Algorithms

Algorithm: what is it?

Example problem: Stable matchings

Goal of algorithm design:
- proof of correctness
- bound on the # steps as a function of the input size (running time)

if A prefers 4 to 7 then A, 7 would forget their partners
⇒ matching is not stable
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
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?
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:
- \( n \)
- \( n^3 \)
- \( \log n \)
- \( 2^n \)
- \( 7n^2 + n^3 / 3 \)
- \( 1 \)
- \( 1 + \log n \)
- \( n \log n \)
- \( n + \log n \)

\( n^4 = O(n^3) \) need to find \( c, n_0 \) s.t.

\[
\begin{align*}
n^2 &\leq c \cdot n^3 \quad \forall n \geq n_0 \\
take \ c = 1, \ n_0 = 1 \\
n^2 \leq n^3 &\quad \forall n \geq 1
\end{align*}
\]

\( \log n = O(1+\log n) \) take \( c = 1, \ n_0 = 1 \) s.t.

\[
\log n \leq c \cdot (1 + \log n) \quad \forall n \geq n_0
\]

\( 1 + \log n \leq c \cdot 1 + \log n \) s.t.

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

\[
1 \leq \log n \quad \forall n \geq 2
\]

Order from asymptotically smallest to largest:

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

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

\[ \frac{7n^2 + \frac{n^3}{3}}{c} \leq n^3 \]

- Take \( c = 3 \), \( n_0 = 1 \)
- Then \( n^3 \leq c \cdot (7n^2 + \frac{n^3}{3}) = 21n^2 + n^3 \)

We claim: \( 7n^2 + \frac{n^3}{3} \leq \frac{22}{3} n^3 \)

\[ \forall n \geq 0 \text{ so we can take } n_0 = 1 \]

Option Andrew: \( c = \frac{22}{3} \)

Option Alex: \( c = 1 \), \( n_0 = 0.5 \)
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 \( n^3 = O(n^{3/3} - 7n^2) \)

we want to find \( c, n_0 \) s.t.

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

take \( c = 9 \) (anything bigger than 3 will work)

we need to show:

\[ n^3 \leq 9 \left( \frac{n^3}{3} - 7n^2 \right) = 3n^3 - 63n^2 \]

\[ 63n^2 \leq 2n^3 \]

\[ \frac{63}{2} \leq n \]

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

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

\[ \log_{10} n = \frac{\log_2 n}{\log_2 10} \]

\[ \log n = O(\log_{10} n) \]

Need to find \( c, n_0 \) such that \( \log n \leq c \cdot \log_{10} n = c \cdot \frac{\log n}{\log 10} \)

Take \( c = \log 10 \)
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 \( 2^n \)

\[ 3^n \neq O(2^n) \]

Proof by contradiction:

Suppose \( 3^n = O(2^n) \) then there is \( c, n_0 \) s.t.

\[ 3^n \leq c \cdot 2^n \quad \forall n \geq n_0 \]

\[ \left( \frac{3}{2} \right)^n \leq c \]

Contradiction

Therefore \( 3^n \neq O(2^n) \)

\[ 2^n = O(3^n) \]

Take \( c = 1 \)

\[ 2^n \leq c \cdot 3^n = 3^n \quad \text{\checkmark DONE} \]
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:

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

we get the same ordering as before but backwards (each function should be $\Omega$ of the next function in the ordering)
Running Time

O(n) - running time of the linear search

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

\[ f(n) = o(g(n)) \quad \text{and} \quad g(n) = O(f(n)) \]

(he same as \( f(n) = \Omega(g(n)) \))

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:

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

Linear

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

\[ \text{constant # steps} \leq c \cdot n \text{ steps} \leq O(n) \]

Also linear:

1. for i=1 to n do
2. something
3. for i=1 to n do
4. something else

\[ \begin{cases} O(n) \\ O(n) + O(n) = O(n) \end{cases} \]
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.

Pseudo code:

1. $counter = 0$
2. generate random line out of $A$
3. for $i=1$ to $n$:
   4. if line crosses $x_i,y_i - x_i+1,y_i+1$ then
   5. $counter++$
6. if $counter$ even return OUT
7. return IN

Note: pseudo code does not deal w. special cases
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.

Pseudo code:

1. \( \text{area} = 0 \)
2. \( \text{for } i = 1 \text{ to } n \text{ do:} \)
3. \( \quad \text{area} += (x_{i+1} - x_i) \cdot \frac{y_i + y_{i+1}}{2} \)
4. \( \text{return area} \)

\( O(n) \)
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

4. for i=1 to n
5. something

\[ O(n^2) + o(n) = O(n^2) \]
A survey of common running times

Example (CONVEX HULL):

Given are $n$ points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ in the plane. Find their convex hull, i.e. smallest convex polygon containing all points.

Running time:

$O(n^2)$ worst case

more precisely,

if the points on the convex hull is $k$, then run. time $O(kn)$
A survey of common running times

Example (CONVEX HULL):

Given are \( n \) points \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\) in the plane. Find their convex hull, i.e. smallest convex polygon containing all points.

1. Sort the points by angle w. respect to a center pt (e.g. average of coords) [polar sorting]
2. Start w. pch (partial convex hull) consisting of the leftmost + next pt
3. Go through the remaining pts (in sorted order)
4. Add the current pt to pch
5. \( \text{last} \times \text{previous} \times \text{previous} \) pt form leftmost angle then
6. Remove previous from pch
7. Return pch

Running time:
- step 1: \( O(n \log n) \)
- step 2: \( O(n) \)
- steps 3-6: looks like \( O(n) \) (correct upperbound)

Actually:
- steps 3-6: every pt is added once and removed at most once

Overall:
\[ O(n \log n) + O(n) + O(n) = O(n \log n) \]
A survey of common running times

Cubic

\[ \text{for } i=1 \text{ to } n \text{ do} \]
\[ \quad \text{for } j=1 \text{ to } n \text{ do} \]
\[ \quad \quad \text{for } k=1 \text{ to } n \text{ do} \]
\[ \quad \quad \quad \text{something} \]
\[ \} \quad O(n^2) \]

\[ \text{for } i=1 \text{ to } \sqrt{n} \text{ do} \]
\[ \quad \text{for } j=1 \text{ to } \sqrt{n} \text{ do} \]
\[ \quad \quad \text{for } k=1 \text{ to } \sqrt{n} \text{ do} \]
\[ \quad \quad \quad \text{something} \]
\[ \quad \quad \quad \text{not cubic} \]
\[ \quad \quad \quad \text{(though it is } O(n^2) \text{, not a tight bound)} \]
\[ \quad \quad \quad \text{not a tight bound} \]

\[ O(\sqrt{n}\sqrt{n}\sqrt{n}) = O(n^{3/2}) \]
A survey of common running times

$O(n^k)$ – polynomial (if $k$ is a constant)

$k$ nested loops ($n$ iterations each)
A survey of common running times

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