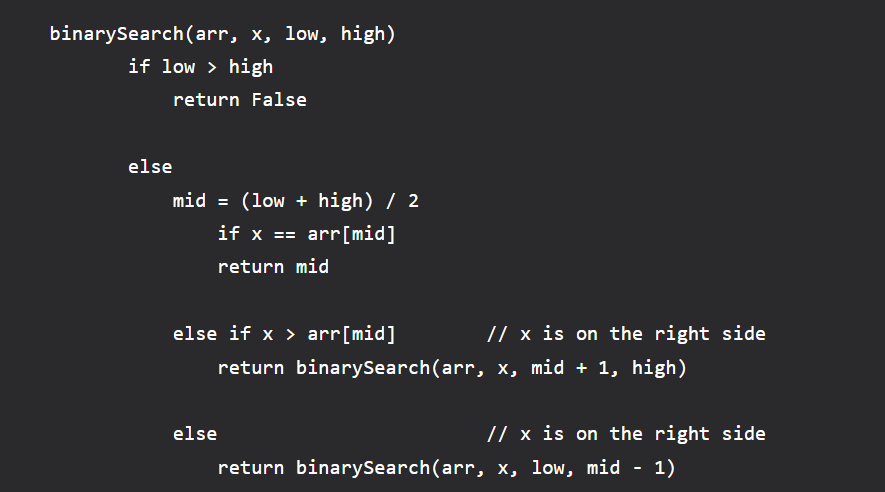
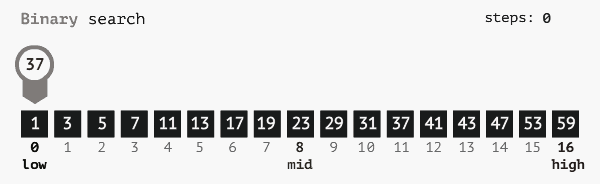
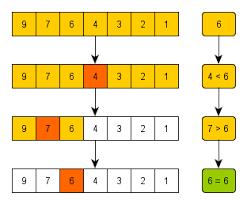
Binary – Search

* **Tìm kiếm nhị phân là thuật toán tìm kiếm 1 giá trị trong mảng đã được sắp xếp.**
* Thuật toán tiến hành so sánh giá trị cần tìm với phần tử đứng giữa mảng. Nếu hai giá trị không bằng nhau, phần nửa mảng không chứa giá trị cần tìm sẽ bị bỏ qua và tiếp tục tìm kiếm trên nửa còn lại, một lần nữa lấy phần tử ở giữa và so sánh với giá trị cần tìm, cứ thế lặp lại cho đến khi tìm thấy giá trị đó. Nếu phép tìm kiếm kết thúc khi nửa còn lại trống thì giá trị cần tìm không có trong mảng.
* Ý tưởng thuật toán: Xét một đoạn trong mảng **arr[left...right]**. Lúc này giá trị của left và right lần luợt là 0 và số phần tử của mảng - 1.
  + So sánh x với phần tử nằm ở vị trí chính giữa của mảng **(mid = (left + right) /2)**. Nếu x bằng arr[mid] thì trả về vị trí và thoát vòng lặp.
  + Nếu **x < arr[mid]**thì chắc chắn x sẽ nằm ở phía bên trái tức là ở khoảng **arr[left .... mid - 1]**.
  + Nếu **x > arr[mid]** thì chắc chắn x sẽ nằm ở phía bên phải mid tức là ở khoảng **arr[mid + 1 ... right]**.
  + Tiếp tục thực hiện chia đôi các khoảng tìm kiếm tới khi nào tìm thấy được vị trí của x trong mảng hoặc khi đã duyệt hết mảng.



* Hình ảnh minh họa:





* Độ phức tạp thuật toán: trường hợp tốt nhất là O(1) – khi tìm kiếm lần đầu tiên và tìm thấy giá trị cần tìm, trường hợp xấu nhất là O(log(n)) – khi không tìm thấy hoặc khi tìm kiếm đến lần cuối cùng.

Ví dụ

Cho 2 số n, k(1 <= n, k <= 105). Có 1 mảng n phần tử đã được sắp xếp theo thứ tự không giảm. Có k truy vấn, mỗi truy vấn là 1 số, kiểm tra số đó có trong mảng hay không. Tất cả các phần tử mảng và truy vấn đều là số nguyên, mỗi phần tử trong số đó không vượt quá |109|. Nếu có in ra “YES”, ngược lại là “NO”.

**Example:**

10 10  
1 61 126 217 2876 6127 39162 98126 712687 1000000000  
100 6127 1 61 200 -10000 1 217 10000 1000000000

