Algorithms

Algorithm: what is it?
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
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** → in the worst case need 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?

\[ A = \begin{array}{c} 5 \ 6 \ 8 \ 9 \ 11 \ 15 \ 17 \ 20 \ 35 \ 100 \end{array} \]

\[ x = 35 \]

binary search \[ O(\log n) \]
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 \( f(n) \leq c \cdot g(n) \)

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

need to find \( c, n_0 \) s.t.

\[ \forall n \geq n_0: \quad n \leq c \cdot n^3 \]

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 \( n^3 = O(7n^2 + n^3/3) \)

need to find \( c, n_0 \) s.t.

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

then,

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

i.e. \( n^3 \leq 21n^2 + n^3 \) ? \( \text{Yes} \)

\( n_0 = 1 \)
Def: **Big-Oh** (asymptotic upper 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 \( f(n) \leq c \cdot g(n) \)

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

need to find \( c, n_0 \) s.t.

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

take \( c = 6 \):

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

when \( n^3 \leq 2n^3 - 42n^2 \) ?

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

\[ 42 \leq n \quad \text{for} \quad n > 0 \]

\[ \text{take} \quad n_0 = 42 \]
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 $\log_{10} n = O(\log n)$

$\log_{10} n \leq \log_2 n \quad \forall n \geq 1$

How about $\log_2 n = O(\log_{10} n)$?

TRUE! we need to find $c, n_0$ s.t.

$\log_2 n \leq c \cdot \log_{10} n \quad \forall n \geq n_0$

$\log_{10} n = \frac{\log_2 n}{\log_2 10}$

take $c = \log_{10} 2 \quad n_0 = 1$
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: what about \( 3^n \) and \( 2^n \)

\[ 3^n = O(2^n) \text{ is } 2^n = O(3^n) \]

YES, take \( c = 1, n_0 = 1 \)

\[ \frac{n \log_2 3}{0} \leq \log_2 c + n \]

\( n \log_2 3 \leq \log_2 c + n \)
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:** \textbf{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 \(c_1 g(n) \leq f(n) \leq c_2 g(n)\)

\[
\log_{10} = \Theta(\log_{10} n) \\
n^3 = \Theta(7n^2 + \frac{n^3}{3})
\]

**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 \( A=(a_x, a_y) \) 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.

counter = 0
for i = 1 to n do :
    if \((x_i,y_i)-(x_{i-1},y_{i-1})\) intersects with the half-line from \( A \):
        counter++
if counter is odd return inside
else return outside

careful with special case:

run.time: \(O(n)\)
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.
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., $O(2^k)$