Binary Search and Variations

- **Idea:** cut search space in half (about half) by asking only one question.
- Pure binary search

$x_1, x_2, \ldots, x_n$ is a sequence of real numbers such that

$$x_1 \le x_2 \le \cdots \le x_n$$

<u>Problem:</u> Given a real number z, we want to find whether z appears in the sequence, and if it does, to find an index i such that $x_i = z$.

Solution: binary search!

Binary search:

Question: "Is $x_{n/2} < z$?"

Yes: binary search range becomes $x_{n/2+1}, \ldots x_n$

No: binary search range becomes $x_1, \ldots, x_{n/2}$

Complexity: $O(\log_2 n)$.

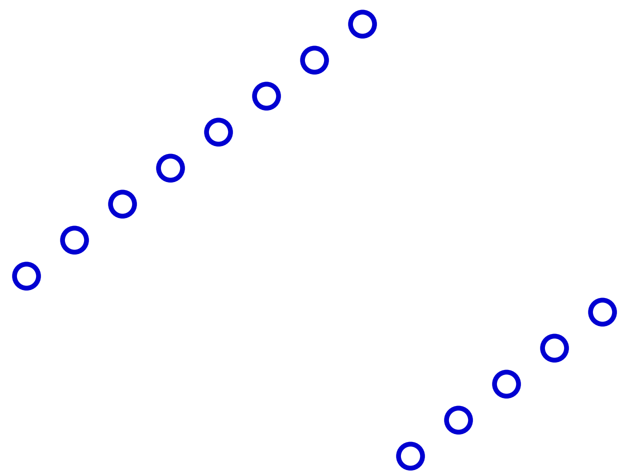
Binary search in a cyclic sequence

Definition. A sequence $x_1, x_2, \ldots, x_n$ is said to be **cyclically sorted** if the smallest number in the sequence is x_i for some unknown i, and the sequence

$$x_i, x_{i+1}, \ldots, x_n, x_1, x_2, \ldots, x_{i-1}$$

is sorted.

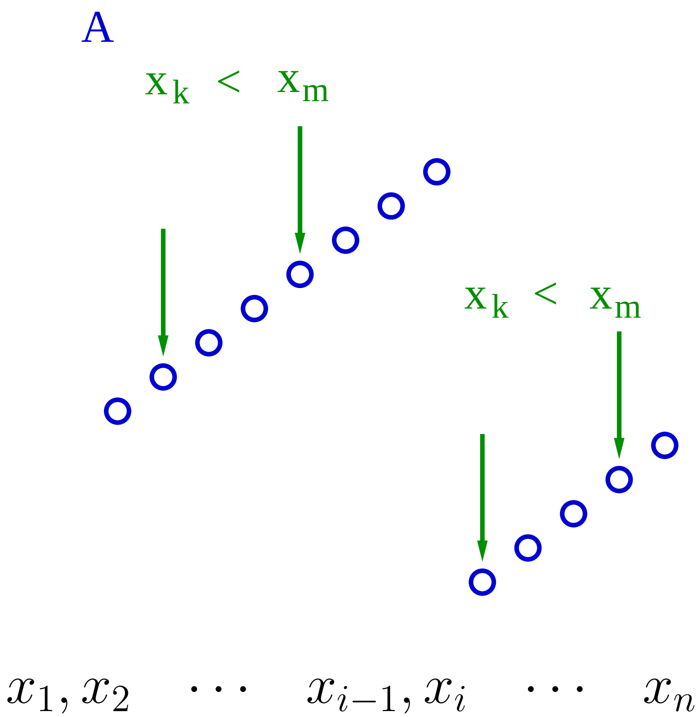
<u>The Problem:</u> Given a cyclically sorted list, find the position of the minimum element in the list (we assume elements are distinct).

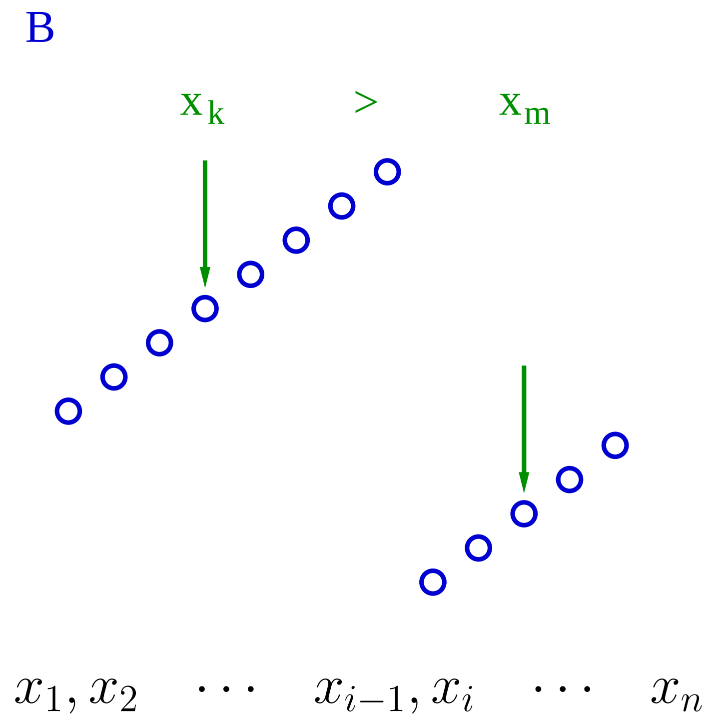


$$x_1, x_2 \quad \cdots \quad x_{i-1}, x_i \quad \cdots \quad x_n$$

Solution: For any two numbers x_k, x_m, such that $k < m$, compare x_k with x_m.

1. If $x_k < x_m$ then i cannot be in the range $k < j \leq m$ (k is possible)
2. If $x_k > x_m$ then i must be in the range $k < j \leq m$





† Is $x_{n/2} < x_n$?

Yes: search range $1 \ldots n/2$. *No:* search range $n/2 + 1, \ldots n$.

† Find the index of the smallest element in $O(\log n)$ time.

Binary search for a special index

The Problem: Given a sorted sequence of distinct integers $a_1, a_2, \ldots, a_n$, determine whether there exists an index i such that $a_i = i$.

If key is not given, binary search cannot be done. The principle still works.

Compare: $a_{n/2}$ with $n/2$

$$
\begin{aligned}
a_{n/2} = n/2 & \quad \text{found!} \\
a_{n/2} < n/2 & \implies a_{n/2-1} \leq a_{n/2} - 1 < n/2 - 1 \\
& \implies a_{n/2-1} < n/2 - 1, \ldots, a_1 < 1 \\
& \implies \text{ search range: } n/2 + 1, \ldots, n \\
a_{n/2} > n/2 & \implies a_{n/2+1} > n/2 + 1, \ldots, a_n > n \\
& \implies \text{ search range: } 1, \ldots, n/2 - 1
\end{aligned}
$$

Can find index i such that $a_i = i$ in $O(\log(n))$ time.

Binary search in sequence of unknown size

Sometimes we *double the search space* instead of halving it.

Consider binary search with size of the sequence unknown!

- Try to find x_i such that $z \leq x_i$,
  then binary search in the range $1, \ldots, i$.

  More specifically,

  † compare z to x_j, $j \geq 1$

  † Assume $x_j < z$, $j \geq 1$, try z and x_{2j}

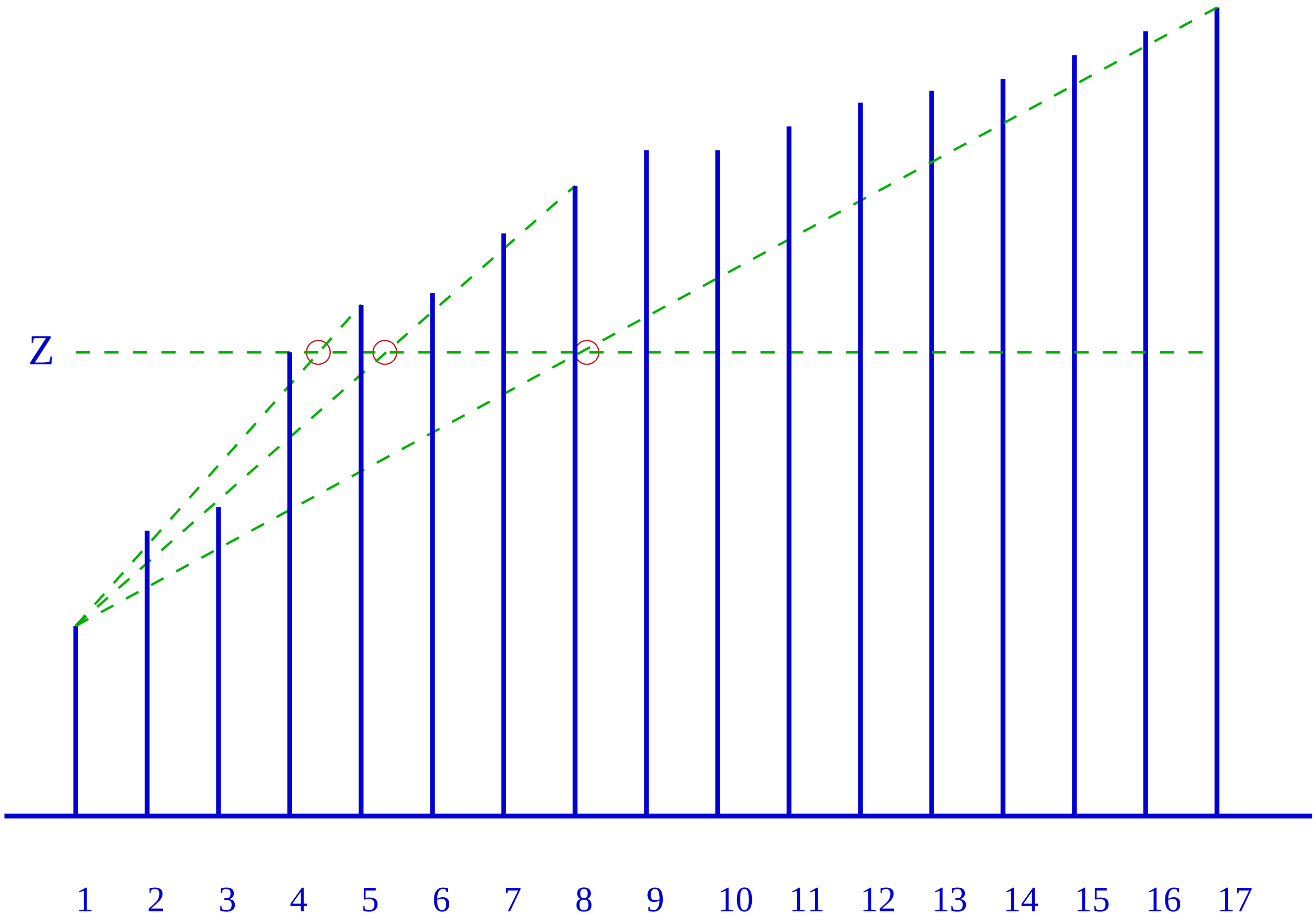
  † If $z \leq x_{2j}$ then $x_j < z \leq x_{2j}$
- Can find z in $O(\log j)$ additional comparisons

If i is the smallest index such that $z \leq x_i$ then:

$O(\log i)$ to find an x_j such that $z \leq x_j$
$O(\log i)$ to find x_i.

Interpolation Search

- In binary search, the search space is always cut in half (which guarantees the logarithmic time).
- However, if we find a value that is very close to the search number z, it seems reasonable to continue the search in that "neighborhood" instead of blindly going to the next half point.
  Example: open a book, try to find a certain page. We want to find page 200 in a 800 page book. We do not start from about half. We try about one-fourth.
- Search range $l, \ldots r$
  If z is close to x_l we should choose something near l.
  We can use the ratio $d = [z - x_l]/(x_r - x_l)$
  We next try $l + d(r - l)$.
- If $z - x_l \cong x_r - z$, this is binary search
- For random keys, interpolation search uses less than $lg(lg(N)) + 1$ comparisons.
- Can be used for very large n.



Interpolation search