2110101 : การทำโปรแกรมคอมพิวเตอร์

การโปรแกรมแบบเวียนเกิด

ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

หัวข้อ

- Recurrences
- Iterative vs. Recursive
- Examples

ความสัมพันธ์เวียนเกิด (Recurrences)

- การเขียนความสัมพันธ์ของจำนวนเต็มในลำดับ

0, 1, 2, 3, 4, ... $a_n = a_{n-1} + 1$ เมื่อ $n > 0$, $a_0 = 0$

3, 5, 7, 9, ... $a_n = a_{n-1} + 2$ เมื่อ $n > 0$, $a_0 = 3$

0, 1, 3, 6, 10, 15, ... $a_n = a_{n-1} + n$ เมื่อ $n > 0$, $a_0 = 0$

0, 1, 1, 2, 3, 5, 8,... $a_n = a_{n-1} + a_{n-2}$ เมื่อ $n > 1$,
$a_0 = 0$, $a_1 = 1$

0, 1, 3, 6, 10, 15, ... (Iterative)

- เขียนโปรแกรมหาตัวที่ n
- รู้ว่า $a_n = (0+1+2+...+n)$

```
import jlab.JLabIO;

public class Sum2N {
  public static void main(String[] args) {
    int n = JLabIO.readInt("n = ");
    System.out.println(a(n));
  }
  static int a(int n) {
    int s = 0;
    for (int i = 0; i <= n; i++)
      s += i;
    return s;
  }
}
```

0, 1, 3, 6, 10, 15, ... (Recursive)

- เขียนโปรแกรมหาตัวที่ n
- รู้ว่า $a_n = a_{n-1} + n$ เมื่อ n > 0, $a_0 = 0$

โปรแกรมเวียนเกิด
(recursive program)

```
import jlab.JLabIO;

public class Sum2N {
  public static void main(String[] args) {
    int n = JLabIO.readInt("n = ");
    System.out.println(a(n));
  }
  static int a(int n) {
    if (n <= 0)
      return 0;
    else
      return a(n-1) + n;
  }
}
```

$$a_n = \begin{cases} 0 & \text{if } n \le 0 \\ a_{n-1} + n & \text{otherwise} \end{cases}$$

0, 1, 1, 2, 3, 5, 8, 13, ... (Iterative)

- เขียนโปรแกรมหาตัวที่ n

```
import jlab.JLabIO;

public class Fibo {
  public static void main(String[] args) {
    int n = JLabIO.readInt("n = ");
    System.out.println(a(n));
  }
  static int a(int n) {
    int p = 0, q = 1, t = n;
    for (int i = 2; i <= n; i++) {
      t = p + q;
      p = q;
      q = t;
    }
    return t;
  }
}
```

0, 1, 1, 2, 3, 5, 8, 13, ... (Recursive)

- เขียนโปรแกรมหาตัวที่ n
- รู้ว่า $a_n = a_{n-1} + a_{n-2}$ เมื่อ n > 1, $a_0 = 0$, $a_0 = 1$

```
import jlab.JLabIO;

public class Fibo {
  public static void main(String[] args) {
    int n = JLabIO.readInt("n = ");
    System.out.println(a(n));
  }
  static int a(int n) {
    if (n < 2) return n
    return a(n-1) + a(n-2);
  }
}
```

$$a_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ a_{n-1} + a_{n-2} & \text{otherwise} \end{cases}$$

โปรแกรมหาผลรวมในอาเรย์ (Iterative)

- ต้องการ d[0] + d[1] + d[2] + ... d[d.length-1]

```
public static int sum(int [] d) {
  int s = 0;
  for (int i = 0; i < d.length; i++)
    s += d[i];
  return s;
}
```

โปรแกรมหาผลรวมในอาเรย์ (Recursive)

- ต้องการ d[0] + d[1] + d[2] + ... d[d.length-1]
- กำหนดให้ s_n = d[0] + d[1] + ... + d[n-1]
- แสดงว่า s_{n-1} = d[0] + d[1] + ... + d[n-2]
- จะได้ว่า $s_n = s_{n-1}$ + d[n-1] เมื่อ n > 0, $s_0 = 0$
- ขอเขียนเป็น
  s(d,n)= s(d,n-1) + d[n-1] เมื่อ n > 0, s(0) = 0

```
public static int sum(int[] d) {
  return sum(d, d.length);
}
private static int sum(int[] d, int n) {
  if (n <= 0) return 0;
  return sum(d, n - 1) + d[n - 1];
}
```

โปรแกรมหาค่ามากสุดในอาเรย์

- กำหนดให้ m_n = max(d[0], d[1], ..., d[n-1])
- แสดงว่า m_{n-1} = max(d[0], d[1], ..., d[n-2])
- ได้ m_n = max(m_{n-1} , d[n-1]) เมื่อ n > 0, s_1 = d[0]
- ขอเขียน m(d,n)= max(m(d,n-1), d[n-1]) n > 0, s(1) = d[0]

```
public static int max(int[] d) {
  return max(d, d.length);
}
private static int max(int[] d, int n) {
  if (n == 1) return d[0];
  int m = max(d, n - 1);
  return (m > d[n-1] ? m : d[n-1]);
}
```

โปรแกรมหาตำแหน่งของค่ามากสุดในอาเรย์

```
public static int max(int[] d) {
  return max(d, d.length);
}
private static int max(int[] d, int n) {
  if (n == 1) return d[0];
  int m = max(d, n - 1);
  return (m > d[n-1] ? m : d[n-1]);
}
```

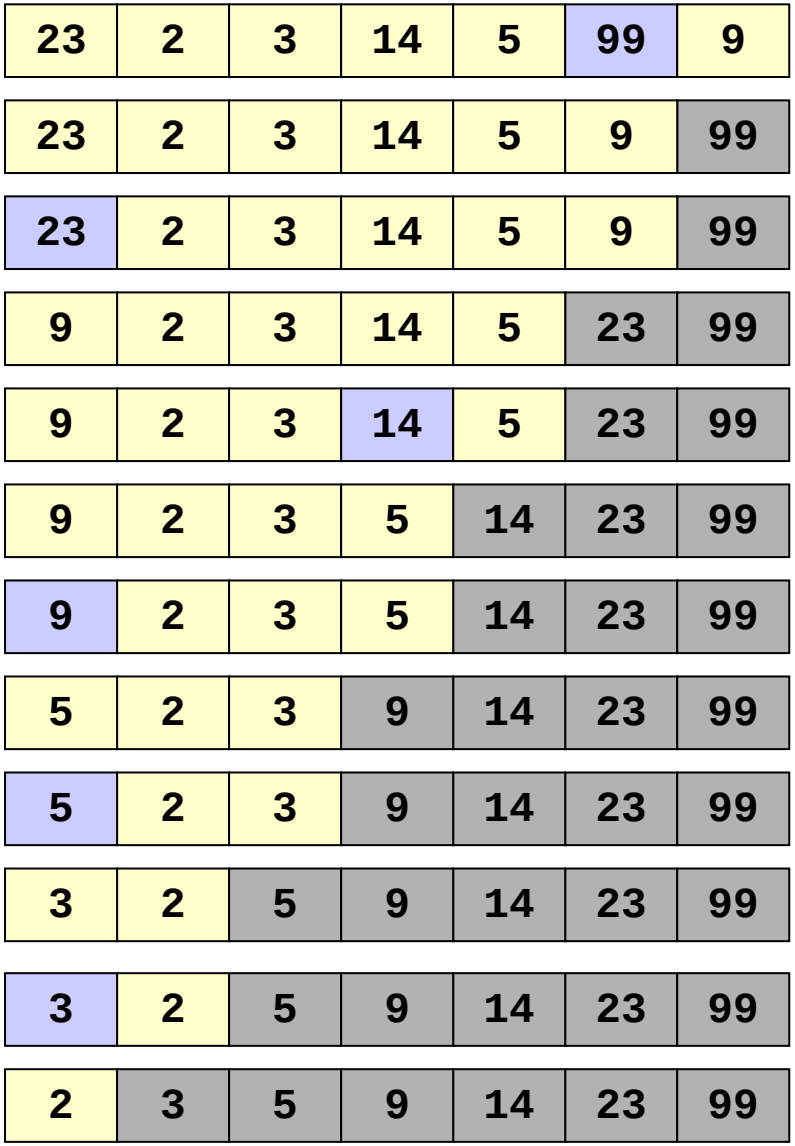
คืนค่ามากสุด

```
public static int maxI(int[] d) {
  return maxI(d, d.length);
}
private static int maxI(int[] d, int n) {
  if (n == 1) return 0;
  int i = maxI(d, n - 1);
  return (d[i] > d[n-1] ? i : n-1);
}
```

คืน index i ที่ d[i] มีค่ามากสุด

การเรียงลำดับแบบเลือก

- มองอาเรย์เป็นสองส่วน
  - ส่วนขวา เรียงเสร็จแล้ว
  - ส่วนซ้าย ยังไม่เรียง
- ส่วนซ้าย
  - หาตัวที่มีค่ามากสุด
  - สลับกับตัวสุดท้าย
- หลังสลับ
  - ส่วนซ้ายลดหนึ่งตัว ส่วนขวาเพิ่มหนึ่งตัว
  - แล้วก็ไปเรียงลำดับส่วนทางซ้ายต่อไป



การเรียงลำดับแบบเลือก

```
public static void selectionSort(int[] data) {
  int i, t;
  for (int k = data.length; k > 1; k--) {
    i = maxI(data, k);
    t = data[i];
    data[i] = data[k - 1];
    data[k - 1] = t;
  }
}
```

Iterative

```
public static void selectionSort(int[] data) {
  selectionSort( data, data.length );
}
private static void selectionSort(int[] data, int n) {
  int i, t;
  if (n < 2) return;
  i = maxI(data, n);
  t = data[i];
  data[i] = data[n-1];
  data[n-1] = t;
  selectionSort( data, n-1 );
}
```

Recursive

การคำนวณ a^p mod m

- a^p mod m เป็นการคำนวณที่ใช้บ่อยในการเข้ารหัสลับ
- ตัวอย่างที่ 1 : 2^{20} % 31 = ?
  - คำนวณ 2^{20} ได้ 1048576 จากนั้น % 31 ได้ 1
- ตัวอย่างที่ 2 : 2^{100} % 31 = ?
  - คำนวณ 2^{101} ได้ 2535301200456458802993406410752 จากนั้น % 31 ได้ 2
  - ทำอีกแบบ : ใช้ 2^{20} % 31 ให้เป็นประโยชน์
    - 2^{101} % 31 = 2 x $(2^{20})^5$ % 31
    - ใช้ความรู้ที่ว่า (ab % m) = (a % m)(b % m)
    - 2^{101} % 31 = 2 x $(2^{20})^5$ % 31 = (2 % 31)$(2^{20}$ % 31$)^5$ = 2

การคำนวณ $a^P \bmod m$

$$a^p \% m = \begin{cases} 1 & \text{if } p = 0 \\ (a^{p/2} \% m)^2 \% m & \text{if } p \text{ is even} \\ 2(a^{(p-1)/2} \% m)^2 \% m & \text{if } p \text{ is odd} \end{cases}$$

```
public static int powerMod(int a, int p, int m) {
  if (p == 0) return 1;
  if (p == 1) return a % m;
  int k = powerMod(a, p / 2, m);
  k = (k * k) % m;
  if ((p % 2) == 1) k = (2 * k) % m;    // p is odd
  return k;
}
```

โปรแกรมแบบเวียนเกิด

- โปรแกรมที่มีการเรียกตัวเอง
- การทำงานแบ่งเป็นกรณีๆ
  - กรณีพื้นฐาน ทำเสร็จได้ทันที
  - กรณีอื่น เรียกตัวเอง โดยขนาดของปัญหาต้องเล็กลง
- เขียนง่าย (?) สวย (?) แต่ทำงานช้ากว่า