Algorithms

Algorithm: what is it?
Algorithms

Algorithm: what is it?

Some representative problems:
- Interval Scheduling
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

\[
\text{for } i = 1 \text{ to } n:\n\quad \text{if } A[i] = x \text{ then return YES}
\]

return NO
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?

  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 \( f(n) \leq c \cdot g(n) \)

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

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 \( n = O(n^3) \)

- We need to find constants \( c, n_0 \) such that for every \( n \geq n_0 \), \( f(n) \leq c \cdot g(n) \).
- Let's take \( c = 1, n_0 = 1 \) such that \( n \leq n^3 \) for all \( n \geq 1 \) and \( n \leq c \cdot n^3 \) holds, therefore \( \square \).
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. \( \forall n \geq n_0 : \quad n^3 \leq c \cdot (7n^2 + \frac{n^3}{3}) \)

let’s take \( c=3, \ n_0=1 : \quad n^3 \leq 3 \cdot (7n^2 + \frac{n^3}{3}) \)

\[ n^3 \leq 21n^2 + n^3 \]

\[ 0 \leq 21n^2 \quad \checkmark \quad \text{always true} \]

\( \square \)
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 \( n^3 = O(n^3/3 - 7n^2) \)

Find \( c, n_0 \) s.t. \( n^3 \leq c \cdot (\frac{n^3}{3} - 7n^2) \)

Let's take \( c = 9 \): we need to show that \( n^3 \leq \frac{4}{3} n^3 - 28n^2 \)

\[ 28n^2 \leq \frac{n^3}{3} \]
\[ 28 \leq \frac{n}{3} \]
\[ 84 \leq n \]

\( \text{take } n = 84 \) done
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 \( \log_{10} n = O(\log n) \)

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

- Find \( c, n_0 \) s.t. \( \log_{10} n \leq c \cdot \log n \)
- Need to show: \( \frac{\log n}{\log_{10} n} \leq c \cdot \log n \)
- Let's take \( c = 1 \), \( n_0 = 2 \) then \( \log n \geq 1 \) \( \forall n \geq n_0 \).
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: what about \( 3^n \) and \( 2^n \)

- \( 2^n = O(3^n) \) holds trivially
- \( 3^n \neq O(2^n) \) is not true because exponential function of a base > 1 goes to oo
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:**

Order these s.t. first = \(\Omega\) (second), second = \(\Omega\) (third), etc.

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

Reverse of earlier:

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

we already showed: $n^3 = O(7n^2 + \frac{n^3}{3})$

we can show $n^3 = \Omega(7n^2 + \frac{n^3}{3}) \iff 7n^2 + \frac{n^3}{3} = O(n^3)$
A survey of common running times

Linear $O(n)$ running time

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): 0(n) running 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.

1. Let \( l \) be a halfline starting at \( A \)
2. \( \text{count} = 0 \)
3. For \( i = 1 \) to \( n \):
   - if \( i \)-th polygon side crosses \( l \):
     - \( \text{count}++ \)
     - (\#) for special cases
4. If \( \text{count} \) even return OUTSIDE
5. Else return INSIDE

Note: problem if passing through a point/vertex of the polygon!
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.

\[
\text{area} = 0; \quad x_{-1} = x_1, y_{-1} = y_1 \\
\text{for } i = 1 \text{ to } n: \quad \text{area} += \frac{(x_{i+1} - x_i)(y_i + y_{i+1})}{2} \\
\text{return } \text{area}; \quad \mathcal{O}(n)
\]
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

\[
\begin{align*}
&\text{for } i=1 \text{ to } n \text{ do} \\
&\quad \text{for } j=1 \text{ to } n \text{ do} \\
&\quad \quad \text{something} \\
&\quad \text{for } i=1 \text{ to } n \text{ do} \\
&\quad \quad \text{for } j=1 \text{ to } n \text{ do} \\
&\quad \quad \quad \text{something} \\
\end{align*}
\]

- constant time

\[
O(n^2)
\]

\[
O(n)
\]

\[
0(1)
\]
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., $O(2^k)$