SEARCHING

การค้นข้อมูล (Searching)

- ข้อมูลที่เก็บไว้ประกอบด้วย
  - คีย์ (Key)
  - ข้อมูลที่เกี่ยวข้องกับคีย์
- การค้นข้อมูล
  - ที่เก็บในหน่วยความจำหลัก (internal search)
  - ที่เก็บในหน่วยความจำสำรอง (external search)
- การค้นแบบลำดับ (sequential search)
- การค้นแบบทวิภาค (binary search)

Sequential Search

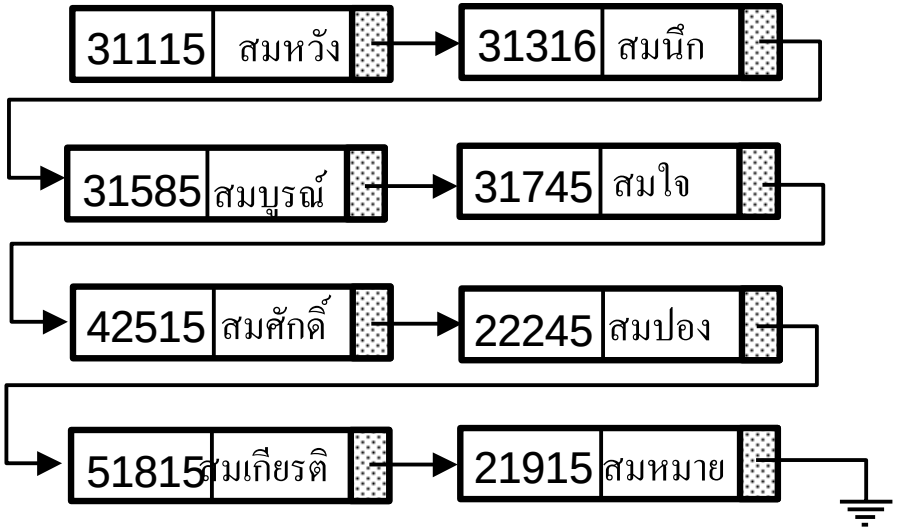
```
typedef  int       KeyType;
typedef  struct    ItemTag {
         KeyType   key;
         char      Name[30];
         ...
} ItemType;

typedef  struct    ListTag {
         int       count;
         ItemType  data[ MAX ];
} ListType;
```

0	31115	สมหวัง
1	31316	สมนึก
2	31585	สมบูรณ์
3	31745	สมใจ
4	42515	สมศักดิ์
5	22245	สมปอง
6	51815	สมเกียรติ
7	21915	สมหมาย

```
int SeqSearch( ListType pList, KeyType TargetKey )
{
  int   loc;

  for ( loc=0; loc<pList->count; loc++ )
    if ( EQ( pList->data[loc].key == TargetKey ) ) return( loc );
  return( -1 );
}
```

Sequential Search

```
typedef  struct    NodeTag {
         ItemType  data;
         struct NodeTag  *pNext;
} NodeType;

typedef  struct    ListTag;
     NodeTag   *pHeader;
} ListType;
```




```
NodeType *SeqSearch( ListType *pList, KeyType TargetKey )
{
 NodeType   *pLoc;

 for ( pLoc=pList->pHeader; ploc!=NULL; ploc=pLoc->pNext )
```

Analysis of Sequential Search

- จำนวนครั้งในการเปรียบเทียบคีย์ระหว่างการค้น เป็นปัจจัยในการวิเคราะห์เวลาในการค้นข้อมูล
- สมมติให้มีข้อมูล n ตัว

  ถ้าหาคีย์ที่ต้องการ ไม่พบ ต้องมีการเปรียบเทียบ n ครั้ง

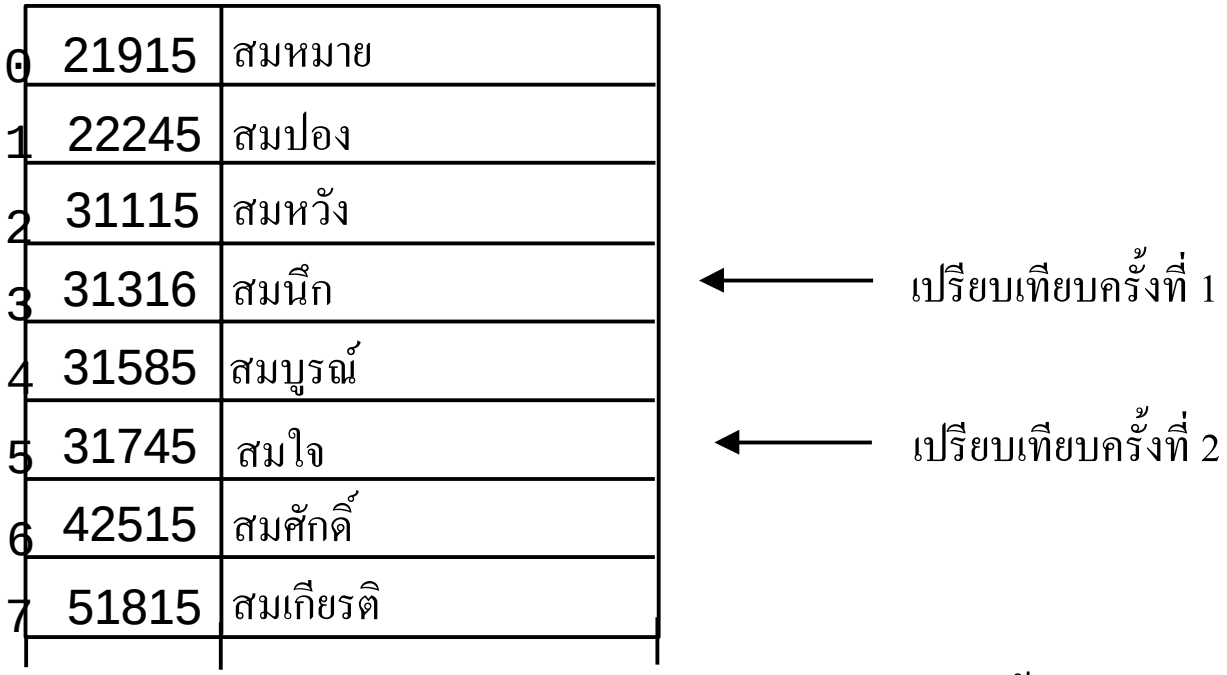
  ถ้าหาคีย์ที่ต้องการ พบ

  จำนวนการเปรียบเทียบขึ้นกับตำแหน่งของข้อมูล

  ให้ความน่าจะเป็นในการค้นข้อมูลแต่ละตัวมีโอกาสเท่ากัน

$$\frac{1+2+3+\ldots+n}{n} = \frac{0.5n(n+1)}{n} = 0.5(n+1)$$

การค้นแบบทวิภาค (Binary Search)



ต้องการทราบชื่อของรหัส 31745 จะมีการเปรียบเทียบข้อมูล 2 ครั้ง

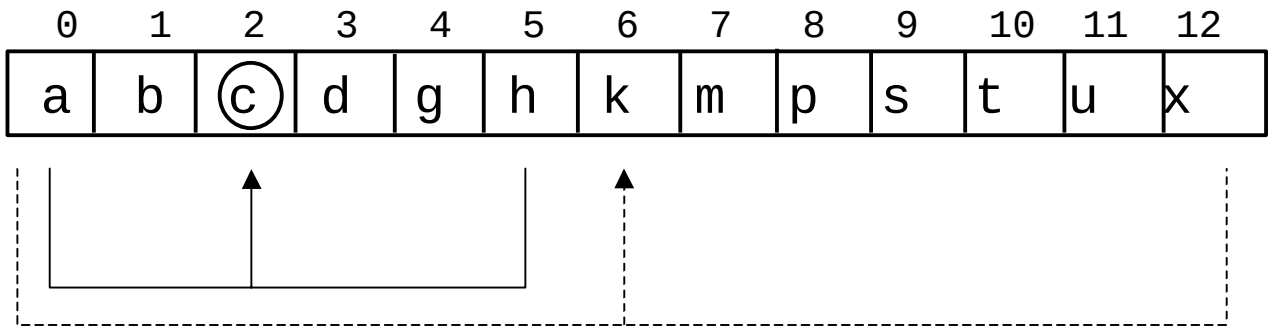
สังเกตุว่าข้อมูลจะต้องถูกเก็บในลักษณะที่

- เรียงลำดับตามคีย์
- สามารถอ้างถึงข้อมูลตัวใดก็ได้ในเวลาที่เท่ากัน

จึงไม่เหมาะกับกรณีที่เก็บข้อมูลแบบ linked list

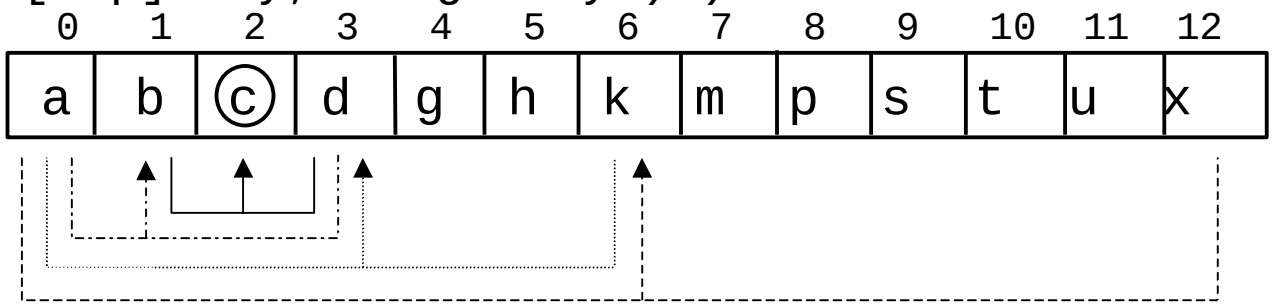
Binary Search : Binary2

```
int Binary2( ListType *pList, KeyType TargetKey )
{
  int        top, bot, mid;

  bot = 0; top = pList->count - 1;
  while ( bot <= top ) {
    mid = (bot + top) / 2;
    if ( EQ( pList->data[mid].key, TargetKey ) )
      return( mid );
    else
    if ( LT( pList->data[mid].key, TargetKey ) )
      bot = mid + 1;
    else
      top = mid - 1;
  }
  return( -1 );
}
```

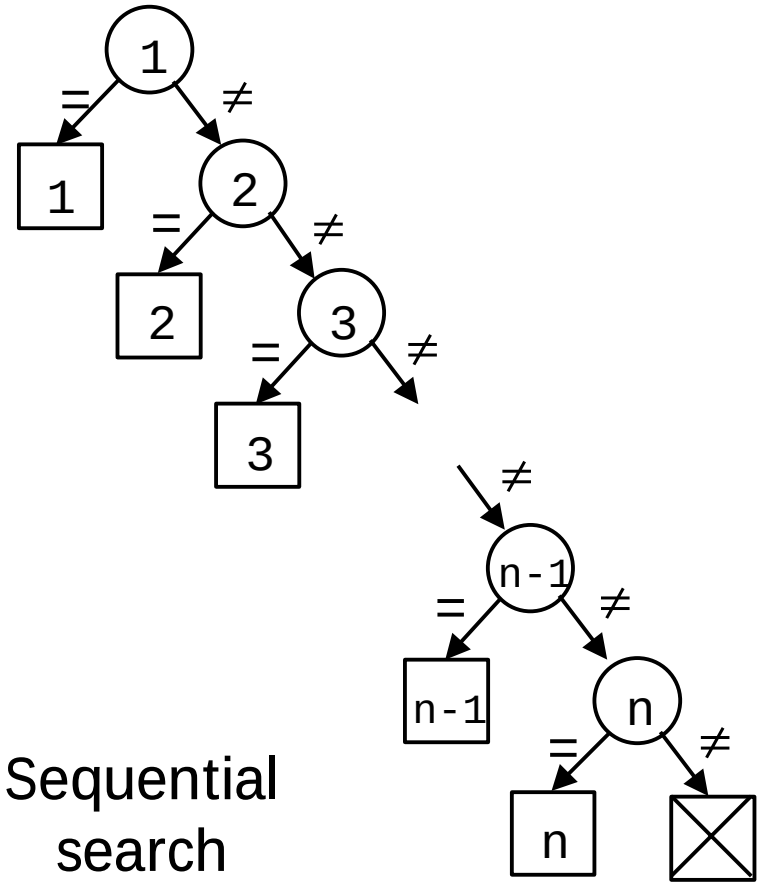


Binary Search : Binary1

```
int Binary1( ListType *pList, KeyType TargetKey )
{
  int        top, bot, mid;

  bot = 0; top = pList->count - 1;
  if ( top == -1 ) return( -1 );
  while ( bot < top ) {
    mid = (bot + top) / 2;
    if ( LT( pList->data[mid].key, TargetKey ) )
      bot = mid + 1;
    else
      top = mid;
  }
  if ( EQ( pList->data[top].key, TargetKey ) )
    return( top );
  else
    return( -1 );
}
```

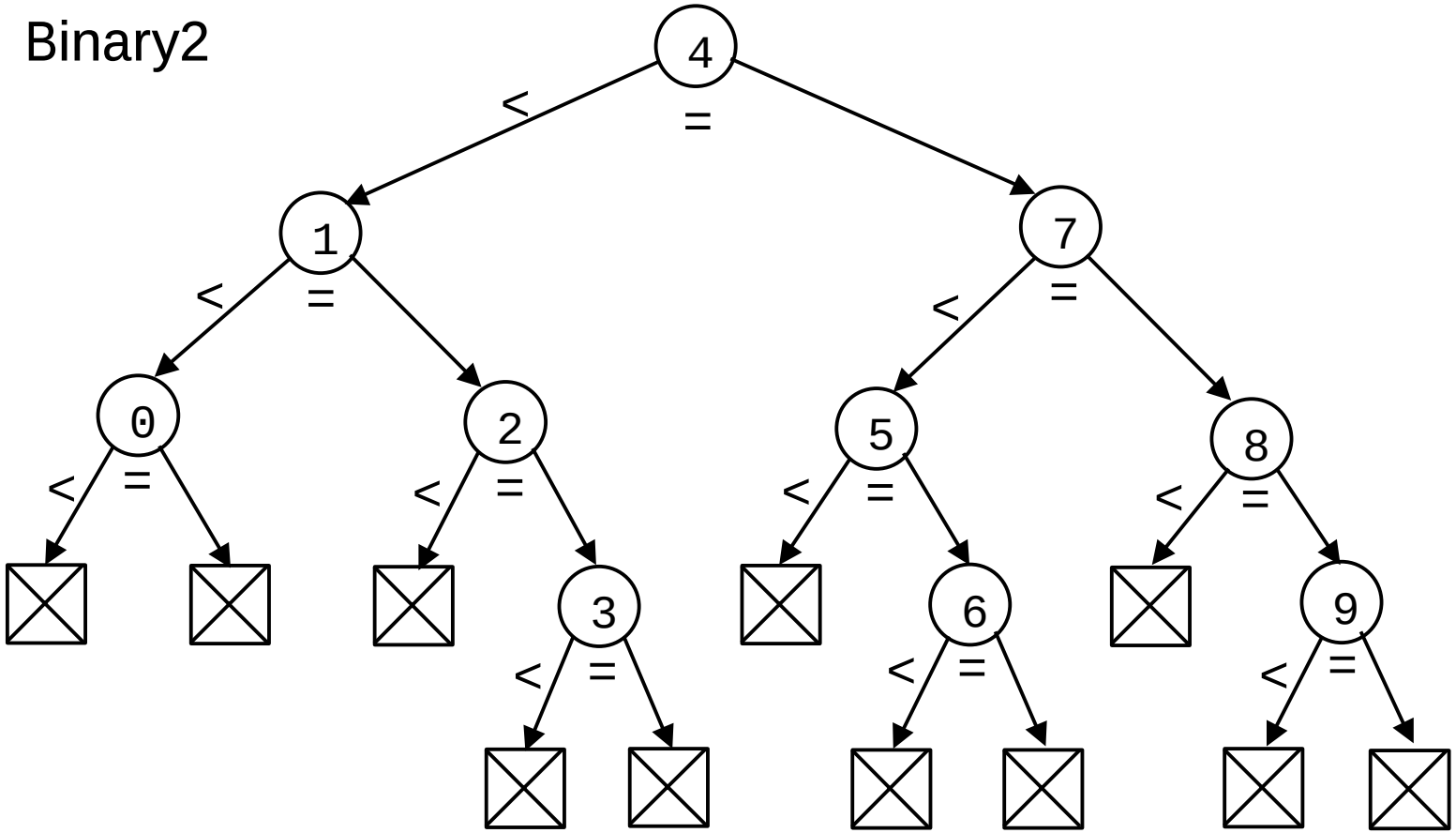


Comparison Trees



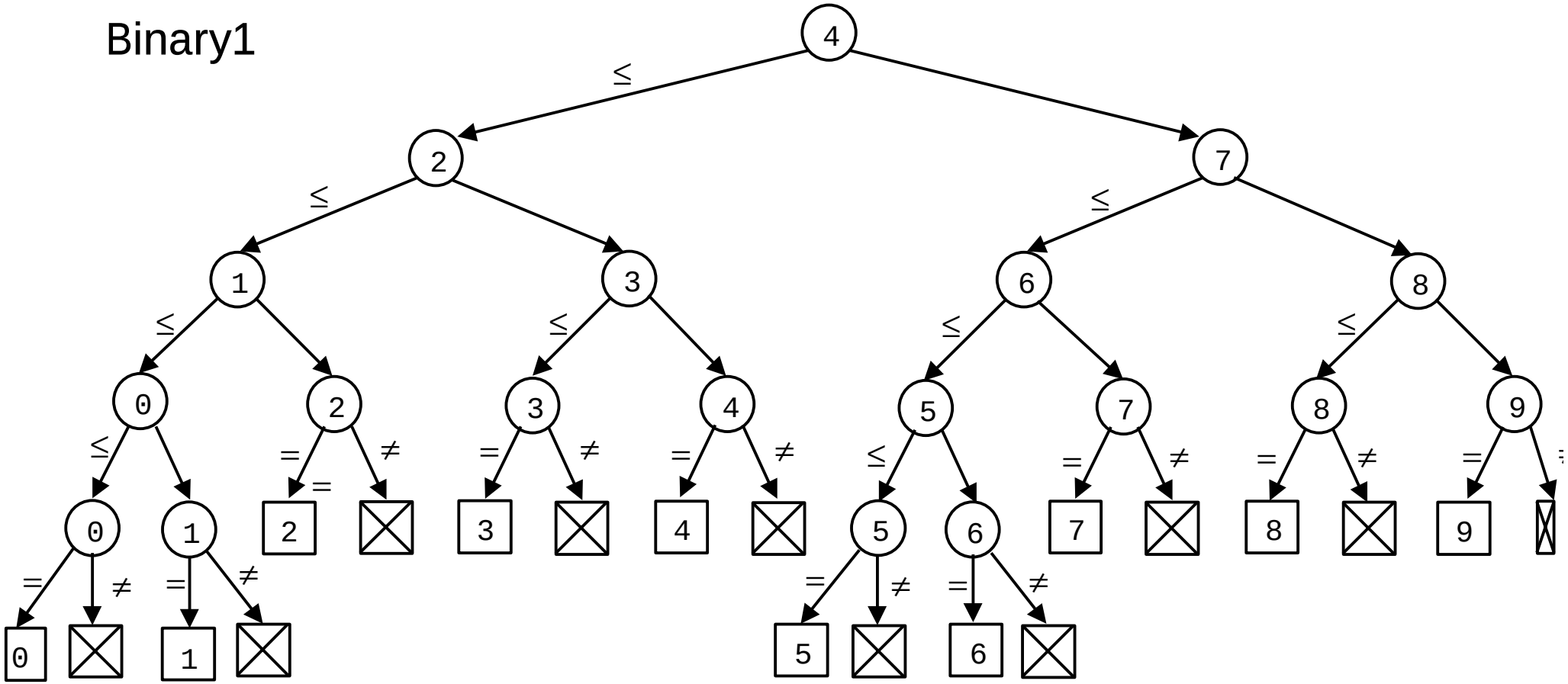
Sequential search

Comparison Trees



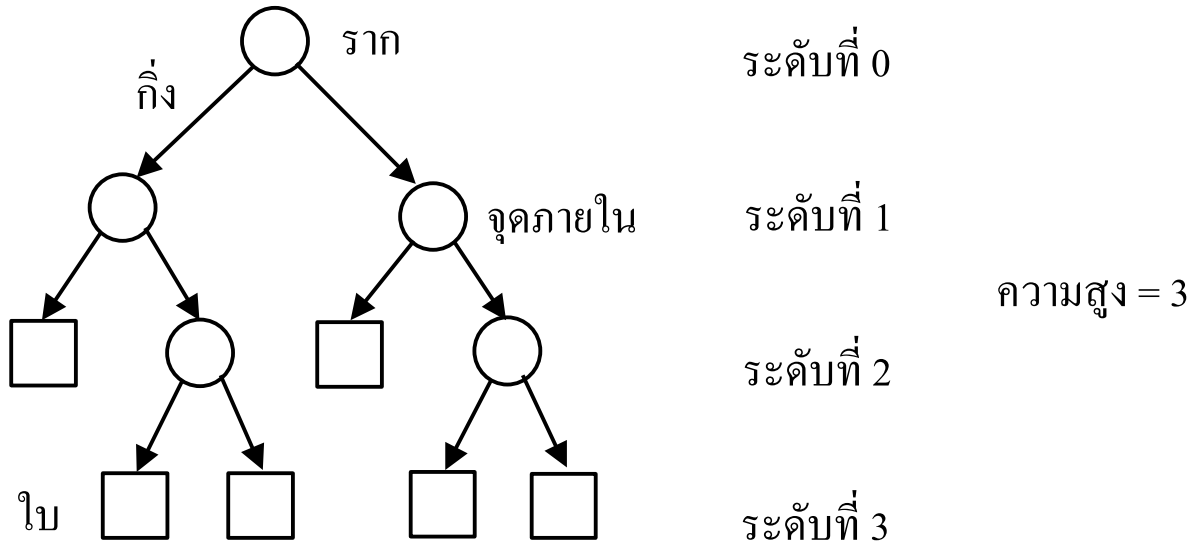
S. Prasitjutrakul 1994

Comparison Trees



Trees

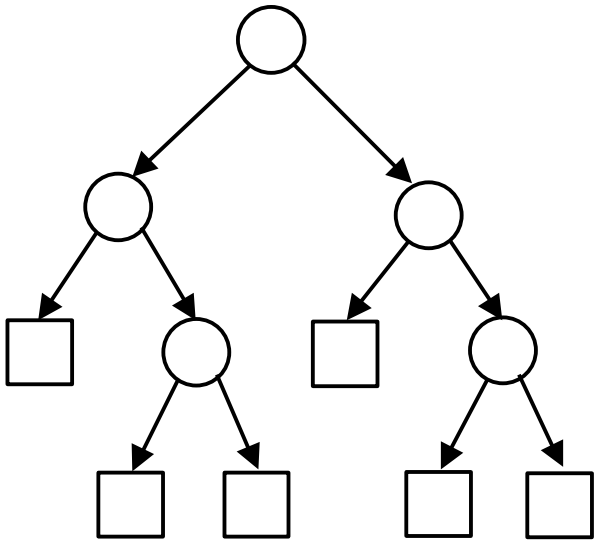
- ราก กิ่ง ใบ จุดภายใน ระดับ ความสูง ของต้นไม้
- ทางเดินจากจุด a ถึง b ในต้นไม้ คือลำดับของกิ่งของต้นไม้ที่ต่อกัน เป็นทางเดินจากจุด a ถึงจุด b
- ต้นไม้ที่สมดุล คือต้นไม้ที่ทุกๆใบในต้นไม้จะอยู่ในระดับที่ต่างกันได้ อย่างมาก เพียง 1 ระดับ



Comparison Trees

- 2-tree คือต้นไม้ที่ทุกๆจุดภายใน ของต้นไม้ จะมีจุดที่เป็นลูก 2 จุด ดังนั้น 2-tree ที่มีความสูง h จะมีจำนวนใบอย่างมาก 2^h ใบ
- ต้นไม้เปรียบเทียบคือต้นไม้แบบ 2-tree
- ต้นไม้เปรียบเทียบของการค้นแบบทวิภาค จะเป็นต้นไม้สมดุล
- ต้นไม้เปรียบเทียบที่มี k ใบ จะมีความสูง h

$$h = \lceil \log_2 k \rceil$$



External & Internal Path Lengths

- External path length - ผลรวมของจำนวนกิ่งของทางเดินทั้งหมดจากรากถึงทุกๆใบของต้นไม้
- Internal path length - ผลรวมของจำนวนกิ่งของทางเดินทั้งหมดจากรากถึงทุกๆจุดภายในของต้นไม้ (ซึ่งคือจุดที่ไม่ใช่ใบ)

External path length

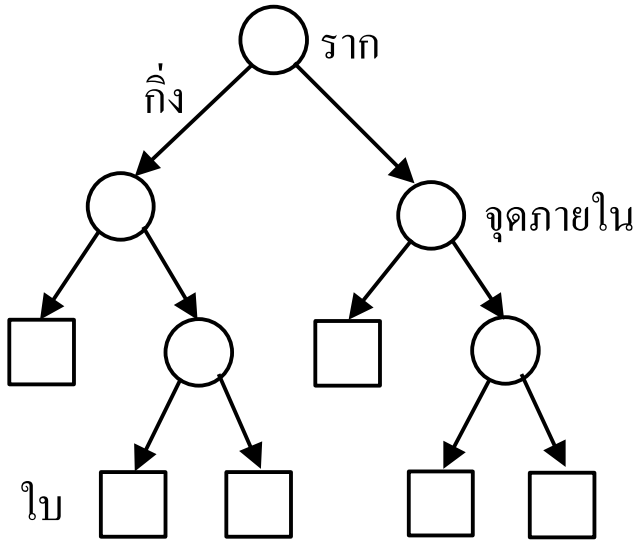
```
E = 2+3+3+2+3+3 = 16
```

Internal path length

```
I = 1+2+1+2 = 6
```

ให้ q แทนจำนวนจุดภายในทั้งหมดของต้นไม้

```
E = I + 2q
```

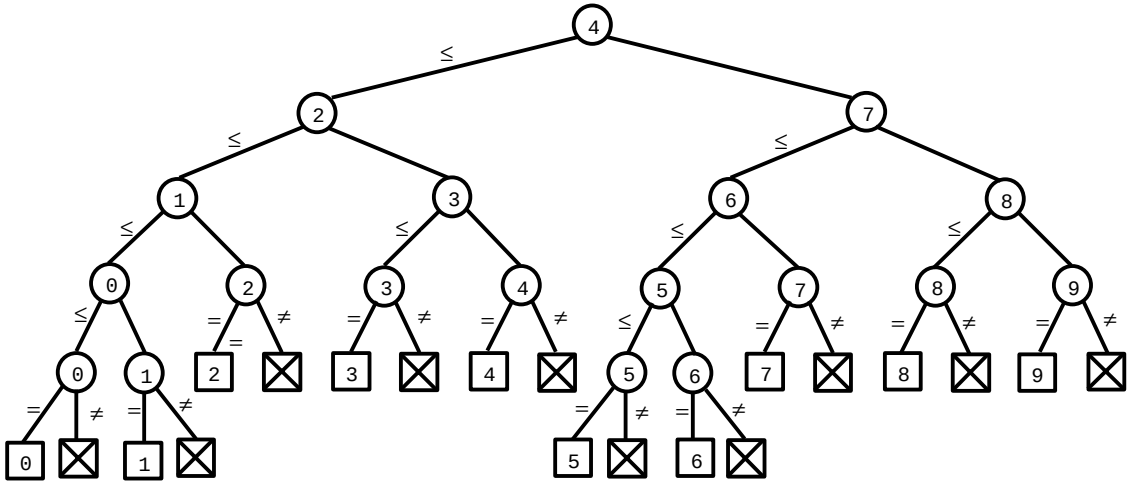


Comparison Counts

E - External path length
I - Internal path length
n - the number of data

	กรณีที่หาพบ	กรณีที่หาไม่พบ
Binary1	$\frac{E}{n}$	$\frac{E}{n}$
Binary2	$\frac{2I}{n} + 1$	$\frac{2E}{n+1}$

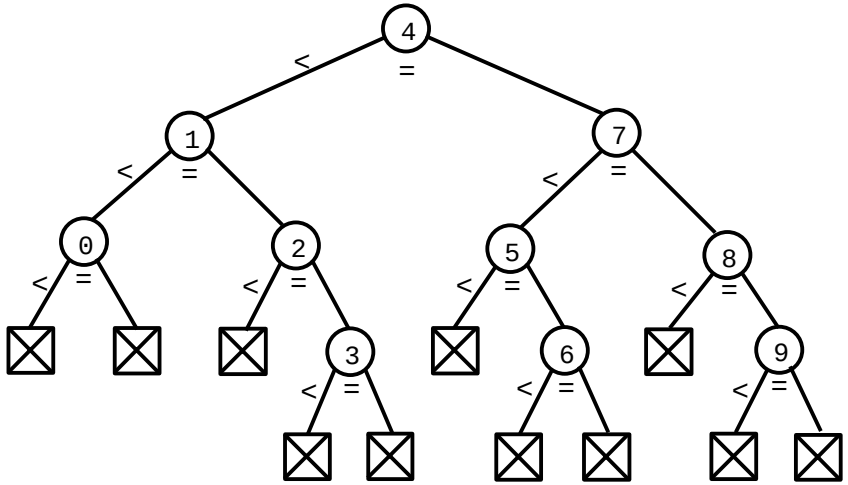
Comparison Counts : Binary1



- ต้นไม้เปรียบเทียบของการค้นข้อมูลจำนวน n ตัว มี 2n ใบ
- จำนวนครั้งในการเปรียบเทียบที่มากที่สุด จะเท่ากับความสูงของ ต้นไม้เปรียบเทียบ
- ความสูงของต้นไม้เปรียบเทียบของ `Binary1` จะประมาณ

$$\log_2 2n = 1 + \log_2 n$$

Comparison Counts : Binary2



ในกรณีที่ไม่พบคีย์ : $2 \log_2(n+1)$

ในกรณีที่พบคีย์ :

$$E = I + 2q$$

$$I = E - 2q$$

$$= (n+1) \log_2(n+1) - 2n$$

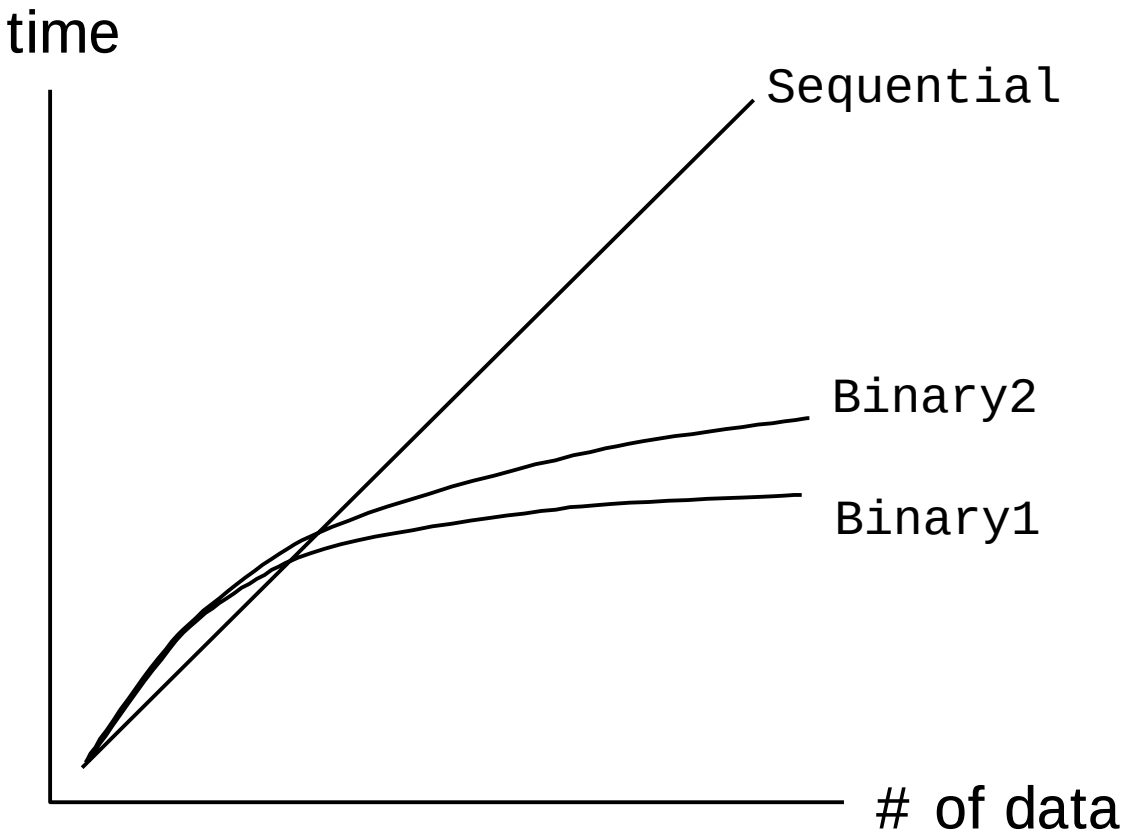
$$\frac{2I}{n} + 1 = \frac{2(n+1)}{n} \log_2(n+1) - 3$$

Sequential vs. Binary Search

Algorithms	จำนวนการเปรียบเทียบเฉลี่ย	
	กรณีที่หาพบ	กรณีที่หาไม่พบ
Sequential	0.5(n+1)	n
Binary1	$\log_2 n + 1$	$\log_2 n + 1$
Binary2	$2\log_2 n - 3$	$2\log_2 n$

n คือจำนวนข้อมูล

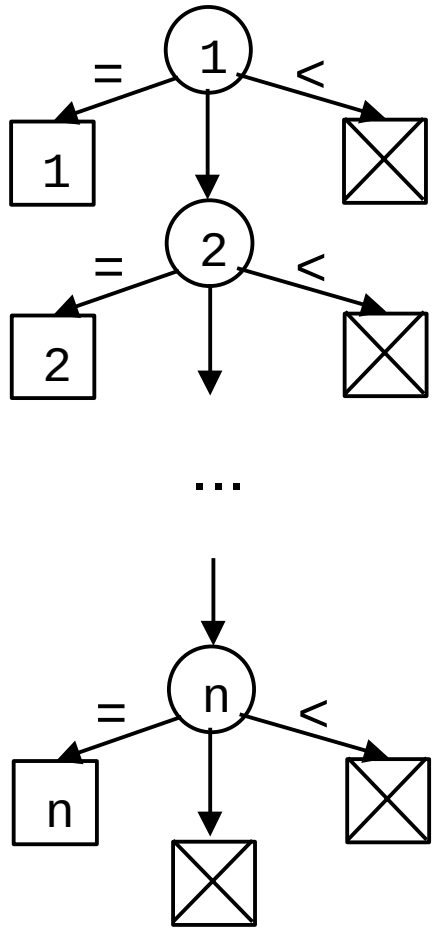
Sequential vs. Binary Search



S. Prasitjutrakul 1994

Sequential Search

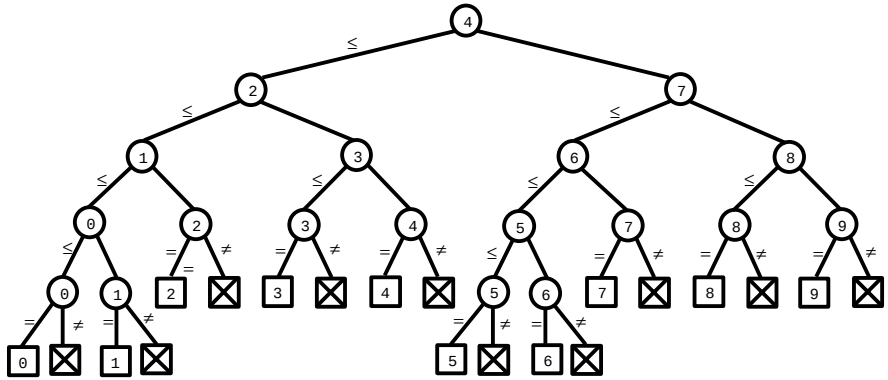
```
int SeqSearch( ListType pList,
               KeyType TargetKey )
{
  int   loc;

  for ( loc=0; loc<pList->count; loc++ ) {
    if ( GT( TargetKey, pList->data[loc].key ) )
      continue;
    if ( EQ( TargetKey, pList->data[loc].key ) )
      return( loc );
    else
      return( -1 );
  }
  return( -1 );
}
```



กรณีหาคีย์พบ : (2+3+...+n+(n+1))/ n = 0.5(n+3)

กรณีหาคีย์ไม่พบ :(2+3+...+n+(n+1) + n)/(n+1) = 0.5(n+3) + n/(n+1)

Lower Bounds for Searching



การค้นข้อมูล n ตัว จะได้ผล 2n+1 กรณี ซึ่งคือ

- กรณีพบคีย์ จะมี n กรณี (เนื่องจากมีข้อมูลอยู่ n ตัว)
- กรณีไม่พบคีย์ จะมี n+1 กรณี ซึ่งคือกรณีที่ คีย์ที่หา

- มีค่าน้อยกว่าตัวแรก, มีค่าระหว่างตัวแรกกับตัวที่สอง,
มีค่าระหว่างตัวที่สองกับตัวที่สาม ,.....
มีค่ามากกว่าตัวที่ n

ใบของต้นไม้เปรียบเทียบแทนผลของการค้น และความสูงของต้นไม้เปรียบเทียบแทน จำนวนครั้ง
การเปรียบเทียบที่มากที่สุด ก่อนที่จะได้ผล ดังนั้น ถ้าต้นไม้มี 2n+1 ใบ จะมีความสูงอย่างน้อย

$(2n+1) \approx 1 + \log n$ ^{2} ^{2}

Lower Bounds for Searching