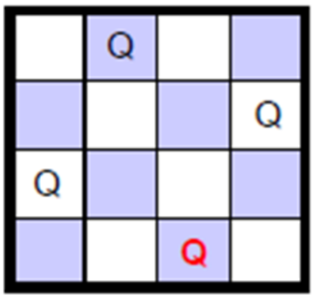
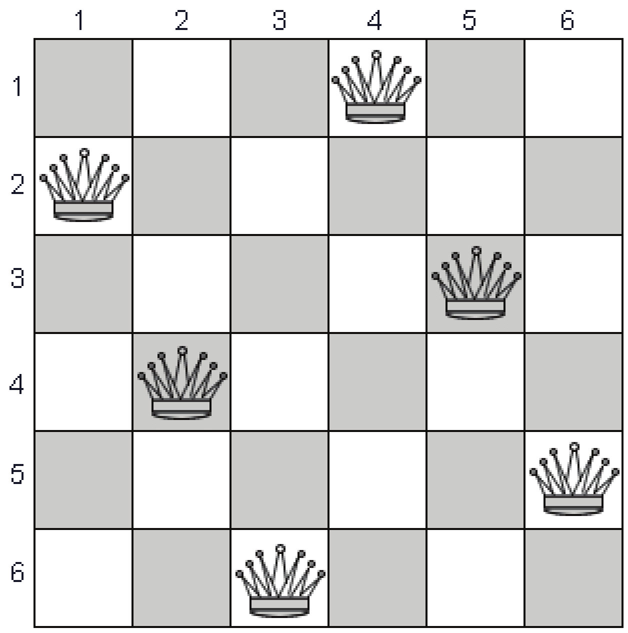
Tutorial 5

Question 1

Using the n-Queens problem we have discussed in class, try to find a backtracking solution if n=6. Clearly indicate the backtracking points.



Answer:

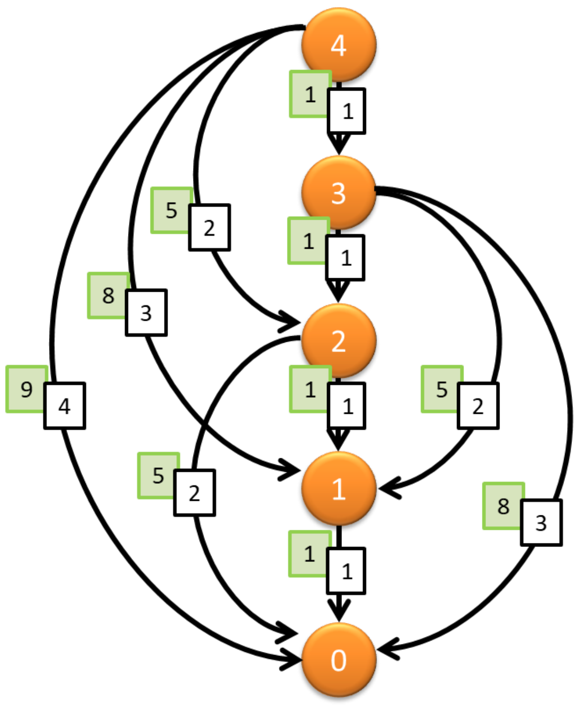


https://www.researchgate.net/figure/A-solution-of-the-6-queens-problem-As-an-example-let-us-consider-the-6-queens-problem_fig1_221787117

Tutorial 5

Question 2

Using the Rod-cutting Problem we have discussed in class for dynamic programming, try to draw a similar tree for a rod with 5m in length and determine the optimal solution.



Answer:

Max cost = 13

Question 3:

Suppose there are n meeting requests, and each meeting has a start time and an end time (st, et). No two meetings can take place if the meeting intervals overlap. The goal of your algorithm is to schedule as many meetings as possible. Example: (1,3) (2,4) (4,6) (3,8) (7,8), (3,4) The meetings scheduled will be (1,3) (3,4) (4,6) (7,8)

Design an algorithm that prints out the meeting numbers that will be scheduled. The inputs are: meeting_number, start_times and end_times. In the example above, it will be: meeting_number = [0,1,2,3,4,5] start_times = [1,2,4,3,7,3] end_times = [3,4,6,8,8,4] Output is 0, 5, 2, 4

Answer:

Given the following:

Request	start	end
0	1	3
1	2	4
2	4	6
3	3	8
4	7	8
5	3	4

Tutorial 5

Sort the above based on end time => SortedList[]:

SortedList	StartTime	EndTime
0	1	3
1	2	4
2	3	4
3	4	6
4	3	8
5	7	8

For the first entry in the SortedList => NewSchedule[]

NewSchedule	StartTime	EndTime
0	1	3

```
LatestNewSchedule = SortedList[0]
For i=1 to end of SortedList[]
{        If SortedList[i].StartTime >= LastestNewSchedule.EndTime
                  Insert SortedList[i] => NewSchedule[]
                  Update LastestNewSchedule to the current entry
}
```

SortedList[1] (2,4) not suitable, since StartTime 2 not >= EndTime 3, skip to next entry
SortedList[2] (3,4) suitable, thus added to NewSchedule

NewSchedule	StartTime	EndTime
0	1	3
1	3	4

SortedList[3] (4,6) suitable, thus added to NewSchedule

NewSchedule	StartTime	EndTime
0	1	3
1	3	4
3	4	6

SortedList[4] (3,8) not suitable, since StartTime 3 not >= EndTime 6, skip to next entry
SortedList[5] (7,8) suitable, thus added to NewSchedule

NewSchedule	StartTime	EndTime
0	1	3
1	3	4
3	4	6
4	7	8