FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110327 ALGORITHM DESIGN
Year II, Second Semester, Midterm Examination, March 11, 2022 13:00-16:00

ชื่อ-นามสกุล.....Jomnoi Z.....เลขประจำตัว..... เลขที่ใน CR58.....

หมายเหตุ

1. ข้อสอบมีทั้งหมด 10 ข้อ ในกระดาษคำถามคำตอบ 7 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 ได้รับสัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

*** ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....

วันที่.....

- **ใช้ดินสอทำข้อสอบได้**
- **นิสิตสามารถใช้ข้อมูลสำหรับ Master Method ในรูปข้างล่างนี้ช่วยในการตอบคำตอบได้**

$$t(n) = at(n/b) + \Theta(n^d) \qquad a \geq 1, b > 1$$

$$t(n) = \begin{cases} \Theta(n^c) & \text{if } n^d < n^c \\ \Theta(n^c \log n) & \text{if } n^d = n^c \\ \Theta(n^d) & \text{if } n^d > n^c \end{cases} \qquad c = \log_b a$$

$$t(n) = at(n/b) + f(n)$$

$$t(n) = \begin{cases} \Theta(n^c) & \text{if } f(n) = O(n^{c-\varepsilon}) \ \textcircled{1} \\ \Theta(n^c \log n) & \text{if } f(n) = \Theta(n^c) \ \textcircled{2} \\ \Theta(f(n)) & \text{if } f(n) = \Omega(n^{c+\varepsilon}) \ \textcircled{3} \\ & a f(n/b) \leq k f(n), k < 1, n \geq n_0 \end{cases} \qquad \varepsilon > 0$$

1. (5 คะแนน) Master Method จาก Recurrence Relation (คำตอบสามารถติด ยกกำลัง ถอดราก ได้) หรือถ้าไม่สามารถแก้ได้โดย Master Method ได้ก็ให้ระบุว่าไม่ได้
   1.1. $T(n) = 3T(n/2) + n^2$ เวลาการทำงานคือ $\Theta(n^2)$
   1.2. $T(n) = 2T(n/2) + n \log n$ เวลาการทำงานคือ $\Theta(n \log^2 n)$
   1.3. $T(n) = 2T(n/4) + n^{0.53}$ เวลาการทำงานคือ $\Theta(n^{0.53})$
   1.4. $T(n) = \sqrt{2}T(n/2) + \log n$ เวลาการทำงานคือ $\Theta(\sqrt{n})$
   1.5. $T(n) = 9T(n/4) - n^3 \log n$ เวลาการทำงานคือ ไม่ได้
2. (4 คะแนน) ในแต่ละข้อย่อยต่อไปนี้จงเลือกคำตอบที่ถูกต้องโดยการกากบาทหน้าข้อที่ถูก (ข้อย่อยละ 1 คะแนน ข้อย่อยใดตอบถูกจะได้ 1 คะแนน ตอบผิดจะได้คะแนน -0.5 หากไม่ตอบจะได้ 0 โดยที่คะแนนรวมของข้อนี้จะไม่น้อยกว่า 0)
   2.1. โปรแกรมย่อยต่อไปนี้คำนวณอะไร

```
int unknown(int p, int q) {
     while (p != q) {
          if (p > q) p-=q; else  q-=p;
     }
     return q;
}
```

   ☐ ผลบวกของ p และ q ☐ เศษของการหาร p ด้วย q
   ☒ ตัวหารร่วมมากของ p และ q ☐ ตัวคูณร่วมน้อยของ p และ q
   2.2. การหาผลลัพธ์ของ x^{20} จะต้องใช้การคูณเลขอย่างน้อยกี่ครั้ง
   ☐ 3 ☐ 4 ☒ 5 ☐ 6
   2.3. ถ้าเรามีเมทริกซ์ M_1, M_2, M_3, M_4 ที่มีขนาดดังต่อไปนี้ 10x100, 100x20, 20x5, 5x80 ตามลำดับ จำนวนครั้งที่น้อยที่สุดที่ใช้ในการคูณตัวเลขเพื่อที่จะคำนวณผลคูณ $M_1M_2M_3M_4$ คือเท่าใด? (ให้ถือว่าการคูณเมทริกซ์ ขนาด MxN และ NxP ต้องใช้การคูณตัวเลข MNP ครั้ง)
   ☐ 15000 ☒ 19000 ☐ 240000 ☐ 30000
   2.4. สำหรับปัญหา Fractional Knapsack ถ้าเรามีของ (มูลค่า,น้ำหนัก) ดังนี้ {(20,5),(30,10),(10,10),(40,20)} และเราสามารถนำของไปได้หนัก 20 หน่วยแล้ว มูลค่าสูงสุดที่จะสามารถนำไปได้จะเป็นเท่าใด
   ☒ 60 ☐ 70 ☐ 80 ☐ 90
3. (8 คะแนน) จงเติมคำในช่องว่างให้เหมาะสม
   3.1. เลขหลักหน่วยของ 27^{117} คือ 7
   3.2. การเขียนโปรแกรมเพื่อแก้ปัญหาด้วย Dynamic Programming สามารถทำได้ด้วยสองรูปแบบหลักๆ คือ Top-Down และ Bottom-Up
   3.3. ให้ $a_1, a_2, \ldots, a_n$ เป็นลำดับตัวเลขจำนวนเต็ม, ลำดับย่อยของลำดับนี้อยู่ในรูป $a_{i1}, a_{i2}, .., a_{ik}$ โดยที่ $1 <= i1 < i2 < \ldots < ik <= n$, ลำดับย่อยที่ลดลง คือลำดับย่อยที่ $a_{i1} > a_{i2} > .. > a_{ik}$ จงเขียน Recurrence Relation ของความยาวของลำดับย่อยที่ยาวที่สุด โดยให้ L(j) คือความยาวของลำดับย่อยที่ยาวที่สุดที่เริ่มด้วย a_j (นั่นก็คือลำดับย่อย $a_{i1} > a_{i2} > .. > a_{ik}$ ที่ยาวที่สุดที่ i1=j นั่นเอง)
   Base case:
   L(n) = 1
   Recurrence Relation สำหรับ L(j), เมื่อ $1 <= j < n$

$$L(j) = \begin{cases} 1 & \text{if } a_j <= \min(a_{j+1}, a_{j+2}, \ldots, a_n) \\ 1 + \max(L(i)) \text{ where } j+1 \le i \le n \text{ and } a_j > a_i & \text{otherwise} \end{cases}$$

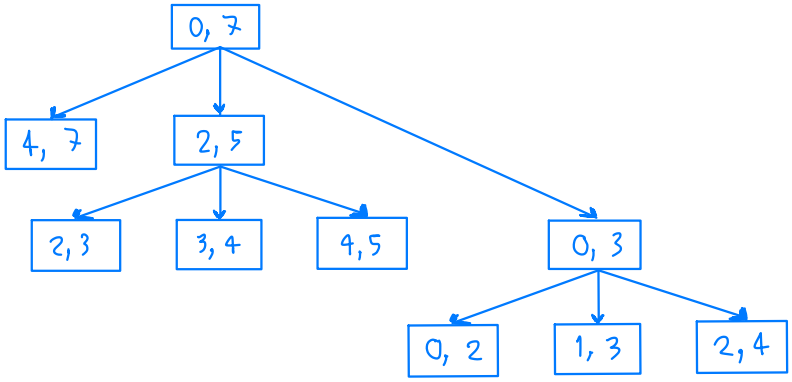
   3.4. ในปัญหา Activity Selection หากมีงานที่มีช่วง [เวลาเริ่ม, เวลาจบ) ดังต่อไปนี้

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
เวลาเริ่ม	1	2	2	3	4	4	6	6	7	8	9	10
เวลาจบ	6	7	5	5	7	7	8	9	10	9	12	14

   จำนวนงานที่สามารถทำได้ที่เวลาไม่ทับซ้อนกันสูงสุดคือ 4 งาน

4. (4 คะแนน) จากส่วนของโปรแกรมด้านขวานี้ จงวาด Recursion Tree เมื่อเราเรียกฟังก์ชัน recur({1,2,3,4,-4,-3,-2,-1},0,7) โดยในแต่ละปมให้ระบุเฉพาะค่า l, r ก็พอ

```
int recur(vector<int> &a,int l,int r) {
  cout << l << " " << r << endl;
  int n = (r-l)+1;
  if (n < 4) return a[(l+r)/2];
  bool ok = true;
  for (int i = l;i <= r;i++) {
    if (a[i] > 0) { ok = false; break; }
  }
  if (ok) return 0;
  int m =n/4;
  int r1 = recur(a,l    ,l+2*m-1);
  int r2 = recur(a,l+1*m,l+3*m-1);
  int r3 = recur(a,l+2*m,r);
  return r1+r2+r3;
}
```



5. (9 คะแนน) ในแต่ละข้อต่อไปนี้ ให้สมมติว่าเรากำลังพยายามแก้ปัญหาด้วยวิธีการ Brute Force จงระบุ candidate solution และ เซ็ตของ candidate solution พร้อมด้วยขนาดของ set ดังกล่าว

5.0. (ตัวอย่าง ปัญหา MCS) มีอาเรย์ A[1..n] เราต้องการหาช่วงติดกันที่ผลรวมของสมาชิกในช่วงมีค่ามากที่สุด

- Candidate solution: คู่อันดับ i,j โดยที่ i <= j
- เซ็ตของ candidate solution: { (i,j) | 1 <= i <= j <= n }
- ขนาดของเซ็ตดังกล่าว: n(n+1)/2

5.1. มีอาเรย์ A[1..n] เราต้องการหาช่วงติดกันที่ผลรวมของสมาชิกในช่วงมีค่ามากที่สุดโดยที่ช่วงดังกล่าวมีสมาชิก k ตัวพอดี

- Candidate solution: คู่อันดับ i, j โดยที่ $i \le j$ และ $j-i+1 = k$
- เซ็ตของ candidate solution: $\{(i,j) \mid 1 \le i \le j \le n \text{ and } j-i+1 = k\}$
- ขนาดของเซ็ตดังกล่าว: $n-k+1$

5.2. สมมติประเทศไทยมีจังหวัดทั้งหมด n จังหวัด (กำหนดเป็นจังหวัด 1 ถึง n) เราต้องเดินทางไปนำเสนอผลงานในจังหวัดที่แตกต่างกันจำนวน k จังหวัดโดยเริ่มต้นที่จังหวัด 1 ให้ D(a,b) เป็นค่าเครื่องบินในการเดินทางจากจังหวัด a ไป b อยากทราบลำดับของจังหวัด k จังหวัดที่เราจะเดินทางไปที่ทำให้ผลรวมของค่าเครื่องบินน้อยที่สุด

- Candidate solution: อาเรย์ความยาว k ตัว ตัวแรกเป็น 1 และตามด้วยการเรียงสับเปลี่ยนของตัวเลข 2 ถึง n ความยาว k-1 ตัว
- เซ็ตของ candidate solution: $\{[1, \_, \_, \ldots, \_] \mid \text{permutation size of } k-1 \text{ from } [2, 3, \ldots n]\}$

5.3. มีอาเรย์ A[1..n] เราต้องการเลือกสมาชิกไม่น้อยกว่า m1 สมาชิก และไม่มากกว่า m2 สมาชิก ที่ทำให้ค่าเฉลี่ยของของที่เลือกน้อยที่สุด

- Candidate solution: คู่อันดับ i, j โดยที่ $i \le j$ และ $m_1 \le j-i+1 \le m_2$
- เซ็ตของ candidate solution: $\{(i,j) \mid 1 \le i \le j \le n \text{ and } m_1 \le j-i+1 \le m_2\}$
- ขนาดของเซ็ตดังกล่าว: $(m_2-m_1+1)\left[n-\frac{(m_1+m_2)}{2}+1\right]$

- **สำหรับข้อที่ 6 เป็นต้นไป เป็นการออกแบบอัลกอริทึม ในแต่ละข้อสามารถตอบโดยการอธิบายอัลกอริทึม โดยใช้รหัสเทียม (Pseudocode) หรือ programming language ภาษาใดที่เคยเรียนมาก็ได้ และต้องวิเคราะห์ประสิทธิภาพในการทำงานของอัลกอริทึมด้วย**
- **คะแนนที่ได้จะแปรตามประสิทธิภาพในการทำงาน**
- **ในทุกข้อให้วิเคราะห์ประสิทธิภาพในการทำงาน โดยตอบเป็นสัญกรเชิงเส้นกำกับด้วย**

6. (10 คะแนน) จงเขียนโปรแกรมหรือ pseudo-code เพื่อแสดงอาเรย์ความยาว n ช่องของเลข**<u>จำนวนเต็มบวก</u>**ที่ผลรวมของตัวเลขทั้งหมดในอาเรย์นั้นมีค่าเป็น m พอดี โดยให้แสดงทุกรูปแบบของอาเรย์ที่เป็นไปได้ที่ตรงตามข้อกำหนดข้างต้น ให้ถือว่า n และ m เป็นข้อมูลนำเข้าของโปรแกรมนี้ และ การแสดงผลอาเรย์ให้เรียกฟังก์ชัน display(x) เมื่อ x คืออาเรย์ที่ต้องการแสดงผลได้เลย ตัวอย่างเช่น ให้ n = 3 และ m = 5 โปรแกรมของเราจะต้องเรียก display 6 ครั้ง ได้แก่ display([1,1,3]) และ display([1,3,1]) และ display([3,1,1]) และ display([1,2,2]), และ display([2,1,2]) และ display([2,2,1]) โดยเราสามารถเรียก display ในลำดับใดก็ได้ แต่ต้องไม่ซ้ำ และครบทุกอันพอดี

```
# include <bits/stdc++.h>
using namespace std;

void recur(int n, int m, int cur_sum, vector <int> &x) {
    if (cur_sum < 0) return;
    if (n == 0) {
        if (cur_sum == 0) display(x);
        return;
    }
    for (int i=1; i<=m; i++) {
        x.push_back(i);
        recur(n-1, m, cur_sum - i, x);
        x.pop_back();
    }
}
int main() {
    cin.tie(nullptr)->sync_with_stdio(false);

    int n, m;
    cin >> n >>m;
    vector <int> x;
    recur(n, m, m, x);
    return 0;
}
```

ประสิทธิภาพเชิงเวลาคือ $O(m^n)$

7. (10 คะแนน) รถยนต์ก้อนพลังงาน เป็นรถยนต์ที่ใช้เชื้อเพลิงเป็นก้อน ๆ โดยเริ่มต้นให้รถอยู่ ณ ตำแหน่ง 0 กม. รถยนต์คันนี้กำลังจะวิ่งเป็นเส้นตรง และบรรทุกก้อนพลังงานหลาย ๆ ก้อนไว้บนรถ สำหรับแต่ละ กม. นั้น เราจะต้องเลือกวิธีการใช้ก้อนพลังงานที่บรรทุกอยู่ สมมติว่า ณ กม.ที่ i นั้น รถมีก้อนพลังงานอยู่ n ก้อน เราจะต้องเลือกจำนวนเต็ม**<u>ไม่ลบ</u>** a และ b โดยที่ a+b = n แล้ว เราจะใช้ a ก้อนมาเป็นพลังงานในการเคลื่อนที่ และจะบรรทุกต่อไป b ก้อน การเคลื่อนที่ของรถยนต์มีกฎคือ หาก a >= b+3 แล้ว รถยนต์จะสามารถเคลื่อนที่ไปยัง กม. i+1 ได้พอดี (ไม่ว่าเราจะใช้ a มากเท่าใดก็ตาม ตราบเท่าที่ a >= b+3 แล้ว รถยนต์จะเคลื่อนที่ไปได้ 1 ช่องพอดีเสมอ ไม่มากหรือน้อยไปกว่านั้น แต่ถ้าหาก a < b+3 แล้ว รถยนต์จะอยู่กับที่ ไม่ได้ขยับไปไหน และมีก้อนพลังงานเหลือเท่ากับ b ตามที่เลือก)

ตัวอย่างการเคลื่อนที่ของรถคันนี้ที่เป็นไปได้เมื่อเริ่มต้นที่ กม. 0 ด้วยก้อนพลังงาน 30 ก้อนเป็นดังนี้

1) ณ กม. 0 เลือก a = 20 และ b = 10 เพื่อเคลื่อนที่ไปยัง กม. 1 โดยเหลือก้อนพลังงาน 10 ก้อนที่ กม. 1
2) ณ กม. 1 เลือก a = 7 และ b = 3 เพื่อเคลื่อนที่ไปยัง กม. 2 โดยเหลือก้อนพลังงาน 3 ก้อนที่ กม. 2
3) ณ กม. 2 เลือก a = 3 และ b = 0 เพื่อเคลื่อนที่ไปยัง กม. 3 โดยเหลือก้อนพลังงาน 0 ก้อนที่ กม. 3
4) ไม่เหลือก้อนพลังงานใด ๆ แล้ว รถยนต์ไม่สามารถเคลื่อนที่ต่อไปได้

จงออกแบบอัลกอริทึมแบบ Divide & Conquer เพื่อตอบคำถามต่อไปนี้ โดยในแต่ละข้อให้ตอบเป็น recurrence relation ตามที่กำหนด โดยในข้อนี้ เครื่องหมาย / จะหมายถึง “การหารปัดเศษทิ้ง” (เช่น 8/3 จะได้ 2) หากต้องการปัดเศษแบบอื่น ให้ระบุให้ชัดเจน (ข้อนี้ไม่ต้องวิเคราะห์ประสิทธิภาพ)

7.1. ให้ distance(n) คือหมายเลข กม. ที่มากที่สุดที่รถคันนี้สามารถไปถึงได้ เมื่อรถคันนี้เริ่มต้นที่ กม. 0 ด้วยก้อนพลังงาน n ก้อนนั้น

Base case: distance(n) = 0 ; n < 3
Recursion case: distance(n) = distance((n-3)/2) + 1 ; n ≥ 3

7.2. ให้ min_fuel(k) คือจำนวนก้อนพลังงานน้อยที่สุด ที่ทำให้รถคันนี้สามารถเคลื่อนที่ไปถึง กม. ที่ k ได้เมื่อเริ่มต้นที่ กม. 0 (นิสิตสามารถตอบ min_fuel(k) โดยใช้ distance(n) ด้วยก็ได้ หรือจะไม่ใช้ก็ได้)

Base case: min_fuel(k) = 0 ; k = 0
Recursion case: min_fuel(k) = 2 * min_fuel(k-1) + 3 ; k > 0

8. (10 คะแนน) ปัญหาแท่งไม้สวย ในข้อนี้คุณมีแท่งไม้ความยาว n หน่วยที่เป็นจำนวนเต็มอยู่ คุณจะต้องตัดสินใจว่าจะหั่นต้นไม้นี้เป็นแท่งๆ อย่างไรเพื่อให้ขายได้มูลค่ามากที่สุด โดยแต่ละแท่งต้องจะยาวเป็นจำนวนเต็ม และแท่งที่ยาว k หน่วยจะมีมูลค่า P[k] บาทเมื่อ 1 <= k <= n (P[k] เป็นจำนวนเต็มแต่ ไม่จำเป็นว่ายิ่งยาวจะยิ่งมูลค่ามาก)
ตัวอย่างเช่น หาก n = 8, และ P[1]=1, P[2]=5, P[3]=8, P[4]=9, P[5]=10, P[6]=17, P[7]=17, P[8]=20 แล้ว มูลค่ามากที่สุดที่เป็นไปคือ 22 (โดยการตัดเป็นยาว 2 และ 6 ซึ่งมีมูลค่า 5 และ 17 ตามลำดับ รวมกันเป็น 22)
จงออกแบบวิธีแก้ปัญหานี้ กำหนดให้ข้อมูลนำเข้าคือ n และ P[1..n] โดยเพื่อความง่าย ให้เพียงคำนวณหามูลค่าสูงสุดที่เป็นไปได้โดยไม่ต้องระบุวิธีตัด

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cin.tie(nullptr)->sync_with_stdio(false);
    int n;
    cin >> n;
    vector<int> p(n+1);
    for(int i=1; i<=n; i++) cin >> p[i];

    vector<int> dp(n+1);
    dp[0] = 0;
    for(int i=1; i<=n; i++) {
        for(int j=1; j<=i; j++) {
            dp[i] = max(dp[i], dp[i-j]+p[j]);
        }
    }
    cout << dp[n];
    return 0;
}
```

ประสิทธิภาพเชิงเวลาคือ $\Theta(n^2)$

9. (10 คะแนน) มีอาเรย์ A[1..n] อยู่ โดยที่ A[i] แต่ละช่องเป็นจำนวนเต็ม (อาจเป็นลบ) เราต้องการเลือกสมาชิกบางตัวมาจากอาเรย์นี้ โดยมีข้อกำหนดคือ ในทุก ๆ 4 ช่องที่ติดกันใด ๆ ในอาเรย์นี้ ห้ามเลือกสมาชิกมากกว่า 2 ตัว (ตัวอย่างเช่นให้ n = 6 เราสามารถเลือกตัวที่ [1,3,5] ก็ได้ [1,2,5,6] ก็ได้ หรือ [3] อย่างเดียวก็ได้ แต่ 2,4,5 ไม่ได้ (เพราะว่า ในช่อง 2 ถึง 5 ซึ่งเป็นช่อง 4 ช่องติดกันนั้นเราเลือกมา 3 ตัวซึ่งมากเกินไป หรืออีกตัวอย่างหนึ่ง ให้ n = 10 เราสามารถเลือก [1,3,5,7,9] ได้ หรือ [1,2,5,6,9,10] ก็ได้ แต่ เลือก [1,5,6,8] ไม่ได้ เพราะช่วง 5 ถึง 8 ซึ่งเป็นช่อง 4 ช่องติดกันนั้นเราเลือกมา 3 ตัวซึ่งมาเกินไป) จงออกแบบอัลกอริทึมที่หาค่าผลรวมสูงสุดที่เป็นไปได้ เมื่อกำหนดให้ A[1..n] และ n เป็นข้อมูลนำเข้า (รับประกันว่า n >= 4) ในกรณีที่ไม่เลือกตัวใดเลยให้ถือว่าผลรวมของของที่เลือกเป็น 0

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cin.tie(nullptr)->sync_with_stdio(false);

    int n;
    cin >> n;
    vector<int> A(n+1);
    for (int i=1; i<=n; i++) cin >> A[i];

    vector<vector<int>> dp(n+1, vector<int>(3));
    for (int i=1; i<=n; i++) {
        for(int j=0; j<3; j++) {
            for(int k=2-j; k<3; k++) dp[i][j] = max(dp[i][j], dp[max(0,i-j-1)][k] + A[i]);
            if (j>0) dp[i][j] = max(dp[i][j], dp[i-1][j-1]);
        }
    }
    cout << max({dp[n][0], dp[n][1], dp[n][2]});
    return 0;
}
```

ประสิทธิภาพเชิงเวลาคือ_____$\Theta(n)$_____

10. (10 คะแนน) ปัญหาทำให้วงเล็บสมดุล เราจะถือว่า string S เป็นการใส่วงเล็บที่สมดุลก็ต่อเมื่อ

- S เป็น string ว่าง หรือ
- S อยู่ในรูปแบบ $S=S_1(S_2)$ โดยที่ S_1 และ S_2 เป็น string ที่แทนการใส่วงเล็บที่สมดุล

ในข้อนี้เรามี string T ที่ความยาว n เป็นเลขคู่และใน string นี้มี (และ) อยู่อย่างละ n/2 ตัว โดย T อาจจะไม่ได้เป็นการใส่วงเล็บที่สมดุลก็ได้ คำถามคือว่าต้นทุนที่ต่ำที่สุดที่ทำให้ T เป็น string ที่แทนการใส่วงเล็บที่สมดุลนั้นเป็นกี่หน่วย เมื่อเราสามารถแก้ไข string ได้ด้วยการสลับที่อักขระที่ติดกันใน string เท่านั้น (ห้ามเพิ่ม ลบ ย้ายที่อื่นใด) โดยการสลับนี้มีต้นทุน 1 หน่วย

ตัวอย่างเช่น หาก T เป็น ())()(คำตอบจะเป็น 2 ซึ่งเกิดจากการสลับตำแหน่งที่ 3 และ 4 ผลลัพธ์คือ ()())(ตามด้วย การลลับตำแหน่งที่ 5 และ 6 ผลลัพธ์ คือ ()()()

หมายเหตุ ให้ T[i] คืออักขระตำแหน่งที่ i ของ T โดยอักขระตัวซ้ายสุดคือ T[1] และอักขระตัวขวาสุดคือ T[n]

จงออกแบบวิธีแก้ปัญหานี้ กำหนดให้ข้อมูลนำเข้าคือ T และ n โดยให้คำนวณหาต้นทุนที่ต่ำที่สุดโดยไม่ต้องระบุวิธี (จะได้คะแนนสูงสุดไม่เกินครึ่งหนึ่งหากเวลาในการคำนวณเป็น $O(n^2)$ และ การที่จะได้เต็มนั้นจำเป็นที่จะต้องใช้เวลาในการคำนวณเป็น O(n))

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cin.tie(nullptr)->sync_with_stdio(false);
    string T;
    cin >> T;
    int n = T.length(), ans = 0;
    T.insert(T.begin(), '#');
    deque <pair<char, int>> dq;
    for (int i=1; i <= n; i++) {
        if (!dq.empty() && dq.back().first == '(' and T[i] == ')') dq.pop_back();
        else if (!dq.empty() && dq.front().first == ')' and T[i] == '(') {
            ans += i - dq.front().second;
            dq.pop_front();
            if (!dq.empty()) dq.front().second++;
        }
        else dq.emplace_back(T[i], i);
    }
    cout << ans;
    return 0;
}

// Do I need to explain how I came up with the solution?! Just greedy :)
```

ประสิทธิภาพเชิงเวลาคือ $\Theta(n)$