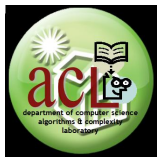
Sequential Tables

Lesson 11

Outline

- Table ADT
- Direct Access Tables
- Static Sequential Tables
  - Structure
  - Searching Records in a SST



Outline

- **Table ADT**
- Direct Access Tables
- Static Sequential Tables
  - Structure
  - Searching Records in a SST

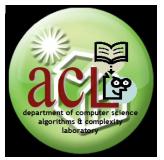


Table ADT

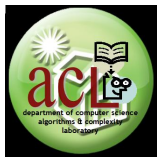
KEY	DATA
k_1	d_1
k_2	d_2
k_3	d_3
$\vdots$	$\vdots$
k_i	d_i
$\vdots$	$\vdots$
k_{n-1}	d_{n-1}
k_n	d_n

Some operations

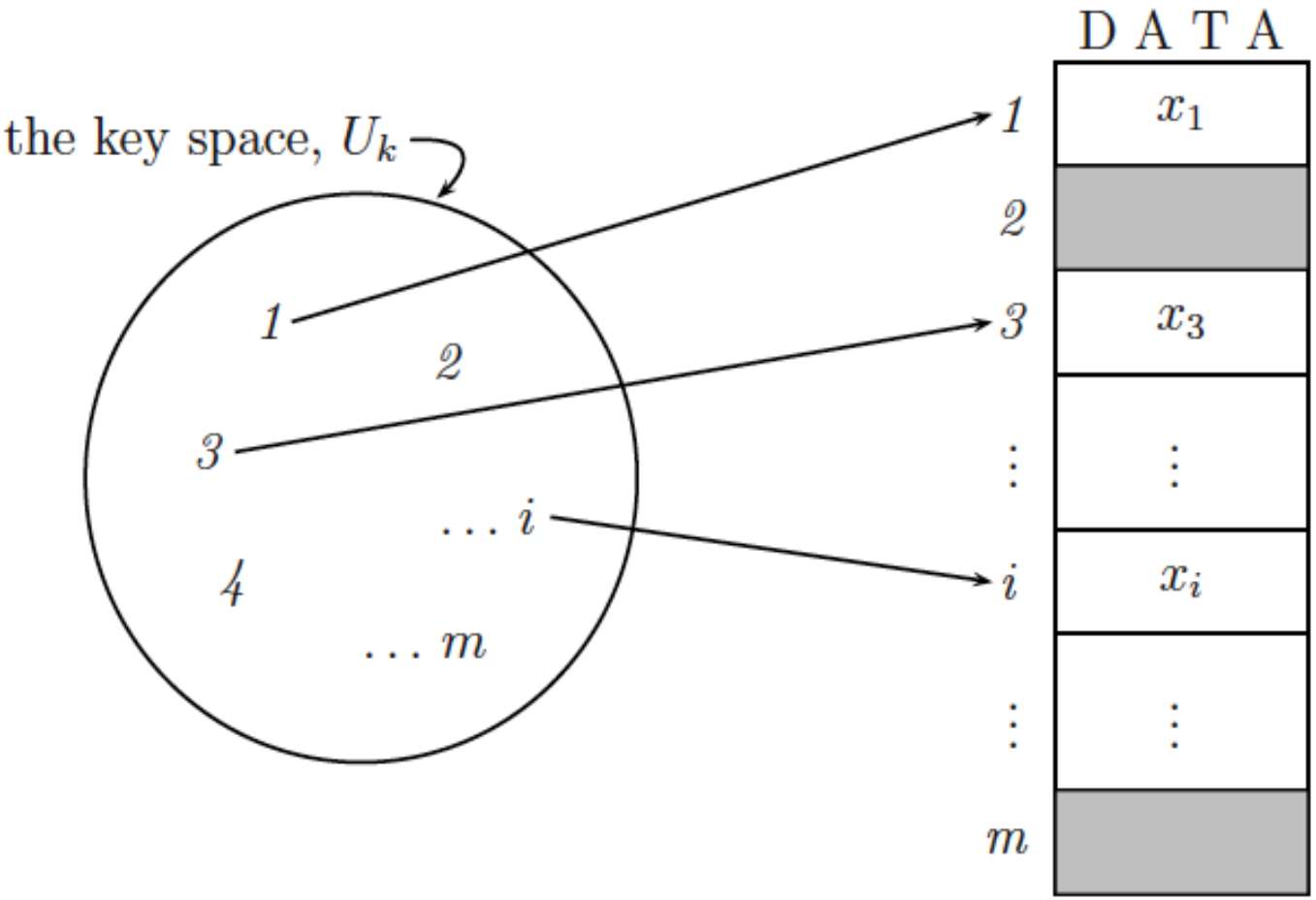
- Find a record (given search key K)
- Insert a new record (k, d) into the table
- Delete a record
- Find the record with smallest / biggest key
- Given a key k, find the record with the next smaller / bigger key

Outline

- Table ADT
- **Direct Access Tables**
- Static Sequential Tables
  - Structure
  - Searching Records in a SST

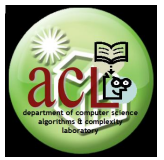


Direct Access Tables



Outline

- Table ADT
- Direct Access Tables
- **Static Sequential Tables**
  - **Structure**
  - **Searching Records in a SST**



Static Sequential Tables

	KEY	DATA
1	k_1	d_1
2	k_2	d_2
$\vdots$	$\vdots$	$\vdots$
i	k_i	d_i
$\vdots$	$\vdots$	$\vdots$
n	k_n	d_n
$\vdots$	$\vdots$	$\vdots$
m	dc	dc

Searching a Record – Example

i	KEY	DATA
1	Au	79
2	Ba	56
3	Cu	29
4	Fe	26
5	Ga	31
6	He	2
7	Hg	80
8	Kr	36
9	Mg	12
10	Ni	28

Searching Records in a SST

- ***Linear Search***
- ***Binary Search***
- ***Multiplicative Binary Search***
- ***Fibonaccian Search***

Linear Search

```
procedure LINEAR_SEARCH_1(𝕋, K)
for i ← 1 to n do
   if K = KEY(i) then return(i)      ▷ successful search
endfor
return(0)                            ▷ unsuccessful search
end LINEAR_SEARCH_1
```

```
procedure LINEAR_SEARCH_2(𝕋, K)
i ← 1
KEY(n + 1) ← K
loop
   if K = KEY(i) then exit
   i ← i + 1
forever
if i ≤ n then return(i)      ▷ successful search
         else return(0)      ▷ unsuccessful search
end LINEAR_SEARCH_2
```

Binary Search

```
procedure BINARY_SEARCH(𝕋, K)
l ← 1;   u ← n
while l ≤ u do
   i ← ⌊(l + u)/2⌋
   case
      :K = KEY(i): return(i)      ▷ successful search
      :K > KEY(i): l ← i + 1      ▷ discard upper half
      :K < KEY(i): u ← i − 1      ▷ discard lower half
   endcase
endwhile
return(0)                         ▷ unsuccessful search
end BINARY_SEARCH
```

Multiplicative Binary Search

Algorithm REARRANGE_FOR_MBS

1. Assign the keys $k_1 < k_2 < k_3 < \ldots < k_n$, to the nodes of a *complete* binary tree on n nodes in *inorder*.
2. Arrange the records in the table according to the sequence of the keys in the binary tree when enumerated in *level order*.

```
procedure MULTIPLICATIVE_BINARY_SEARCH(𝕋, K)
i ← 1
while i ≤ n do
   case
      :K = KEY(i): return(i)          ▷ successful search
      :K < KEY(i): i ← 2 * i          ▷ go left
      :K > KEY(i): i ← 2 * i + 1      ▷ go right
   endcase
endwhile
return(0)                             ▷ unsuccessful search
end BINARY_SEARCH
```

Fibonaccian Search

procedure FIBONACCIAN_SEARCH($\mathbb{T}, K$)
$i \leftarrow F_k$; $p \leftarrow F_{k-1}$; $q \leftarrow F_{k-2}$
$m \leftarrow F_{k+1} - 1 - n$; **if** $K > k_i$ **then** $i \leftarrow i - m$
loop
  case
    :$K = KEY(i)$: **return**(i)
    :$K < KEY(i)$: [**if** $q = 0$ **then return**(0)
      else [$i \leftarrow i - q$; $t \leftarrow p$; $p \leftarrow q$; $q \leftarrow t - q$]]
    :$K > KEY(i)$: [**if** $p = 1$ **then return**(0)
      else [$i \leftarrow i + q$; $p \leftarrow p - q$; $q \leftarrow q - p$]]
  endcase
forever
end FIBONACCIAN_SEARCH

Thank you
for your attention.

Questions ...