



Introduction to Searching

SHUTCHON PREMCHAIWATT

แนะนำการค้นหา (Introduction to Searching)

ความหมายของการค้นหา: การค้นหาคือกระบวนการค้นหาข้อมูลในชุดข้อมูล (data set) เพื่อหาค่าที่ตรงกับสิ่งที่ต้องการ

ประเภทของการค้นหา:

การค้นหาเชิงเส้น (**Linear Search**)

การค้นหาแบบไบนารี (**Binary Search**)

ความสำคัญของการค้นหา: การเลือกวิธีค้นหาที่เหมาะสมสามารถทำให้การค้นหาข้อมูลมีประสิทธิภาพมากขึ้น



Linear Search

หลักการทำงานของ Linear Search:

- ตรวจสอบรายการทีละรายการจากเริ่มต้นจนถึงจบ
- ใช้ได้กับรายการที่ไม่ได้เรียงลำดับ

ตัวอย่างการค้นหาลำบากๆ:

- ค้นหาชื่อในรายการชื่อ

Linear Search - การเขียนโค้ด

```
def linear_search(arr, target):  
    for index, value in enumerate(arr):  
        if value == target:  
            return index  
    return -1 # ถ้าหาไม่เจอ
```

Good

```
def linear_search(arr, target):  
    result = -1  
    for index, value in enumerate(arr):  
        if value == target:  
            result = index  
    return result
```

Bad

คำอธิบาย: enumerate(arr) ใช้เพื่อให้ได้ทั้ง index และ value ของรายการใช้
if เพื่อตรวจสอบว่าค่าปัจจุบันตรงกับ target หรือไม่

ทำไม code ทางด้านซ้ายถึงไม่ดี ?

Linear Search - Complexity Analysis

- **Time Complexity:**

- $O(n)$: การค้นหาอาจต้องตรวจสอบทุกรายการในกรณีที่เลวร้ายที่สุด

- **Space Complexity:**

- $O(1)$: ใช้พื้นที่หน่วยความจำคงที่ (นอกจากพื้นที่เก็บข้อมูล)

Binary Search - พื้นฐาน

หลักการทำงานของ **Binary Search:**

ค้นหารายการที่เรียงลำดับ
แล้ว

ใช้การแบ่งครึ่งเพื่อหาค่าที่
ต้องการ

ข้อกำหนด:

รายการต้องเรียงลำดับ

Binary Search - การเขียนโค้ด

คำอธิบาย: `left` และ `right` กำหนดขอบเขตการค้นหา
`mid` คำนวณตำแหน่งกลางเปรียบเทียบ
`arr[mid]` กับ `target` และปรับขอบเขตตาม
ผลลัพธ์

```
def binary_search(arr, target):  
    # กำหนดขอบเขตการค้นหาเริ่มต้น  
    left, right = 0, len(arr) - 1  
  
    # ตรวจสอบดูว่ายังมีช่วงที่ต้องค้นหา  
    while left <= right:  
        # หาค่าตำแหน่งกึ่งกลางของช่วงปัจจุบัน  
        mid = (left + right) // 2  
  
        # ถ้าพบค่าที่ต้องการ ส่งคืนตำแหน่งนั้น  
        if arr[mid] == target:  
            return mid  
        # ถ้าค่ากลางน้อยกว่าเป้าหมาย ปรับขอบซ้ายให้อยู่หลังตำแหน่งกลาง  
        elif arr[mid] < target:  
            left = mid + 1  
        # ถ้าค่ากลางมากกว่าเป้าหมาย ปรับขอบขวาให้อยู่ก่อนตำแหน่งกลาง  
        else:  
            right = mid - 1  
  
    # ถ้าไม่พบค่าที่ต้องการในอาร์เรย์ ส่งคืน -1  
    return -1
```



Binary Search - Complexity Analysis

Time Complexity:

- **$O(\log n)$** : จำนวนขั้นตอนลดลงครึ่งหนึ่งในแต่ละรอบ

Space Complexity:

- **$O(1)$** : ใช้พื้นที่หน่วยความจำคงที่ (นอกจากพื้นที่เก็บข้อมูล)


การเปรียบเทียบ Linear Search และ Binary Search

Linear Search:

- ใช้ได้กับข้อมูลที่ไม่เรียงลำดับ
- ง่ายและเข้าใจง่าย
- ช้ากว่าเมื่อข้อมูลมีขนาดใหญ่

Binary Search:

- ต้องการข้อมูลที่เรียงลำดับ
- มีประสิทธิภาพสูงกว่าเมื่อข้อมูลมีขนาดใหญ่
- ต้องใช้เวลาจัดเรียงข้อมูลก่อน



ตัวอย่างการค้นหาข้อมูล

ค้นหาในฐานข้อมูล:

- การใช้เทคนิคการค้นหาที่มีประสิทธิภาพในฐานข้อมูลขนาดใหญ่

ค้นหาในไฟล์:

- การค้นหาข้อมูลในไฟล์ขนาดใหญ่ เช่น การค้นหาข้อความในไฟล์ log

การค้นหาในโครงสร้างข้อมูล
(Searching in
Data Structures)

การค้นหาในโครงสร้างข้อมูล:

- การค้นหาในโครงสร้างข้อมูลที่ซับซ้อน เช่น กราฟและต้นไม้

ตัวอย่าง:

- การค้นหาใน Binary Search Tree (BST)