„Introduction to Algorithms"

Syllabus

2. Searching
   - Linear Search
   - Neighbor Search
   - Binary Search
   - Uniqueness
   - Hashing
   - Order Statistics/ Median
   - 1D Range Searching

- Specification
- Primitives: semantics and cost
- Design
- Analysis
- (Optimality)

2. Searching

***linear* search**

Specification: Fix set X.

Input: $N\in\mathbb{N}$ and finite sequence $x_1,\ldots x_N\in X$ as well as $y\in X$.

in array $x[1\ldots N]$

Output: $n\in\{1,\ldots N\}$ such that $x_n = y$ or $n=0$ if $x_j \neq y$

Primitives: Access $\mathbb{N}\ni n\rightarrow x[n]\in X$ cost 1

Comparison "$x=y$?" cost 1

Index integer arithmetic cost 1

Algorithm:

for $n=1\ldots N$
  if $x[n]=y$ then return n;
return 0.

runtime $O(N)$

correctness: √

2. Searching

neighbor search

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1,\ldots x_N \in X$ as well as $y \in X$.

in array $x[1\ldots N]$

Output: $n \in \{0,\ldots N\}$ such that $x_n \leq y < x_{n+1}$

where $x_{N+1} := \infty$, $x_0 := -\infty$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ — cost 1

ordered comparison "$x \leq y$?" — cost 1

Index integer arithmetic — cost 1

if $x[1] > y$ then return 0;
for $n=1\ldots N$
  if $x[n] > y$ then return n-1;
return N.

runtime $O(N)$

correctness: √

2. Searching

binary search

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1,\ldots x_N \in X$ as well as $y \in X$.

in array $x[1\ldots N]$ with $x_1 \leq \ldots \leq x_N$

Output: smallest $r \leq N+1$ such that $y < x_r$

where $x_{N+1} := \infty$, $x_0 := -\infty$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$ — cost 1

ordered comparison "$x \leq y$?" — cost 1

Index integer arithmetic — cost 1

l:=0; r:=N+1;
while l+1<r do
  $n := \lfloor (l+r)/2 \rfloor$;
  if $y>x[n]$ then r:=n else l:=n;

runtime $O(\log N)$

correctness: $x[l] \leq y < x[r]$

time: $r'-l' \leq \lfloor (r-l)/2 \rfloor$

2. Searching

uniqueness

Specification: Fix set X with total order $\leq$.

Input: $N\in\mathbb{N}$ and finite sequence $x_1,\ldots x_N\in X$ in array $x[1\ldots N]$ with $x_1\leq\ldots\leq x_N$

Output: **1** if all elements are distinct: $\forall i,j$: $x_i=x_j \Rightarrow i=j$

0 if some element is repeated: $\exists i\neq j$: $x_i=x_j$

Primitives: Access $\mathbb{N}\ni n\rightarrow x[n]\in X$ cost 1

ordered comparison "$x\leq y$?" cost 1

Uniq1 ($x[1..N]$)
For m:=2 to N do
  For k:=1 to m-1 do
    If $x[m]=x[k]$ Return **0**;
Return **1**;

runtime $O(N^2)$

Uniq2 ($x[1..N]$)
For m:=1 to $N-1$ do
  If $x[m]=x[m+1]$ Return **0**;
Return **1**;

runtime $O(N)$

2. Searching

uniqueness

Specification: Fix set $X=\{0,\ldots M\text{-}1\}$

Input: $N\in\mathbb{N}$ and finite sequence $x_1,\ldots x_N\in X$ in array $x[1\ldots N]$

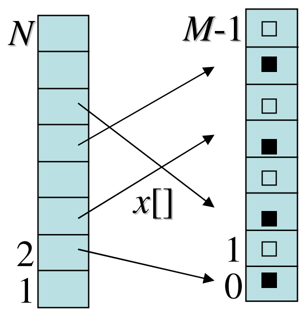
Output: **1** if all elements are distinct: $\forall i,j$: $x_i=x_j \Rightarrow i=j$

0 if some element is repeated: $\exists i\neq j$: $x_i=x_j$

Primitives: Access $\mathbb{N}\ni n\rightarrow x[n]\in X$ cost 1

Uniq3 ($x[1..N]$)
Boolean array *present*[0…M-1]; // initialized with **0/false**
For n:=1 to N do
  If *present*[$x[n]$] then return **0**;
  present[$x[n]$] := **1**;
Endfor; return **1**

runtime $O(N)$
memory $O(M)$



2. Searching

hashing

Specification: Fix set $X=\{0,\dots M\text{-}1\}$

Input: $N\in\mathbb{N}$ and finite sequence $x_1,\dots x_N\in X$ — in array $x[1\dots N]$

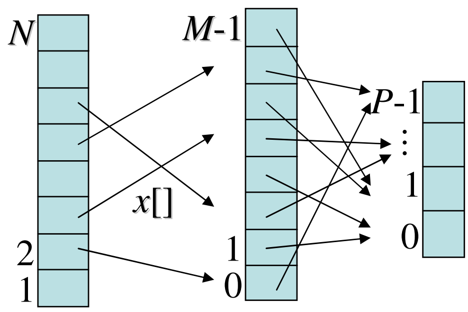
Output: **1** if all elements are distinct: $\forall i,j$: $x_i=x_j \Rightarrow i=j$
0 if some element is repeated: $\exists i\neq j$: $x_i=x_j$

Primitives: Arithmetic (on values!)

Fix prime $P\geq\Omega(N)$ and $k\in\{1,\dots P\text{-}1\}$ random

Integer array $y[0\dots P\text{-}1]$; // initialized with **0**s
For n:=1 to N do ; $p := x[n]\cdot k \mod P$;
  While $y[p]\neq 0$ do
    If $y[p]=x[n]$ then return **0**;
    $p := (p+1) \mod P$; Endwhile ;
  $y[p] := x[n]$;
Endfor; return **1**

runtime $O(\mathbf{?})$
memory $O(P)$



2. Searching

order statistics/ median

Specification: Fix set X with total order $\leq$.

Input: $N\in\mathbb{N}$ and finite sequence $x_1,\dots x_N\in X$ — in array $x[1\dots N]$
as well as $K\in\{1,\dots,N\}$. — with $x_1\leq\dots\leq x_N$

Output: $m\in\{1,\dots N\}$ s.t. $\#\{n : x_n < x_m\} < K \leq \#\{n : x_n \leq x_m\}$

Primitives: Access $\mathbb{N}\ni n\to x[n]\in X$ cost 1
ordered comparison "$x\leq y$?" cost 1

OrderStat1 ($x[1..N]$; K) runtime $O(N^2)$
For m:=1 to N do
  a:=0; b:=0; For n:=1 to N do
    If $x[n] < x[m]$ then $a:=a+1$;
    If $x[n] \leq x[m]$ then $b:=b+1$;
  If $a < K \leq b$ then Return m;

OrderStat2 ($x[1..N]$; K)
Return K;
runtime $O(1)$

2. Searching

1D Range Counting

Specification: Fix set X with total order $\leq$.

Input: $N \in \mathbb{N}$ and finite sequence $x_1, \ldots x_N \in X$ as well as $y < z \in X$.

Output: $\#\{ m : y < x_m \leq z \}$

Primitives: Access $\mathbb{N} \ni n \to x[n] \in X$

ordered comparison "$x \leq y$?"

in array $x[1 \ldots N]$

with $x_1 \leq \ldots \leq x_N$

r=**BinSearch**($x[]$, y): smallest $r \leq N+1$ such that $y < x_r$

Range1 ($x[]$; y , z)
For m:=1 to N do
If $y < x[m] \leq z$
then output m;

runtime $O(N)$

Range2 ($x[]$; y , z)
l := BinSearch($x[]$; y)
r := BinSearch($x[]$; z)
If r-$l \leq 0$ then Return 0
else Return r-l;

runtime $O(\log N)$