

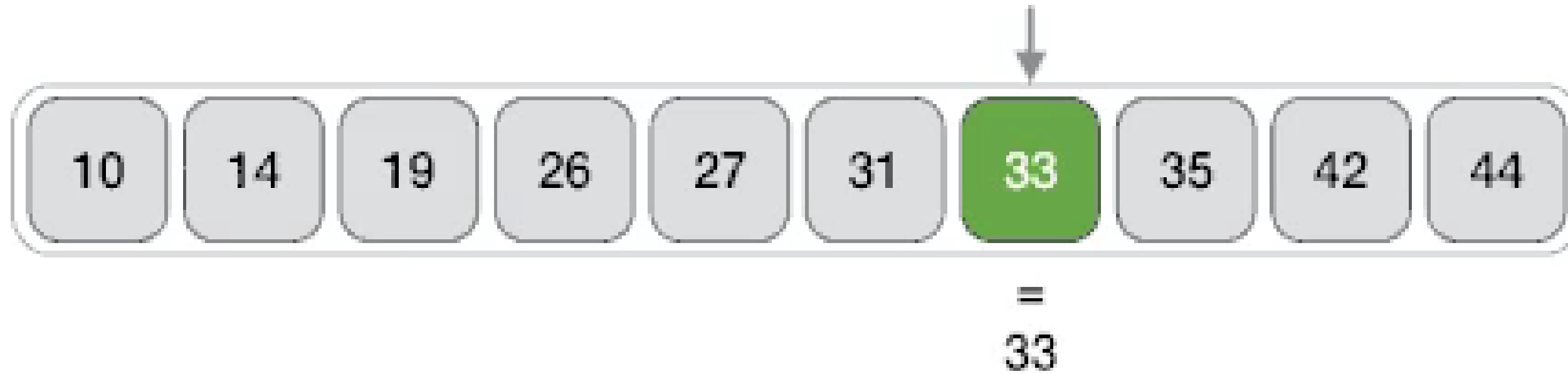
基礎演算法

下學期最後一次社課

兩個基礎的搜尋演算法

1.線性搜尋(暴力)

Linear Search



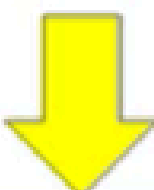
**線性搜尋是一種簡單而直觀的搜尋方法，適用於小型資料集合
或未排序的資料。**

**時間複雜度為 $O(n)$ ，其中 n 是資料集合中的元素數量。最壞情
況下，需要搜索整個集合。**

**線性搜尋適用於資料量不大且搜尋操作不需要高
效率的情況。**

2.二元搜尋(Binary search)

Search for 47



0	4	7	10	14	23	45	47	53
---	---	---	----	----	----	----	----	----

概念簡單，搜尋高效，達到對數執行時間 $O(\log n)$

不需額外實作資料結構或配置記憶體空間。

只能搜尋已排序的序列。

題目

給定一整數陣列 $a[0]$ 、 $a[1]$ 、...、 $a[99]$ 且 $a[k]=3k+1$ ，以 $value=100$ 呼叫以下兩函式，假設函式 **f1** 及 **f2** 之 **while** 迴圈主體分別執行 $n1$ 與 $n2$ 次 (i.e, 計算 **if** 敘述執行次數，不包含 **else if** 敘述)，請問 $n1$ 與 $n2$ 之值為何？ 註： $(low + high)/2$ 只取整數部分。

```
int f1(int a[], int value) {
    int r_value = -1;
    int i = 0;
    while (i < 100) {
        if (a[i] == value) {
            r_value = i;
            break;
        }
        i = i + 1;
    }
    return r_value;
}
```

```
int f2(int a[], int value) {
    int r_value = -1;
    int low = 0, high = 99;
    int mid;
    while (low <= high) {
        mid = (low + high)/2;
        if (a[mid] == value) {
            r_value = mid;
            break;
        }
        else if (a[mid] < value) {
            low = mid + 1;
        }
        else {
            high = mid - 1;
        }
    }
    return r_value;
}
```

- (A) $n1=33, n2=4$
- (B) $n1=33, n2=5$
- (C) $n1=34, n2=4$
- (D) $n1=34, n2=5$

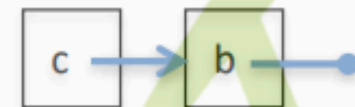
16. List 是一個陣列，裡面的元素是 `element`，它的定義如右。List 中的每一個 `element` 利用 `next` 這個整數變數來記錄下一個 `element` 在陣列中的位置，如果沒有下一個 `element`，`next` 就會記錄 -1。所有的 `element` 串成了一個串列 (linked list)。例如在 `list` 中有三筆資料

1	2	3
data = 'a'	data = 'b'	data = 'c'
next = 2	next = -1	next = 1

它所代表的串列如下圖



`RemoveNextElement` 是一個程序，用來移除串列中 `current` 所指向的下一個元素，但是必須保持原始串列的順序。例如，若 `current` 為 3 (對應到 `list[3]`)，呼叫完 `RemoveNextElement` 後，串列應為



請問在空格中應該填入的程式碼為何？

- (A) `list[current].next = current ;`
- (B) `list[current].next = list[list[current].next].next ;`
- (C) `current = list[list[current].next].next ;`
- (D) `list[list[current].next].next = list[current].next ;`

```
struct element {
    char data;
    int next;
}

void RemoveNextElement (element
list[], int current) {
    if (list[current].next != -1) {
        /*移除 current 的下一個 element*/
        
    }
}
```

下面哪組資料若依序存入陣列中，將無法直接使用二分搜尋法搜尋資料？

(A) a, e, i, o, u

(B) 3, 1, 4, 5, 9

(C) 10000, 0, -10000

(D) 1, 10, 10, 10, 100

演算法

54. 給定一個 1x8 的陣列 **A**，**A** = {0, 2, 4, 6, 8, 10, 12, 14}。右側函式 **Search(x)** 真正目的是找到 **A** 之中大於 **x** 的最小值。然而，這個函式有誤。請問下列哪個函式呼叫可測出函式有誤？

- (A) **Search(-1)**
- (B) **Search(0)**
- (C) **Search(10)**
- (D) **Search(16)**

```
int A[8]={0, 2, 4, 6, 8, 10, 12, 14};

int Search (int x) {
    int high = 7;
    int low = 0;
    while (high > low) {
        int mid = (high + low)/2;
        if (A[mid] <= x) {
            low = mid + 1;
        }
        else {
            high = mid;
        }
    }
    return A[high];
}
```

什麼是貪心演算法？

貪心算法是一種每一步都選擇當前最優解的策略，旨在通過一系列局部最優選擇達到全局最優。

在生活中舉個例子

假設你身邊有很多很多的零錢
你要幫忙換錢
如何使用最少的零錢來達成目標

題目大意：

已知有 1 、 5 、 10 、 50 、 100 、 500 、 1000 元的幣值（有紙鈔以及硬幣）。

現在每列輸入給定一正整數 n （ $1 \leq n \leq 50000$ ），請找出用最少的紙鈔或硬幣組成金額 n 的方法。

範例輸入：

552

1246

10000

範例輸出：

$552 = 500 * 1 + 50 * 1 + 1 * 2$

$1246 = 1000 * 1 + 100 * 2 + 10 * 4 + 5 * 1 + 1 * 1$

$10000 = 1000 * 10$

解題思維：

雖然是硬幣問題，如[此題](#)。

但是這題的幣值可以使用貪心演算法（Greedy Algorithm）——先盡量使用面額較大的硬幣、紙鈔，然後才使用面額較小的。

參考解答

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  // 用來存儲每種面值的硬幣數量
5  int back[7] = {0};
6  // 硬幣面值
7  int mlist[7] = {1000, 500, 100, 50, 10, 5, 1};
8
9  // 計算如何用最少的硬幣來達成目標金額
10 void moneychange(int a) {
11     for (int i = 0; i < 7; i++) {
12         back[i] = a / mlist[i]; // 計算使用當前面值的硬幣數量
13         a = a % mlist[i];      // 更新剩餘金額
14     }
15 }
16
17 int main() {
18     int n = 0;
19     cin >> n; // 輸入目標金額
20     moneychange(n);
21     cout << n << " = ";
22
23     bool first = true; // 用來控制是否輸出“ + ”
24     for (int i = 0; i < 7; i++) {
25         if (back[i] != 0) {
26             if (!first) {
27                 cout << " + ";
28             }
29             cout << mlist[i] << "*" << back[i];
30             first = false; // 第一次輸出之後設置為 false
31         }
32     }
33     cout << endl;
34     return 0;
35 }

```

**感謝各位一學期的參與
有緣在見~**