$O(n \log n)$ time algorithm to find the longest monotonically increasing subsequence of sequence of n numbers

sequence element

For new $X[i]$, if $X[i]$ > last element in sequence S
add $X[i]$ into S
else $X[i] \leq$ last element in sequence S
find smallest element that is $> X[i]$
$S[k] < X[k]$
$X[i] \leq S[k+1]$
replace $S[k+1]$ with $X[i]$

$X = \langle x_1, x_2, \ldots, x_n \rangle$

length of S = length of longest increasing subsequence of X

LIS(X, n)
$L = 0$
for $i = 1$ to n
binary search for largest positive $j \leq L$ such that $X[M[j]] < X[i]$
$P[i] = M[j]$
if $j == L$ or $X[i] < X[M[j+1]]$
$M[j+1] = i$
$L = \max(L, j+1)$

$M[j]$ stores position k of smallest value $X[k]$ such that there is increasing subsequence of length j ending at $X[k]$ on range $k \leq i$

$P[k]$ stores position of predecessor of $X[k]$ in longest increasing subsequence ending at $X[k]$

L is length of LIS

Last item of longest sequence is in $X[M[L]]$
Second last item of longest sequence is in $X[P[M[L]]]$

Algorithm performs single binary search for n sequence elements $\Rightarrow O(n \log n)$