Algorithms

Algorithm: what is it?
Algorithm: what is it?

Some representative problems:
- Interval Scheduling

Diagram:
- Math 10am-12pm
- Karate 1pm
- Physics
- Dance 11:30am-6pm
Algorithms

Algorithm: what is it?

Some representative problems:
- Interval Scheduling
- Bipartite Matching
Algorithms

Algorithm: what is it?

Some representative problems:
- Interval Scheduling
- Bipartite Matching
- Independent Set

(largest) a set of vertices with no edges between them
Algorithms

Algorithm: what is it?

Some representative problems:
- Interval Scheduling
- Bipartite Matching
- Independent Set
- Area of a Polygon
Algorithms

How to decide which algorithm is better?

Search problem

Input: a sequence of n numbers (in an array A) and a number x
Output: YES, if A contains x, NO otherwise

- linear search
  - about n steps
  - O(n)
Algorithms

How to decide which algorithm is better?

Search problem

Input: a sequence of n numbers (in an array A) and a number x
Output: YES, if A contains x, NO otherwise

What if A is already sorted?

\[
A = 1 5 7 9 10 15 16 17 30
\]

\[
x = 17 \quad \text{binary search} \quad O(\log n)
\]
Running Time

O(n) - running time of the linear search
O(log n) - running time of the binary search

Def: **Big-Oh (asymptotic upper bound)**

\[ f(n) = O(g(n)) \] if there exists a constant \( c > 0 \) and a constant \( n_0 \) such that for every \( n \geq n_0 \) we have \( f(n) \leq c g(n) \)

Examples:

\[ n, \ n^3, \ \log n, \ 2^n, \ 7n^2 + n^3/3, \ 1, \ 1 + \log n, \ n \log n, \ n + \log n \]
Running Time

Def: **Big-Oh (asymptotic upper bound)**

\[ f(n) = O(g(n)) \] if there exists a constant \( c > 0 \) and a constant \( n_0 \) such that for every \( n \geq n_0 \) we have \( f(n) \leq c g(n) \)

Example: Prove that \( n = O(n^3) \)

Take \( c = 1 \)

\( n_0 = 1 \)
**Running Time**

**Def**: **Big-Oh** (asymptotic upper bound)

\[ f(n) = O(g(n)) \] if there exists a constant \( c > 0 \) and a constant \( n_0 \) such that for every \( n \geq n_0 \) we have \( f(n) \leq c \cdot g(n) \)

**Example**: Prove that \( n^3 = O(7n^2 + n^3/3) \)

1. Let \( c = 3 \), \( n_0 = 0 \)

   \[ n^3 \leq 3 \left( 7n^2 + \frac{n^3}{3} \right) = 21n^2 + n^3 \]

2. Is this true?

   \[ n^3 = O\left( \frac{n^3}{3} - 7n^2 \right) \]

   Let \( c = 6 \)

   \[ n^3 \leq 6 \left( \frac{n^3}{3} - 7n^2 \right) = 2n^3 - 42n^2 \]

   Want to find \( n_0 \) s.t.

   \[ \forall n \geq n_0 : \]

   \[ n^3 \leq 2n^3 - 42n^2 \]

   \[ 42n^2 \leq n^3 \]

   \[ 42 \leq n \]

   Assuming \( n \geq 0 \)

   Take \( n_0 = 42 \).
Running Time

Def: Big-Oh (asymptotic upper bound)

\[ f(n) = O(g(n)) \text{ if there exists a constant } c > 0 \text{ and a constant } n_0 \text{ such that for every } n \geq n_0 \text{ we have } f(n) \leq c g(n) \]

Example: Prove that \( \log_{10} n = O(\log n) \)

\[ \text{done on the board} \]
Running Time

Def: **Big-Oh** (asymptotic upper bound)

\[ f(n) = O(g(n)) \text{ if there exists a constant } c > 0 \text{ and a constant } n_0 \text{ such that for every } n \geq n_0 \text{ we have } f(n) \leq c g(n) \]

Example: what about \(3^n\) and \(2^n\)

*done on the board*
Running Time

\( O(n) \) - running time of the linear search

\( O(\log n) \) - running time of the binary search

Def: **Big-Omega (asymptotic lower bound)**

\( f(n) = \Omega(g(n)) \) if there exists a constant \( c > 0 \) and a constant \( n_0 \) such that for every \( n \geq n_0 \) we have \( f(n) \geq c \cdot g(n) \)

Examples:

\( n, \ n^3, \ \log n, \ 2^n, \ 7n^2 + n^3/3, \ 1, \ 1 + \log n, \ n \log n, \ n + \log n \)
Running Time

$O(n)$ - running time of the linear search

$O(\log n)$ - running time of the binary search

Def: **Theta (asymptotically tight bound)**

$f(n) = \Theta(g(n))$ if there exists constants $c_1, c_2 > 0$ and a constant $n_0$ such that for every $n \geq n_0$ we have $c_1 g(n) \leq f(n) \leq c_2 g(n)$

Examples:

$n$, $n^3$, $\log n$, $2^n$, $7n^2 + n^3/3$, $1$, $1 + \log n$, $n \log n$, $n + \log n$
A survey of common running times

**Linear**

1. for i=1 to n do
2. something

**Also linear:**

1. for i=1 to n do
2. something
3. for i=1 to n do
4. something else
A survey of common running times

Example (linear time):

Given is a point $A=(a_x, a_y)$ and $n$ points $(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)$ specifying a polygon. Decide if $A$ lies inside or outside the polygon.

\begin{itemize}
  \item [\textbf{idea:}]
    \begin{align*}
      \text{counter} &= 0 \\
      \text{for } i = 1 \text{ to } n \text{ do} \\
      &\quad \text{if } (x_i,y_i) - (x_0,y_0) \text{ intersects with the half-line:} \\
      &\quad \quad \text{counter} + \\
      &\quad \text{if counter odd return inside} \\
      &\quad \text{else return outside}
    \end{align*}
\end{itemize}

$O(n)$
A survey of common running times

Example (linear time):

Given are $n$ points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ specifying a polygon. Compute the area of the polygon.

Idea 1:
- triangulate the polygon and add the areas of the triangles

Running time:
- more than linear, e.g. can be easily in $O(n^2)$ steps
A survey of common running times

Example (linear time):

Given are \( n \) points \((x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\) specifying a polygon. Compute the area of the polygon.

\[\text{pink} = \text{add} \]
\[\text{while} = \text{subtract}\]
A survey of common running times

$O(n \log n)$

1. for $i=1$ to $n$ do
2. for $j=1$ to $\log(n)$ do
3. something

Or:

1. for $i=1$ to $n$ do
2. $j=n$
3. while $j>1$ do
4. something
5. $j = j/2$
A survey of common running times

**Quadratic**

1. for i=1 to n do
2. for j=1 to n do
3. something
A survey of common running times

Cubic
A survey of common running times

\( O(n^k) \) - polynomial (if \( k \) is a constant)
A survey of common running times

Exponential, e.g., $O(2^k)$