ឥន្ស្យាស្ថានសន្តប៉ូល
Saint Paul Institute

Group 2

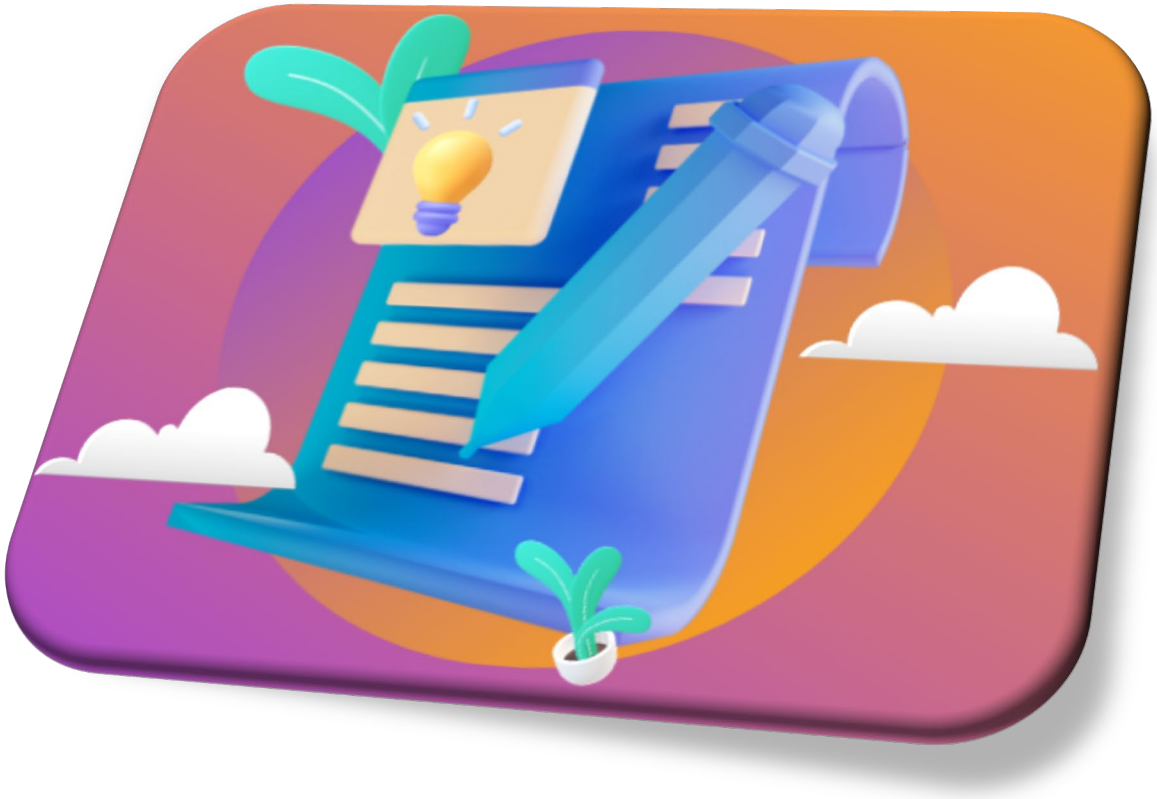
Topic: Searching Algorithms

Teacher's Name: Seng Ponleu

Member:

1. Chea Chanbormey
2. Yong Pisey
3. Men Both
4. Tum Leangseng
5. Ni Socheata
6. Soeun Sovannarong

Content



Introduction

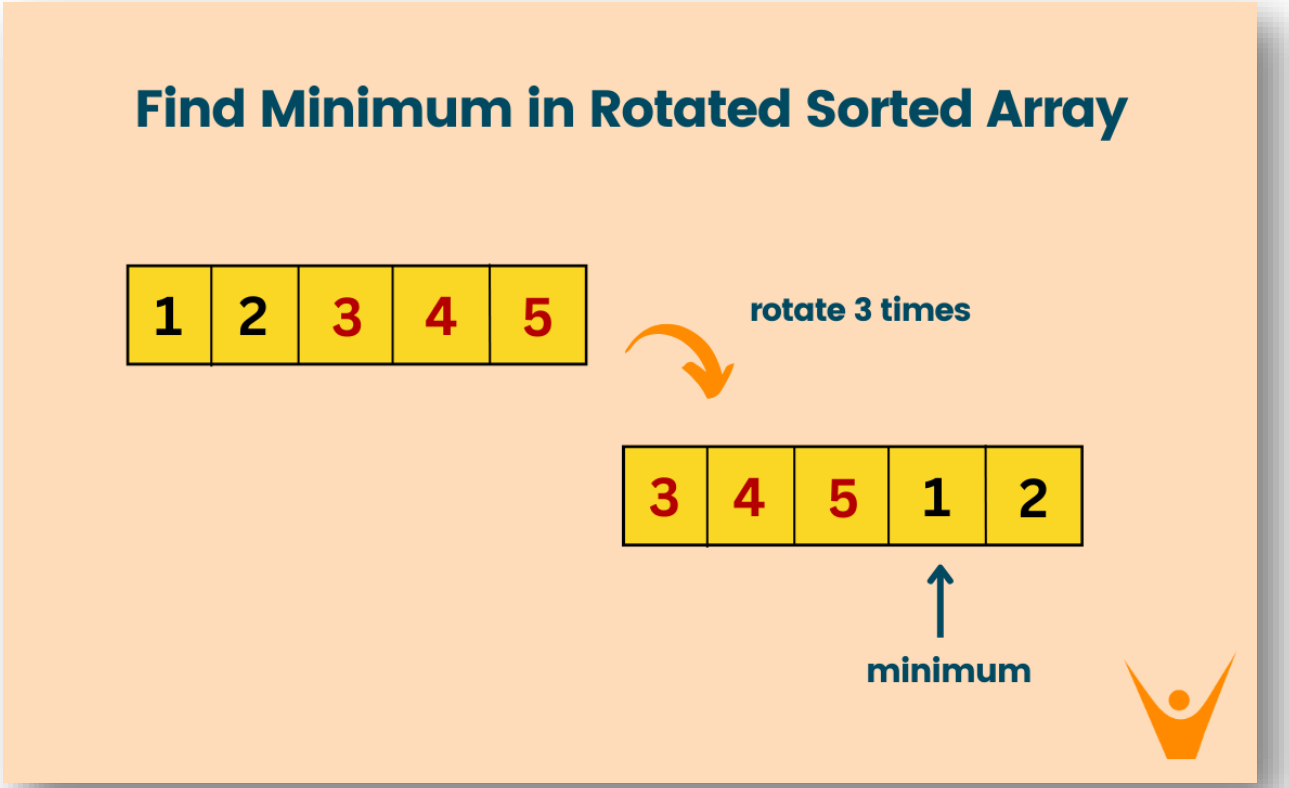
Searching algorithms គឺជា វិធីសាស្ត្រដែលប្រើសម្រាប់ស្វែងរកទិន្នន័យជាក់លាក់មួយនៅក្នុងសំណុំទិន្នន័យ។

❖ **Searching algorithms** អាចស្វែងរកតាមវិធីសាស្ត្រពីរយ៉ាងគឺ

- Linear Search : គឺជាវិធីសាស្ត្រស្វែងក្នុងការស្វែងរកទិន្នន័យដែលមានលក្ខណៈសាមញ្ញដោយធ្វើការត្រួតពិនិត្យធាតុ(node)ទៅតាមលំដាប់ជាប់គ្នារហូតដល់រកឃើញតម្លៃជាក់លាក់។
- Binary Search : Binary Search គឺជាវិធីសាស្ត្រដែលបំបែក Array ជាពាក់កណ្តាល ម្តងហើយម្តងទៀត, ដើម្បីស្វែងរកតម្លៃគោលដៅ តម្លៃជាក់លាក់(target value)។

Minimum in a Sorted and Rotated Array

- **Minimum in a Sorted** គឺជាតម្លៃដែលតូចបំផុតនៅក្នុង array ដែលបានតម្រៀប។
- **Rotated Array** គឺជាការតម្រៀបធាតុឡើងវិញដែលមានពីរទីតាំងខាងឆ្វេងឬខាងស្តាំ។



Minimum in a Sorted and Rotated Array(Cont)

❑ Coding in Linear Search

```cpp
#include <iostream>
#include <vector>
using namespace std;

int findMin(vector<int>& arr) {

    int res = arr[0];

    // Traverse over arr[] to find minimum element
    for (int i = 1; i < arr.size(); i++)
        res = min(res, arr[i]);

    return res;
}
```

Minimum in a Sorted and Rotated Array(Cont)

❑ Coding in Linear Search

```
int main() {
    vector<int> arr = {5, 6, 1, 2, 3, 4};
    int n = arr.size();

    cout <<"minimum element is: "<< findMin(arr) << endl;
    return 0;
}
```

❑ Output

```
minimum element is: 1

--------------------------------
Process exited after 0.009851 seconds with return value 0
Press any key to continue . . .
```

Maximum in a Sorted and Rotated Array

Maximum in a Sorted and Rotated Array(Cont)

❑ Coding in Linear Search

```cpp
#include <iostream>
#include <vector>
using namespace std;

// Function to find the maximum value
int findMax(vector<int>& arr) {

    int res = arr[0];

    // Traverse over arr[] to find maximum element
    for (int i = 1; i < arr.size(); i++)
        res = max(res, arr[i]);

    return res;
}
```

Maximum in a Sorted and Rotated ArrayCont()

❑ Coding in Linear Search

```
int main() {
    vector<int> arr = {5, 6, 1, 2, 3, 4};

    cout <<"maximum element is: "<< findMax(arr) << endl;
    return 0;
}
```

❑ Output

```
maximum element is: 6

--------------------------------
Process exited after 0.402 seconds with return value 0
Press any key to continue . . .
```

Missing and Repeating in an Array

pisey

Missing and Repeating in an Array(Cont)

- **Coding in Linear Search**

pisey

Missing and Repeating in an Array(Cont)

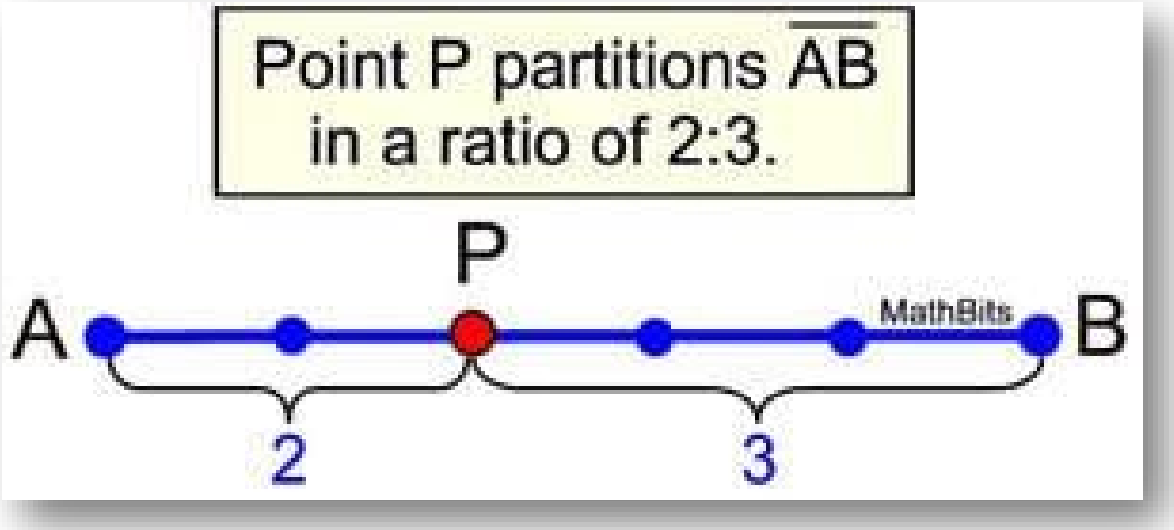
- **Coding in Linear Search**

pisey

- **Output**

Partition Point

- Partition Point : ចំណុចជាក់លាក់នៅលើផ្នែកបន្ទាត់ដែលបែងចែកផ្នែកទៅជាផ្នែកតូចជាងពីរជាមួយនឹងសមាមាត្រដែលបានកំណត់។
- សមាមាត្រនេះបង្ហាញពីរបៀបដែលផ្នែកត្រូវបានបំបែក។



Partition Point (Cont)

❑ Coding to use Nested Loops

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

int findElement(vector<int> &arr) {

    // Iteratte through each elements
    for(int i = 1; i < arr.size() - 1; i++) {

        // to store maximum in left
        int left = INT_MIN;
        for(int j = 0; j < i; j++) {
            left = max(left, arr[j]);
        }
```

Partition Point (Cont)

❑ Coding to use Nested Loops

```
        // to store minimum in right
        int right = INT_MAX;
        for(int j = i + 1; j < arr.size(); j++) {
            right = min(right, arr[j]);
        }

        // check if current element is greater
        // than left and smaller than right (or equal)
        if(arr[i] >= left && arr[i] <= right) {
            return arr[i];
        }

    }

    return -1;
}
```

Partition Point (Cont)

❑ Coding to use Nested Loops

```
int main() {
    vector<int> arr = {5, 1, 4, 3, 6, 8, 10, 7, 9};
    cout << findElement(arr);
    return 0;
}
```

❑ Output

```
6
--------------------------------
Process exited after 0.8916 seconds with return value 0
Press any key to continue . . .
```

First repeating

First repeating (Cont)

❑ Coding to use Naive approach

```cpp
#include <bits/stdc++.h>
using namespace std;
// Function to find the index of the first
// repeating element in a vector
int firstRepeatingElement(vector<int> &arr)
{
    int n = arr.size();
    // Nested loop to check for repeating elements
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                return i;
            }
        }
    }
    // If no repeating element is found, return -1
    return -1;
}
```

First repeating (Cont)

❑ Coding to use Naive approach

```
int main()
{
    vector<int> arr = {10, 5, 3, 4, 3, 5, 6};
    int index = firstRepeatingElement(arr);
    if (index == -1)
        cout << "No repeating found!" << endl;
    else
        cout << "First repeating is " << arr[index] << endl;
    return 0;
}
```

❑ Output

```
First repeating is 5

--------------------------------
Process exited after 0.4015 seconds with return value 0
Press any key to continue . . .
```

common elements in 3 sorted

common elements in 3 sorted (Cont)

❑ Coding to use Tree Map

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

vector<int> commonElements(vector<int> &arr1, vector<int> &arr2, vector<int> &arr3)
{
    // hash table to mark the common elements
    map<int, int> mp;

    // Mark all the elements in the first Array with value=1
    for (int ele : arr1)
    {
        mp[ele] = 1;
    }
```

common elements in 3 sorted (Cont)

❑ Coding to use Tree Map

```
// Mark all the elements which are common in first and
// second Array with value = 2
for (int ele : arr2)
{
    if (mp.find(ele) != mp.end() && mp[ele] == 1)
        mp[ele] = 2;
}

// Mark all the elements which are common in first,
// second and third Array with value = 3
for (int ele : arr3)
{
    if (mp.find(ele) != mp.end() && mp[ele] == 2)
        mp[ele] = 3;
}
```

common elements in 3 sorted (Cont)

❑ Coding to use Tree Map

```
    // Store all the common elements
    vector<int> commonElements;
    for (auto ele : mp)
    {
        if (ele.second == 3)
            commonElements.push_back(ele.first);
    }

    // Return the common elements which are common in all
    // the three sorted Arrays
    return commonElements;
}
```

common elements in 3 sorted (Cont)

❑ Coding to use Tree Map

```
int main()
{
    // Sample Input
    vector<int> arr1 = {1, 5, 10, 20, 30};
    vector<int> arr2 = {5, 13, 15, 20};
    vector<int> arr3 = {5, 20};

    vector<int> common = commonElements(arr1, arr2, arr3);
    if (common.size() == 0)
        cout << -1;
    for (int i = 0; i < common.size(); i++)
    {
        cout << common[i] << " ";
    }

    return 0;
}
```

common elements in 3 sorted (Cont)

❑ Output

```
5 20
--------------------------------
Process exited after 0.3887 seconds with return value 0
Press any key to continue . . .
```

THANK YOU

Any Questions?