FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110327 ALGORITHM DESIGN
Year II, Second Semester, Midterm Examination, March 8, 2019 13:00-16:00

---

ชื่อ-นามสกุล.....เลขประจำตัว.....ตอนเรียนที่.....เลขที่ใน CR58.....

<u>หมายเหตุ</u>

1. ข้อสอบมีทั้งหมด 9 ข้อ ในกระดาษคำถาม 6 หน้า
2. **ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ**
3. **ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ**
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. **นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับ สัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

**ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

*** ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ ***

**ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือหรือให้ความช่วยเหลือ ในการทำข้อสอบนี้**

ลงชื่อนิสิต.....

วันที่.....

1. (10 คะแนน) จงตอบคำถามต่อไปนี้สั้น ๆ ไม่ต้องอธิบาย

- $\log 1 + \log 2 + \log 3 + ... + \log (n-1) + \log n$ = $\Theta($ $\log(n!)$ $)$
- $1 + 2 + 3 + ... + n$ = $\Theta($ $n^2$ $)$
- การเรียงลำดับข้อมูลที่เรียงลำดับอยู่แล้ว $n$ ตัวด้วย selection sort ใช้เวลา = $\Theta($ $n^2$ $)$
- การเรียงลำดับข้อมูลที่เรียงลำดับอยู่แล้ว $n$ ตัวด้วย insertion sort ใช้เวลา = $\Theta($ $n$ $)$
- การเรียงลำดับข้อมูลที่เรียงลำดับอยู่แล้ว $n$ ตัวด้วย merge sort ใช้เวลา = $\Theta($ $n \log n$ $)$
- ให้ $T(n) = T(n-2) + \Theta(n)$, $T(0) = \Theta(1)$ จะได้ว่า $T(n)$ $T(n) = 3T(n-1)+1$ = $\Theta($ $n^2$ $)$
- ให้ $T(n) = 3T(n-1) + \Theta(1)$, $T(0) = \Theta(1)$ จะได้ว่า $T(n)$ $1+3+9+...+3^n$ = $\Theta($ $3^n$ $)$
- ให้ $T(n) = 3T(n/3) + \Theta(1)$, $T(0) = \Theta(1)$ จะได้ว่า $T(n)$ = $\Theta($ $n$ $)$
- การหา median of medians of fives ของข้อมูล $n$ ตัว ด้วยวิธีที่เรียนมา ใช้เวลา = $\Theta($ ____ $)$
- การหา longest common subsequence ของสตริงที่ยาว $n$ สองตัวในกรณีแย่สุด ด้วยวิธี dynamic programming ที่เรียนมา ใช้เวลา = $\Theta($ ____ $)$

2. (10 คะแนน) จงวิเคราะห์ว่าอัลกอริทึมข้างล่างนี้ใช้เวลา**เป็น Θ อะไรของตัวแปร n** (ให้การดำเนินการพื้นฐานเช่น * / + – และอื่น ๆ ใช้เวลา $\Theta(1)$ และการหาร / เป็นการหารแบบปัดเศษทิ้ง) **แสดงวิธีทำด้วยในช่องทางขวา**

| | |
|---|---|
| `EE(A[1..n][1..n], B[1..n][1..n]) {`<br>`  C = new array[1..n][1..n]`<br>`  for (i=0; i<n; i++) {`<br>`  for (j=0; j<n; j++) {`<br>`  C[i][j] = 0`<br>`  for (k=0; k<n; k++)`<br>`  C[i][j] += A[i][k]*B[k][j]`<br>`  }`<br>`  }`<br>`  return C`<br>`}` | $\Theta(n^3)$ |
| `RR( n ) {`<br>`  if (n==2) return 2*n`<br>`  for (k=0; k<n; k++)`<br>`  for (i=0; i<k; i++)`<br>`  s += 2*k - i*i + 4`<br>`  for (k=0; k<4; k++)`<br>`  s += RR(n/2)`<br>`  return s`<br>`}` | $4(\frac{n}{2}) + \Theta(n^2)$<br>$\Theta(n^2)$ |
| `MM( d[1..n], t[1..n] ) {`<br>`  i = 1; j = n; k = 1`<br>`  m = n/2`<br>`  while (i<=m and j<=n) {`<br>`  if (d[i] < d[j]) t[k++] = d[i++]`<br>`  else t[k++] = d[j++]`<br>`  }`<br>`  while (i<=m) t[k++] = d[i++]`<br>`  while (j<=n) t[k++] = d[j++]`<br>`  for (k = 1; k < n; k++) d[k] = t[k]`<br>`}` | $\Theta(\frac{n}{2})$ $\Theta(n)$<br>$\Theta(n)$<br>$\Theta(n)$ |
| `SS( d[1..n] ) { # ทุกช่องใน d เก็บจำนวนเต็มบวก`<br>`  SS( d, n, new array_of_zeros[n] )`<br>`}`<br>`SS( d[1..n], m, A[1..n] ) {`<br>`  if (m < 1) return 1`<br>`  if (m < 4) return d[m]`<br>`  if (A[m] > 0) return A[m]`<br>`  x1 = d[m]*SS(d, m-1, A)`<br>`  x2 = d[m-1]*SS(d, m-2, A)`<br>`  x3 = d[m-2]*SS(d, m-3, A)`<br>`  A[m] = x1 + x2 + x3`<br>`  return A[m]`<br>`}` | $\Theta(n)$ |
| `SO(d[1..n]) {`<br>`  for (k = 0; k < n; k++)`<br>`  insertion_sort( d )`<br>`}` | $\Theta(n^2)$ |

```
}
SS( d[1..n], m, A[1..n] ) {
  if (m < 1) return 1
  if (m < 4) return d[m]
  if (A[m] > 0) return A[m]
  x1 = d[m]*SS(d, m-1, A)
  x2 = d[m-1]*SS(d, m-2, A)
  x3 = d[m-2]*SS(d, m-3, A)
  A[m] = x1 + x2 + x3
  return A[m]
```

$\Theta(n)$

3. (10 คะแนน) จงตอบคำถามต่อไปนี้ในช่องว่างที่กำหนดให้ ไม่ต้องแสดงวิธีทำ

- $7^{47}$ mod 11 มีค่าเท่ากับ .....6.....
- median-of-medians of 5 ของ [1,3,29,4,5], [14,11,19,18,21], [32,12,25,7,24], [13,17,2,15,20], [21,29,26,8,9] คือ .....
- การ merge ข้อมูล [1, 3, 8, 9] กับ [2, 4, 6, 7] จะเกิดการเปรียบเทียบข้อมูลจาก 2 อาเรย์นี้ .....6..... ครั้ง
- ให้ X = "ABCABC", ค่าของ Y ที่ทำให้การหา longest common subsequence ของ X กับ Y แบบ top-down (ไม่มี memoization) ทำงานได้เร็วสุด ๆ คือ .....
- ให้ v = [15, 20, 20, 24, 40] และ w = [5, 4, 2, 4, 8] แทนมูลค่าและน้ำหนักของของ 5 ชิ้น และให้มีถุงที่รับน้ำหนักไม่เกิน 10 ต้องเลือกชิ้นไหน เท่าไร แบบ fractional จึงได้มูลค่ารวมสูงสุด .....
- ให้ v = [15, 20, 20, 24, 40] และ w = [5, 4, 2, 4, 8] แทนมูลค่าและน้ำหนักของของ 5 ชิ้น และให้มีถุงที่รับน้ำหนักไม่เกิน 10 ต้องเลือกชิ้นไหน แบบ 0/1 จึงได้มูลค่ารวมสูงสุด .....2,3,4.....
- ให้ A = [(0,3), (5,8), (2,4), (4,5), (5,7), (7,9), (3,6)] แทนรายการของเวลาเริ่มและเวลาสิ้นสุดของกิจกรรมต่าง ๆ เราต้องเลือกกิจกรรมใดบ้าง (ที่ไม่ซ้อนเหลื่อมกัน) จึงได้จำนวนกิจกรรมมากสุด .....1,4,5,6.....
- ให้ F = [('a',1), ('b',2), ('c',4), ('d',8), ('e',16), ('f',32), ('g',64)] แทนรายการของตัวอักษรและความถี่ของข้อมูลชุดหนึ่ง ถ้าเราเข้ารหัสตัวอักษรชุดนี้ด้วยรหัสแบบ Huffman จะใช้ข้อมูลจำนวน ..... บิตในการแทนข้อมูลชุดนี้ทั้งหมด
- มีข้อมูลอยู่ 1,000 ตัวที่เรียงลำดับจากมากมาน้อยแล้ว (ซ้ายไปขวา) ในอาเรย์หนึ่ง การค้นข้อมูลในอาเรย์นี้ด้วย binary search จะพิจารณาข้อมูลในอาเรย์ไม่เกิน .....9..... ตัว
- ให้ $A_1$, $A_2$, $A_3$ และ $A_4$ คือเมทริกซ์ขนาด 5×100, 100×2, 2×100 และ 100×5 ตามลำดับ ถ้าต้องการหาผลคูณของ $A_1 \times A_2 \times A_3 \times A_4$ ควรจัดลำดับการคูณอย่างไรจึงจะหาผลคูณได้เร็วสุด ..... (ตอบโดยใส่วงเล็บเพื่อกำหนดลำดับการคูณ)

4. (6 คะแนน) จากตารางค่าของคำตอบของปัญหาที่ใช้กำหนดการพลวัตต่อไปนี้ จงสร้างตารางของการตัดสินใจที่ถูกต้อง

- Longest Common Subsequence (เติม ↑ หรือ ← หรือ ↖)

| | | A | A | B | C | A | C | A |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
| C | 0 | 1 | 1 | 2 | 3 | 3 | 3 | 3 |
| A | 0 | 1 | 2 | 2 | 3 | 4 | 4 | 4 |
| C | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| C | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| A | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 6 |

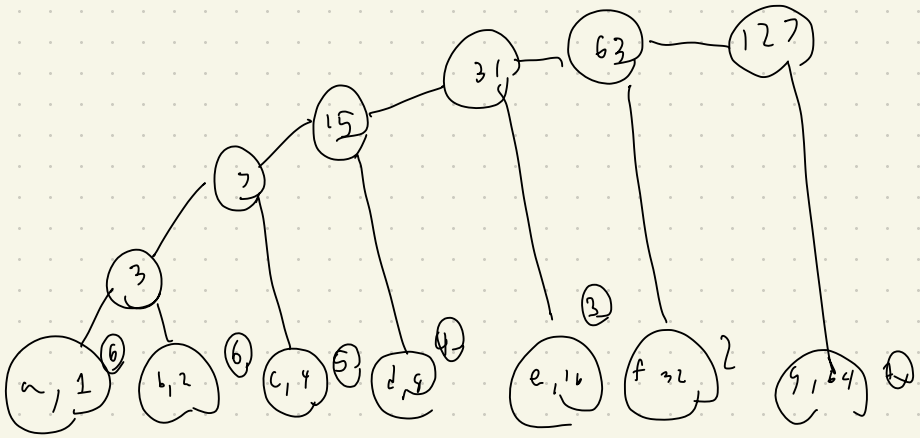
| | | A | A | B | C | A | C | A |
|---|---|---|---|---|---|---|---|---|
| A | | ↖ | ←↖ | ← | ← | ← | ← | ← |
| B | | ↑ | ←↑ | ↖ | ← | ← | ← | ← |
| C | | ↑ | ←↑ | ↑ | ↖ | ← | ←↖ | ← |
| A | | ↑ | ↖ | ←↑ | ↑ | ↖ | ← | ←↖ |
| C | | ↑ | ↑ | ←↑ | ↖↑ | ↑ | ↖ | ← |
| C | | ↑ | ↑ | ↑ | ↖↑ | ↑ | ↖↑ | ←↑ |
| A | | ↑ | ↖↑ | ←↑ | ↑ | ↖↑ | ↑ | ↖ |

- 0/1 Knapsack (เติม ✓ หรือ ✘ แทนการเลือกไม่เลือก)

| | capacity | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| empty | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| value=12 weight=4 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 |
| value=10 weight=6 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 |
| value=8 weight=5 | 0 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 |
| value=14 weight=3 | 0 | 0 | 0 | 14 | 14 | 14 | 14 | 26 | 26 |
| value=7 weight=1 | 0 | 7 | 7 | 14 | 21 | 21 | 21 | 26 | 33 |
| value=9 weight=6 | 0 | 7 | 7 | 14 | 21 | 21 | 21 | 26 | 33 |
| value=9 weight=2 | 0 | 7 | 9 | 16 | 21 | 23 | 30 | 30 | 33 |

| | capacity | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| empty | | | | | | | | | |
| value=12 weight=4 | | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✓ |
| value=10 weight=6 | | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ |
| value=8 weight=5 | | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ |
| value=14 weight=3 | | ✘ | ✘ | ✘ | ✓ | ✓ | ✓ | ✓ | ✓ |
| value=7 weight=1 | | ✓ | ✘ | ✓ | ✓ | ✘ | ✓ | ✓ | ✘ |
| value=9 weight=6 | | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ | ✘ |
| value=9 weight=2 | | ✘ | ✓ | ✓ | ✘ | ✓ | ✓ | ✓ | ✓ |

ให้ F = [('a',1), ('b',2), ('c',4), ('d',8), ('e',16), ('f',32), ('g',64)] แทนรายการของตัวอักษรและความถี่ของข้อมูลชุดหนึ่ง ถ้าเราเข้ารหัสตัวอักษรชุดนี้ด้วยรหัสแบบ Huffman จะใช้ข้อมูลจำนวน ..... บิตในการแทนข้อมูลชุดนี้ทั้งหมด




6 + 12 + 20 + 32 + 44 + 64 + 64

70 + 112 + 64

128 + 48 + 176

246

1000 500 250 125

7 15 31 63

3 1

$\lceil \log_2 1000 \rceil = 9$

$7^{47} \bmod 11$ มีค่าเท่ากับ

$$\left(7^{7} \bmod 11 \cdot \left(\left(7^{23} \bmod 11\right)^{4} \bmod 11\right)^{1}\right) \bmod 11 \qquad 23 \bmod 11 = 6$$

$$\left(7^{7} \bmod 11 \left(\left(7^{11} \bmod 11\right)^{5} \bmod 11\right)^{1}\right) \bmod 11 = 7$$

$$\left(\left(7^{7} \bmod 11\right) \left(\left(7^{5} \bmod 11\right)^{2} \bmod 11\right)^{1}\right) \bmod 11 = 7$$

$$\left(\left(7 \bmod 11\right)\left(\left(7^{2} \bmod 11\right)^{2} \bmod 11\right)\right) \bmod 11 = 10$$

$7^{2} \bmod 11 = 5$

3

ให้ $A_1$, $A_2$, $A_3$ และ $A_4$ คือเมทริกซ์ขนาด 5×100, 100×2, 2×100 และ 100×5 ตามลำดับ ถ้าต้องการหาผลคูณของ $A_1 \times A_2 \times A_3 \times A_4$ ควรจัดลำดับการคูณอย่างไรจึงจะหาผลคูณได้เร็วสุด $(A_1 A_2)(A_3 A_4)$ (ตอบโดยใส่วงเล็บเพื่อกำหนดลำดับการคูณ)

1000 +

$(5 \times 2) \times (2 \times 100)$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1000 | 2000 | 2100 |
| 2 | | 0 | 20,000 | 2000 |
| 3 | | | 0 | 1000 |
| 4 | | | | 0 |

5. (10 คะแนน) ให้ D เป็นอาเรย์ขนาด *n* ช่อง ภายในเก็บจำนวนเต็ม 0 ถึง *n* แต่มีค่าหนึ่งหายไป (ที่ไม่ใช่ *n*) ข้อมูลใน D เรียงจากน้อยไปมากแล้ว เช่น D = [0,1,2,4,5,6,7] มี 3 หายไป, D = [1,2,3,4,5,6] มี 0 หายไป โดยไม่มีกรณีไม่มีตัวหาย เช่น D = [0,1,2,3,4,5])
จงเขียนรหัสเทียมของอัลกอริทึมที่ใช้เวลา $O(\sqrt{n})$ เพื่อหาข้อมูลที่หายไปในอาเรย์ D

```
missing( D[0..n-1] ) {

    int L = 0
    int R = n-1
    while ( L < R)
    {
        int mid = (L+R)/2 ;
        if ( mid == D[mid] )
        {
            L = mid+1 ;
        }
        else
        {
            R = mid ;
        }
    }
    Return R ;
}
```

ยกตัวอย่างประกอบ

6. (10 คะแนน) ให้ D คืออาเรย์ขนาด *n* ช่องที่เก็บจำนวนเต็ม จงเขียนรหัสเทียมของอัลกอริทึมที่ใช้เวลา $O(\log n)$ เพื่อหาค่า "peak" (ขอค่า peak สักหนึ่งค่า) ใน D โดย peak คือค่าในอาเรย์ที่มีค่ามากกว่าหรือเท่ากับ ค่าของตัวก่อนหน้าทางซ้ายหนึ่งตัวและตัวถัดไปทางขวาหนึ่งตัว (ถ้าไม่มีตัวก่อนหน้าหรือตัวถัดไป ก็พิจารณาอีกข้างหนึ่งที่มีก็พอ) เช่น D = [9, 7, 7, 99, 4, 5, 6, 6, 5, 8] มี 9, 99, 6 และ 8 เป็น peak

```
peak( D[0..n-1] ) {

    int L = 0 ;
    int R = n-1 ;
    while ( 1 )
    {
        int m = (L+R)/2 ;
        int comp1 = max( 0, m-1)
        int comp2 = min( n-1, m+1)
        if ( D[mid] >= D[comp1] and D[mid] >= D[comp2])
        {
            return D[mid];
        }
        else
        {
            if ( D[comp2] > D[mid])
            {
                L = mid+1;
            }
            else if (D[comp1] > D[mid])
            {
                R = mid-1 ;
            }
        }
    }
}
```

ยกตัวอย่างประกอบ

7. (10 คะแนน) จากความสัมพันธ์เวียนบังเกิดข้างล่างนี้ จงเขียนรหัสเทียมเพื่อแก้ปัญหานี้ด้วย bottom up dynamic programming

$$F(i,j) = \begin{cases} 0 & \text{if } i == 0 \text{ or } j == n \\ \max(F(i-1,j), F(i, j+p[i]) + q[i]) & \text{if } j + p[i] \le n \\ F(i-1,j) & \text{otherwise} \end{cases}$$

รับประกันว่า $p[i] > 0$ และ $0 \le i, j \le n$

```
F( p[0..n], q[0..n] ) {




}
```

8. (10 คะแนน) นัทที่ต้องการขับรถจากกิโลเมตรที่ 1 ไปกิโลเมตรที่ $n$ ($n \ge 2$) รถยนต์คันนี้มีถังน้ำมันซึ่งจุได้ $v$ ลิตร ($v \ge 1$) แต่กินน้ำมันมาก คือ ต้องใช้น้ำมัน 1 ลิตรในการเดินทาง 1 กิโลเมตร ตอนเริ่มเดินทางไม่มีน้ำมันเลย โชคดีที่มีปั๊มน้ำมันทุก ๆ หลักกิโลเมตร โดยปั๊มน้ำมันที่หลักกิโลเมตรที่ $k$ ขายน้ำมันในราคา $P_k$ บาทต่อลิตร คำถามคือ นัทที่ต้องจ่ายค่าน้ำมันน้อยสุดกี่บาท เพื่อที่จะเดินทางถึงจุดหมายที่กิโลเมตรที่ $n$ ได้ และน้ำมันหมดถังพอดี โดยนัทที่สามารถเติมน้ำมันที่หลักกิโลเมตรใดก็ได้

เช่น ถ้า $n = 4$, $v = 2$ และ $P = [1,2,3,4]$ คำตอบคือ 4 เพราะ ค่าน้ำมันน้อยสุดคือ เติมน้ำมัน 2 ลิตรที่ กม. 1 (2 ลิตร×1 บาท/ลิตร = 2 บาท) ขับถึง กม. 2 เหลือ 1 ลิตร เติมอีก 1 ลิตร (1 ลิตร×2 บาท/ลิตร = 2 บาท) ขับถึง กม. 4 ถึงจุดหมายและน้ำมันหมดพอดี

จงเขียนรหัสเทียมของอัลกอริทึมสำหรับแก้ปัญหาข้างบนนี้ด้วยกำหนดการพลวัต (dynamic programming) โดยต้องระบุความสัมพันธ์เวียนบังเกิด (recurrence), ขนาดของตารางที่ต้องใช้ และคำอธิบายย่อ ๆ

```
min_cost( P[1..n], n, v ) { # P[k] contains oil price at k-th kilometer.




}
```

ความสัมพันธ์เวียนบังเกิด:

ขนาดของตาราง:

คำอธิบายย่อ ๆ (ยกตัวอย่างประกอบ):

7. (10 คะแนน) จากความสัมพันธ์เวียนบังเกิดข้างล่างนี้ จงเขียนรหัสเทียมเพื่อแก้ปัญหานี้ด้วย bottom up dynamic programming

$$F(i,j) = \begin{cases} 0 & \text{if } i == 0 \text{ or } j == n \\ \max(F(i-1,j), F(i,j+p[i]) + q[i]) & \text{if } j + p[i] \le n \\ F(i-1,j) & \text{otherwise} \end{cases}$$

```
F [p[0...n] q[0...n]
{
  int save [n+1] [n+1];
  for int i=0 to n :
      for int j = n to 0 :
          if ( i = 0 or j = n )
              save [i] [j] = 0
          else if (j + p[i] ≤ n )
              save [i] [j] = max ( save [i-1] [j], save [ i ] [ j + p[i] ] + q[i] )
          else
              save [i] [j] = save [i-1] [j]

  return save [n] [n]
}
```

0, 0

F (1, p[i])

9. ให้ $D = [d_1, d_2, d_3, ..., d_n]$ เป็นรายการที่แต่ละช่องเก็บเลขโดด 0 ถึง 9 เช่น [5,1,8,7,1,1] นิยามให้ $D_{i,j}$ คือจำนวนที่ได้จากการนำเลขโดดใน $D$ มาต่อกันตั้งแต่ตัวที่ $i$ ถึง $j$ เช่น $D_{2,2} = 1$, $D_{2,3} = 18$, $D_{2,4} = 187$ เป็นต้น

- (5 คะแนน) จงเขียนรหัสเทียมของอัลกอริทึม `max_2(D)` ที่ใช้เวลา O($n$) เพื่อหา $D_{i,j}$ ที่มีจำนวนหลักมากสุดที่มีค่าที่หารด้วย 2 ลงตัว ถ้าไม่มี ให้คืน 0 (วิเคราะห์ประสิทธิภาพการทำงานเชิงเวลาด้วย) เช่น `max_2( [5,1,8,7,1,1] )` ได้ผลคือ `518`

```
max_2( D[1..n] ) {
    int tmp = 1;
    for int i = 2; i ≤ n ; i++)
      if( D[i] % 2 = 0)
        tmp = i;
      )
}
if(D[tmp] % 2 != 0)
    return 0
else;
  int sum = 0
  for (i = 1 to tmp)
    sum = sum*10 + D[i]
  return sum
```

วิเคราะห์เวลาการทำงาน: $\Theta(n)$

คำอธิบายหลักการทำงาน (ยกตัวอย่างประกอบ):

- (10 คะแนน) จงเขียนรหัสเทียมของอัลกอริทึม `max_3(D)` ที่ใช้เวลา O($n$) เพื่อหา $D_{i,j}$ ที่มีจำนวนหลักมากสุดที่มีค่าที่หารด้วย 3 ลงตัว ถ้าไม่มี ให้คืน 0 (วิเคราะห์ประสิทธิภาพการทำงานเชิงเวลาด้วย) เช่น `max_3( [5,1,8,7,1,1] )` ได้ผลคือ `18711` (ข้อแนะนำ: จำนวนเต็ม $N$ หารด้วย 3 ลงตัว เมื่อผลบวกของเลขโดดทุกตัวใน $N$ หารด้วย 3 ลงตัว เช่น 1+8+7+1+1 = 18 หารด้วย 3 ลงตัว ดังนั้น 18711 หารด้วย 3 ลงตัว)

```
max_3( D[1..n] ) {
    return max_3(D[1...n], 1, n)
}
max_3( D[1..n], start, stop )
  {
    if(start == stop)
    { if(D[start]%3 == 0)
        return D[start];
      return 0;
    }
    int sum = 0
    for(int i=start;i<=stop;i++
      sum = sum*10 + D[i];
    if(sum%3 == 0)
      sum = 0
      for in ti=start to stop
        sum = sum*10 + D[i];
      return sum
    else
      return max(max_3(D[1..n], start+1, stop), max_3(D[1..n], start, stop-1))
}
```

วิเคราะห์เวลาการทำงาน: $T(n) = 2T(n-1) + n$ $\quad T(n) = n(2^n) \Rightarrow \Theta(n2^n)$

$2T(n-1) = 4T(n-2) + 2n$

คำอธิบายหลักการทำงาน (ยกตัวอย่างประกอบ):