Algorithms

Algorithm: what is it?
Algorithms

Algorithm: what is it?

Some representative problems:
- Interval Scheduling
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
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

\[ A = \begin{array}{ccccccc}
5 & 7 & 3 & 2 & 6 & 1 & 107 \\
\end{array} \]

\[ x = 107 \]

O(n) steps
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?

$A = \begin{pmatrix} 1 & 7 & 8 & 13 & 20 & 23 & 25 & 30 & 102 \end{pmatrix}$

$x = 107$

$O(\log n)$

We mean $\log_2 n$
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 $f(n) \leq c \cdot g(n)$

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

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

\[ \frac{1 + \log n}{f(n)} = O\left(\frac{\log n}{g(n)}\right) \]

\[ f(n) = O(g(n)) \text{ if } \exists c > 0 \text{ and } n_0 \text{ s.t. } \forall n \geq n_0 : f(n) \leq c \cdot g(n) \]

\[ f(n) = 1 + \log n \quad \leq \quad c \cdot g(n) = c \cdot \log n \]

Take \( c = 2 \):

\[ 1 + \log n \quad \leq \quad 2 \cdot \log n \]

\[ 1 \leq \log n \quad \checkmark \]
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 \( n = O(n^3) \)

\[
\begin{align*}
  n &\leq c \cdot n^3 & c=1 \\
  n &\leq n^3 & \checkmark \\
  1 &\leq n^2 & \forall n \geq 1 & n_0=1
\end{align*}
\]
**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 \( n^3 = O(7n^2 + n^3/3) \)

We have to find \( c, n_0 \):

\[ n^3 \leq c \cdot (7n^2 + \frac{n^3}{3}) \quad \forall n \geq n_0 \]

Take \( c = 3 \)

\[ n^3 \leq 3 \cdot (7n^2 + \frac{n^3}{3}) = 21n^2 + n^3 \quad \forall n \geq 0 \]
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 \( n^3 = O(n^3/3 - 7n^2) \)

\[
\begin{align*}
n^3 & \leq c \left( \frac{n^3}{3} - 7n^2 \right) \\
c & = 3 \quad \text{?} \quad \text{does not work} \\
\text{take } c & = 30 \\
n^3 & \leq 30 \left( \frac{n^3}{3} - 7n^2 \right) = 10n^3 - 210n^2 \\
210n^2 & \leq 9n^3 \\
210 & \leq 9n \\
\Rightarrow & \quad \frac{210}{9} \leq n \\
& \quad n = \frac{70}{3}
\end{align*}
\]
Running Time

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

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

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

And that \( \log n = O(\log_{10} n) \)

\[
\log_{10} n \leq c \cdot \log n \leq \frac{\log n}{\log 10} \leq c \cdot \log n \quad \text{take } c = \frac{1}{\log 10}
\]

\[
\log(n^2) = O(\log n) \quad \log^2 n = (\log n)^2 \leq O(\log n)
\]
Running Time

Def: Big-Oh (asymptotic upper bound)

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

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

\[ \log^2 n \neq O(\log n) \quad \text{by const} \]

\[ \log n = O(\log^2 n) \]

we need to find \( c \) s.t. \( \ldots \) (e.g. \( c = 1 \) works)

\[ \log n \leq c \cdot \log^2 n \]

\[ 1 \leq c \cdot \log n \]

since \( n_0 = 2 \) then \( \log n \geq 1 \), therefore \( c = 1 \) works \( \checkmark \)
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: what about \( 3^n \) and \( 2^n \)

\[
2^n = O(3^n) \\
\text{need to find } c, n_0: \\
2^n \leq c \cdot 3^n \quad \forall n \geq n_0 \\
c = 1 \quad n_0 = 1
\]

\[
3^n \not\in O(2^n) \\
\text{suppose, by contradiction, } \exists c, n_0: \\
3^n \leq c \cdot 2^n \quad \forall n \geq n_0 \\
\text{take log (increasing in)}
\]
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 $f(n) \geq c g(n)$

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 $O(n)$

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 $A=\left(a_x, a_y\right)$ 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.

Idea:
1. Consider a line from $A$ to an outside point.
2. Count the number of intersections of the polygon with the line. If even, $A$ is outside; if odd, $A$ is inside.

Special cases:
1. Intersection through a vertex of the polygon but both sides on the same side of the line:
   - Don't count as an intersection.
2. Different sides:
   - Count as intersection.
3. Same side:
   - Don't count as intersection.
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.
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

The pseudo-code makes
$c \cdot \frac{n(n+1)}{2}$ steps
$= c \cdot \frac{n^2}{2} + cn$
$= \Theta(n^2)$

inside j loop: c constant steps
A survey of common running times

Cubic
A survey of common running times

\( O(n^k) \) - polynomial (if \( k \) is a constant)

\( k \) nested for loops (\( n \) iterations each)

NP complete if the set is unbounded

e.g. if looking for an independent set of size \( 7 \):

check all sets of 7 vertices

\( \Rightarrow \) can be done w. 7 nested for loops
A survey of common running times

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

\textit{e.g.} try all sets to find the largest independent set