UCSC Silicon Valley Extension

C Programming, Advanced

Assignment 1 : Recursion and time complexity
Instructor : Radhika Grover

Provide the source code and output for all programming problems and test your programs on the test cases provided in the folder *Test cases for Assignment 1***.** *The sample output is also provided for the programming problems.*

5. **Time Complexity Analysis and Algorithm Design**

1. Given an array of *n* integers and an integer *S*, write an algorithm that only returns true if the array contains two numbers whose sum is *S*.

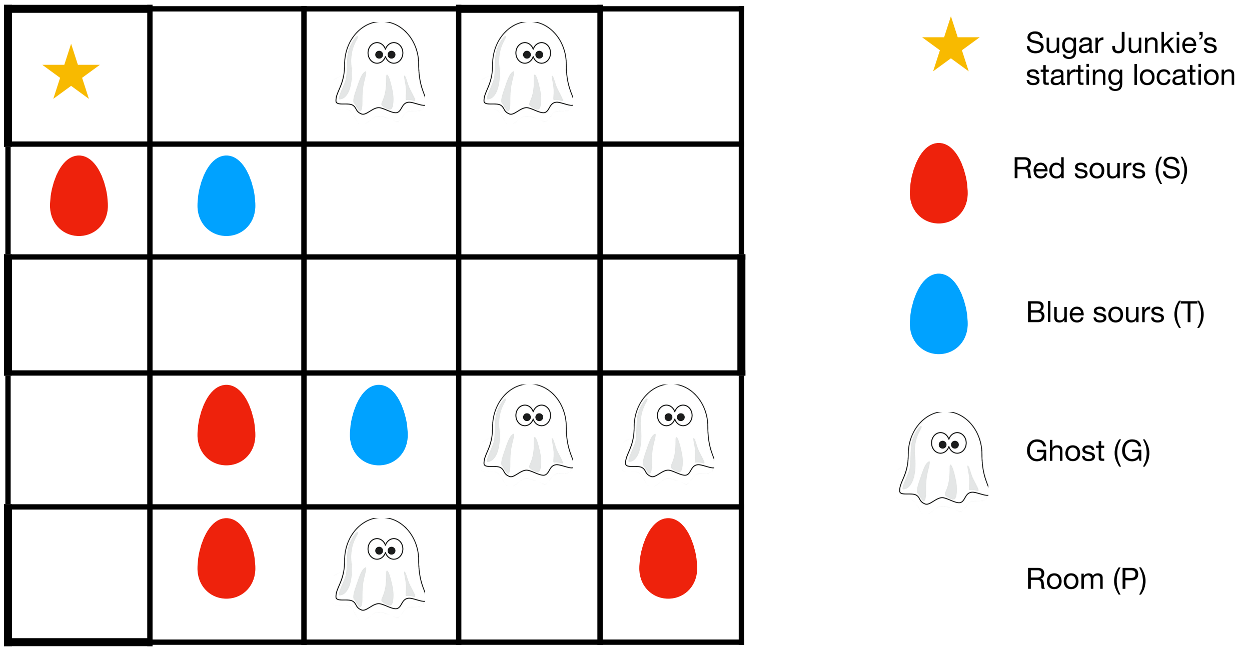
(a) Find the recurrence relation of an algorithm that uses an exhaustive search and solve it to find the (best-case and worst-case) running time of the algorithm.

(b) Can you find a better algorithm to solve the problem? What is the time complexity (best case and worst case) of the algorithm? What is the space complexity of the algorithm?

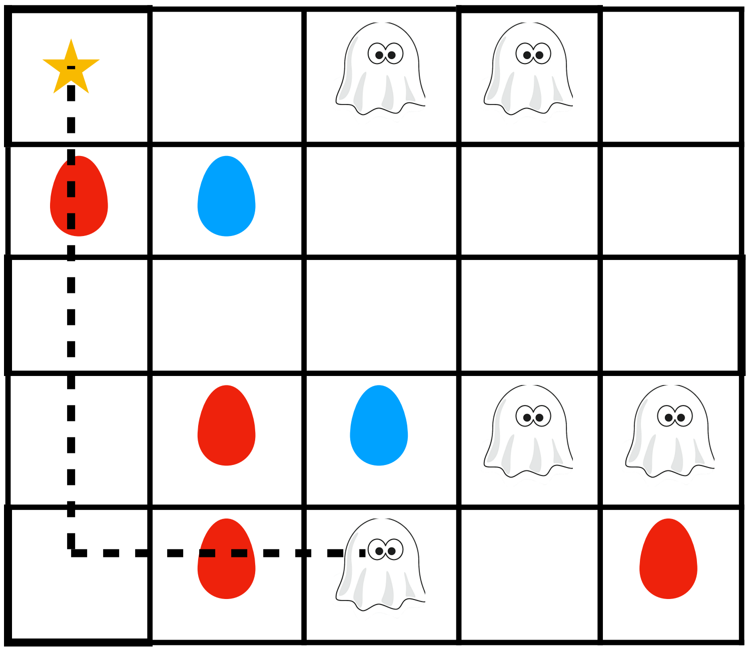
(c) Implement both algorithms and determine the execution times of the corresponding program for different input sizes with randomized inputs. Plot the values on a graph and find the equation of the curve that fits the data points. Do the results agree with the theoretical analysis of part (c)?

6. Halloween Candy Binge

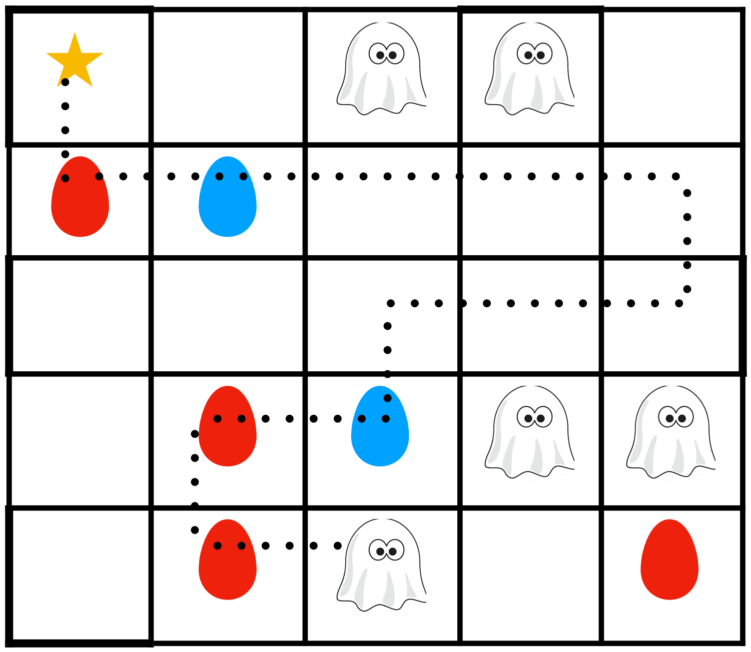
Help Sugar Junkie eat candy to score the maximum number of points! Each room is shown with a square in the grid. A room may contain a packet of red or blue sour candy, and Sugar Junkie gets 2 points on eating a packet of red sours and 3 points on eating a packet of blue sours. Moves are in the N, S, E, and W directions along adjacent rooms called a path. Some rooms contain ghosts and Sugar Junkie freezes upon reaching a room with a ghost. Note that Sugar Junkie cannot visit any room more than once on a given path. Given a starting location, determine the maximum number of points on any path collected until a room with a ghost is reached.



For example, see two possible paths in the next two figures. Sugar Junkie is initially at (0,0) The dashed path goes through two rooms with red sours before a ghost is reached, so the total points collected are 4.



The dotted path below goes through 3 red and 2 blue sours, so the total points are this path are 12. This is the maximum number of points that can be collected through any path from the given start point, and is displayed as the solution. Sugar Junkie cannot revisit any of the rooms on this path.   So, the path (0,0), (1,0), (2,0), (1,0) is invalid because (1,0) is revisited.



Format of input file (halloween_candy_binge_input.txt):

5 #Number of test cases
5 5 #rows columns (grid size)
0 0 #starting (x, y) coordinates of Sugar Junkie
P P T G P #Elements of the grid (P = empty room, G = ghost, S = red sour, T = blue sour)
T S P P P
P P P P P
P T S T T
P T T P T

7. VIP Riders

You are writing a carpool scheduling application that selects a set of riders for a driver based on their distance from the driver and the cost of the ride. Each rider is at a distance *d* miles from the driver and will pay *c* dollars for the ride. The application selects a set of riders so that their total distance from the driver is not greater than some specified limit of *D* miles with the goal of maximizing the total revenue for the ride. The program should display the revenue for the schedule.

For example, there are four riders with the following *d* and *c* values:

The limit *D* is 10 miles. Then the carpool application can select the riders with (*d*, *c*) values (6, 50) , (3, 20) and (1, 7). The total revenue is 50 + 20 + 7 = 77.

(b) Modify your program in (a) to meet a limit on the maximum number of riders that can be selected for the carpool.

Suppose that the maximum number of riders is limited to 2. Then, the application can select (6, 50) and (3, 20). The total revenue is 70.

(c) Some riders are marked as VIP riders and must be the only rider in the carpool. Modify the algorithm in (b) so that it can find the maximum revenue by choosing either a VIP rider or a group of non-VIP riders.

(d) Determine the execution time for each test case. What is the running time of your algorithm?

The input file for VIP_riders.txt contains the following data:

```
5 # number of test cases
10 # max distance D
2 # max number of riders
5 # number of riders
6  50 0 # d=6  c=50 isVIP = 0 for rider 1
5 100 1 # d=6  c=50 isVIP = 1 for rider 2
3 30 0 # d=6  c=50 isVIP = 0 for rider 3
1 40 0 # d=6  c=50 isVIP = 0 for rider 4
4 90 0 # d=6  c=50 isVIP = 0 for rider 5
```