第六周作业

中等

1）最小路径和（亚马逊、高盛集团、谷歌在半年内面试中考过）

https://leetcode-cn.com/problems/minimum-path-sum/

```
  // 方法一
  func minPathSum(grid [][]int) int {
    m := len(grid)
    n := len(grid[0])
    dp := make([][]int, m)
    for i:= 0; i<m; i++ {
      dp[i] = make([]int, n)
    }
    for i:=0; i<m; i++ {
      for j:=0; j<n; j++ {
        if i == 0 && j == 0 {
          dp[i][j] = grid[i][j]
        } else if i == 0 {
          dp[i][j] = dp[i][j-1] + grid[i][j]
        } else if j == 0 {
          dp[i][j] = dp[i-1][j] + grid[i][j]
        } else {
          dp[i][j] = min(dp[i-1][j], dp[i][j-1]) + grid[i][j]
        }
     }
  }
  return dp[m-1][n-1]
}
func min(a, b int)int {
  if a < b {
  return a
  }
  return b
}

//方法二
func minPathSum(grid [][]int) int {
  m := len(grid)
  n := len(grid[0])
  for i:=0; i<m; i++ {
    for j:=0; j<n; j++ {
      if i == 0 && j == 0 {
        continue
      } else if i == 0 {
        grid[i][j] = grid[i][j-1] + grid[i][j]
      } else if j == 0 {
        grid[i][j] = grid[i-1][j] + grid[i][j]
      } else {
        grid[i][j] = min(grid[i-1][j], grid[i][j-1]) + grid[i][j]
      }
    }
  }
  return grid[m-1][n-1]
}
func min(a, b int)int {
  if a < b {
    return a
  }
  return b
}
```

2）解码方法（亚马逊、Facebook、字节跳动在半年内面试中考过）

https://leetcode-cn.com/problems/decode-ways/

```
func numDecodings(s string) int {
  if s[0] == '0' {
    return 0
  }
  if len(s) == 1 {
    return 1
  }
  dp := make([][]int, len(s))
  for i := 0; i < len(s); i++ {
    dp[i] = make([]int, 2)
  }
  dp[0][0] = 1
  dp[0][1] = 0
  for i:=1; i< len(s); i++ {
    nums := s[i-1: i+1]
    if i == 1 {
      if s[i] == '0' {
        dp[i][0] = 0
      } else {
        dp[i][0] = 1
      }
      if nums >= "1" && nums <= "26" {
        dp[i][1] = 1
      } else {
        dp[i][1] = 0
      }
      continue
    }
    if s[i] == '0' {
      dp[i][0] = 0
    } else {
      dp[i][0] = dp[i-1][0] + dp[i-1][1]
    }
    if nums >= "1" && nums <= "26" {
      dp[i][1] = dp[i-2][0] + dp[i-2][1]
    } else {
      dp[i][1] = 0
    }
  }
  return dp[len(s)-1][0] + dp[len(s)-1][1]
}
```

3)最大正方形（华为、谷歌、字节跳动在半年内面试中考过）

https://leetcode-cn.com/problems/maximal-square/

```
// 方法一
func maximalSquare(matrix [][]byte) int {
  m := len(matrix)
  n := len(matrix[0])
  dp := make([][]int, m+1)
  maxSlid := 0
  for i:=0; i<= m; i++ {
    dp[i] = make([]int, n+1)
  }
  fmt.Printf("dp is %v:", dp)
  for i:=1; i<=m; i++ {
    for j:=1; j<=n; j++ {
      if matrix[i-1][j-1] == '1'{
        dp[i][j] = min(min(dp[i-1][j-1], dp[i-1][j]), dp[i][j-1]) + 1
        maxSlid = max(maxSlid, dp[i][j])
      }
    }
  }
  return maxSlid * maxSlid
}
func max(a, b int) int {
  if a > b {
    return a
  }
  return b
}
func min(a, b int) int {
  if a < b {
    return a
  }
  return b
}
```

4）任务调度器（Facebook 在半年内面试中常考）

https://leetcode-cn.com/problems/task-scheduler/

```
// 方法一
func leastInterval(tasks []byte, n int) int {
  length := len(tasks)
  if length <= 1 {
    return length
  }
  m := map[byte]int{}
  maxC := 0
  for _, c := range tasks {
    m[c]++
    if maxC < m[c] {
      maxC = m[c]
    }
  }
  cnt := 0
  for _, v := range m {
    if v == maxC {
      cnt++
    }
  }
  return max(length, (maxC - 1) * (n+1) + cnt)
}
func max(a, b int) int {
  if a > b {
    return a
  }
  return b
}

// 方法二
func leastInterval(tasks []byte, n int) int {
  length := len(tasks)
  if length <= 1 {
    return length
  }
  m := make([]int, 26)
  maxC := 0
  for _, c := range tasks {
    m[c-'A']++
  }
  for _, v := range m {
    if v > maxC {
      maxC = v
    }
  }
  cnt := 0
  for _, v := range m {
    if v == maxC {
      cnt++
    }
  }
  return max(length, (maxC - 1) * (n+1) + cnt)
}
func max(a, b int) int {
  if a > b {
    return a
  }
  return b
}
```

5）回文子串（Facebook、苹果、字节跳动在半年内面试中考过）

https://leetcode-cn.com/problems/palindromic-substrings/

```
// 方法一
func countSubstrings(s string) int {
  length := len(s)
  dp := make([][]bool, length)
  for i:=0; i< length; i++ {
    dp[i] = make([]bool, length)
  }
  cnt := 0
  for j:=0; j<length; j++ {
    for i:=0; i<=j; i++ {
    if s[i] == s[j] && (j-i<2 || dp[i+1][j-1]) {
      dp[i][j] = true
      cnt++
    }
    }
  }
  return cnt
 }

//方法二
func countSubstrings(s string) int {
  length := len(s) * 2 - 1
  cnt := 0
  for center:=0; center < length; center++ {
    left := center / 2
    right := left + center % 2
    for left >= 0 && right < len(s) && s[left] == s[right] {
      cnt++
      left--
      right++
    }
  }
  return cnt
}
```

困难

1）最长有效括号（字节跳动、亚马逊、微软在半年内面试中考过）

https://leetcode-cn.com/problems/longest-valid-parentheses/

```
func longestValidParentheses(s string) int {
  length := len(s)
  dp := make([]int, length)
  maxCnt := 0
  for i:=0;i<length;i++{
    if s[i] == ')' {
      if i-1 >= 0 && s[i-1] == '(' {
        dp[i] = 2
        if i - 2 >= 0 {
          dp[i] = dp[i-2] + dp[i]
        }
      }
      if i-1 >= 0 && dp[i-1] > 0 {
        if i-dp[i-1]-1 >= 0 && s[i-dp[i-1]-1] == '(' {
          dp[i] = dp[i-1] + 2
          if i-dp[i-1]-2 >= 0 {
            dp[i] = dp[i] + dp[i-dp[i-1]-2]
          }
        }
      }
    }
  }
  for _, v := range dp {
    if v > maxCnt {
      maxCnt = v
    }
  }
  return maxCnt
}
```

2) 编辑距离（字节跳动、亚马逊、谷歌在半年内面试中考过）

https://leetcode-cn.com/problems/edit-distance/

```
func minDistance(word1 string, word2 string) int {
  m := len(word1)
  n := len(word2)
  dp := make([][]int, m+1)
  for i := 0; i <= m; i++ {
    dp[i] = make([]int, n+1)
    dp[i][0] = i
  }
  for i:= 0; i<= n; i++ {
    dp[0][i] = i
  }
  for i := 1; i <=m; i++ {
    for j := 1; j <= n; j++ {
      if word1[i-1] == word2[j-1] {
        dp[i][j] = dp[i-1][j-1]
      } else {
        dp[i][j] = min(min(dp[i-1][j-1],dp[i][j-1]), dp[i-1][j]) + 1
      }
    }
  }
  return dp[m][n]
}

func min(a, b int) int {
  if a > b {
    return b
  }
  return a
}
```

3） 矩形区域不超过 K 的最大数值和（谷歌在半年内面试中考过）

https://leetcode-cn.com/problems/max-sum-of-rectangle-no-larger-than-k/

```
// 方法一 四层for循环 固定左右边界
func maxSumSubmatrix(matrix [][]int, k int) int {
  ans := math.MinInt32
  row := len(matrix)
  col := len(matrix[0])
  for left := 0; left < col; left++ {
    for right := left; right < col; right++ {
      rSum := make([]int, row)
      for i := 0; i < row; i++ {
        for j := left; j<=right; j++ {
          rSum[i] += matrix[i][j]
        }
      }
      res := helper(rSum, k)
      if res > ans {
        ans = res
      }
    }
  }
  return ans
}
```

```
func helper(nums[]int, k int) int {
  max := math.MinInt32
  for i:= 0; i< len(nums); i++ {
    sum := 0
    for j := i; j < len(nums); j++ {
      sum += nums[j]
      if sum > max && sum <= k {
        max = sum
      }
    }
  }
  return max
}

// 方法二： 三层for循环 固定左右边界
func maxSumSubmatrix(matrix [][]int, k int) int {
  ans := math.MinInt32
  row := len(matrix)
  col := len(matrix[0])
  for left := 0; left < col; left++ {
    rSum := make([]int, row)
    for right := left; right < col; right++ {
      for i := 0; i < row; i++ {
        rSum[i] += matrix[i][right]
      }
      res := helper(rSum, k)
      if res > ans {
        ans = res
      }
    }
  }
  return ans
}
func helper(nums[]int, k int) int {
  max := math.MinInt32
  sum := 0
  for _, num := range nums {
    if sum > 0 {
      sum += num
    } else {
      sum = num
    }
    if sum > max {
      max = sum
    }
  }
  if max <= k {
    return max
  }
  max = math.MinInt32
  for i:= 0; i< len(nums); i++ {
    sum := 0
    for j := i; j < len(nums); j++ {
      sum += nums[j]
      if sum > max && sum <= k {
        max = sum
      }
    }
  }
  return max
}
```

4）青蛙过河（亚马逊、苹果、字节跳动在半年内面试中考过）

https://leetcode-cn.com/problems/frog-jump/

```
func canCross(stones []int) bool {
  m := map[int]bool{}
  return helper(m, stones, 0, 0)
}

func helper(m map[int]bool, stones []int, index int, k int) bool {
  key := index * 1000 + k
  if m[key] {
    return false
  } else {
    m[key] = true
  }
  for i:=index+1; i<len(stones); i++ {
    gap := stones[i] - stones[index]
    if gap >= k-1 && gap <= k+1 {
      if helper(m, stones, i, gap) {
        return true
      }
    }
    if gap > k+1 {
      return false
    }
    if gap < k-1 {
      continue
    }
  }
  return index == len(stones) -1
}
```

5）分割数组的最大值（谷歌、亚马逊、Facebook 在半年内面试中考过）

https://leetcode-cn.com/problems/split-array-largest-sum/

```
func splitArray(nums []int, m int) int {
  // m 确定子数组个数
  // 方法：二分查找，子数组和的边界[max(nums), sums(nums)]]
  // 通过二分查找找到mid值，即刚好实现数组和最少
  lmax := nums[0]
  rmax := 0
  for _, num := range nums {
    if lmax < num {
      lmax = num
    }
    rmax += num
  }
  for lmax < rmax {
    cnt := 1
    mid := lmax + (rmax - lmax) / 2
    tmp := 0
    for _, num := range nums {
      tmp += num
      if tmp > mid {
        tmp = num
        cnt++
      }
    }
    if cnt > m {
      lmax = mid + 1
    } else {
      rmax = mid
    }
  }
  return lmax
}
```

6）学生出勤记录 II （谷歌在半年内面试中考过）

https://leetcode-cn.com/problems/student-attendance-record-ii/

```
func checkRecord(n int) (ans int) {
    const mod int = 1e9 + 7
    dp := make([][2][3]int, n+1) // 三个维度分别表示：长度，A 的数量，结尾连续 L 的数量
    dp[0][0][0] = 1
    for i := 1; i <= n; i++ {
        // 以 P 结尾的数量
        for j := 0; j <= 1; j++ {
            for k := 0; k <= 2; k++ {
                dp[i][j][0] = (dp[i][j][0] + dp[i-1][j][k]) % mod
            }
        }
        // 以 A 结尾的数量
        for k := 0; k <= 2; k++ {
            dp[i][1][0] = (dp[i][1][0] + dp[i-1][0][k]) % mod
        }
        // 以 L 结尾的数量
        for j := 0; j <= 1; j++ {
            for k := 1; k <= 2; k++ {
                dp[i][j][k] = (dp[i][j][k] + dp[i-1][j][k-1]) % mod
            }
        }
    }
    for j := 0; j <= 1; j++ {
        for k := 0; k <= 2; k++ {
            ans = (ans + dp[n][j][k]) % mod
        }
    }
    return ans
}
```