2110211 โครงสร้างข้อมูลเบื้องต้น

บทนำ

สมชาย ประสิทธิ์จูตระกูล

เรียนอะไรบ้าง ?

- วิธีสร้าง
- วิธีวิเคราะห์ประสิทธิภาพ
- วิธีการนำไปใช้งาน
- ของโครงสร้างข้อมูลพื้นฐานประเภทต่าง ๆ

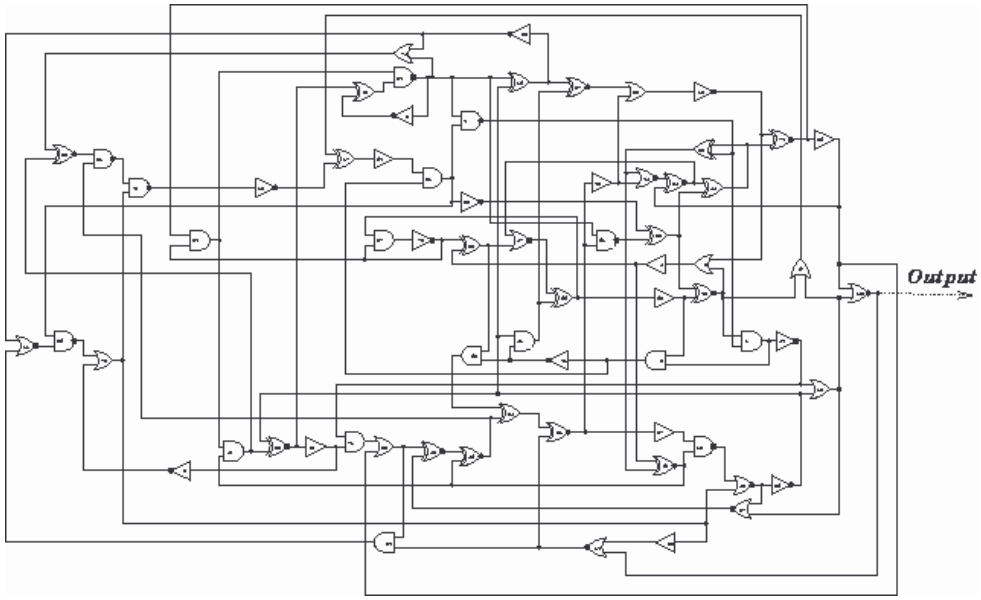
ต้องรู้อะไรมาก่อน ?

- 2110101 Computer Prog.
- 2110210 Programming Meth.
- 2110200 Discrete Structures
- 2110250 Computer Organization

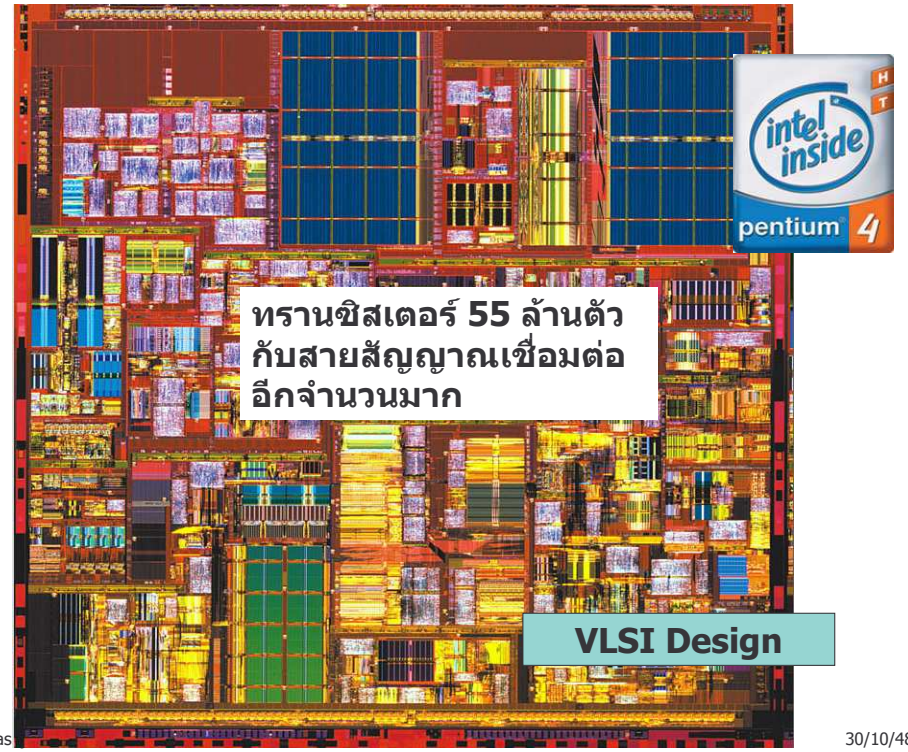
โครงสร้างข้อมูลคืออะไร

- วิธีการจัดเก็บและจัดการข้อมูลให้
  - ตรงตามความต้องการ
  - รวดเร็ว
  - ประหยัดเนื้อที่
  - เข้าใจง่าย

จะเขียนโปรแกรมจัดเก็บและจัดการอย่างไร ?

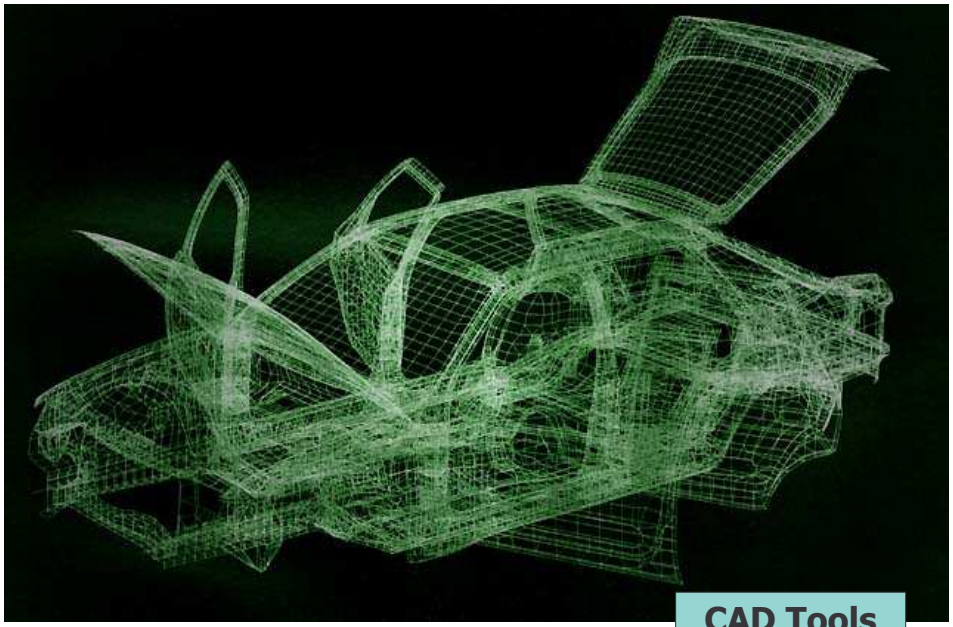


Logic Design Tools



VLSI Design

Games



CAD Tools

และอื่น ๆ

- Google เก็บเอกสารอย่างไร ทำให้ค้นได้รวดเร็ว
- jvm เก็บออปเจกต์ต่าง ๆ คลาสต่าง ๆ ตัวแปรต่าง ๆ thread ต่าง ๆ ไว้อย่างไร
- Word processor เก็บตัวอักษร คำ ข้อความ ย่อหน้า สูตร รูป และอื่น ๆ ในเอกสารภายในหน่วยความจำอย่างไร ขณะที่เรากำลังใช้งาน
- ...

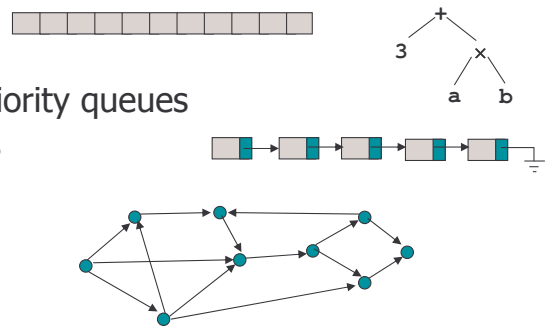
ล้วนเป็นปัญหาที่เกี่ยวกับโครงสร้างข้อมูล

ข้อมูลต่าง ๆ

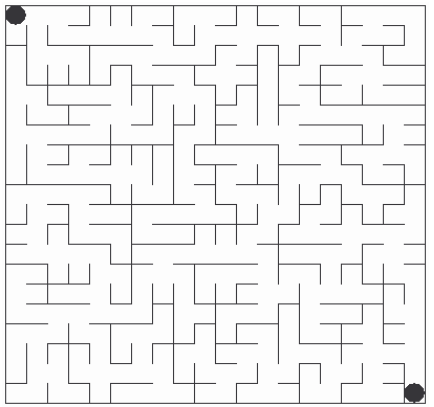
- บางชนิด ตัวภาษามีให้แล้ว (primitive)
  - int, double, char, boolean, ...
- บางชนิดคลังคำสั่งของระบบมีให้ใช้ (class)
  - String, Color, BigDecimal, ArrayList, HashMap, ...
- และมีอีกมากมายที่ต้องออกแบบและสร้างเอง

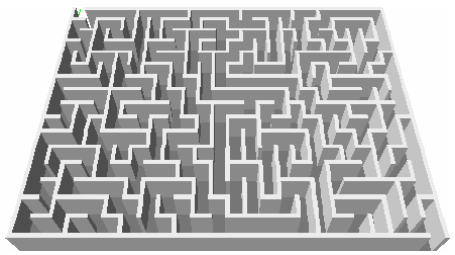
2110211 : นำเสนอโครงสร้างข้อมูล

- พื้นฐาน
- คนอื่น ๆ เขาคิดและใช้กันแพร่หลาย
- มีให้ใช้อยู่แล้วในคลังคำสั่ง
- เช่น
  - arrays
  - linked lists
  - stacks, queues, priority queues
  - trees, search trees
  - hash tables
  - disjoint sets
  - graphs

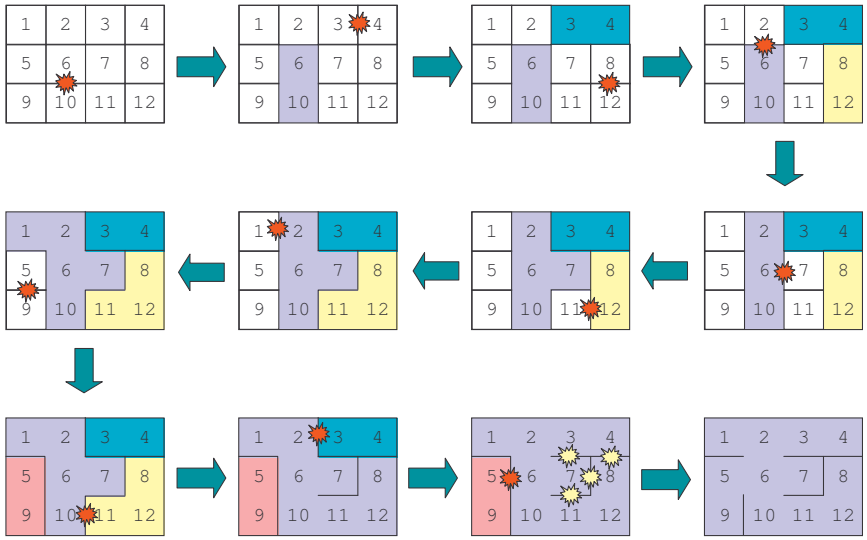


ตัวอย่าง : การสร้างเขาวงกต

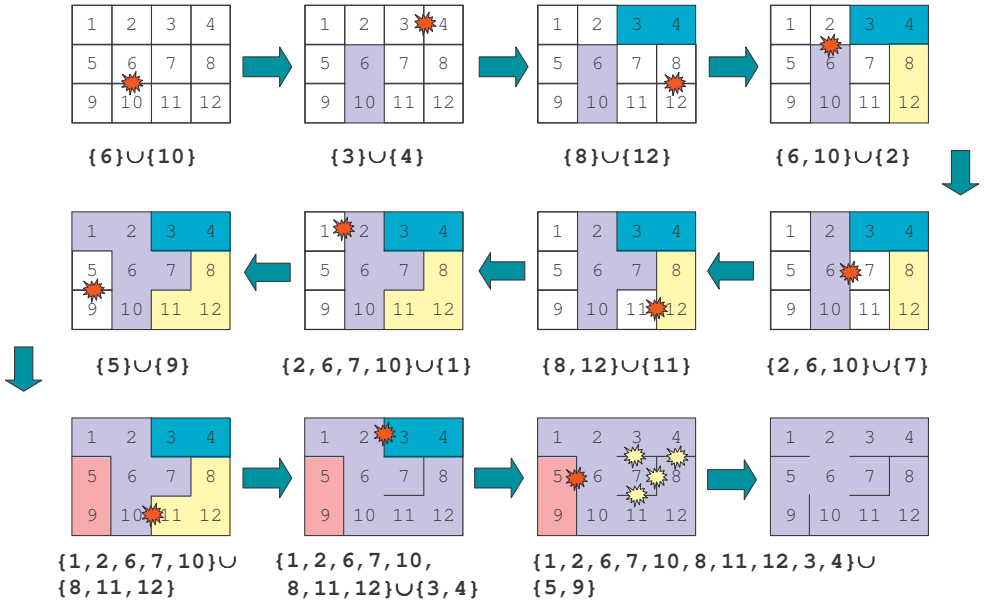




วิธีสร้างเขาวงกต



วิธีสร้างเขาวงกต



วิธีสร้างเขาวงกต

- ต้องการที่เก็บข้อมูลแบบเซต (disjoint sets)
  - มีการยูเนี่ยนเซตสองเซต
  - มีการค้นว่าข้อมูลที่ให้สองตัว อยู่ในเซตเดียวกันหรือไม่

```
void randomlyRemoveWalls(Walls[] walls, int numRooms) {
  // {0}, {1}, {2}..., {numRooms-1}
  DisjointSets sets = new DisjointSets(numRooms);

  for (int i=0; i<walls.length; i++) {
    int s1 = sets.find(walls[i].getRoom1Number());
    int s2 = sets.find(walls[i].getRoom2Number());
    if (s1 != s2) {
        sets.union(s1, s2);
        walls[i].setVisible(false);
    }
  }
}
```

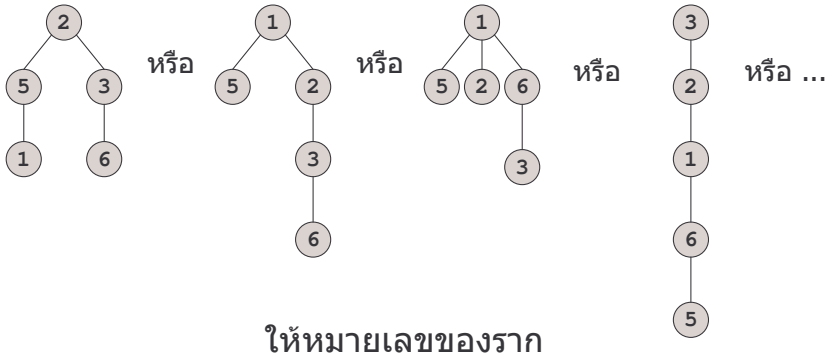
เริ่มต้นให้แต่ละห้องมีหมายเลขกำกับ

DisjointSets ของจำนวนเต็ม 0 ถึง n-1

```
public class DisjointSets {
  ...
  public DisjointSets(int n) {
    ...
  }
  public void union(int s1, int s2) {
    ...
  }
  public int find(int x) {
    ...
  }
}
```

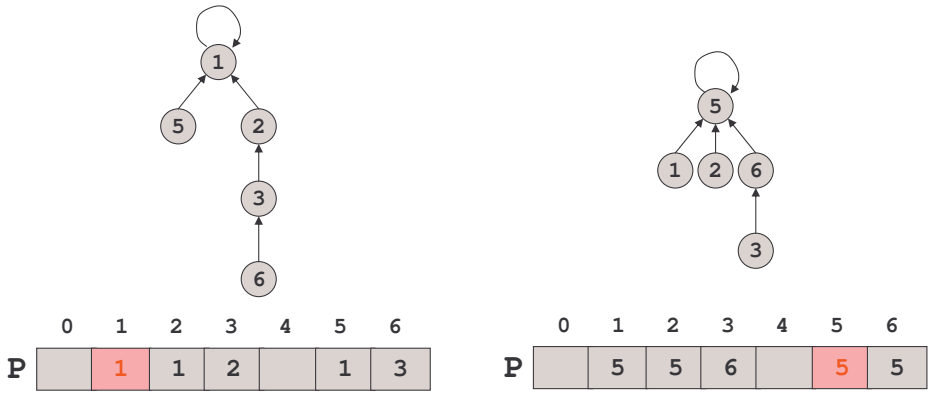
วิธีแทนเซตด้วยต้นไม้

{2,5,1,3,6}



ให้หมายเลขของราก
แทน หมายเลขเซต

วิธีแทนต้นไม้ด้วยอาเรย์

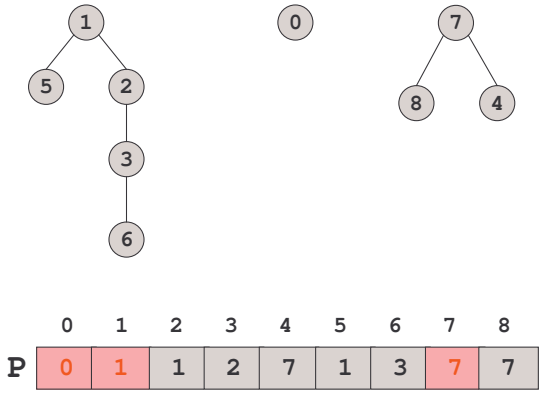


parent node ของ k อยู่ที่ P[k]

ถ้า P[k] มีค่าเป็น k แสดงว่า k เป็นราก

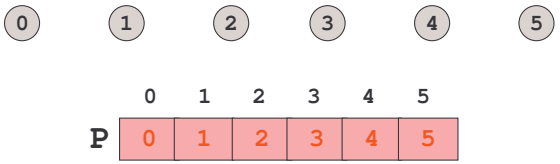
หลาย ๆ เซตใช้อาเรย์ร่วมกัน

{2,5,1,3,6}, {0}, {7,8,4}



เริ่มต้น ให้เซตละหนึ่งสมาชิก

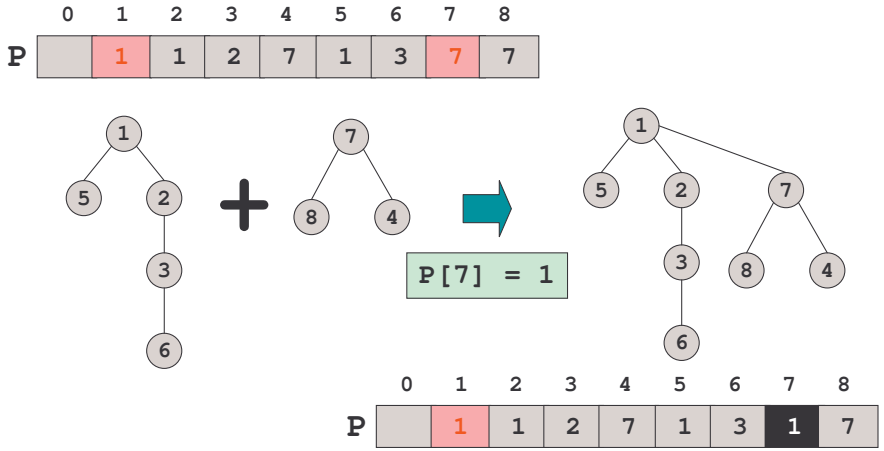
{0}, {1}, {2}, {3}, {4}, {5}




```
public class DisjointSets {
  private int[] P;
  public DisjointSets(int n) {
    P = new int[n];
    for(int i=0; i<n; i++) P[i] = i;
  }
  ...
}
```

วิธียูเนี่ยนเซตสองเซต

- รวมต้นไม้สองต้นให้เป็นต้นเดียว
  - นำรากของต้นหนึ่ง มาต่อเป็นลูกของรากอีกต้นหนึ่ง



วิธียูเนี่ยนเซตสองเซต

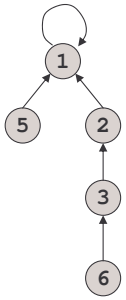
- รวมต้นไม้สองต้นให้เป็นต้นเดียว
  - นำรากของต้นหนึ่ง มาต่อเป็นลูกของรากอีกต้นหนึ่ง

```
public class DisjointSets {
  private int[] P;
  ...
  public void union(int s1, int s2) {
    P[s1] = s2;
  }
}
```


วิธีหาว่าข้อมูล x เป็นสมาชิกของเซตหมายเลขใด

จาก x วิ่งไล่ขึ้นไปจนถึงราก ก็จะได้หมายเลขเซต

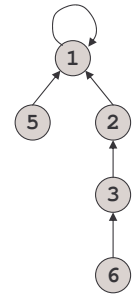
```
public class DisjointSets {
  private int[] P;
  ...
  public int find(int x) {
    while( P[x] != x ) {
      x = P[x];
    }
    return x;
  }
}
```



วิธีหาว่าข้อมูล x เป็นสมาชิกของเซตหมายเลขใด

ถ้า x ไม่ใช่ราก
หมายเลขเซตของ x ย่อมเท่ากับหมายเลขเซตของ P[x]

```
public class DisjointSets {
  private int[] P;
  ...
  public int find(int x) {
    if (P[x] == x) return x;
    return findSet( P[x] );
  }
}
```

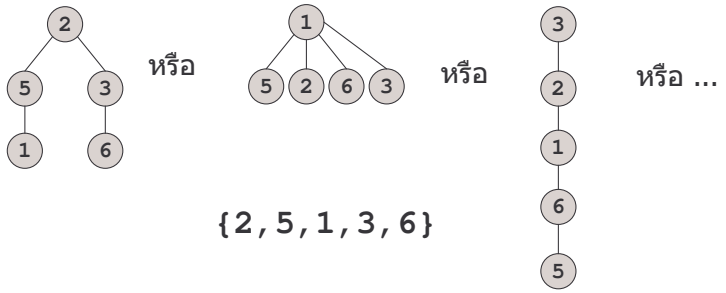


DisjointSets

```
public class DisjointSets {
  private int[] P;

  public DisjointSets(int n) {
    P = new int[n];
    for(int i=0; i<n; i++) P[i] = i;
  }

  public void union(int s1, int s2) {
    P[s1] = s2;
  }

  public int find(int x) {
    if (P[x] == x) return x;
    return find( P[x] );
  }
}
```

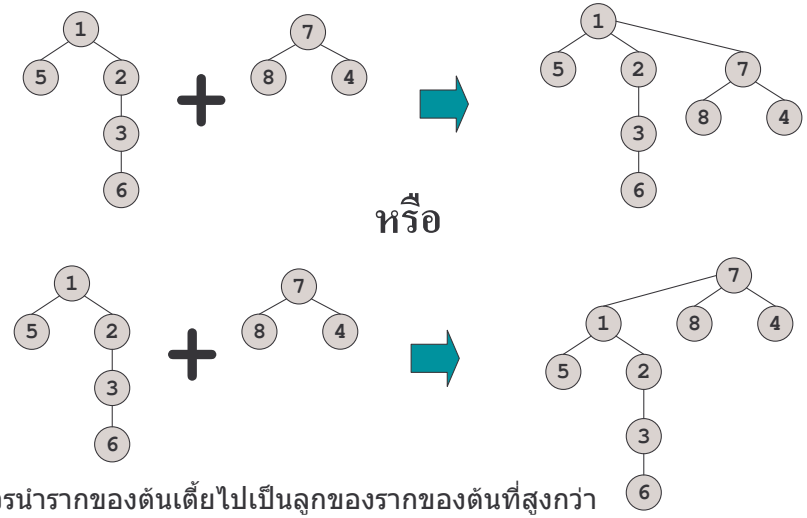

ประสิทธิภาพการทำงาน

- union : ใช้เวลาคงตัวเสมอ
- find : การทำงานขึ้นกับว่าหาสมาชิกตัวที่อยู่ใกล้หรือไกลจากราก
  - เซตหนึ่งแทนได้ด้วยต้นไม้หลากหลายแบบ
  - ควรเลือกแบบที่ได้ต้นไม้เตี้ย ๆ



ทำอย่างไรให้ต้นไม้เตี้ย ๆ

- ต้องระวังตอน union

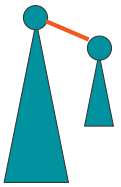


ควรนำรากของต้นเตี้ยไปเป็นลูกของรากของต้นที่สูงกว่า

วิธี union แบบฉลาด

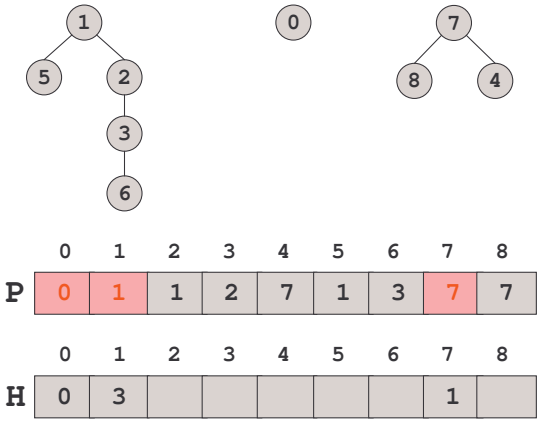
```
public class DisjointSets {
  private int[] P;
  ...

  public void union(int s1, int s2) {
    if (height(s1) > height(s2)) {
      P[s2] = s1;
    } else {
      P[s1] = s2;
    }
  }
}
```

แนวคิด



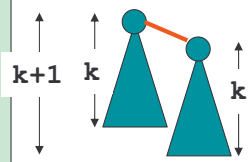
จะรู้ความสูงของต้นไม้ได้อย่างไร ?

- ความสูงหาจากอาเรย์ P ได้ แต่ช้า
- มีที่เก็บความสูง แล้วคอยใส่ค่าให้ถูกต้อง



วิธี union แบบอาศัยความสูง

```
public class DisjointSets {
  private int[] P;
  private int[] H;
  ...
  public void union(int s1, int s2) {
    if (H[s1] > H[s2]) {
      P[s2] = s1;
    } else {
      P[s1] = s2;
      if (H[s1] == H[s2]) H[s2]++;
    }
  }
}
```



ต้นไม้สูงขึ้นเมื่อรวมต้นไม้ที่สูงเท่ากัน

DisjointSets แบบยูเนี่ยนอาศัยความสูง

```
public class DisjointSets {
  private int[] P, H;

  public DisjointSets(int n) {
    P = new int[n];
    for(int i=0; i<n; i++) {P[i] = i; H[i] = -1;}
  }
  public void union(int s1, int s2) {
    if (H[s1] > H[s2]) {
      P[s2] = s1;
    } else {
      P[s1] = s2;
      if (H[s1] == H[s2]) H[s2]++;
    }
  }
  public int find(int x) {
    if (P[x] == x) return x;
    return find( P[x] );
  }
}
```


การบ้าน

- การยูเนี่ยนแบบอาศัยความสูง ทำให้ได้ต้นไม้เตี้ย ๆ จริงหรือ ?
- ลองเริ่มสร้าง disjoint sets ที่มี 16 เซต
- แล้วค่อย ๆ union เซตไปเรื่อย ๆ 15 ครั้ง สุดท้ายก็ต้องเหลือเพียงเซตเดียว
- คำถาม :
  - จงหาลำดับการ union เซต 15 ครั้งที่ทำให้ได้ต้นไม้ที่สูงสุด ๆ เท่าที่จะทำได้ (เมื่อใช้ union แบบฉลาด)
  - พอจะสรุปเป็นกรณีทั่วไปได้หรือไม่ว่า disjoint sets ที่มีสมาชิกทั้งหมดรวม n ตัว จะแทนด้วยต้นไม้ที่สูงไม่เกินเท่าใด (ตอบเป็นฟังก์ชันของ n)

การบ้าน

- ออกแบบใหม่ให้ DisjointSets ใช้แค่อาเรย์เดียว
  - ข้อสังเกต : อาเรย์ P เก็บแต่จำนวนไม่ติดลบ ไม่คุ้มเลย!!!

```
public class DisjointSets {
  private int[] P;
  private int[] H;
  ...
}
```

```
public class DisjointSets {
  private int[] PH;
  ...
}
```

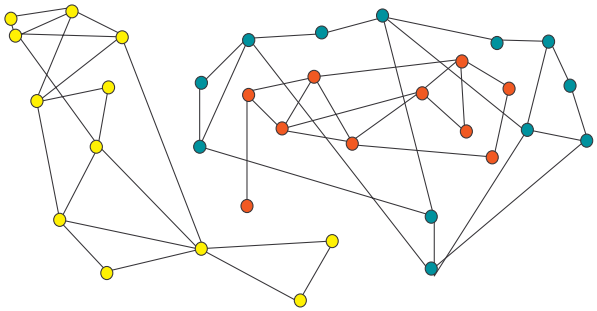

การบ้าน

- ถ้าเปลี่ยน find ให้ทำงานดังแสดงข้างล่างนี้ อะไรจะเกิดขึ้น ?

```
public class DisjointSets {
  private int[] P, H;

  ...

  public int find(int x) {
    if (P[x] == x) return x;
    return P[x] = find( P[x] );
  }
}
```


การบ้าน

- จงใช้ disjoint sets ช่วยแก้ปัญหา
  - การหา connected components ของ undirected graph



การบ้าน

- จงใช้ disjoint sets ช่วยแก้ปัญหา clustering