Leetcode 200: islands

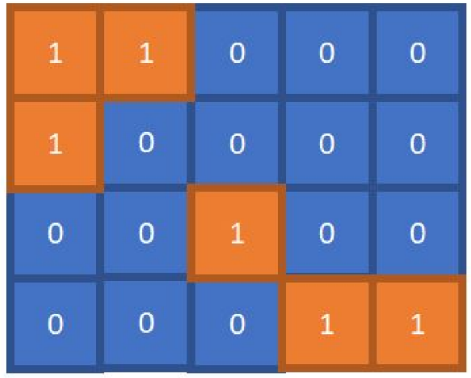
Banin, M.; Kochanova, K.; Skrebkov, A.; Vodopyanov, D.

January 18, 2018

Number of Islands

Task: Given a 2d grid map of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example: given matrix



A number of islands is 3.

Capturing state in lambdas

```
function<int(int, int)> sink =
    [&sink, &grid](int i, int j){
        if (i < 0 || i >= grid.size() ||
            j < 0 || j >= grid[0].size() ||
            grid[i][j] == '0') return 0;

            grid[i][j] = '0';
            sink(i + 1, j); sink(i - 1, j);
            sink(i, j + 1); sink(i, j - 1);
            return 1;
    };

int islands = 0;
for (int i = 0; i < grid.size(); i++)
    for (int j = 0; j < grid[0].size(); j++)
        islands += sink(i, j);
return islands;
```

Disjoint set union

```
parent = vector<size_t>(
            (grid.size() + 1) * (grid[0].size() + 1));
iota(parent.begin(), parent.end(), 0);
grid.push_back(vector<char>(grid[0].size() + 1, '0'));

for(size_t i = 0; i < grid.size() - 1; ++i){
  grid[i].push_back('0');
  for(size_t j = 0; j < grid[i].size(); ++j){
    if(grid[i][j] == '1' && grid[i + 1][j] == '1')
      union_sets(ix(i, j), ix(i + 1 , j));
    if(grid[i][j] == '1' && grid[i][j + 1] == '1')
      union_sets(ix(i, j), ix(i, j + 1));
  }
}
```

Number of Islands: BFS with std::queue

- Time complexity: O(mn).
- Create a queue of coordinates of ‘1’ elements.
- In the loop: add coordinates of edges to the queue, if edge is ‘1’. Mark visited elements as ‘x’ and remove their coordinates from the queue.

```
queue<pair<int, int>> q;
q.emplace(x, y);
while (!q.empty()) {
        int i = q.front().first;
        int j = q.front().second;
        q.pop();
        if (grid[i][j] != visited_symbol) {
        grid[i][j] = visited_symbol;
        if (i + 1 < width && grid[i + 1][j] == '1’)
        q.emplace(i + 1, j);
        if (j + 1 < height && grid[i][j + 1] == '1’)
        q.emplace(i, j + 1);
        ...
```

Asymptotic performance

	Runtime
DFS	$O(n)$
BFS	$O(n)$
DSU	$O(n\alpha(n)) - O(n \log n)$

where $\alpha(n)$ is an inverse Ackerman function.

DSU in $O(\log n)$

```
size_t find_parent(size_t v){
  if (v == parent[v])
    return v;
  return parent[v] = find_parent(parent[v]);
};

void union_sets (int a, int b) {
  a = find_parent (a);
  b = find_parent (b);
  if (a == b) return;
  if(rand() & 1)
    parent[b] = a;
  else
    parent[a] = b;
};
```