

Francis' answer to the Alien Invasion Problem

How to approach

One of the approaches to solving this problem is to assume that the events of destroying cities and killing aliens happen separately. That is, the problem can be divided by two steps following:

- Step1: At the same time, all alive aliens wander around the cities as defined in the problem.
- Step2: Then, their moving are evaluated (destroyed and killed) simultaneously as defined.

This assumption is very important. If each alien wandered individually at a different time and those destructive events happened separately, this problem would have become much more complex.

With the help of this assumption, I just need to introduce only two basic data structure.

```
1 # world map info
2 world_map := {k: v}, where k: "city" and v: "route info of each city"
3             v = {p: q}, where p: "direction such as 'north', 'south', ..."
4             q: "the adjacent cities connected with roads"
5
6 # aliens' position info: tracking aliens every turn
7 aliens_spot := {k: v}, where k: "city where aliens are invading"
8             v: "alien identification" in terms of v = set(alien_id)
9
```

I avoided OOP style coding as it can be solved with a simple data structure introduced above.
For more detail, please see the commented code below.

Note1: Individually Implemented using two languages, both *Go*, and *Python*. It seems a little funny as almost the same algorithm was used in each. Please kindly consider them.

Note2: Based on the Christopher's answer (**Q:** *Does "kill and destroy" happen when the number of aliens is exactly equal to two? A:* *"kill and destroy" happens with two or more aliens.*), every city with two or more aliens was destroyed and all aliens in it were killed, when evaluating each turn right after moving of aliens. Printing out event messages is closely connected to this fundamental definition.

How to run

```
1 # To run with other map files, modify the variable "FILE_MAP" in the src file
2
3 ### Go
4 $ cd go
5
6 $ go build alien_invasion.go
7 $ ./alien_invasion NUM_OF_ALIENS
8 # or
9 $ go run alien_invasion.go NUM_OF_ALIENS
10
11 # simple unit test
12 $ go test -v
13
14 ### Python
15 $ cd python
16
17 # Python 3.6 or later is recommended
18 $ python3 alien_invasion.py NUM_OF_ALIENS
```

Some comments with code (based on Go implementation)

```
1  /*
2  Note: here assumed that ONLY VALID DATA will be given as a map file.
3  In order to focus on the algorithm itself, I skipped all validations of
4  input files and command-line arguments.
5  */
6  package main
7
8  import (
9      "fmt"
10     "bufio"
11     "os"
12     "strings"
13     "strconv"
14     "time"
15     "math/rand"
16 )
17
18 type Roads map[string]string
19 type WorldMap map[string]Roads
20 type Intruders map[int]bool
21 type AlienSpot map[string]Intruders
22
23 func parse_world_map(mapfile string) WorldMap {
24     f, err := os.Open(mapfile)
25     if err != nil { panic(err) }
26     defer f.Close()
27
28     world_map := make(WorldMap)
29     scanner := bufio.NewScanner(f)
30     for scanner.Scan() {
31         line := strings.TrimSpace(scanner.Text())
32         // skipping blank line in the world map file
33         if len(line) == 0 { continue }
34         tokens := strings.Fields(line)
35         city, roads := tokens[0], tokens[1:]
36         world_map[city] = make(Roads)
37         for _, road := range roads {
38             split := strings.Split(road, "=")
39             direction, dest := split[0], split[1]
40             world_map[city][direction] = dest
41         }
42     }
43     return world_map
44 }
45
46 func dump_world_map(world_map WorldMap) {
47     fmt.Println("\n----- THE REMAINING WORLD -----")
48     if len(world_map) == 0 { fmt.Println("All cities are destroyed.") }
49     for city, roads := range world_map {
50         roads_string := ""
51         for direction, dest := range roads {
52             roads_string += fmt.Sprintf(" %s=%s", direction, dest)
53         }
54         fmt.Printf("%s%s\n", city, roads_string)
55     }
56 }
57
58 func dump_destroy_event(city string, intruders Intruders) {
59     ids := make([]string, 0, len(intruders))
60     for id, _ := range intruders { ids = append(ids, fmt.Sprintf("alien %d", id)) }
```

```

61     destroyer := strings.Join(ids[:len(ids)-1], ", ")
62     destroyer += fmt.Sprintf(" and %s", ids[len(ids)-1])
63     fmt.Printf("%s has been destroyed by %s!\n", city, destroyer)
64 }
65
66 func init_alien_invasion(num_aliens int, world_map WorldMap) AlienSpot {
67     rand.Seed(time.Now().Unix())
68     alien_spot := make(AlienSpot)
69     if len(world_map) == 0 { return alien_spot }
70     cities := make([]string, 0, len(world_map))
71     for k := range world_map { cities = append(cities, k) }
72     for id := 1; id <= num_aliens; id++ {
73         invaded := cities[rand.Intn(len(cities))]
74         move_alien_into_city(id, invaded, alien_spot)
75     }
76     return alien_spot
77 }
78
79 func move_alien_from_city(alien_id int, city string, alien_spot AlienSpot) {
80     delete(alien_spot[city], alien_id)
81     if len(alien_spot[city]) == 0 { delete(alien_spot, city) }
82 }
83
84 func move_alien_into_city(alien_id int, city string, alien_spot AlienSpot) {
85     _, ok := alien_spot[city]
86     if ok {
87         alien_spot[city][alien_id] = true
88     } else {
89         alien_spot[city] = Intruders{alien_id: true}
90     }
91 }
92
93 func wander_randomly(world_map WorldMap, alien_spot AlienSpot) {
94     for from_city, intruders := range alien_spot {
95         // No wandering when a alien gets trapped
96         if _, ok := world_map[from_city]; !ok { continue }
97
98         cities := make([]string, 0, len(world_map[from_city]))
99         for _, v := range world_map[from_city] { cities = append(cities, v) }
100        if len(cities) == 0 { continue }
101        for alien_id, _ := range intruders {
102            // random selection based on roads connected to cities
103            into_city := cities[rand.Intn(len(cities))]
104            // pull aliens out of an existing location
105            move_alien_from_city(alien_id, from_city, alien_spot)
106            // place aliens into the newly selected location
107            move_alien_into_city(alien_id, into_city, alien_spot)
108        }
109    }
110 }
111
112 func destroy_and_kill(world_map WorldMap, alien_spot AlienSpot) {
113     for city, intruders := range alien_spot {
114         // if the number of aliens at the same place is less than 2, do nothing
115         if len(intruders) < 2 { continue }
116
117         // otherwise, destroy the city and kill aliens in it
118         for _, vicinity := range world_map[city] {
119             // finding all related routes and destroy it
120             for direction, c := range world_map[vicinity] {
121                 if c == city { delete(world_map[vicinity], direction) }
122             }
123         }

```

```

124         // remove the city from world map and from the alien tracker
125         delete(world_map, city)
126         delete(alien_spot, city)
127
128         // print out an event log based on the problem definition
129         dump_destroy_event(city, intruders)
130     }
131 }
132
133 func main() {
134     if len(os.Args) != 2 {
135         fmt.Fprintf(os.Stderr, "Usage: %s <num of aliens>\n", os.Args[0])
136         return
137     }
138     const FILE_MAP = "../worldmap/universe.txt"
139     const MAX_MOVES_ALIENS = 10000
140     num_aliens, _ := strconv.Atoi(os.Args[1])
141
142     // parse the map file given
143     world_map := parse_world_map(FILE_MAP)
144
145     // initialize the aliens position
146     alien_spot := init_alien_invasion(num_aliens, world_map)
147     destroy_and_kill(world_map, alien_spot)
148
149     moves := 0
150     for {
151         // exit conditions
152         if moves > MAX_MOVES_ALIENS { break }
153         if len(world_map) == 0 { break }
154         if len(alien_spot) == 0 { break }
155
156         // each alien wanders randomly each turn
157         wander_randomly(world_map, alien_spot)
158
159         // update world map and aliens' position by definition
160         // destroy cities and kill aliens if needed.
161         destroy_and_kill(world_map, alien_spot)
162         moves++
163     }
164     dump_world_map(world_map)
165 }

```