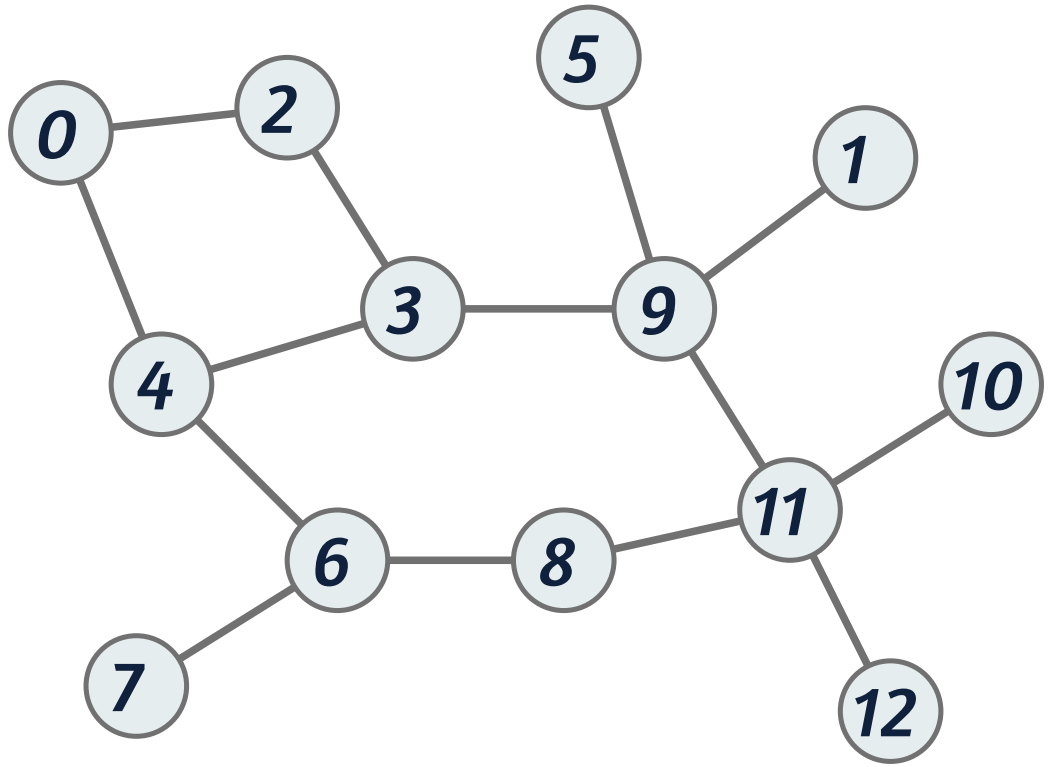
Compute the total cost of the list of items

```
// prices and quantities
int items [N][2] = {
    { 150,  1 },
    { 299,  1 },
    {  10,  5 },
    { 115,  2 },
    {  85,  0 },
    { 449,  0 },
    { 275, 12 },
    { 185,  8 }
};
```

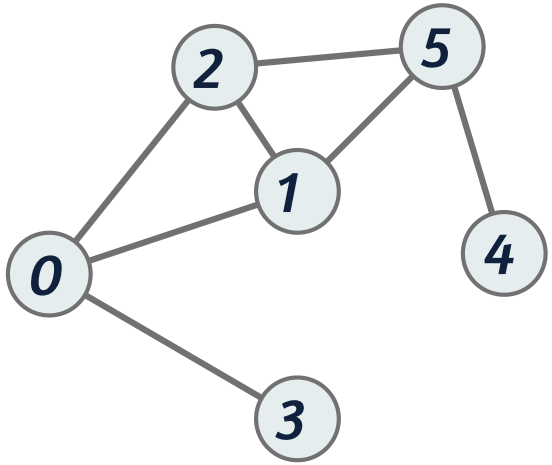
Question: How to compute the total cost of the items?

```
int main() {
    const int N = 8;
    // prices and quantities
    int items [N][2] = {
        { 150,  1 },
        { 299,  1 },
        {  10,  5 },
        { 115,  2 },
        {  85,  0 },
        { 449,  0 },
        { 275, 12 },
        { 185,  8 }
    };
    int tot = 0;
    for (int i = 0; i < N; i++) {
        tot = tot + items[i][0] * items[i][1];
    }
}
```

N cities are connected by roads



Question: Can we somehow use arrays to store the information about the connections between the cities?



Adjacency matrix

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

```
const int N = 6;
int adjacent [N][N] = {
 { 0, 1, 1, 1, 0, 0 },
 { 1, 0, 1, 0, 0, 1 },
 { 1, 1, 0, 0, 0, 1 },
 { 1, 0, 0, 0, 0, 0 },
 { 0, 0, 0, 0, 0, 1 },
 { 0, 1, 1, 0, 1, 0 },
};
```

N cities are connected by roads

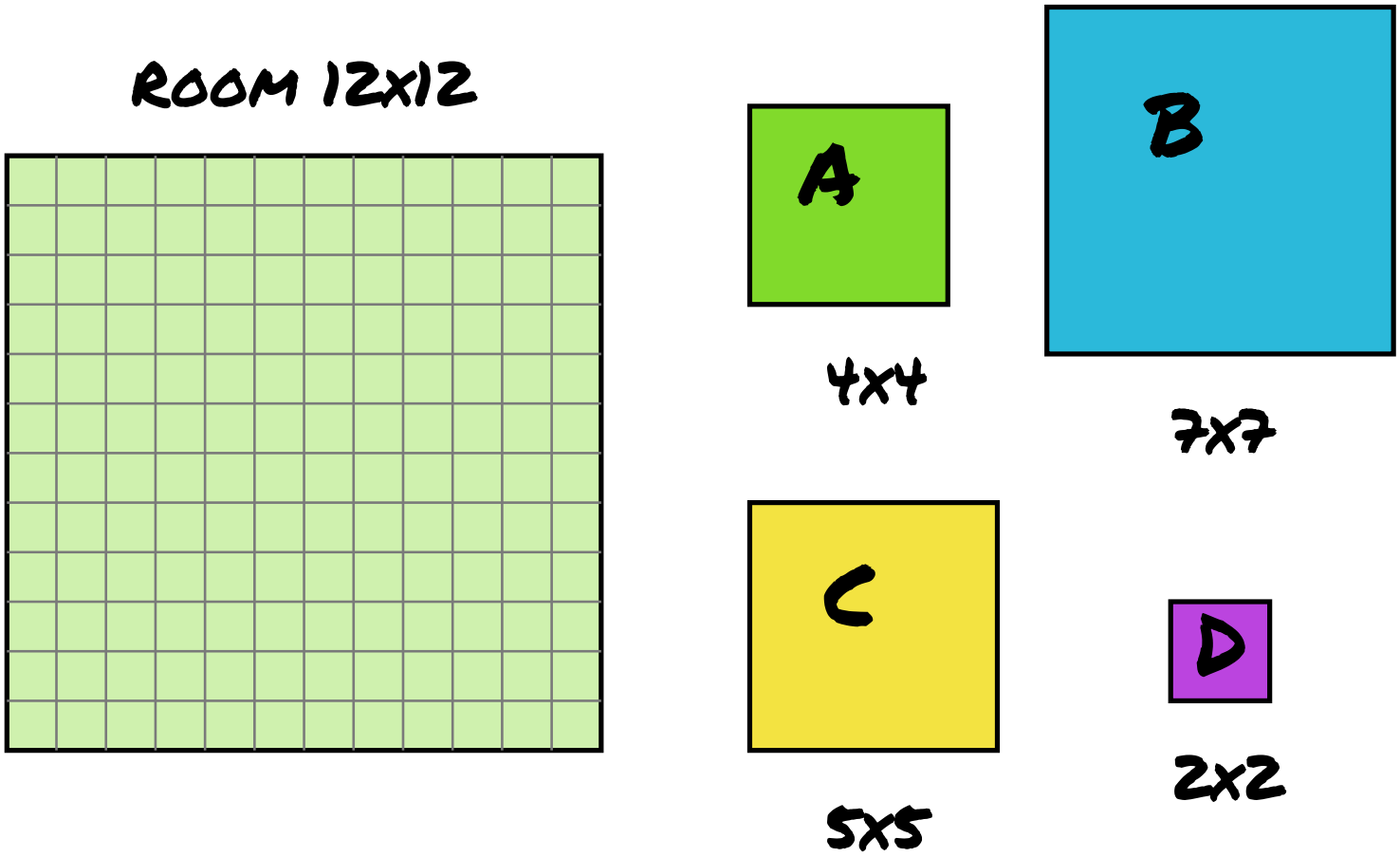
Question 1: How to find all the cities that are reachable from the city 0 in one hop?

N cities are connected by roads

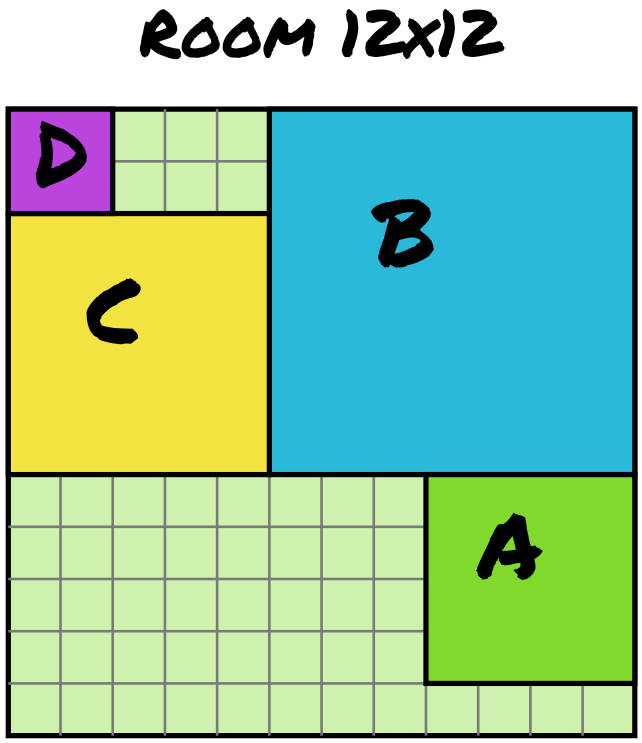
Question 1: How to find all the cities that are reachable from the city 0 in one hop?

Question 2: In at most two hops?

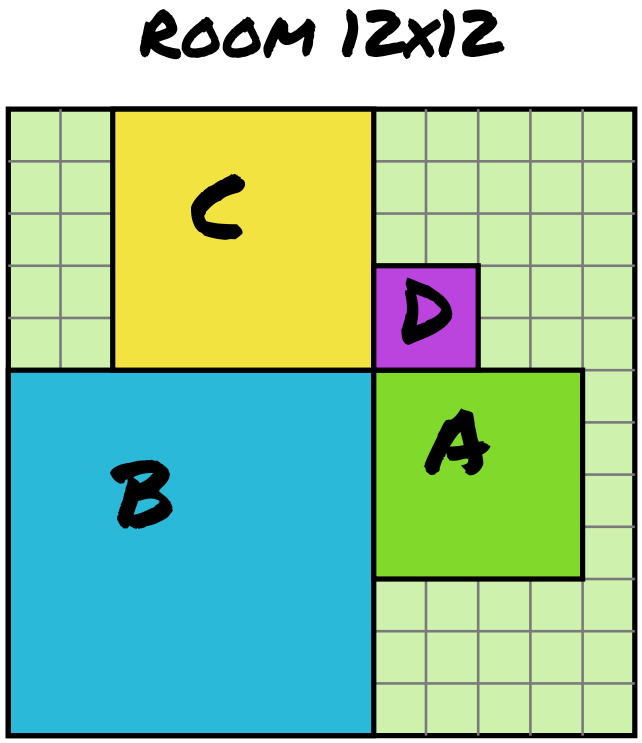
Packing square boxes



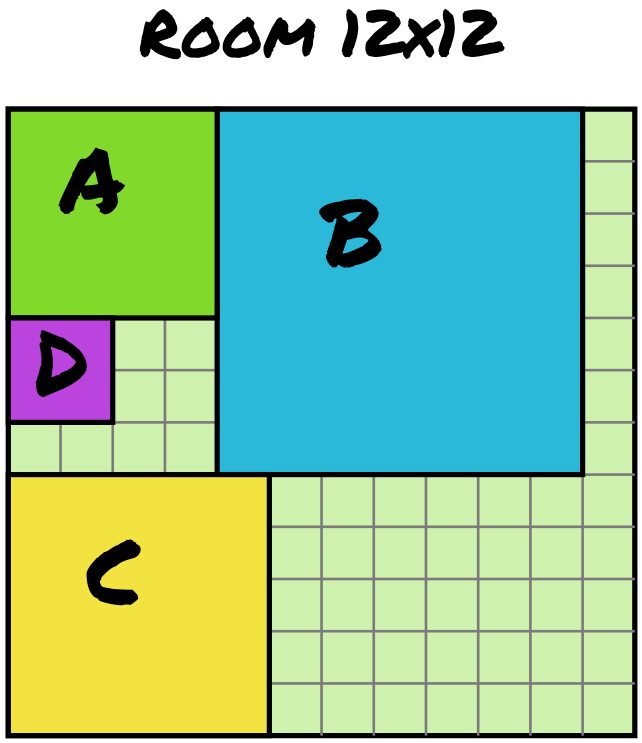
Packing square boxes



Packing square boxes

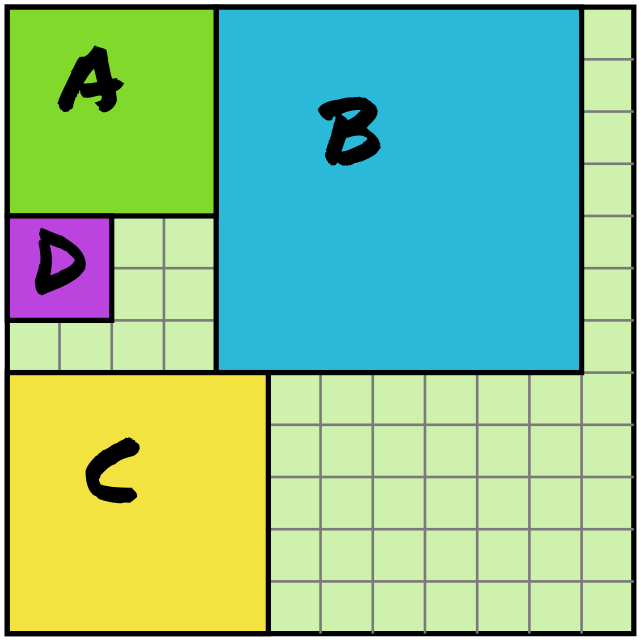


Packing square boxes



Packing square boxes

Room 12x12



Program Output:

A	A	A	A	B	B	B	B	B	B	B	
A	A	A	A	B	B	B	B	B	B	B	
A	A	A	A	B	B	B	B	B	B	B	
A	A	A	A	B	B	B	B	B	B	B	
D	D			B	B	B	B	B	B	B	
D	D			B	B	B	B	B	B	B	
				B	B	B	B	B	B	B	
C	C	C	C	C							
C	C	C	C	C							
C	C	C	C	C							
C	C	C	C	C							
C	C	C	C	C							

Packing square boxes

```
// given four square boxes
const int BOXES = 4;
int size[BOXES]   = {4, 7, 5, 2};          // box size
char label[BOXES] = {'A', 'B', 'C', 'D'}; // label

// put them in the room 12x12
const int ROWS = 12;
const int COLS = 12;
char room[ROWS][COLS];
```

The goal: To come up with a program that tries to pack the room with the given square boxes.

Example output:

```
A A A A . B B B B B B B
A A A A . B B B B B B B
A A A A . B B B B B B B
A A A A . B B B B B B B
. . . . . B B B B B B B
. . . . . B B B B B B B
C C C C C B B B B B B B
C C C C C . . . . . . .
C C C C C . D D . . . .
C C C C C . D D . . . .
C C C C C . . . . . . .
. . . . . . . . . . . .
```