排序與搜尋

Sorting and Searching

本課程由以下贊助商贊助辦理

奧義智慧科技
Powered by CyCraft

DEVCORE

少年圖靈計畫
Young Turing Program

TEAMT5
Persistent Cyber Threat Hunters

iTSA

主題

- 排序
- 建表
- 二分搜
- 各種二分搜技巧
- 雙指針

排序

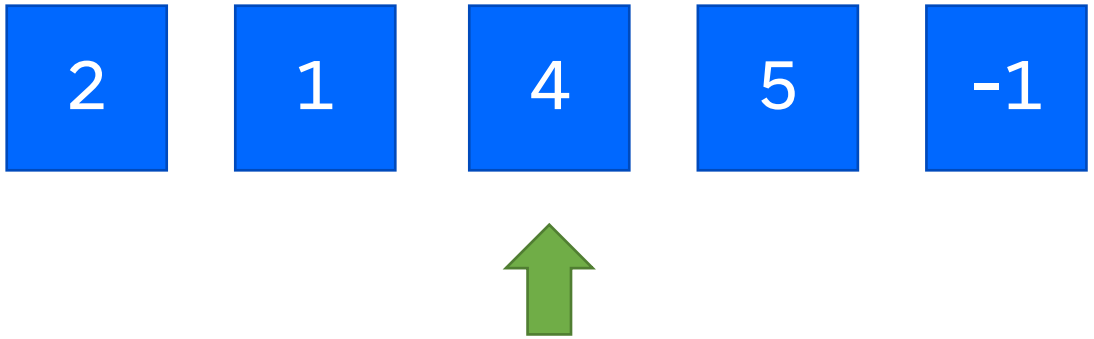
- Bubble sort
- Merge sort
- C++內建sort用法
- 例題

Bubble Sort

- 邊掃邊交換比較大的數字
- 當前最大的會被移動到最右邊
- 掃n-1次


2
4
1
5
-1


2
4
1
5
-1

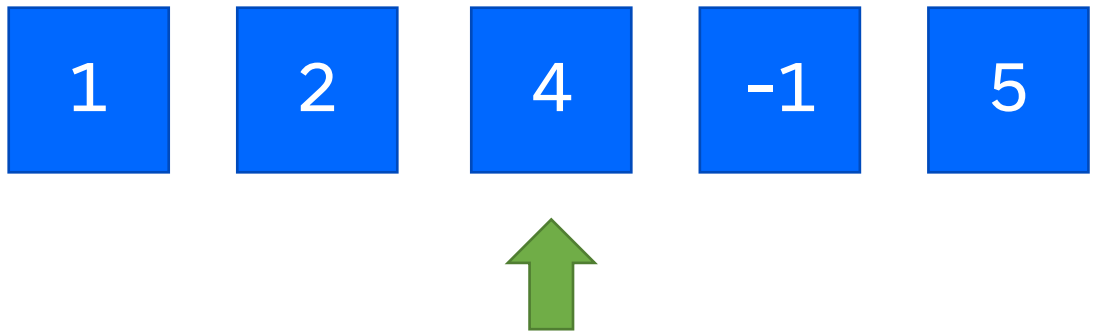

2
1
4
5
-1


2
1
4
5
-1


2
1
4
-1
5


2
1
4
-1
5

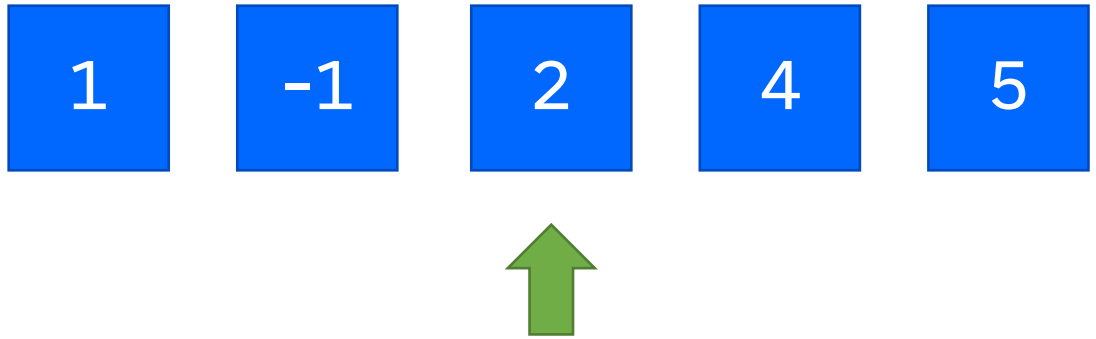

1
2
4
-1
5


1
2
4
-1
5


1
2
-1
4
5


1
2
-1
4
5


1
2
-1
4
5


1
-1
2
4
5


1
-1
2
4
5


-1
1
2
4
5


-1
1
2
4
5

Code

```
for (int j = 0; j < n - 1; j++) {
    for (int i = 0; i < n - 1 - j; i++) {
        if (a[i] > a[i + 1]) swap(a[i], a[i + 1]);
    }
}
```

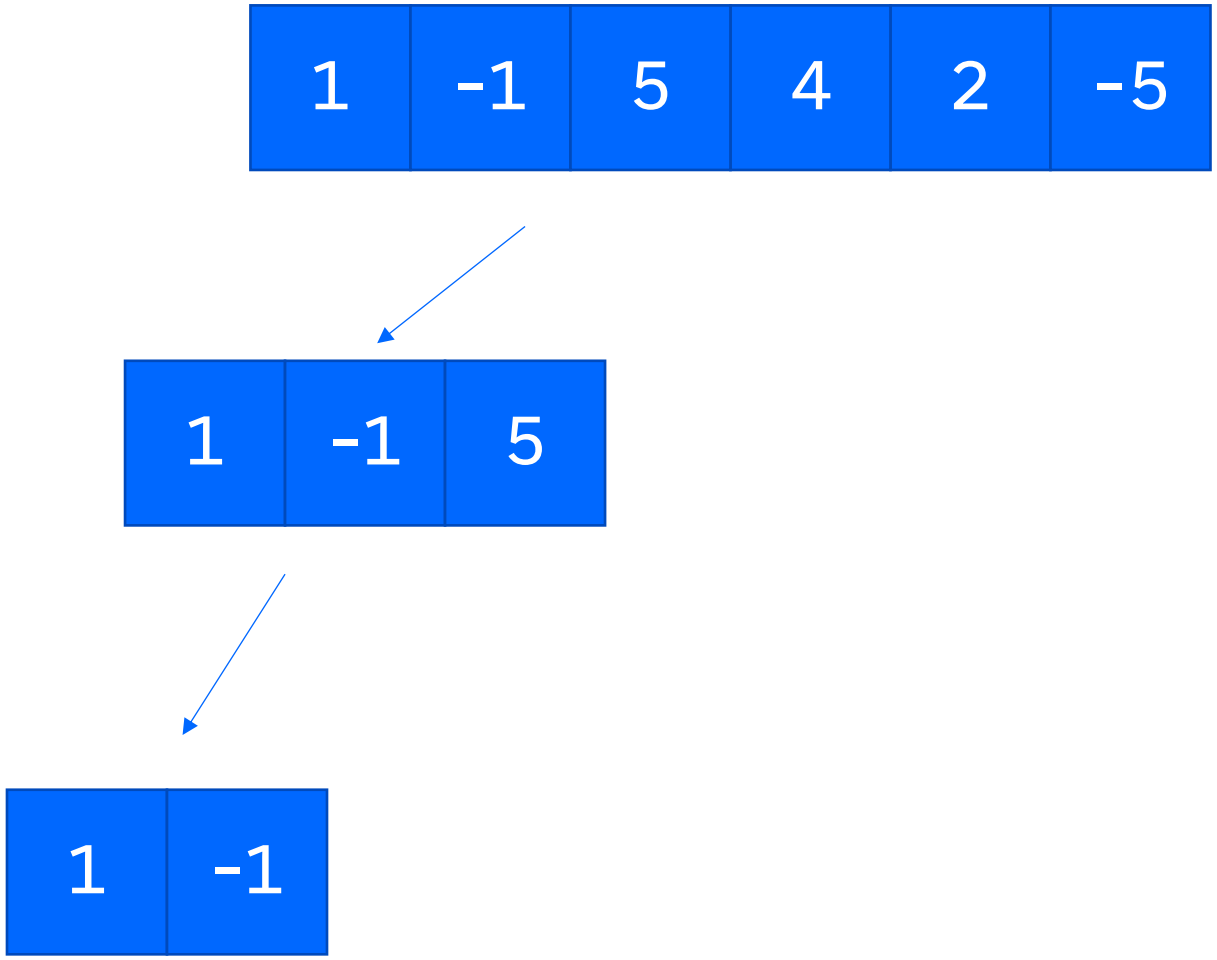
Merge Sort

- 切一半，分別遞迴做排序
- 合併兩半

1	-1	5	4	2	-5

1	-1	5	4	2	-5

1	-1	5

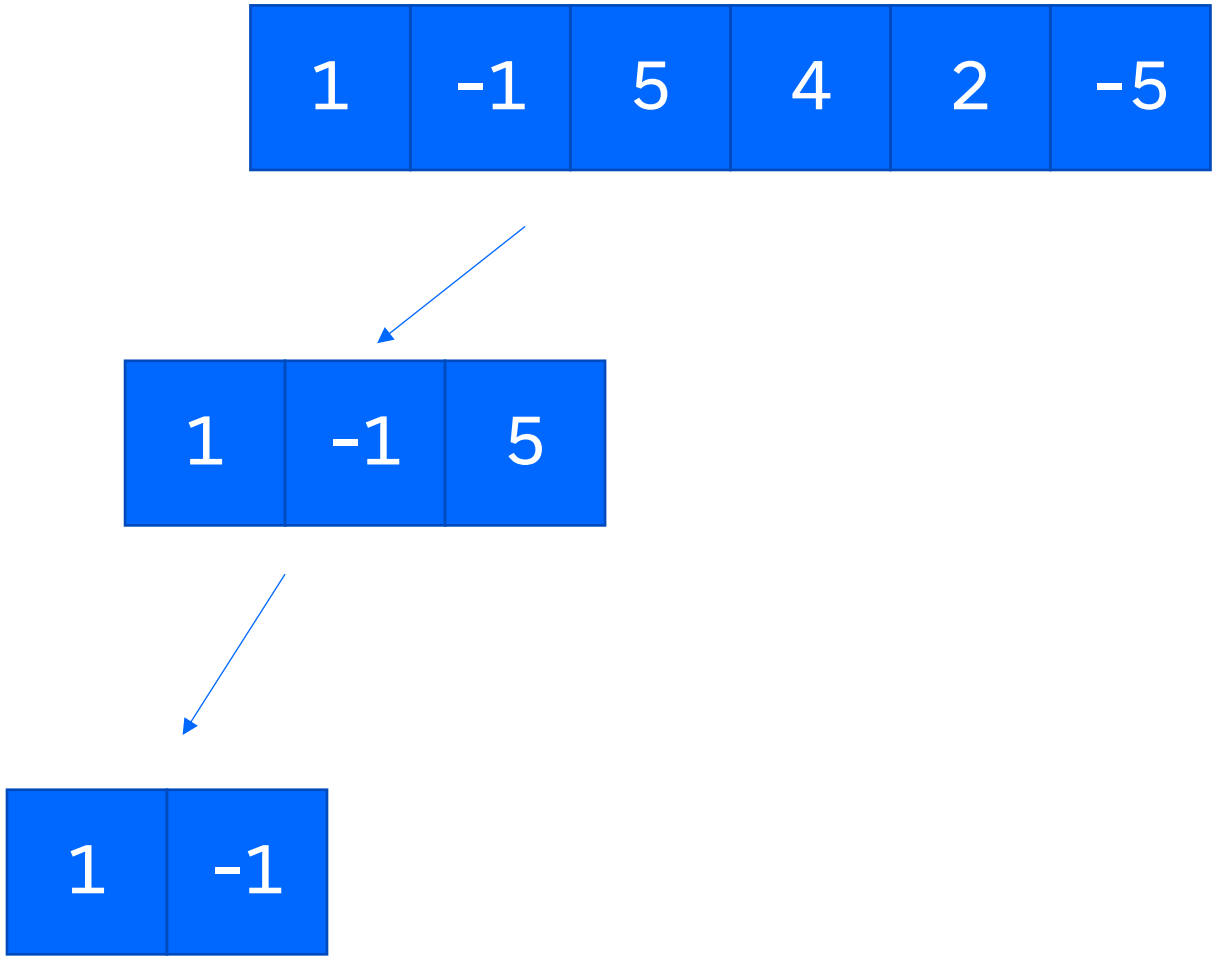

1 -1 5 4 2 -5
1 -1 5
1 -1

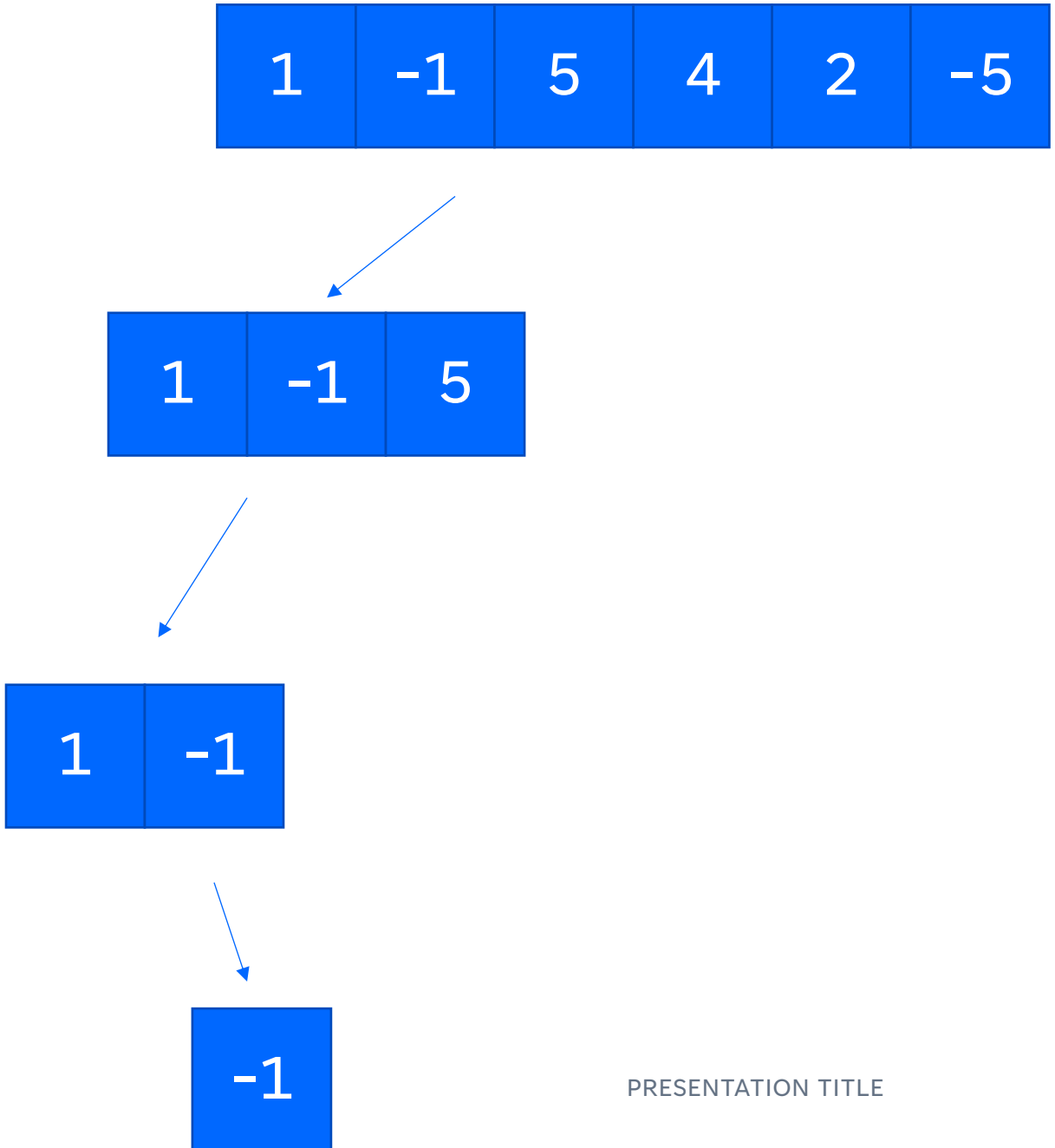
1	-1	5	4	2	-5

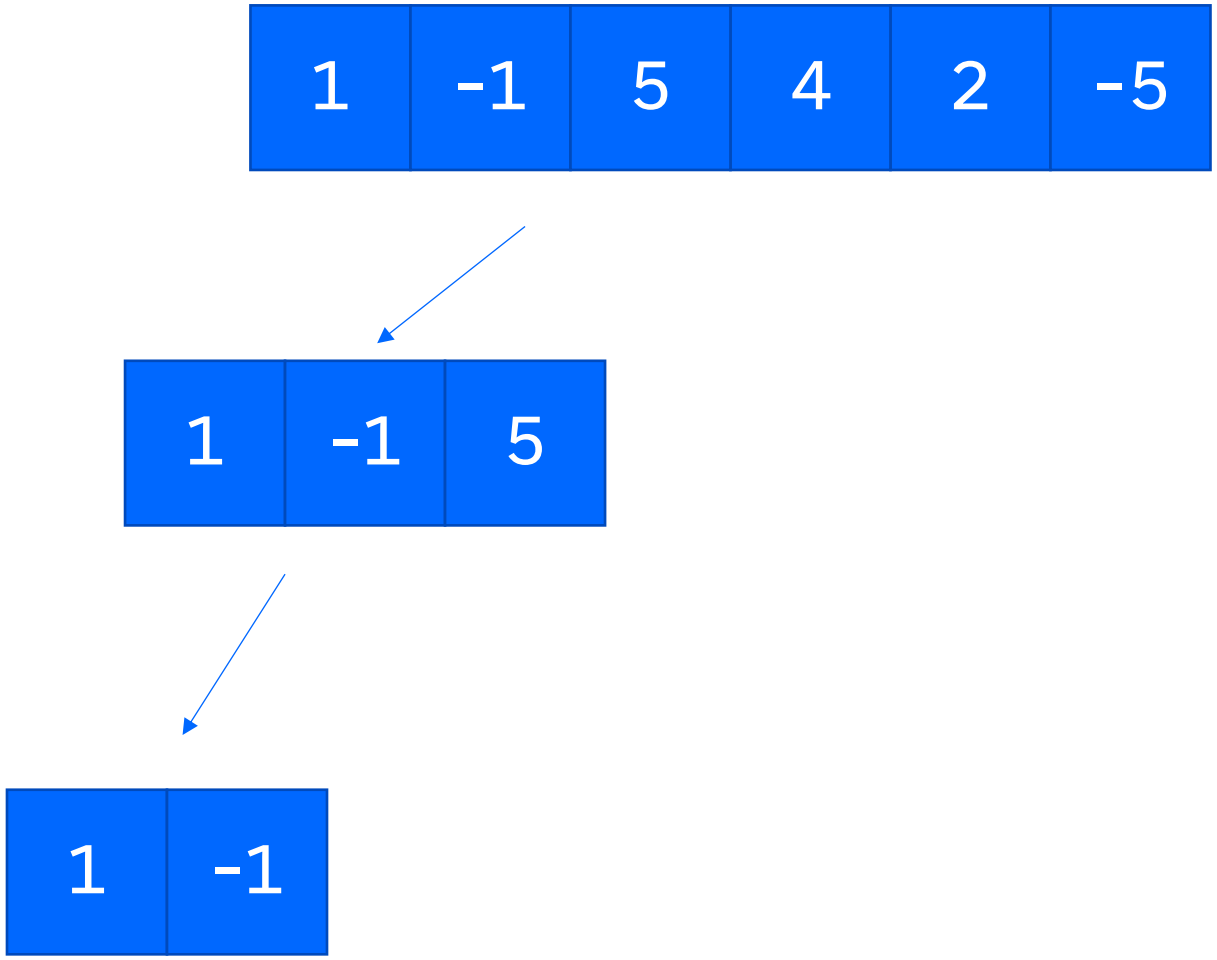
1	-1	5

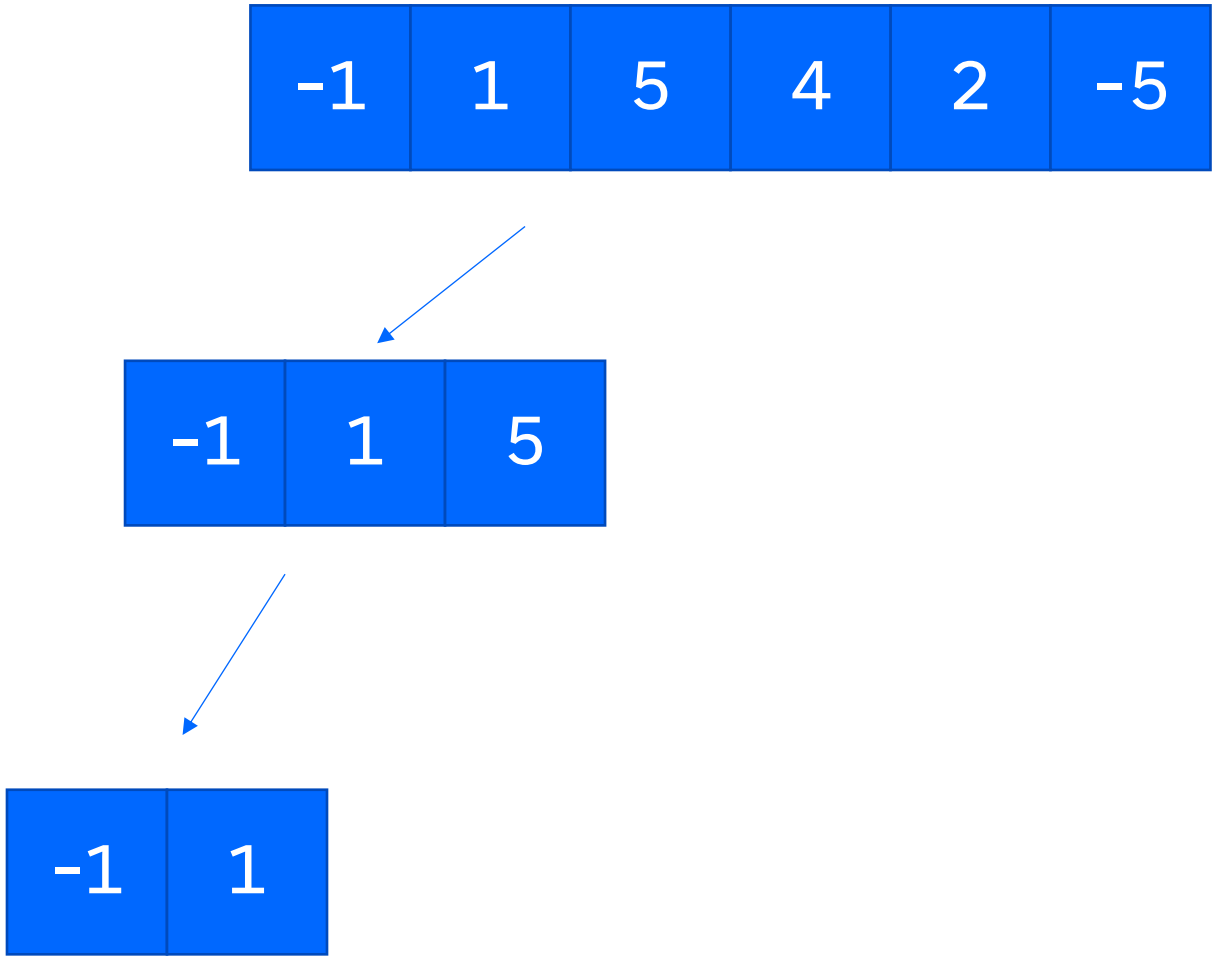
1	-1

1


1 -1 5 4 2 -5
1 -1 5
1 -1

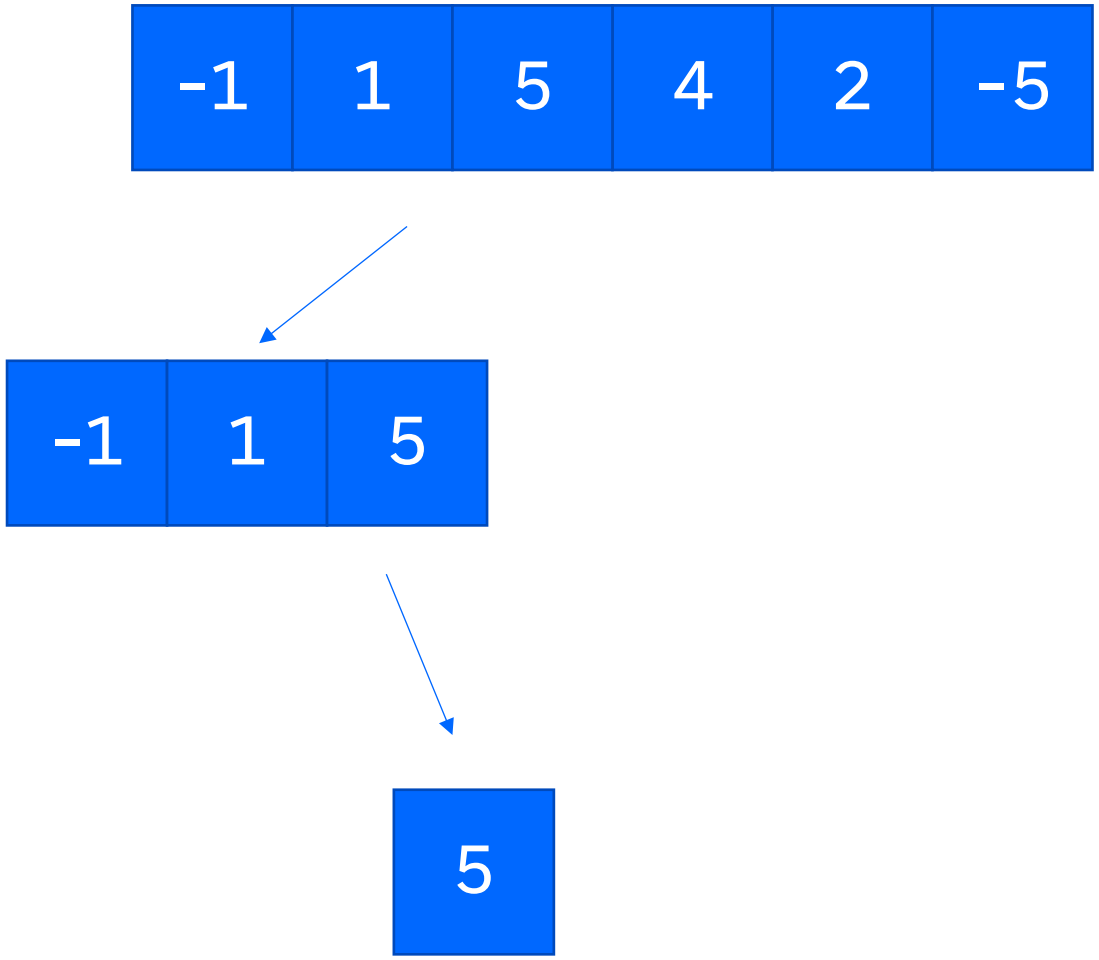

1 -1 5 4 2 -5
1 -1 5
1 -1
-1


1 -1 5 4 2 -5
1 -1 5
1 -1


-1 1 5 4 2 -5
-1 1 5
-1 1

-1	1	5	4	2	-5

-1	1	5


-1 1 5 4 2 -5
-1 1 5
5

-1	1	5	4	2	-5

-1	1	5

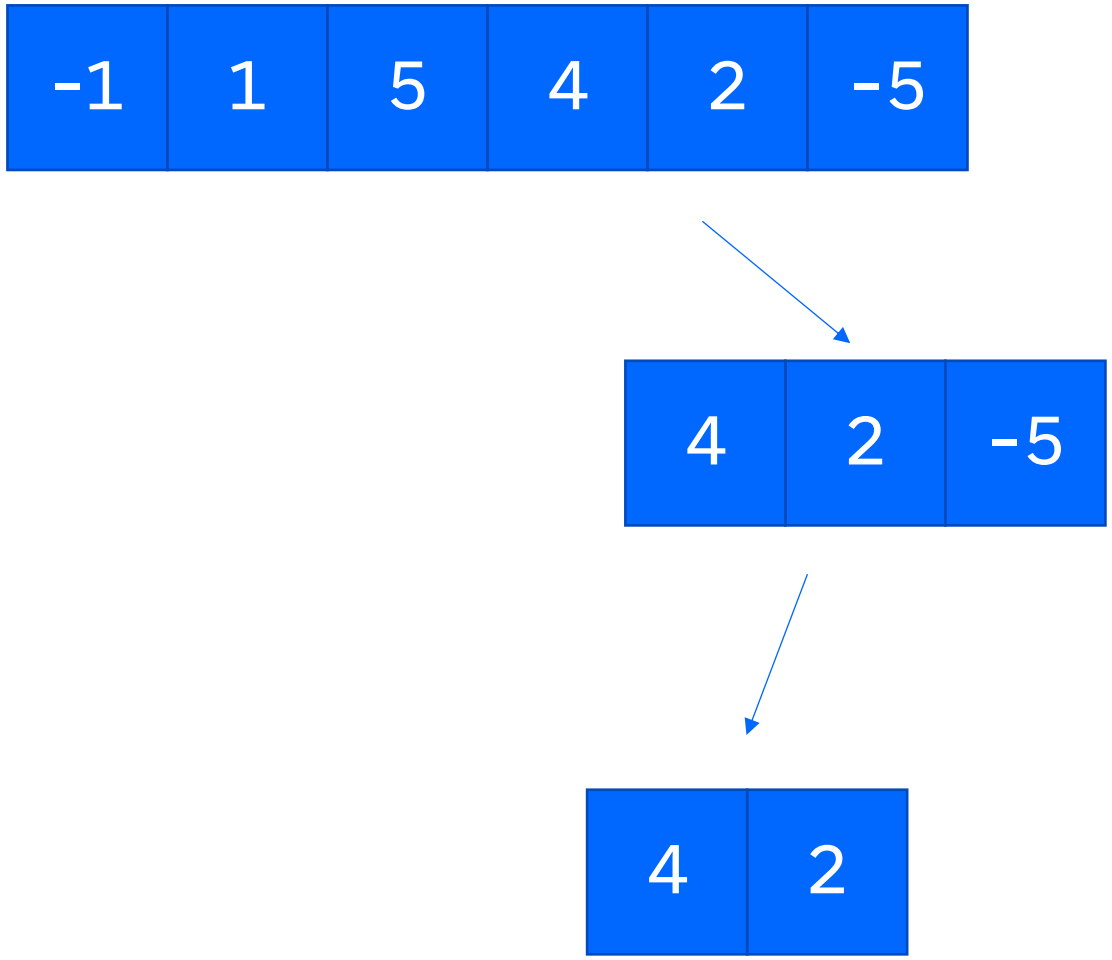
-1	1	5	4	2	-5

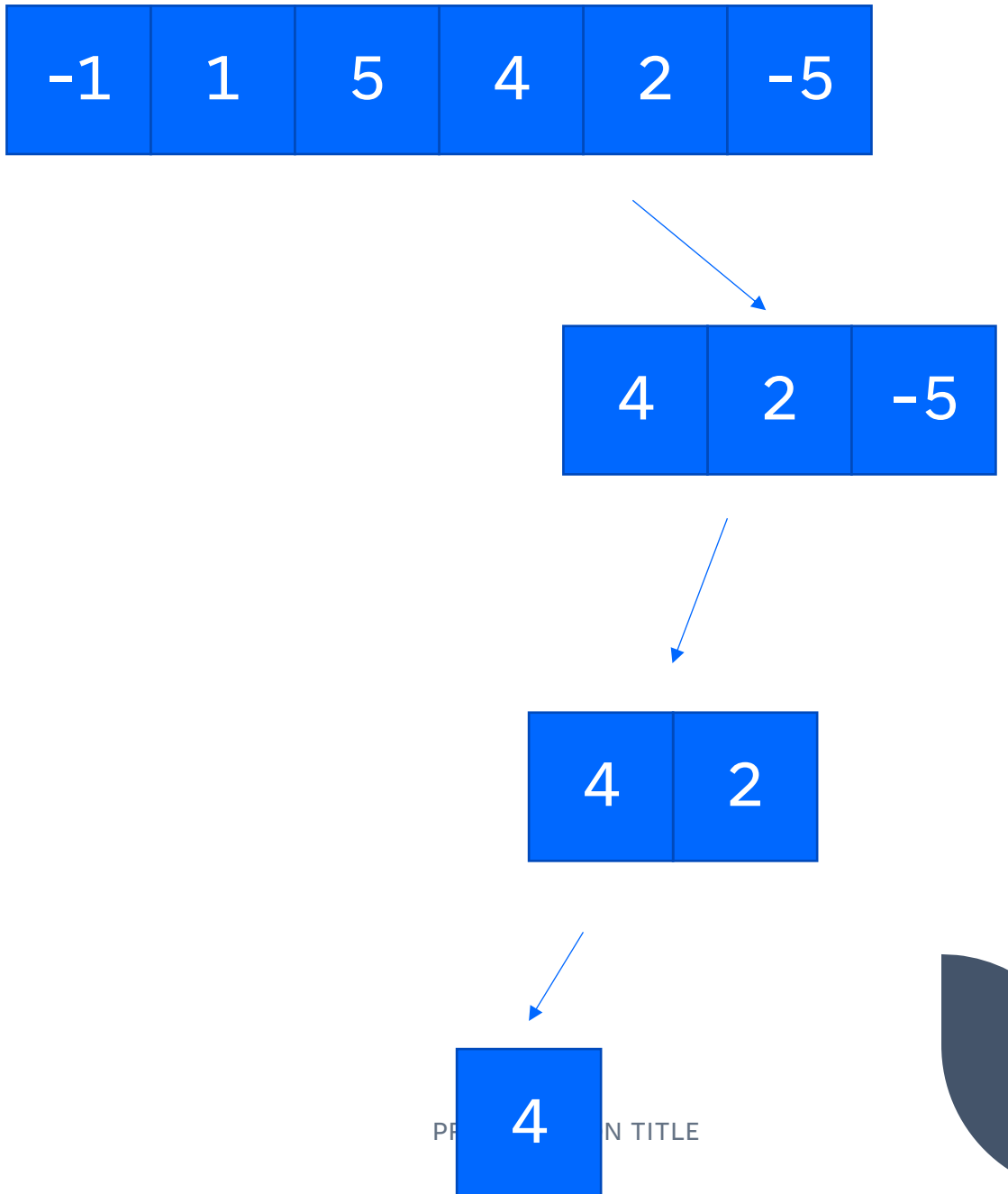
-1	1	5

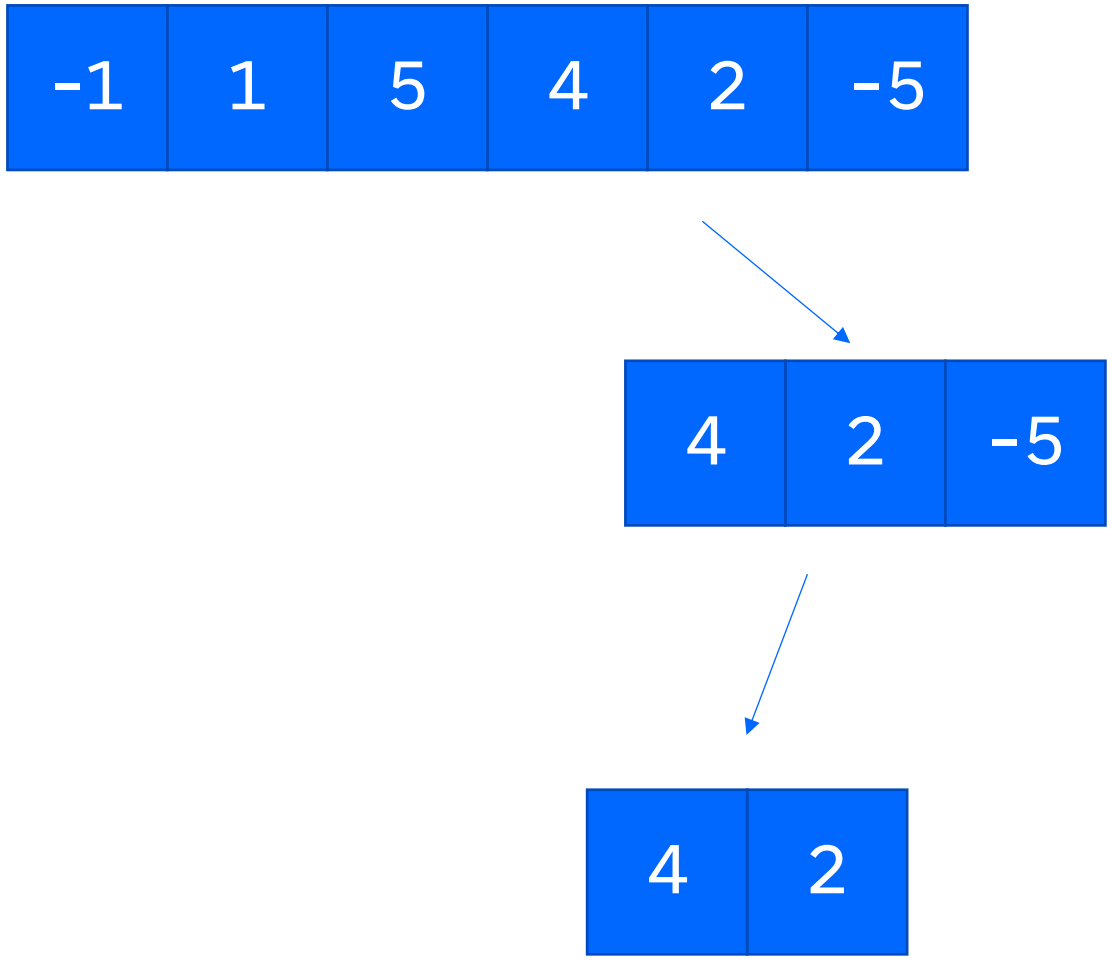
-1	1	5	4	2	-5

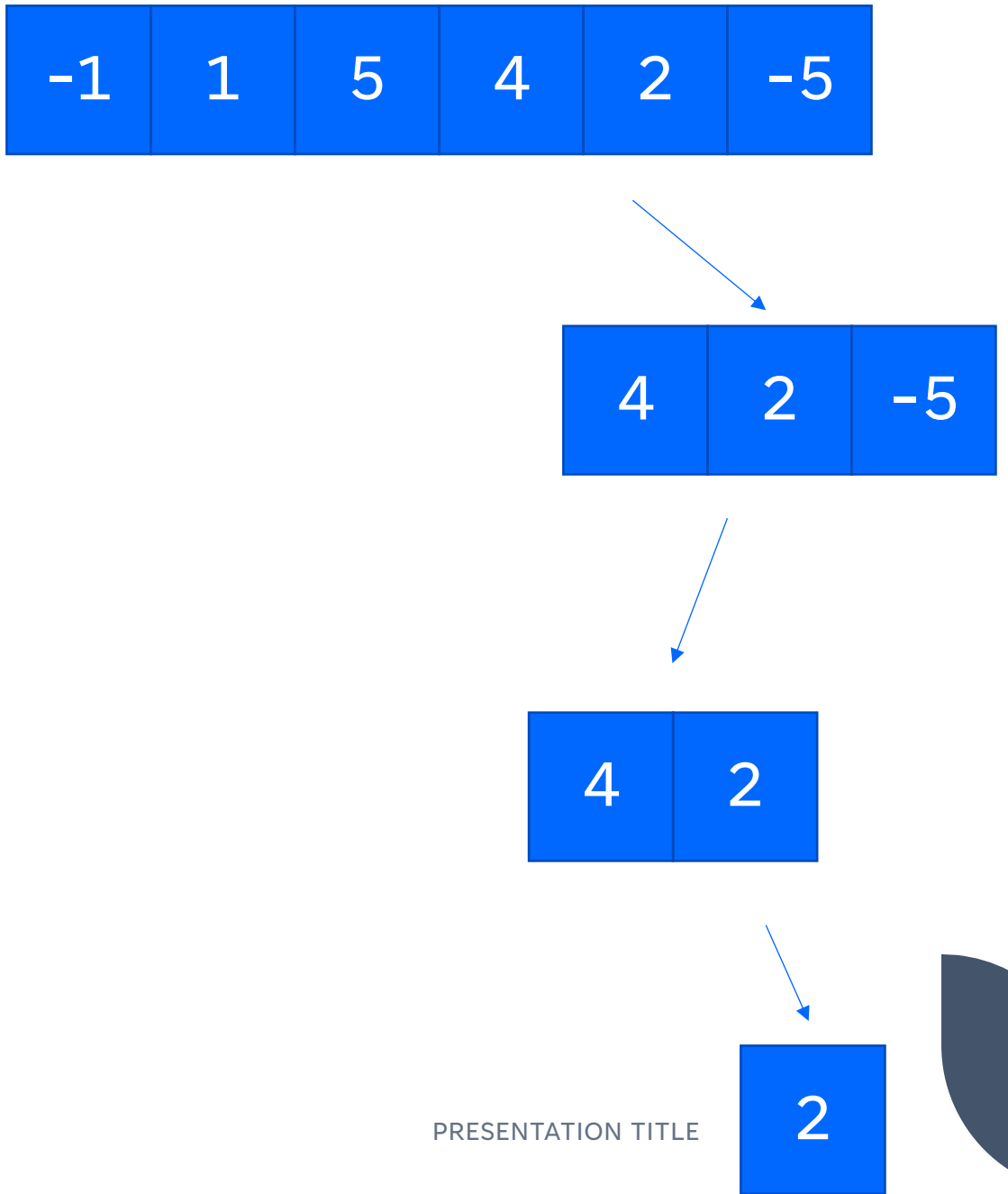
-1	1	5	4	2	-5

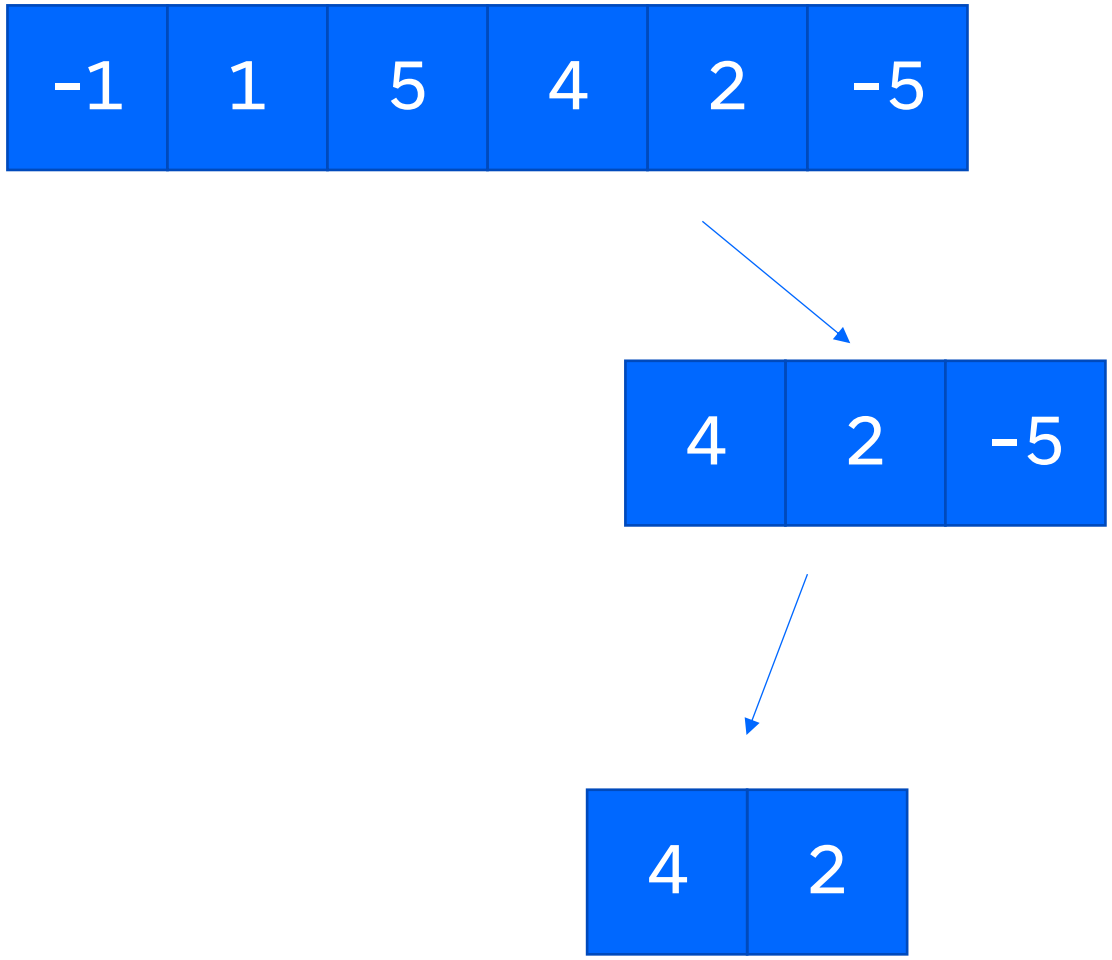
4	2	-5

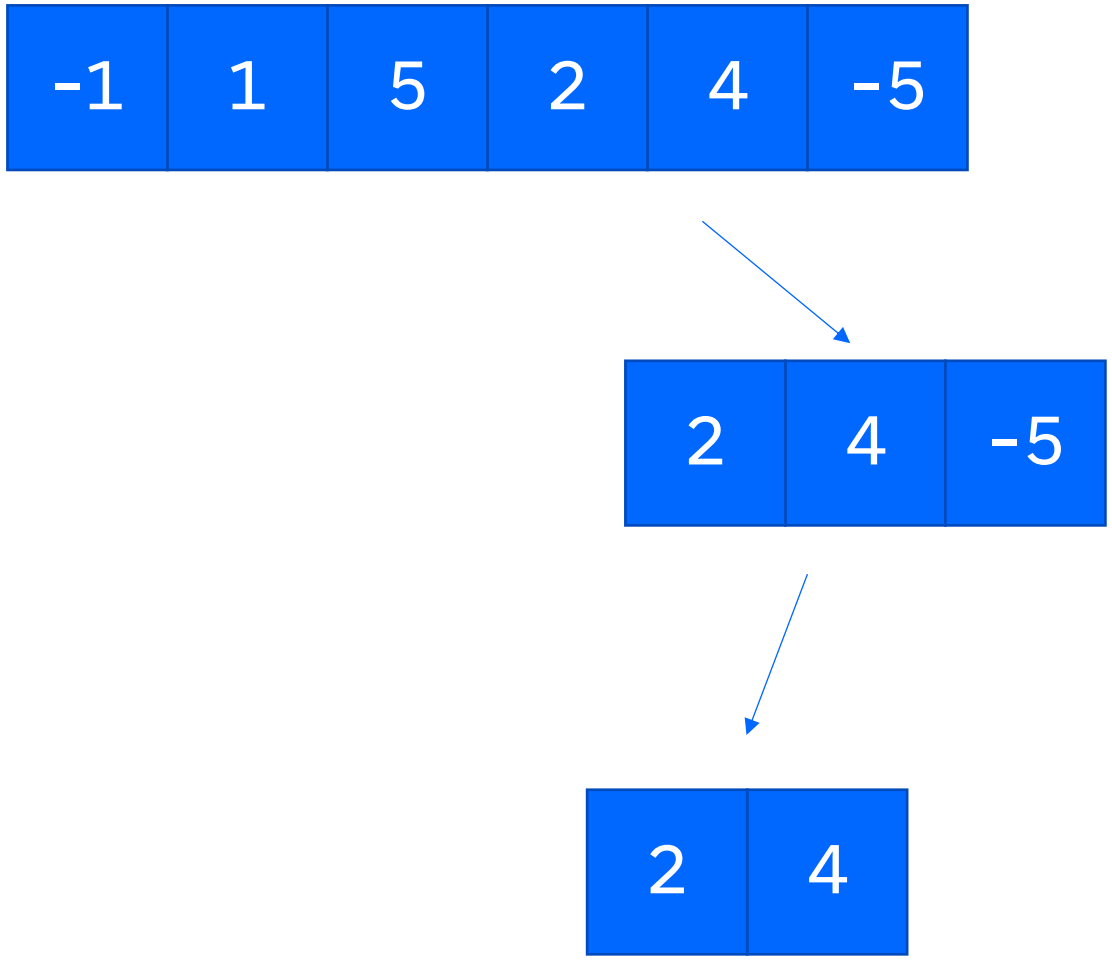

-1
1
5
4
2
-5
4
2
-5
4
2


-1 1 5 4 2 -5
4 2 -5
4 2
4


-1
1
5
4
2
-5
4
2
-5
4
2

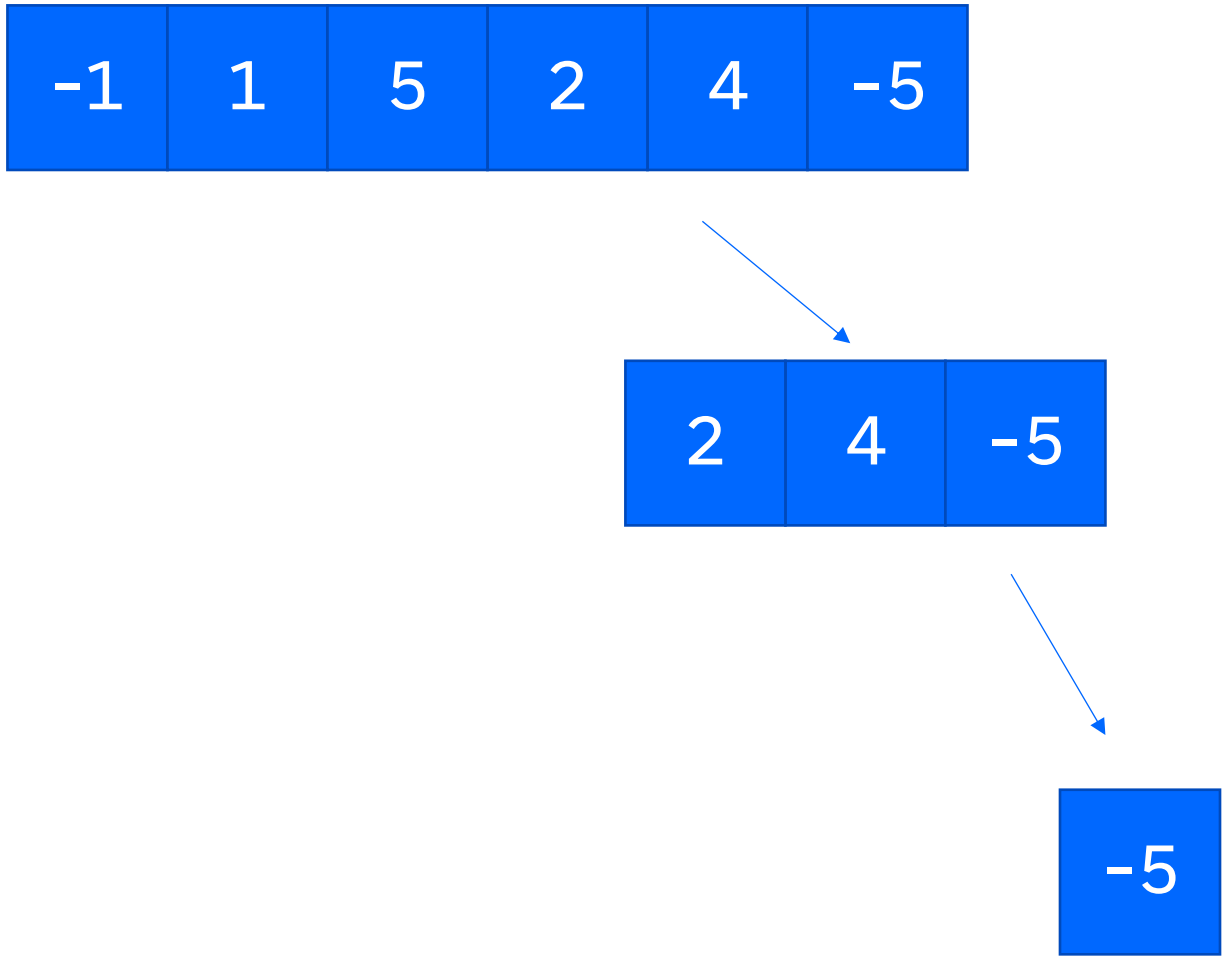

-1 1 5 4 2 -5
4 2 -5
4 2
2


-1
1
5
4
2
-5
4
2
-5
4
2


-1
1
5
2
4
-5
2
4
-5
2
4

-1	1	5	2	4	-5

2	4	-5


-1
1
5
2
4
-5
2
4
-5
-5

-1	1	5	2	4	-5

2	4	-5

-1	1	5	-5	2	4

-5	2	4

-1	1	5	-5	2	4

-5	-1	1	2	4	5

合併(Merge)

- 每次找當前左半邊或右半邊最小的數字加入到tmp陣列
- 複製tmp陣列回原本的陣列

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5					

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5					

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1				

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1				

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1			

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1			

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1	2		

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1	2		

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1	2	4	

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1	2	4	

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1	2	4	5

左半邊

-1	1	5

右半邊

-5	2	4

tmp陣列

-5	-1	1	2	4	5

Code

```
void merge_sort(int l, int r) {
    if (l == r) return;

    int mid = (l + r) / 2;

    merge_sort(l, mid);
    merge_sort(mid + 1, r);

    int pl = l, pr = mid + 1, idx = l;
    while (pl <= mid && pr <= r) {
        if (a[pl] < a[pr]) {  // 左半邊
            tmp[idx++] = a[pl];
            pl++;
        } else {  // 右半邊
            tmp[idx++] = a[pr];
            pr++;
        }
    }
    while (pl <= mid) tmp[idx++] = a[pl++];
    while (pr <= r) tmp[idx++] = a[pr++];
    for (int i = l; i <= r; i++) a[i] = tmp[i];
}
```

C++内建sort()

```
vector<int> a(10);
sort(a.begin(), a.end());
sort(a.begin(), a.begin() + 5);
int b[10];
sort(b, b + 10);
```

Compare Function

```
bool cmp(int a, int b) {
    return a > b;
}

int main() {
    vector<int> a(10);
    sort(a.begin(), a.end(), cmp);
}
```

Compare Function

```
sort(a.begin(), a.end(), greater<int>());
sort(a.begin(), a.end(), less<int>());  //預設
```

stable_sort()

- 如元素相同，相對位置不變
- C++ Sort : n<=16 : insert sort(元素保持stable)
  n >16 : quick sort(元素不一定會stable)

```
stable_sort(a.begin(), a.end());
stable_sort(a.begin(), a.end(), cmp);
```

進階用法

```
sort(a.begin(),a.end(),[&](int i,int j){
    return i>j;
});
```

例題-ZJ a737

- 給一堆一維坐標，你可以選擇一個點，使得該點和所有給定座標距離總和最小，問距離總和為多少。
- [2 8 6 4]

  選擇7，距離總和為：$|2-7|+|8-7|+|6-7|+|4-7|=10$

  選擇5，距離總和為：$|2-5|+|8-5|+|6-5|+|4-5|=8$（最小值）

中位數

- 性質：中位數和其他點的距離總和會最小
- 排序
- 中位數求法

  奇數：最中間的數字

  偶數：最中間的兩個數字相加再除以2

Code

```
void solve() {
    int n, mid;
    cin >> n;
    vector<int> a(n);
    for (auto &i : a) cin >> i;

    sort(all(a));

    if (n % 2 == 0) //偶數
        mid = (a[n / 2] + a[n / 2 - 1]) / 2;
    else  // 奇數
        mid = a[n / 2];

    int ans = 0;
    for (auto i : a) ans += abs(mid - i);
    cout << ans << '\n';
}
```

建表

- 費式數列
- 質數篩法
- 例題

優點與使用時機

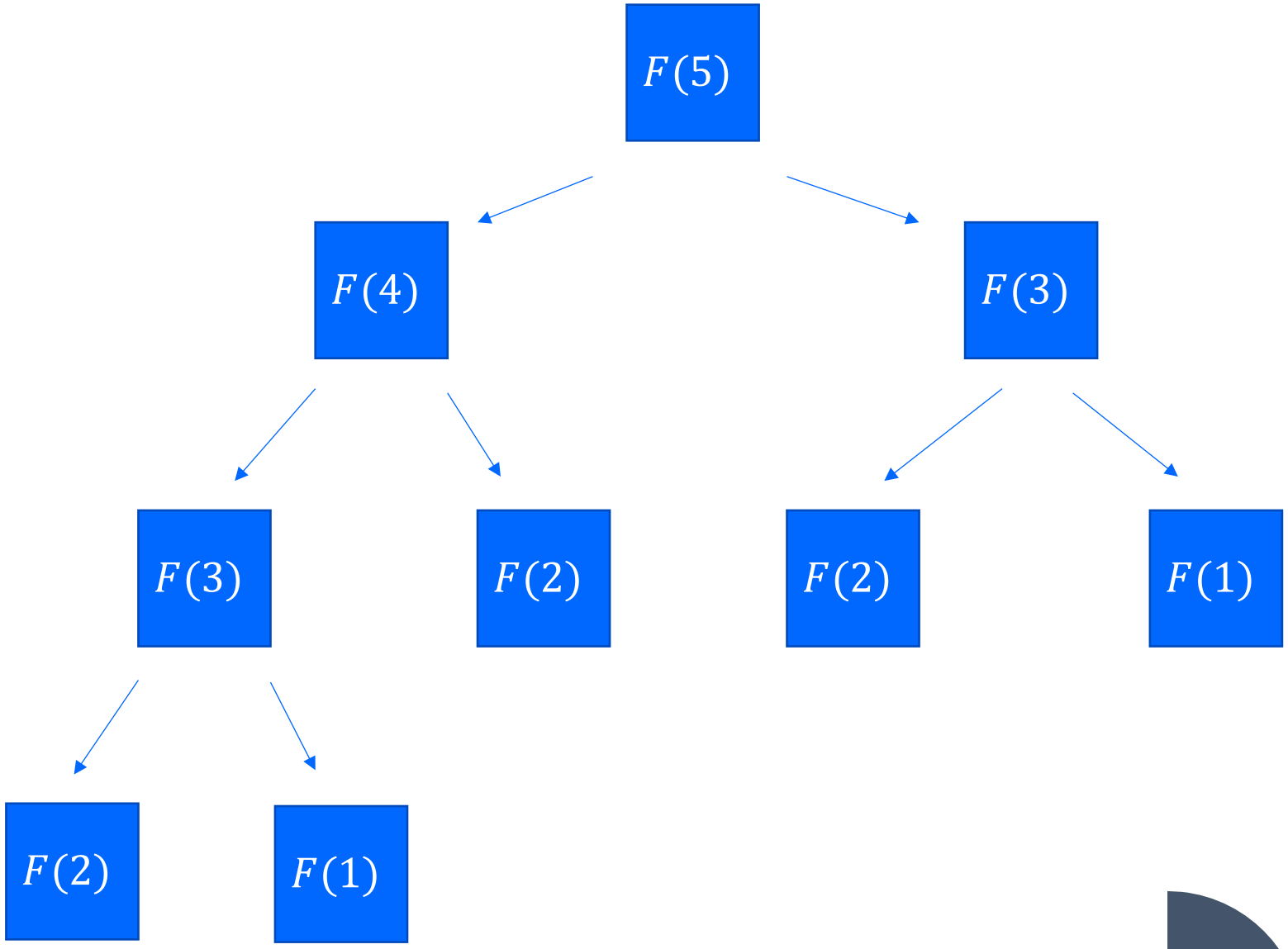
- 預處理
- 查詢$O(1)$
- 多組詢問
- 值域範圍<=1e6

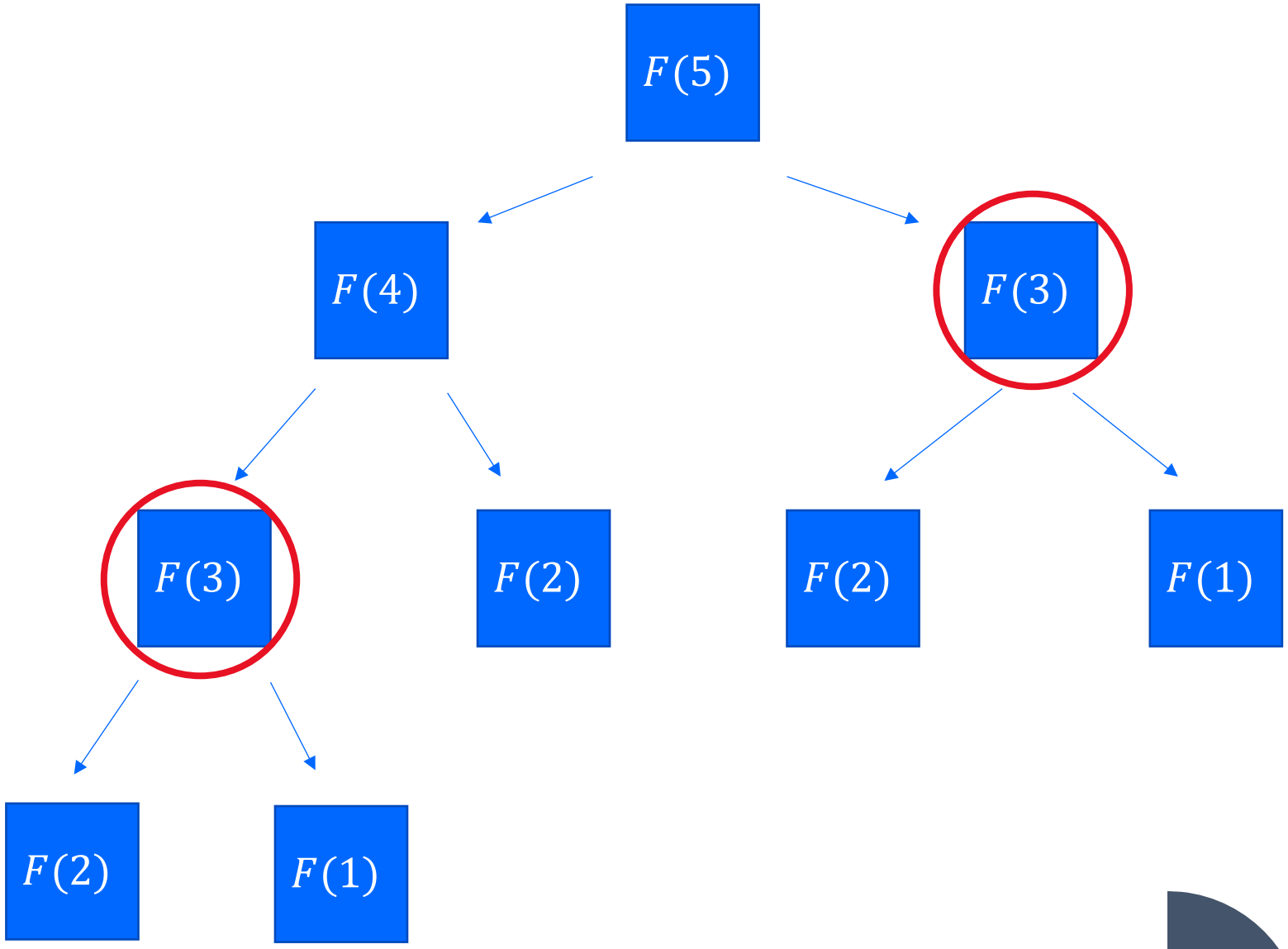
費式數列定義

- $F(n) = f(n-1) + f(n-2)$

```
int f(int n) {
    if(n == 1) return 1;
    if(n == 2) return 1;

    return f(n - 1) + f(n - 2);
}
```


$F(5)$
$F(4)$
$F(3)$
$F(3)$
$F(2)$
$F(2)$
$F(1)$
$F(2)$
$F(1)$


$F(5)$
$F(4)$
$F(3)$
$F(3)$
$F(2)$
$F(2)$
$F(1)$
$F(2)$
$F(1)$

Code

```
int f(int n) {
    if(n == 1) return 1;
    if(n == 2) return 1;
    if(ans[n] != -1) return ans[n];

    ans[n] = f(n - 1) + f(n - 2);
    return ans[n];
}
```

改寫一下

```
fi[1] = 1;
fi[2] = 1;
for (int i = 3; i < N; i++) {
    fi[i] = fi[i - 1] + fi[i - 2];
}
```

質數

- 定義：只能被1和自己整除

```
bool isPrime(int n) {
    if (n < 2) return false;
    if (n == 2) return true;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return false;
    }
    return true;
}
```

質數建表－埃式篩法

- 當遇到質數時，把它的倍數都篩掉
- 12=2*2*3=2*6

Code

```
vector<bool> isprime(N+1, true);
isprime[0] = false;
isprime[1] = false;
for (int i = 2; i <= N; i++) {
    if (isprime[i]) {
        for (int j = i * 2; j <= N; j += i) {
            isprime[j] = false;
        }
    }
}
```

C. Gunjyo 與骰子 (Gunjyo and dice)

- 給 target ，將三個數字(1~100)相乘或相加
- 問有幾種湊法
- 有Q(2e5)組詢問
- Target=2，有5種湊法：{1·1+1},{1+1·1},{2·1·1},{1·2·1},{1·1·2}

Code

```
for (int i = 1; i <= 100; i++) {
    for (int j = 1; j <= 100; j++) {
        for (int k = 1; k <= 100; k++) {
            ans[i * j * k]++;
            ans[i + j + k]++;
            ans[i * j + k]++;
            ans[i + j * k]++;
        }
    }
}

int Q, N;
cin >> Q;
while (Q--) {
    cin >> N;
    cout << ans[N] << '\n';
}
```

二分搜

- 原理
- 實作
- 複雜度計算

二分搜原理

- 排序
- 分成兩半，每次往左半或右半繼續搜尋
- 注意無窮迴圈

在序列中找到target

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 9$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 9$

$$mid = \frac{1+9}{2} = 5\,, array[5] = 9$$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 9$

$$mid = \frac{1+9}{2} = 5\,, array[5] = 9$$

因為 target(4) < array[mid] (9)，所以往左邊搜，$R = mid - 1 = 4$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 4$

$$mid = \frac{1+9}{2} = 5\,, array[5] = 9$$

因為 target(4) < array[mid] (9)，所以往左邊搜，$R = mid - 1 = 4$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 4$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 4$

$$mid = \frac{1+4}{2} = 2\,, array[2] = 2$$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 4$

$$mid = \frac{1+4}{2} = 2\,, array[2] = 2$$

因為 array[mid] (2) < target(4) ，所以往右邊搜，$L = mid + 1 = 3$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 3$
$R = 4$

$$mid = \frac{1+4}{2} = 2\,, array[2] = 2$$

因為 array[mid] (2) < target(4) ，所以往右邊搜，$L = mid + 1 = 3$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 3$
$R = 4$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 3$
$R = 4$

$$mid = \frac{3+4}{2} = 3\,, array[3] = 4$$

Target：4

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 3$
$R = 4$

$$mid = \frac{3+4}{2} = 3\,, array[3] = 4$$

$array[mid](4) = target(4)$, 找到了！！

Code

```
int binary_search(int target) {
    int l = 0, r = n - 1;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (a[mid] == target) {
            return mid;
        } else if (target < a[mid]) {  // 往左搜
            r = mid - 1;
        } else {  // 往右搜
            l = mid + 1;
        }
    }
    return -1;
}
```

序列中找<=target最大的數

注意

- $mid = \frac{L+R+1}{2} = ceil(\frac{L+R}{2})$ ，要取高位
- 否則會變成無窮迴圈

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 9$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 9$

$$mid = \frac{1 + 9 + 1}{2} = 5\,, array[5] = 9$$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 1$
$R = 9$

$$mid = \frac{1 + 9 + 1}{2} = 5\,, array[5] = 9$$

因為 array[mid] (9) < target(11)，右邊可能會有更好的解，$\mathrm{L} = mid = 5$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 9$

$$mid = \frac{1 + 9 + 1}{2} = 5\,, array[5] = 9$$

因為 array[mid] (9) < target(11)，右邊可能會有更好的解，$\mathrm{L} = mid = 5$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 9$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 9$

$$mid = \frac{5 + 9 + 1}{2} = 7\,, array[7] = 12$$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 9$

$$mid = \frac{5 + 9 + 1}{2} = 7\,, array[7] = 12$$

因為 target(11) < array[mid] (12)，解一定在左邊，$\mathrm{R} = mid - 1 = 6$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 6$

$$mid = \frac{5 + 9 + 1}{2} = 7\,, array[7] = 12$$

因為 target(11) < array[mid] (12)，解一定在左邊，$\mathrm{R} = mid - 1 = 6$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 6$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 6$

$$mid = \frac{5 + 6 + 1}{2} = 6\,, array[6] = 10$$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 5$
$R = 6$

$$mid = \frac{5 + 6 + 1}{2} = 6\,, array[6] = 10$$

因為 array[mid] (10) < target(11) ，右邊可能會有更好的解，$\mathrm{L} = mid = 6$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 6$
$R = 6$

$$mid = \frac{5+6+1}{2} = 6\,, array[6] = 10$$

因為 array[mid] (10) < target(11) ，右邊可能會有更好的解，$\mathrm{L} = mid = 6$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 6$
$R = 6$

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 6$
$R = 6$

只剩一個數字，判斷是否符合條件：array[6]=10 < target(11)

Target：11

array	1	2	4	7	9	10	12	15	20
index	1	2	3	4	5	6	7	8	9

$L = 6$
$R = 6$

只剩一個數字，判斷是否符合條件：array[6]=10 < 11

找到了！！答案：10

Code

```
int binary_search(int target) {
    int l = 0, r = n - 1;
    while (l < r) {
        int mid = (l + r + 1) / 2;
        if (a[mid] <= target) {  // 往右搜
            l = mid;
        } else {  // 往左搜
            r = mid - 1;
        }
    }
    if (a[l] <= target) {
        return a[l];  // 合法
    } else {
        return -1;  // 找不到
    }
}
```

練習題

- Lower bound (>=target中最小的數)
- Upper bound (> target中最小的數)

Lower bound(>=target中最小的數)

```
int l = 0, r = n - 1;
while (l < r) {
    int mid = (l + r) / 2;
    if (a[mid] < target)  // 不合法
        l = mid + 1;
    else
        r = mid;
}

if (target <= a[l])
    cout << a[l] << '\n';
else
    cout << -1 << '\n';
```

Upper bound (> target中最小的數)

```
int l = 0, r = n - 1;
while (l < r) {
    int mid = (l + r) / 2;
    if (a[mid] <= target)  // 不合法
        l = mid + 1;
    else
        r = mid;
}

if (target < a[l])
    cout << a[l] << '\n';
else
    cout << -1 << '\n';
```

C++內建函式庫

- Vector

```
lower_bound(a.begin(),a.end(),value);
upper_bound(a.begin(),a.end(),value);
```

- Set

```
s.lower_bound(value);
s.upper_bound(value);
```

複雜度計算

- $O(\log2(n))$ ，log2(1e9)=29.897
- Checker: $O(n * \log2(n))$，n=1e6的話大約2e8

各種二分搜技巧

- 跳躍式二分搜
- 二分搜保險寫法
- 二分搜經典題
- 二分搜答案

跳躍式二分搜

- 從0開始，dis為跳躍距離，每次判斷a[cur+dis]有沒有符合條件，有就跳，沒有就把dis減半，重複執行，直到dis=0。

```
int binary_search(int target) {  //找<=target中最大的數
    int cur = 0;
    for (int dis = n - 1; dis > 0; dis /= 2) {
        while (cur + dis < n && a[cur + dis] <= target) {  // 可以跳就跳
            cur += dis;
        }
    }
    if (a[cur] <= target) {
        return a[cur];  // 合法
    } else {
        return -1;  // 找不到
    }
}
```

二分搜保險寫法

- 為了不要變成無窮迴圈，合法時直接更新答案，並把範圍縮小。

```
int binary_search(int target) {  // 找<=target中最大的數
    int l = 0, r = n - 1, ans = -inf;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (a[mid] <= target) {  // 合法，往右搜
            ans = max(ans, a[mid]);
            l = mid + 1;
        } else {  // 往左搜
            r = mid - 1;
        }
    }
    return ans == -inf ? -1 : ans;
}
```

二分搜經典題

- 找$\sqrt{n}$ 的整數
- Leetcode 33. Search in Rotated Sorted Array
- Leetcode 34. Find First and Last Position of Element in Sorted Array

找$\sqrt{n}$ 的整數

- 大數實作開根號

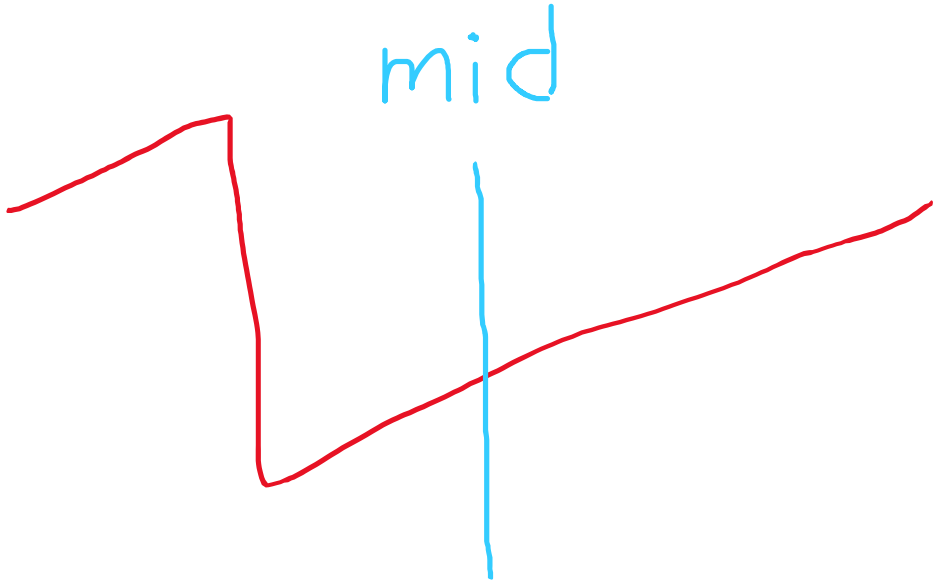
```
int l = 1, r = 1e9;
while (l < r) {
    int mid = (l + r + 1) / 2;
    if (mid * mid > n)
        r = mid - 1;
    else
        l = mid;
}
cout << l << '\n';
```

Search in Rotated Sorted Array

- 原本為排序好的陣列[1 2 3 4 5 6 7]
- 題目會隨機找一個地方進行旋轉 [5 6 7 1 2 3 4]
- 給一target
- 找出target的位置
- $O(\log n)$

觀察性質

- [L, R] 遞增：正常二分搜判斷
- [L, R] 有斷層：斷層在左半邊，右邊嚴格遞增 [5 6 0 1 2 3 4]
- [L, R] 有斷層：斷層在右半邊，左邊嚴格遞增 [2 3 4 5 6 0 1]



Code

```
int l = 0, r = nums.size() - 1;
while (l <= r) {
    int mid = (l + r) / 2;
    if (nums[mid] == target) return mid;

    if (nums[l] <= nums[r]) {  // 遞增
        if (target < nums[mid])
            r = mid - 1;
        else
            l = mid + 1;
    } else if (nums[l] > nums[mid]) {  // 斷層段在左邊
        if (nums[mid] <= target && target <= nums[r])
            l = mid + 1;
        else
            r = mid - 1;
    } else {  // 斷層段在右邊
        if (nums[l] <= target && target <= nums[mid])
            r = mid - 1;
        else
            l = mid + 1;
    }
}
return -1;
```

Find First and Last Position of Element in Sorted Array

- 排序好的序列 [1 2 4 6 8 8 8 10]
- target 8
- 問targe在序列中起始和結束的索引值 ans=(4, 6)
- $O(\log n)$

性質分析

一般二分搜判斷：

target < middle : r=mid-1

middle < target : l=mid+1

搜起始點：

middle=target 時往左搜, r=mid

搜結束點：

middle=target 時往右搜, l=mid

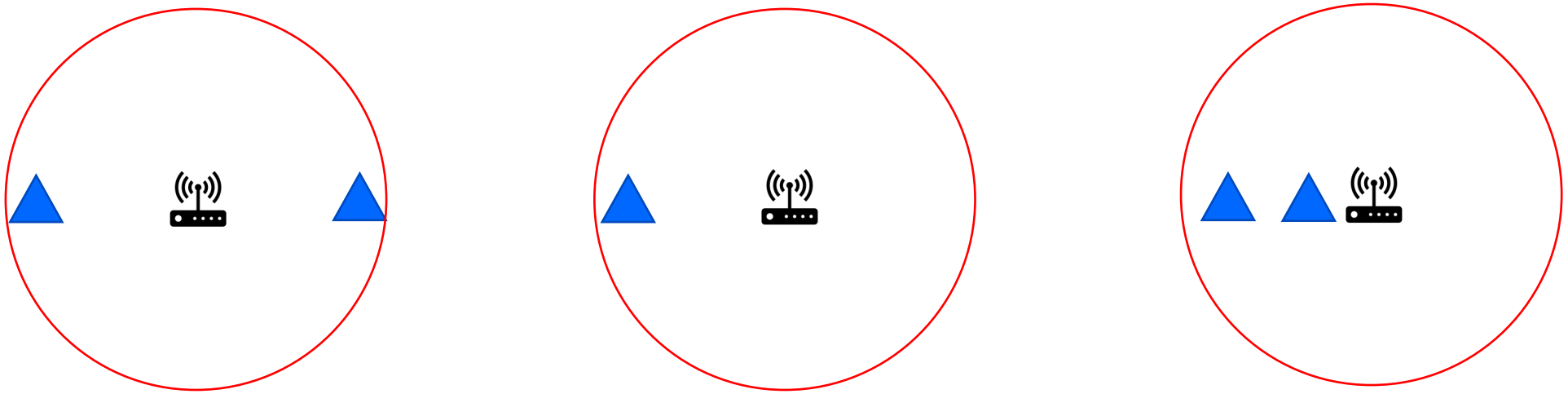
Code

```
int l = 0, r = nums.size() - 1;
while (l < r) {  //find left bound
    int mid = (l + r) / 2;
    if (target == nums[mid])
        r = mid;
    else if (target < nums[mid])
        r = mid - 1;
    else
        l = mid + 1;
}
ansL = l;
l = 0, r = nums.size() - 1;
while (l < r) {  //find right bound
    int mid = (l + r + 1) / 2;
    if (target == nums[mid])
        l = mid;
    else if (target < nums[mid])
        r = mid - 1;
    else
        l = mid + 1;
}
ansR = l;
```

二分搜答案

- 猜答案
- 枚舉答案：TLE
- 具備單調性：二分搜

例題：APCS基地台



客戶

基地台

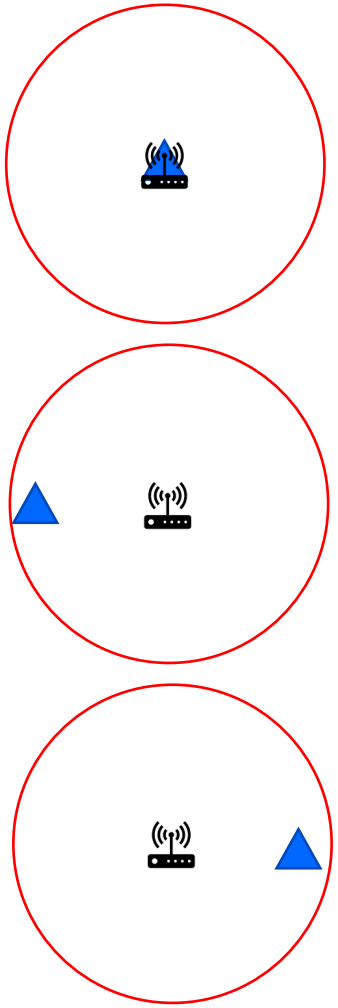
觀察答案

- $R = \infty$: 可行
- $R = 0$: 不可行
- 答案分佈： 000000011111
- 具備單調性 : 二分搜

```
while(l<r){
    int mid=(l+r)/2;
    if(check(mid)){
        r=mid;
    }else{
        l=mid+1;
    }
}
```

判斷解是否可行-最少基地台數量

- 客戶位置
- 基地台半徑



Code

```
bool check(int R){
    int right_bound=-1,cnt=0;
    for(auto x:a){
        if(x<=right_bound) continue;
        right_bound=x+R;
        cnt++;
    }
    return cnt<=k;
}
```

練習時間

雙指針

- 字串翻轉
- 快慢指針
- 對撞指針
- 例題

字串翻轉

gindoC evoL I

I Love Coding

g	n	i	d	o	C		e	v	o	L		I

I	n	i	d	o	C		e	v	o	L		g

I	n	i	d	o	C		e	v	o	L		g

I		i	d	o	C		e	v	o	L	n	g

I		i	d	o	C		e	v	o	L	n	g

I		L	d	o	C		e	v	o	i	n	g

I		L	d	o	C		e	v	o	i	n	g

I		L	o	o	C		e	v	d	i	n	g

I		L	o	o	C		e	v	d	i	n	g

I		L	o	v	C		e	o	d	i	n	g

I		L	o	v	C		e	o	d	i	n	g

I		L	o	v	e		C	o	d	i	n	g

I		L	o	v	e		C	o	d	i	n	g

Code

```
int l = 0, r = s.length() - 1;
while (l < r) {
    swap(s[l], s[r]);
    l++, r--;
}
```

快慢指針-字串匹配

- 建立兩個指針，分別指向兩個陣列的開頭
- 遇到匹配的字母才一起動

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

s: Pneumonoultramicroscopicsilicovolcanoconiosis

t: ultra

Code

```
bool match(string s, string t) {
    int ps = 0, pt = 0;
    while (ps < s.length()) {
        if (s[ps] == t[pt])
            pt++;
        else
            pt = 0;

        if (pt == t.length()) return true;
        ps++;
    }
    return false;
}
```

對撞指針-兩數和問題

- 給一整數序列和一target，問哪兩個數相加會等於target

序列： 9 2 11 3 8 5　　target=8

解答：3+5=8

怎麼解

- 枚舉$O(n^2)$：TLE
- 排序
- 雙指標

Target:8

2	3	5	9	11

Sum:13

Target:8

2	3	5	9	11

Sum:11

Target:8

2	3	5	9	11

Sum:7

Target:8

2	3	5	9	11

Sum:8

找到了！！

Code

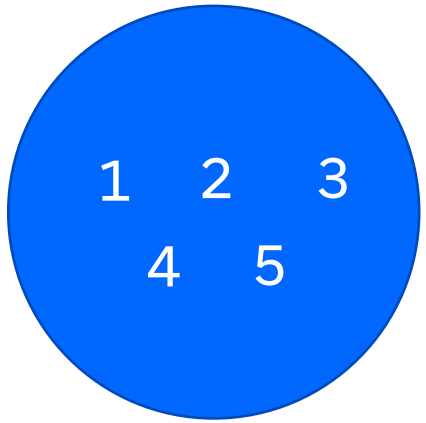
```
sort(a, a + n);
int l = 0, r = n - 1;
while (l < r) {
    int sum = a[l] + a[r];
    if (sum == target) break;
    if (sum > target) r--;
    if (sum < target) l++;
}
cout << a[l] << ' ' << a[r] << '\n';  // 3 5
```

例題：Colten 與風原的餅乾大戰爭

- 有兩堆餅乾，每堆中餅乾的價錢都相同
- 每塊餅乾都有各自的滿意度
- 問最少花費得到至少T的滿意度

T=10

4 塊



9塊

input

```
5 4 9 10
1 2 3 4 5
8 1 3 4 7
```

output

```
12
```

Code

```
int p1 = 0, p2 = 0, sum = 0, ans = inf;
while (p1 <= n) {
    while (p2 > 0 && sum - b[p2 - 1] >= t) sum -= b[--p2];  //多退
    while (p2 < n && sum < t) sum += b[p2++];               //少補

    if (sum >= t) ans = min(ans, p1 * c1 + p2 * c2);
    sum += a[p1++];
}
```

例題

https://codeforces.com/edu/course/2/lesson/9/3/practice/contest/307094/problem/A

https://cses.fi/problemset/task/1084

https://cses.fi/problemset/task/1090

例題-A. Looped Playlist

- 給一序列和k
- 每次只能往右走(最後一個接第一個)，可重複走(繞一圈)
- 自己決定起點
- 問起點和最少要走多少步才能讓路徑上的數字總和>=k

input

```
9 10
1 2 3 4 5 4 3 2 1
```

output

```
3 3
```

input

```
5 6
3 1 1 1 4
```

output

```
5 2
```

input

```
3 100
10 10 10
```

output

```
1 10
```

想法

- 判斷要走起圈：K/sum，剩餘K%sum
- 枚舉起點，判斷終點
- 環形 小訣竅：複製陣列接在後面

Code

```
int base = (k / sum) * n;
k %= sum;

int idx = 0, ans = inf;
for (int i = 1, cnt = 0, j = 1; i <= n; i++) {
    cnt -= a[i - 1]; //左界往右移一格
    while (j <= 2 * n && cnt < k) cnt += a[j++];  //總和不夠，往右拉數字進來
    if (cnt < k) break;  //剩餘數字已經不夠用了，直接break
    if (j - i < ans) {  //找到更短的
        ans = j - i;
        idx = i;
    }
}
cout << idx << ' ' << base + ans << '\n';
```

例題-Ferris Wheel

- N個小朋友要坐摩天輪，每個小朋友有不同的體重
- 一節車廂可坐1~2人，並且有承重限制
- 問最少需要多少節車廂才能讓所有人坐完

```
Input:
4 10
7 2 3 9

Output:
3
```

想法

- 考慮最大的，要馬自己做要馬跟別人做
- 最大的根誰配會最有可能成功？最小的
- 最大配最小
- 1 2 5 6 7 8, k=10

Code

```
sort(all(a));
int l = 0, r = n - 1, ans = 0;
while (l <= r) {
    ans++;
    if (l == r) break;

    if (a[l] + a[r] <= k)
        l++, r--;
    else
        r--;
}
```

例題-Apartments

- 給a b兩整數序列，ai和bi可配對iff ai-k<=bi<=ai+k
- 求最大配對數

```
Input:
4 3 5
60 45 80 60
30 60 75

Output:
2
```

想法

- 排序
- 貪心：由左往右掃，能配就配
- 考慮最左邊的bi，拿越左邊ai的來配越好，不要跟後面的搶

Code

```
sort(all(a));
sort(all(b));
int ans = 0;
for (int i = 0, j = 0; j < m; j++) {  //枚舉bi
    while (i < n && a[i] + k < b[j]) i++;  //找到一組合法的區間
    if (a[i] - k <= b[j] && b[j] <= a[i] + k) {  //檢查有沒有在區間裡
        ans++;
        i++;
    }
}
```