# THUẬT TOÁN TÌM KIẾM NHỊ PHÂN

## 2.1. Thuật toán tìm kiếm nhị phân là gì ?

Thuật toán tìm kiếm nhị phân là một thuật toán cơ bản nhất trong tất cả các thuật toán tìm kiếm, đóng vai trò quan trọng trong việc giải quyết các vấn đề tìm kiếm trong các mảng dữ liệu đã được sắp xếp, sử dụng nguyên tắc chia để trị để hiệu quả hóa quá trình tìm kiếm.

Thuật toán bắt đầu việc so sánh giá trị ở giữa mảng dữ liệu với giá trị cần tìm kiếm. Nếu chúng bằng nhau, tìm kiếm kết thúc. Ngược lại, thuật toán sẽ xác định xem giá trị cần tìm kiếm lớn hơn hay nhỏ hơn giá trị ở giữa mảng, từ đó quyết định loại bỏ nửa mảng không cần thiết đi. Quá trình này sẽ được lặp lại và mỗi lần loại bỏ một nửa mảng thì phạm vi tìm kiếm cũng sẽ giảm theo.

## 2.2. Tại sao lại cần có thuật toán nhị phân?

Những lý do sau đây làm cho tìm kiếm nhị phân trở thành lựa chọn tốt hơn để sử dụng làm thuật toán tìm kiếm:

- Tìm kiếm nhị phân hoạt động hiệu quả trên dữ liệu được sắp xếp bất kể kích thước của dữ liệu
- Thay vì thực hiện tìm kiếm bằng cách duyệt qua dữ liệu theo trình tự, thuật toán nhị phân truy cập ngẫu nhiên dữ liệu để tìm phần tử cần thiết. Điều này làm cho chu kỳ tìm kiếm ngắn hơn và chính xác hơn.
- Tìm kiếm nhị phân thực hiện so sánh dữ liệu được sắp xếp dựa trên nguyên tắc sắp xếp so với sử dụng so sánh bằng nhau, phương pháp này chậm hơn và hầu như không chính xác.
- Sau mỗi chu kỳ tìm kiếm, thuật toán chia kích thước của mảng thành một nửa, do đó ở lần lặp tiếp theo, nó sẽ chỉ hoạt động ở nửa còn lại của mảng.

### 2.3. Ý tưởng triển khai thuật toán

Thuật toán tìm kiếm nhị phân là một thuật toán khá thông dụng và chỉ dùng được với một mảng đã sắp xếp. Để triển khai thuật toán này, phải chắc chắn rằng mảng đã được sắp xếp. Sau đây là ý tưởng triển khai của thuật toán:

- Chia đôi mảng và gọi 2 phần chia đôi đó là left và right.
- Phần tử đứng ở giữa left và right được gọi là mid.
- Sau đó dựa vào phần tử mid để tìm xem giá trị cần tìm nó nằm trên mảng left hay mảng right.
- Nếu giá trị cần tìm nằm trên left thì sẽ loại bỏ mảng right và chỉ thực hiện tìm kiếm trên left và ngược lại.


*Hình 2. 1 - Hình ảnh mô phỏng thuật toán tìm kiếm nhị phân*

### 2.4. Cách thức hoạt động của thuật toán tìm kiếm nhị phân

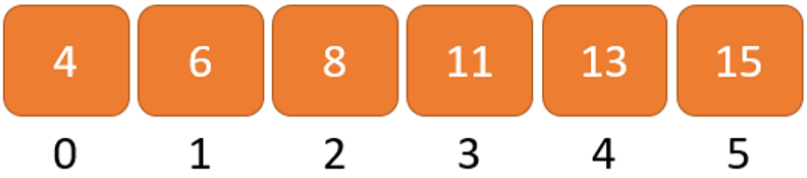
Thuật toán tìm kiếm nhị phân có thể được thực hiện theo 2 phương sau đâu:

+ Phương pháp sử dụng vòng lặp.
+ Phương pháp sử dụng đệ quy.

Sự lựa chọn giữa hai phương pháp này thường phụ thuộc vào ngôn ngữ lập trình và yêu cầu cụ thể của ứng dụng cần thực hiện.

### 2.1.1. Các bước chung cho cả 2 phương pháp:

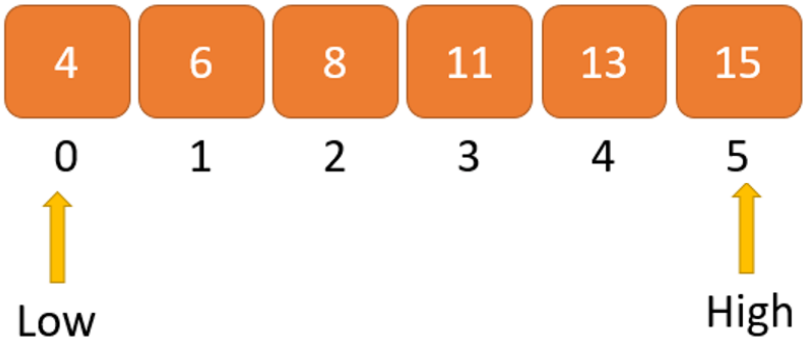
- Bước 1: Giả sử ta có mảng chứa phần tử cần tìm có dạng như sau:




*Hình 2. 2 - Hình ảnh minh họa Bước 1 của thuật toán tìm kiếm nhị phân*

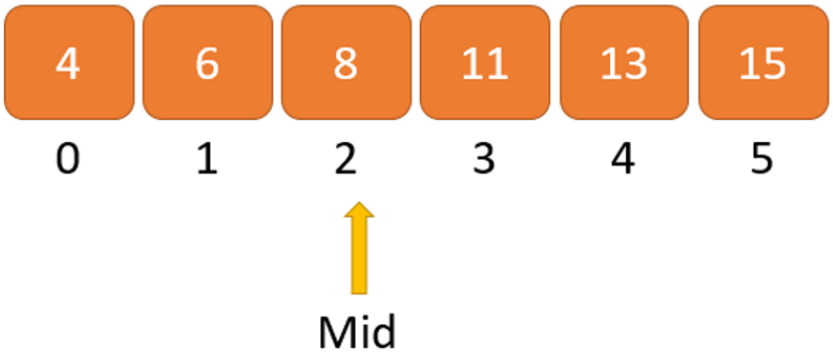
  - Gọi x = 6 là phần tử cần tìm.

- Bước 2: Đặt hai con trỏ low và high lần lượt ở các vị trí đầu mảng và cuối mảng




*Hình 2. 3 - Hình ảnh minh họa Bước 2 của thuật toán tìm kiếm nhị phân*

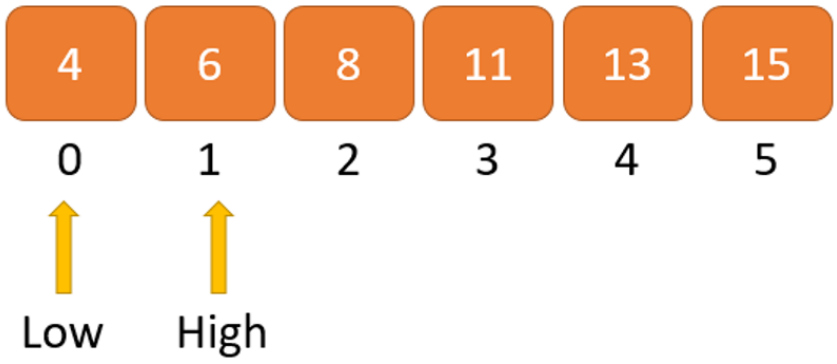
- Bước 3: Tìm phần tử giữa ở giữa mảng tức là (arr[high + low)/2]) = 8




*Hình 2. 4 - Hình ảnh minh họa Bước 3 của thuật toán tìm kiếm nhị phân*

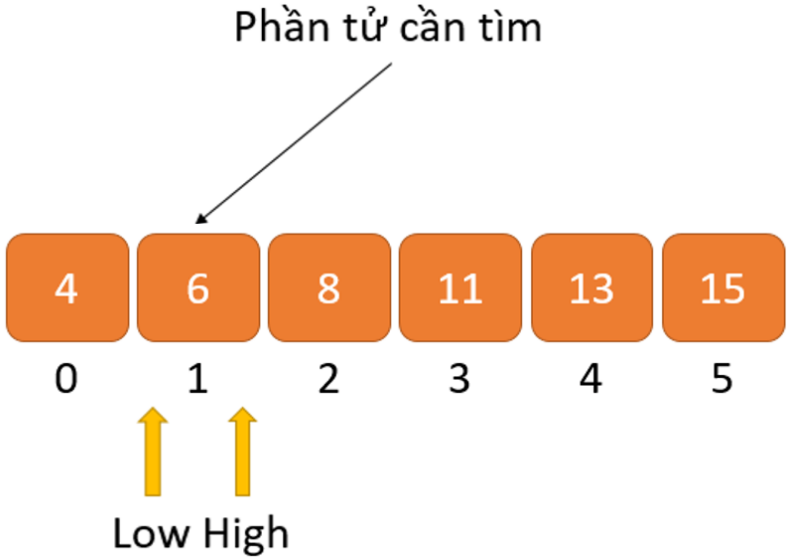
- Bước 4: Nếu x == mid thì trả về mid. Nếu không thì, so sánh phần tử cần tìm với m.

- Bước 5: Nếu x > mid, ta sẽ so sánh x với phần tử ở giữa của các phần tử ở phí bên phải của phần tử mid. Điều này được thiệc hiện bằng cách đặt low = mid + 1.
- Bước 6: Mặt khác, so sánh x với phần tử ở giữa của các phần tử ở bên trái của phần tử mid. Điều này được thực hiện bằng cách thiết lập high = mid – 1.




*Hình 2. 5 - Hình ảnh minh họa Bước 6 của thuật toán tìm kiếm nhị phân*

- Bước 7: Lặp lại từ bước 3 đến bước 6 cho đến khi low = high.
- Bước 8: Và cuối cùng ta sẽ tìm được phần tử cần tìm là 6.




*Hình 2. 6 - Hình ảnh minh họa Bước 8 của thuật toán tìm kiếm nhị phân*

### 2.1.2. Cài đặt python của thuật toán tìm kiếm nhị phân bằng vòng lặp

```
n, x = map (int, input () .split () )
A = list(map(int, input() .split()))

t = 0
p = n – 1
kq = -1
while (kq == -1):
    #Xác định vị trí i giữa t và p
        i = (t + p) // 2
    #Kiểm tra nếu x bằng phần tử số i
        if A[i] == x:
            kq = i #Đặt điều kiện để dừng vòng lặp
    #Nếu x lớn hơn, bỏ qua nửa bên trái
        elif A[i] < x:
        t = i + 1
    #Nếu x nhỏ hơn, bỏ qua nửa bên phải
        else:
        p = i – 1
print(“Phần tử số”, kq, “có giá trị”, x)
```

- Kiểm thử:

| | |
|---|---|
| Input: 7 10<br>Output: 4 7 8 10 14 21 22<br>Phần tử số 3 có giá trị 10 | Input: 7 22<br>Output: 4 7 8 10 14 21 22<br>Phần tử số 6 có giá trị 22 |

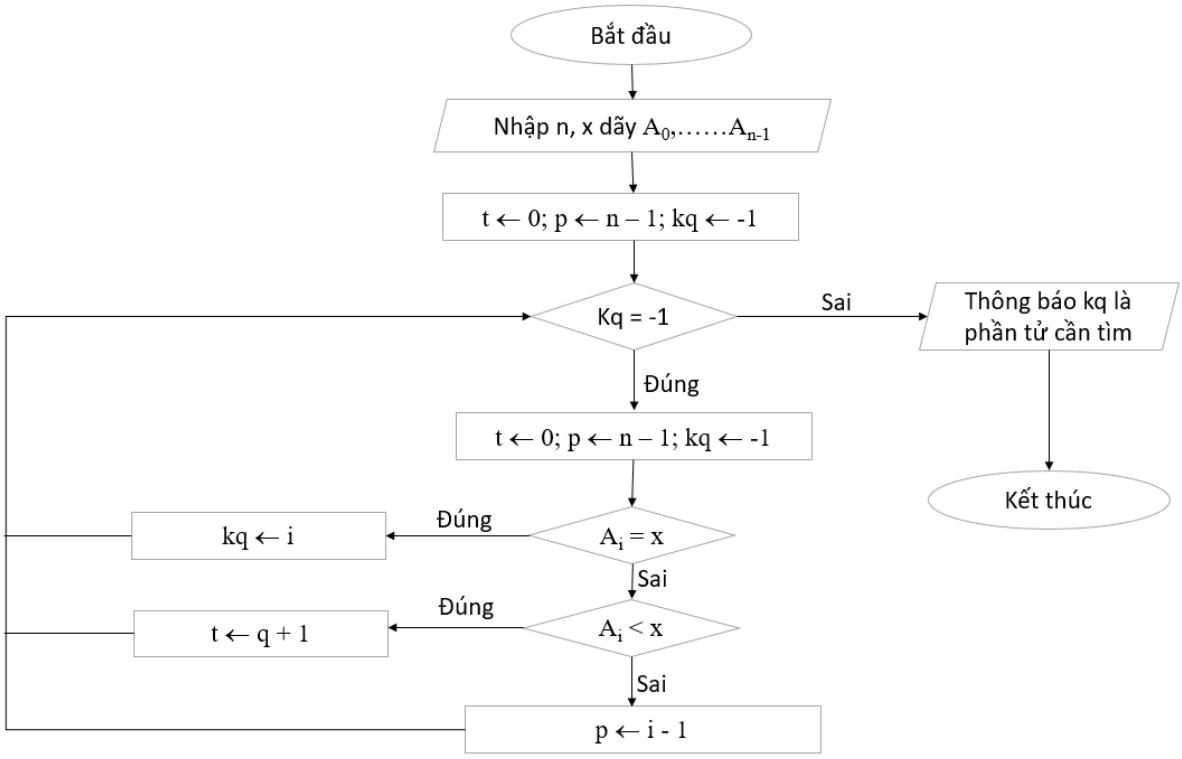
### 2.1.3. Cài đặt python của thuật toán tìm kiếm nhị phân bằng đệ qui

```
def TKNP(A,t,p,x):
    if t <= p:
        #Xác định vị trí giữa t và p
        i = (t+p)//2
        #Kiểm tra nếu x bằng phần tử số i
        if A[i] == x:
            return i
        #Nếu x nhỏ hơn, bỏ qua nửa bên phải,
        #gọi đệ quy TKNP cho phần bên trái
        elif A[i] > x:
            return TKNP (A, t, i-1, x)
         #Nếu x lớn hơn, bỏ qua nửa bên trái,
        #gọi đệ quy TKNP cho phần bên phải
        else:
            return TKNP (A, i+1, p, x)
    else:
          #Không tồn tại phần tử cần tìm trong mảng
        return -1
n,x = map(int,input().split())
A = list(map(int,input().split()))
#Gọi đệ quy
kq = TKNP(A,0,len(A)-1,x)
if kq == -1:
    print("Không tồn tại phần tử có giá trị",x)
else:
    print("Phần tử số",kq,"có giá trị",x)
```

➢ Kiểm thử:

| | |
|---|---|
| Input: 6 21<br>Output: 4 7 8 10 14 21<br>Phần tử số 5 có giá trị 21 | Input: 6 11<br>Output: 4 7 8 10 14 21<br>Không tồn tại phần tử có giá trị 11 |

### 2.5. Sơ đồ khối các bước thực hiện của thuật toán nhị phân (Binary search)




*Hình 2. 7 - Hình ảnh sơ đồ khối của thuật toán tìm kiếm nhị phân*

### 2.6. Bài tập

**Bài tập 1:** Viết chương trình nhập vào 1 mảng số nguyên và một số nguyên k, hãy đếm xem có bao nhiêu số nguyên bằng k. Nhập tiếp 2 số x < y và đếm xem có bao nhiêu số lớn hơn x và nhỏ hơn y.

**Bài tập 2:** Cài đặt thuật toán tìm kiếm nhị phân theo kiểu đệ qui.

**Bài tập 3:** Viết chương trình nhập một mảng các số nguyên từ bàn phím, nhập 1 số nguyên S, hãy đếm xem có bao nhiêu cặp số của mảng ban đầu có tổng bằng S, có hiệu bằng S.

**Bài tập 4:** Viết chương trình sinh một dãy các số nguyên. Hãy tim và đưa ra vị trí, giá trị của số dương đầu tiên trong dãy, số nguyên tố cuối cùng trong dãy.