File access

2 standard methods for accessing data:

- sequential access
- random access: access via index or ID (key) for data element

Sequential access API

(API = Application Programming Interface: collection of methods).

Sequential access API

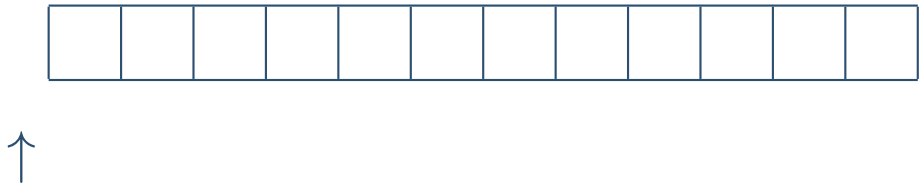
(API = Application Programming Interface: collection of methods).

Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()

Sequential access API

(API = Application Programming Interface: collection of methods).

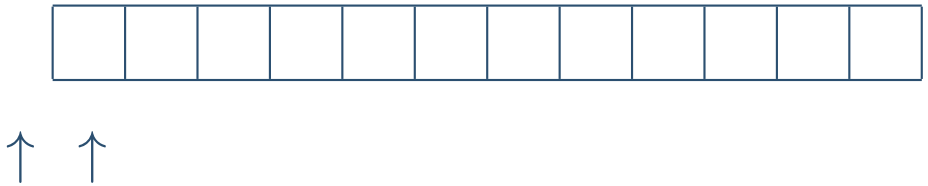
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

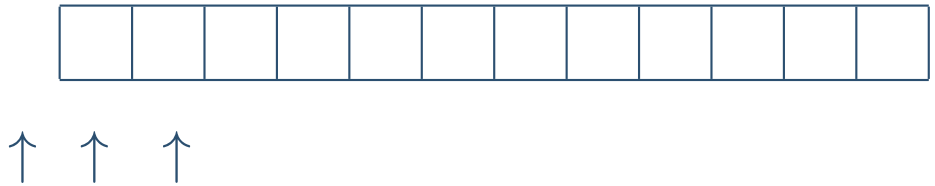
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

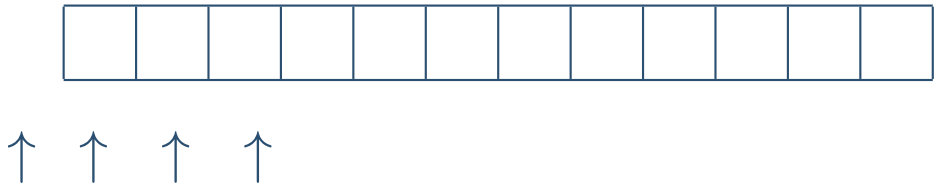
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

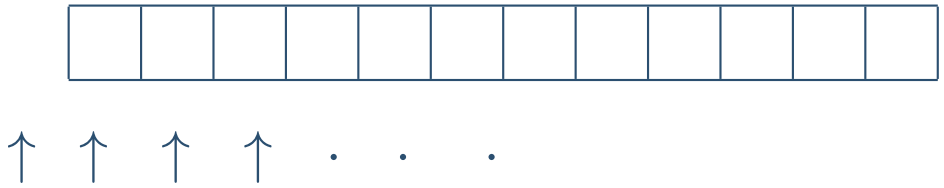
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

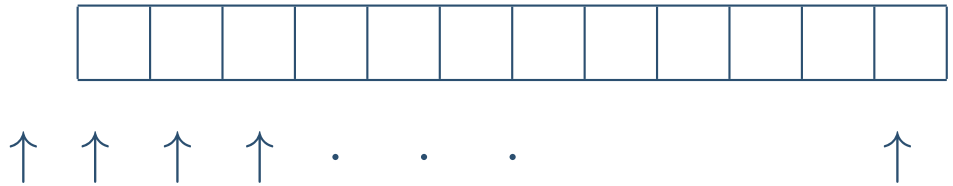
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

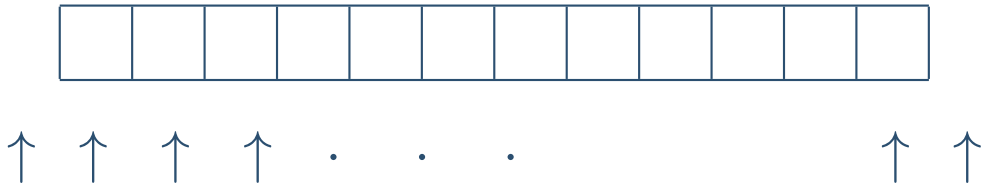
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

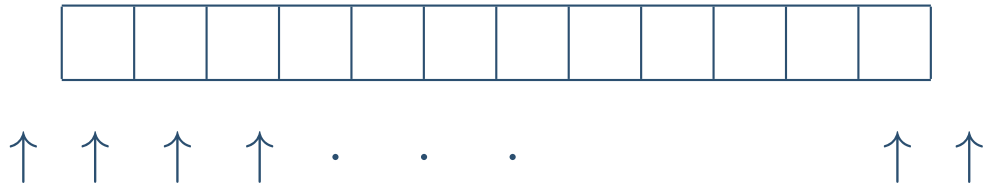
Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Sequential access API

(API = Application Programming Interface: collection of methods).

Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()

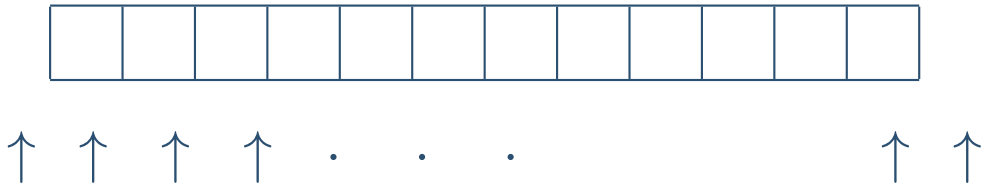


Writing:
Operations: `writeNext()`, open(), close()

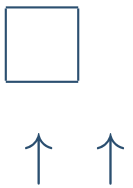
Sequential access API

(API = Application Programming Interface: collection of methods).

Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



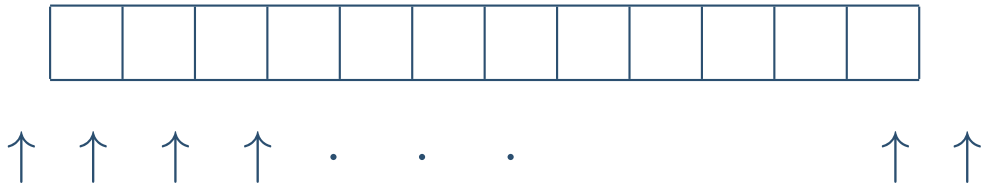
Writing:
Operations: `writeNext()`, open(), close()



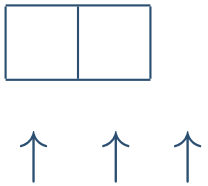
Sequential access API

(API = Application Programming Interface: collection of methods).

Reading:
Operations: `readNext()`, isEndOfFile(), open(), close()



Writing:
Operations: `writeNext()`, open(), close()



Random access API

random access: access via ID (key) for data element

Operations:
findElm(ID)
insertElm(ID,elementData)
deleteElm(ID)
open()
close()

Examples:

- dictionaries in Python
- arrays in Java — with ID = index in array

Questions

1. What can be done using only Sequential access?
2. How can one implement Random access?

Importance

Most data sources can be accessed by Sequential access.
Some can only be accessed sequentially.

- hard disk
- CD, DVD
- tape
- streaming (over the Internet)
- data generated on-the-fly, by another program
- data in an array

Sequential algorithms

What can be done using only Sequential access?

Sequential algorithms

What can be done using only Sequential access?

- Sequential search?
- Insertion Sort?
- Find maximum entry?
- Selection Sort?
- Find sum and average?

Find sum and average


```
procedure SumAverage(A)
    open(A)
    (s, n) ← SumAve(A, 0, 0)
    close(A)
    return(s, s/n)

procedure SumAve(A, s, n)
    if (isEndOfFile(A)) then
        return(s, n)
    else
        SumAve(A, s+readNext(A), n + 1)
```

Find sum and average

Invariants: s is sum of first n entries of A.
Next entry of A is the $n+1$st.

Therefore, SumAverage computes the correct result.

Fundamental operation: readNext —
done length(A) times, once in each recursive call, except nth.

Find sum and average


```
procedure SumAverage(A)
    open(A)
    (s, n) ← SumAve(A, 0, 0)
    close(A)
    if (n > 0) return(s, s/n)
    else report empty file
```

```
procedure SumAve(A, s, n)
    if (isEndOfFile(A)) then
        return(s, n)
    else
        SumAve(A, s+readNext(A), n + 1)
```

Sequential algorithms

What can be done using only Sequential access?

- Sequential search?
- Insertion Sort?
- Find maximum entry?
- Selection Sort?
- Find sum and average?
- Merging 2 sorted lists into 1?
- Mergesort?

Merging 2 lists

Input: 2 lists, A and B are both sorted.
Output: 1 sorted list, C, containing the entries of A and B.

Merging 2 lists

Input: 2 lists, A and B are both sorted.
Output: 1 sorted list, C, containing the entries of A and B.

Merge Step:

- Compare current records of A and B.
- Put smallest in C.
- Advance to next record in A or B, whichever had that smallest entry.

Merging 2 lists


```
procedure MergeFiles(A, B, C):
    open(A); open(B); open(C); fA,fB,fC ← false;
    if (isEndOfFile(A) and isEndOfFile(B)) then Stop with C empty
    if (not isEndOfFile(A)) then currentA ← readNext(A); fA← true;
    if (not isEndOfFile(B)) then currentB ← readNext(B); fB ← true;
    while (fA and fB) do
        if (currentA ≤ currentB) then
            writeNext(currentA,C)
            if (not isEndOfList(A)) then currentA ← readNext(A)
            else fA ← false
        else
            writeNext(currentB,C)
            if (not isEndOfList(B)) then currentB ← readNext(B)
            else fB ← false
    Starting with the current record in the input file which is not at EOF,
        copy the remaining records to C
    close(A); close(B); close(C)
```

Merging 2 arrays

procedure MergeArrays(A, B, C):
    $i, j, k \leftarrow 1$
    if (length(A)$= 0$ and length(B)$= 0$) **then** Stop with C empty
    if (length(A)> 0) **then** currentA $\leftarrow A[1]$
    if (length(B)> 0) **then** currentB $\leftarrow B[1]$
    while (length(A)$\geq i$ and length(B)$\geq j$) **do**
        { Merge Step() }
        if (currentA $\leq$ currentB) **then**
            $c[k] \leftarrow$ currentA; $k \leftarrow k + 1$; $i \leftarrow i + 1$;
            if (length(A)$\geq i$) **then** currentA $\leftarrow A[i]$
        else
            $c[k] \leftarrow$ currentB; $k \leftarrow k + 1$; $j \leftarrow j + 1$;
            if (length(B)$\geq j$) **then** currentB $\leftarrow B[j]$)
    Starting with the current record in the array which is not finished,
        copy the remaining records to C

Merge Sort

procedure MergeSort(A, f, l):
{ Input: Array A with first index f and last index l }
{ Output: Sorted array, A, with same entries as input A }

```
if (f < l) then
    m ← (f + l) div 2
    MergeSort(A, f, m)
    MergeSort(A, m + 1, l)
    MergeArrays(A[f..m], A[m + 1..l], C)
    Copy C to A
```

MergeSort(A, 1, length(A));

Analysis of Merge Sort

Let $T(n)$ be the maximum number of comparisons MergeSort uses if length(A)$= n$.

Let $M(m_A, m_B)$ be the maximum number of comparisons MergeArrays uses if length(A)$= m_A$ and length(B)$= m_B$.

$$T(n) \leq T(\lceil \tfrac{n}{2} \rceil) + T(\lfloor \tfrac{n}{2} \rfloor) + M(\lceil \tfrac{n}{2} \rceil, \lfloor \tfrac{n}{2} \rfloor)$$

Need to calculate $M(m_A, m_B)$.

Analysis of Merge Sort

As an aside, what is the minimum number of comparisons done by MergeArrays, given m_A and m_B?

A. $\min(m_A, m_B) - 1$

B. $\min(m_A, m_B)$

C. $\max(m_A, m_B) - 1$

D. $\max(m_A, m_B)$

E. $m_A + m_B - 1$

Vote at m.socrative.com. Room number 415439.

Analysis of Merge Sort

What is $M(m_A, m_B)$?

A. $m_A + m_B - 1$

B. $m_A + m_B$

C. $m_A + m_B + 1$

D. $m_A \cdot m_B$

E. $m_A \cdot m_B + 1$

Vote at m.socrative.com. Room number 415439.

Analysis of Merge Sort

$$\begin{aligned} T(n) &\leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + M\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right) \\ &\leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor - 1\right) \\ &\leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n - 1 \end{aligned}$$

$T(n) \in \Theta(n \log n)$.