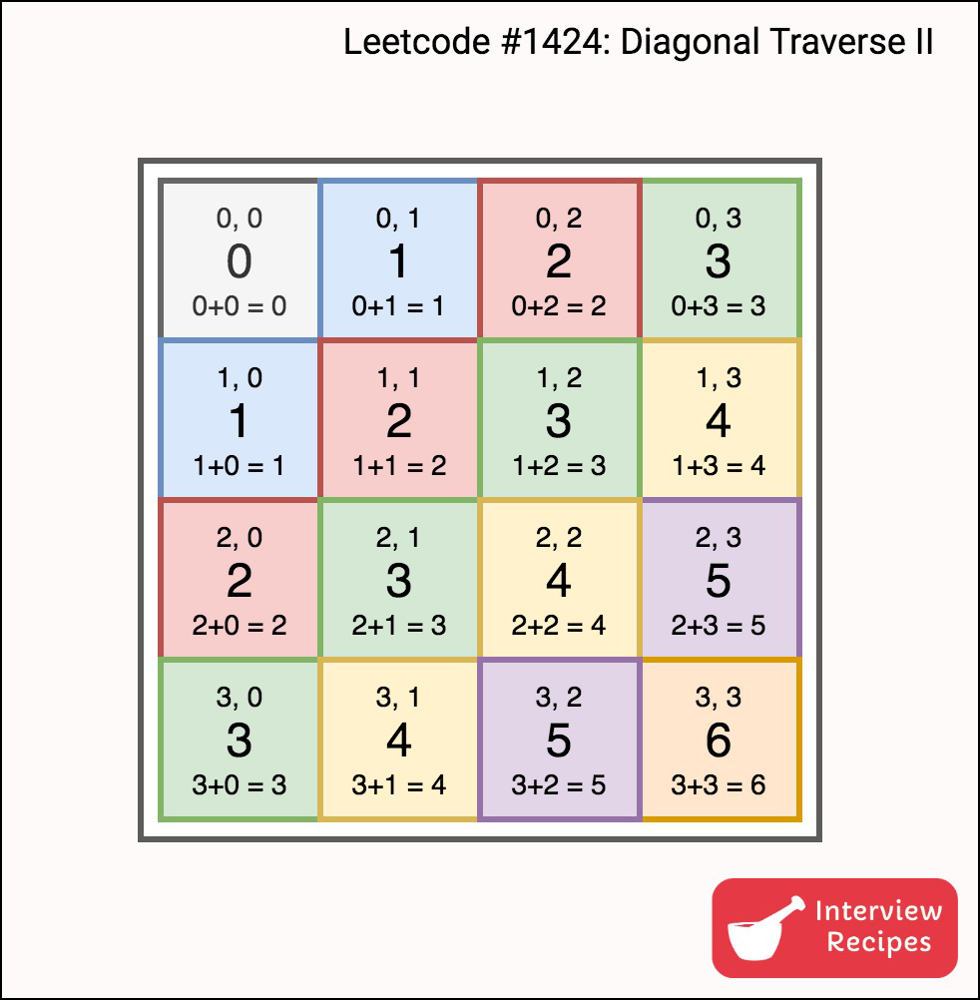
[**https://leetcode.com/problems/diagonal-traverse-ii/discuss/597741/Clean-Simple-Easiest-Explanation-with-a-picture-and-code-with-comments**](https://leetcode.com/problems/diagonal-traverse-ii/discuss/597741/Clean-Simple-Easiest-Explanation-with-a-picture-and-code-with-comments)

**Key Idea**  
In a 2D matrix, elements in the same diagonal have same sum of their indices.



So if we have all elements with same sum of their indices together, then it’s just a matter of printing those elements in order.

**Algorithm**

1. Insert all elements into an appropriate bucket i.e. nums[i][j] in (i+j)th bucket.
2. For each bucket starting from 0 to max, print all elements in the bucket.  
   **Note**: Here, diagonals are from bottom to top, but we traversed the input matrix from first row to last row. Hence we need to print the elements in reverse order.

Please upvote if you liked the idea, it would be encouraging. If you like this, checkout my blog/website for posts on coding interviews, programming, data structures and algorithms, problem solving.

**C++**

vector<int> findDiagonalOrder(vector<vector<int>>& nums) {

vector<int> answer;

unordered\_map<int, vector<int>> m;

int maxKey = 0; // maximum key inserted into the map i.e. max value of i+j indices.

for (int i=0; i<nums.size(); i++) {

for (int j=0; j<nums[i].size(); j++) {

m[i+j].push\_back(nums[i][j]); // insert nums[i][j] in bucket (i+j).

maxKey = max(maxKey, i+j); //

}

}

for (int i=0; i<= maxKey; i++) { // Each diagonal starting with sum 0 to sum maxKey.

for (auto x = m[i].rbegin(); x != m[i].rend(); x++) { // print in reverse order.

answer.push\_back(\*x);

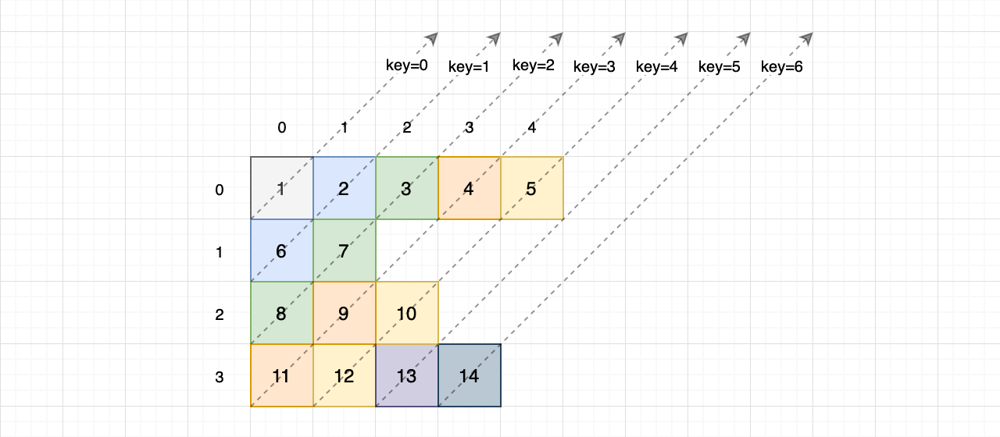
}

}

return answer;

}

<https://leetcode.com/problems/diagonal-traverse-ii/discuss/597698/JavaC%2B%2B-HashMap-with-Picture-Clean-code-O(N)>

Example:  


**Java**

class Solution {

public int[] findDiagonalOrder(List<List<Integer>> nums) {

int n = 0, i = 0, maxKey = 0;

Map<Integer, List<Integer>> map = new HashMap<>();

for (int r = nums.size() - 1; r >= 0; --r) { // The values from the bottom rows are the starting values of the diagonals.

for (int c = 0; c < nums.get(r).size(); ++c) {

map.putIfAbsent(r + c, new ArrayList<>());

map.get(r + c).add(nums.get(r).get(c));

maxKey = Math.max(maxKey, r + c);

n++;

}

}

int[] ans = new int[n];

for (int key = 0; key <= maxKey; ++key) {

List<Integer> value = map.get(key);

if (value == null) continue;

for (int v : value) ans[i++] = v;

}

return ans;

}

}