Lec 10: Summations + Algorithms

Sequence: an ordered list.

\[ S = \{a_j\} \text{ where } j \in \mathbb{N} \text{ or } j \in \mathbb{Z}^+ \]

Typically want the underlying formula to generate the sequence.

i.e. arithmetic sequence \[ a_j = a + dj \]

for \( a, d \in \mathbb{R} \) given. (often \( a, d \in \mathbb{Z} \))

geometric sequence \[ a_j = ar^j \]

for \( a, r \in \mathbb{R}, j \in \mathbb{N}, \mathbb{Z}^+ \)

Note is \( r = 1, a_j = a \) for all \( j \)

\( r = 1 \), \( S = \{a, a, a, \ldots\} \)

\( r \neq 1 \), \( S = \{a, ar, ar^2, ar^3, \ldots\} \).

Summations

\[ a_0 + a_1 + a_2 + \cdots + a_n = \sum_{j=0}^{n} a_j \]

more generally

\[ a_m + a_{m+1} + \cdots + a_n = \sum_{j=m}^{n} a_j \]

Summation indices are flexible.
\[
\sum_{j=0}^{n} a_j = a_0 + \sum_{j=1}^{n} a_j = \sum_{j=0}^{n-1} a_j + a
\]

pull out first term

\[
= a_0 + \sum_{j=1}^{n-1} a_j + a_n, \text{ etc.}
\]

Recall important sums:

\[
S_{\text{Int}}(n) = \sum_{j=0}^{n} j = \sum_{j=1}^{n} j = \frac{n(n+1)}{2}
\]

\[
S_{\text{Geo}}(n) = \sum_{j=0}^{n} a \cdot r^j = \begin{cases} 
  a(n+1) & \text{if } r = 1 \\
  \frac{ar^{n+1} - a}{r-1} & \text{if } r \neq 1
\end{cases}
\]

(recall we calculated this by considering \( rS_{\text{Geo}}(n) \)).

What if instead of \( j=0 \) we start w/ \( j=1 \)?

\[
S_{\text{Geo}}(n) = \sum_{j=1}^{n} a \cdot r^j
\]

Approach 1: \( S_{\text{Geo}}(n) = S'_{\text{Geo}}(n) + a \)
Telescopic sum

\[ S = \sum_{j=1}^{N} (a_j - a_{j-1}) \]

\[ = (a_1 - a_0) + (a_2 - a_1) + (a_3 - a_2) + \ldots + (a_{n-1} - a_{n-2}) + (a_n - a_{n-1}) \]

\[ = -a_0 + a_n \]

Important series:

\[ e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{j=0}^{\infty} \frac{x^j}{j!} \]

(Recall 0! = 1).

Why \( e^x = 1 + x \) if \( |x| \ll 1 \). For \( x \in \mathbb{R} \),

\[ e^{ix} = \cos x + i \sin x \quad (\text{Recall } i^2 = -1) \]

Collect real terms:

\[ \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots \]

Collect imaginary terms

\[ \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \]
1) **Algorithm** a procedure for performing a computation. Takes an input and maps it to an output.

Every input (of the correct type) is mapped deterministically to one output.

(i.e. a function)

2) **Properties:**
   - Definite
   - Correct
   - Effective
   - Generalizable

3) *Fundamental constructs of an algorithm:*
   
   - *if-then statements* (the implication)
   - *while loops* (while \( p = T \))
   - *for loops* (for specified \( x \in U \))

4) **Simple algorithm.**

Find the maximum value in an unordered list

```
\begin{array}{c|c}
\hline
a_0 \quad & \quad a_n \\
\hline
\end{array}
```

Best case scenario: \( m \)

* Must search every element

# of operations grows linearly with \( n \).
5. Searching a list

Is it ordered? 
\{ No \rightarrow \text{linear search} \\
Yes \rightarrow \text{binary search} \}

Linear search.
locate an element $x$ in list $a$.

Start from the beginning & stop once I find an item $x$ in list $a$ (or else exit the list).

Is $a_j = x$ for any $j$?

How long does linear search take?

Best case: $x = a_1$ (one "while" loop execution)
Worst case: $x$ is not in list. (n "while" loops)
Average case: $x = a_{n/2}$ (search half the list) (n/2 "while" loops).

(worst case grows linearly $\propto n$

Owe case grows linearly $\propto n$).
If the list is ordered, we can do much better.

Binary search: (list where elements are ordered from smallest to biggest)

look at middle item, am

\[ m = \left\lfloor \frac{n+1}{2} \right\rfloor \]

Is \( x > am? \)
- yes: then any \( a_j = x \) will need \( j > m \).
- no: then any \( a_j = x \) will need \( j \leq m \).

Recurse if for instance \( x > am \)

left with \( a_{m+1} \) to \( a_n \)

If \( x > am' \), where \( am' \) is the new middle item?
- if yes then any \( a_j = x \) will need \( j > m' \)
- if no \( " " " " " " " " " j \leq m' \)
Each iteration reduces the length of the list in half.

For a list of length $2^k$, max of $k$ operations.