Dynamic Programming

DP memoisation recursion

Max Square in Binary Matrix

```
1 1 1
0 1 1
1 0 0

Ans: 2
```

We will define the subproblem as:

$$DP[i][j] = max\ size\ square\ with\ one\ corner\ as\ (i,j)$$

The recurrence relation is given by $DP[i][j] =$

$$\begin{cases} 0 & grid[i][j] == 0 \\ 1 + max(\begin{matrix} DP[i-1][j], \\ DP[i][j-1], \\ DP[i-1][j-1] \end{matrix}) & otherwise \end{cases}$$

Max rectangle in a histogram

Although one can easily come up with a naive solution with O(n^2) time complexity. A linear solution using stacks also exists.

```
1 2 6 4 6 3

Ans: 12
```

Algorithm:

```
Insert a pair (-1, -INF) in the stack

while (i < array size)
  if value on top of stack less than current value
    - push the current value
  else
    - keep popping stack until the value on top of stack is less than the current value
    - calculate the area for each element popped using the current index as the right index and index of the prev element in the stack as the left index
    - push the current value on stack
end while

while stack not empty
  - pop value from stack and calculate the area using the current index as the right
```

```
    index and index of the prev element in the stack as the left index
14  end while
15
16  return the max area
```

```
 1  #define INF 1000000007
 2
 3  typedef pair<int,int> ii;
 4
 5  int maxRectangle(vector<int> &arr){
 6      int ans = 0;
 7      stack<ii> st;
 8      st.push(make_pair(-1, -INF));
 9      for(int i = 0; i < arr.size(); ++i){
10          if(st.top().second < arr[i]){
11              st.push(make_pair(i, arr[i]));
12              continue;
13          }
14          while(st.top().second >= arr[i]){
15              int idx = i - 1;
16              int val = st.top().second;
17              st.pop();
18              int len = idx - st.top().first;
19              ans = max(ans, val * len);
20          }
21          st.push(make_pair(i, arr[i]));
22      }
23      while(!st.empty()){
24          int idx = arr.size() - 1;
25          int val = st.top().second;
26          st.pop();
27          if(val == -INF)
28              break;
29          int len = idx - st.top().first;
30          ans = max(ans, val * len);
31      }
32      return ans;
33  }
```

Max Rectangle in Binary Matrix

We can use the solution of the previous problem to solve this problem.
For each row we calculate the height histogram of 1's and then run the `maxRectangle(vector<int> arr)` routine to calculate the max area in the current row.
We return the overall max area.