Project 03: Golomb Ruler

Contents

1 Description

1.1 Strategy

To find the optimal solution, we have made use of the fact that the distance between any 2 pairs of markers should be unique. The maximum number of combinations of 2 markers possible out of M markers is M C 2, i.e. M * (M - 1) / 2. As each of these values has to be unique, the minimum length of the ruler has to be M C 2.
Hence, if L < M C 2, we return -1, [ ] as no solution exists.
Else, for all values from M C 2 to L, we search for a solution. If a solution is found, we return the length and positions.
If all such values are exhausted and no solution is obtained, we return -1, [ ].

1.2 Backtracking

If all the positions are assigned then the length and the corresponding positions are returned. If not, every marker is assigned a position from its domain and is checked for consistency. Consistency includes, checking if the position is already allocated to some other marker and the distance between the marker and previously assigned markers is unique. If it is not consistent and it's domain is exhausted then it is backtracked to change the position of the previous marker.

1.3 Forward Checking

If all the positions are assigned then the length and the corresponding positions are returned. If domain of any of the marker is empty then it is backtracked to assign different position to the previous marker. If not, every marker is assigned a position from its domain and is checked for consistency. Consistency includes, the distance between the marker and previously assigned markers should be unique. If it is not consistent different position is assigned to the current marker and again checked for consistency. Now if, it becomes consistent, that position is removed from the domains of future markers. Also, position from the domains of the markers are pruned using the following constraints.

1. Remove the positions from the domains of all markers where positions = allocated positions + distances iff positions <= L

2. Remove the positions from the domains of all markers where positions = allocated positions - distances iff positions >= 0

2 Statistics

2.1 Test Cases - Set 1

All (L, M) pairs given as input are such that L is the optimal length for number of markers M.

L	M	BT		FC		Output
		Nodes Expanded	Time (seconds)	Nodes Expanded	Time (seconds)	(length, positions)
6	4	6	0.012	5	0.016	`(6, {1: 0, 2: 1, 3: 4, 4: 6})`
11	5	3615	0.076	1354	0.076	`(11, {1: 0, 2: 1, 3: 4, 4: 9, 5: 11})`
17	6	170294	3.212	58462	4.244	`(17, {1: 0, 2: 1, 3: 4, 4: 10, 5: 12, 6: 17})`
25	7	10095777	318.326	4064085	350.380	`(25, {1: 0, 2: 1, 3: 4, 4: 10, 5: 18, 6: 23, 7: 25})`

2.2 Test Cases - Set 2

All (L, M) pairs given as input are such that L is lesser than the optimal length for number of markers M.

L	M	BT		FC		Output
		Nodes Expanded	Time (seconds)	Nodes Expanded	Time (seconds)	(length, positions)
5	4	0	0.024	0	0.008	`(-1, [])`
4	4	0	0.004	0	0.012	`(-1, [])`
10	5	3602	0.056	1346	0.072	`(-1, [])`
8	5	0	0.012	0	0.016	`(-1, [])`
16	6	170227	3.220	58435	4.108	`(-1, [])`
10	6	0	0.008	0	0.012	`(-1, [])`
24	7	10094986	659.529	4063786	357.212	`(-1, [])`
15	7	0	0.012	0	0.016	`(-1, [])`

2.3 Test Cases - Set 3

All (L, M) pairs given as input are such that L is greater than the optimal length for number of markers M.

L	M	BT		FC		Output
		Nodes Expanded	Time (seconds)	Nodes Expanded	Time (seconds)	(length, positions)
7	4	6	0.014	5	0.016	`(6, {1: 0, 2: 1, 3: 4, 4: 6})`
9	4	6	0.010	5	0.012	`(6, {1: 0, 2: 1, 3: 4, 4: 6})`
12	5	3615	0.060	1354	0.076	`(11, {1: 0, 2: 1, 3: 4, 4: 9, 5: 11})`
14	5	3615	0.076	1354	0.084	`(11, {1: 0, 2: 1, 3: 4, 4: 9, 5: 11})`
18	6	170294	3.526	58462	4.324	`(17, {1: 0, 2: 1, 3: 4, 4: 10, 5: 12, 6: 17})`
20	6	170294	3.245	58462	4.548	`(17, {1: 0, 2: 1,`

						3: 4, 4: 10, 5: 12, 6: 17})
26	7	10095777	318.456	4064085	364.508	(25, {1: 0, 2: 1, 3: 4, 4: 10, 5: 18, 6: 23, 7: 25})
28	7	10095777	325.154	4064085	375.836	(25, {1: 0, 2: 1, 3: 4, 4: 10, 5: 18, 6: 23, 7: 25})

3 Analysis

1. The number of nodes expanded increases exponentially for backtracking as well as forward checking.
2. Here, it can be observed that when an optimal solution does not exist, forward checking fails early, resulting in consistently lesser number of nodes expanded than plain backtracking.
3. However, due to the added commands for pruning, forward checking requires more time in seconds.
4. In addition, if L is lesser than M C 2, the search tree is not constructed at all, as explained in the strategy. Hence, it returns in constant time.