Algorithms

Algorithm: what is it?

Stable matchings example problem

Two things to convince ourselves about:
- correctness
- bound the #steps (running time)
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
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 $f(n) \leq c g(n)$

Examples:

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

$1, \log n, 1+\log n, n, n+\log n, n\log n, 7n^2+\frac{n^3}{3}, n^3$

$n^2 = O(n^3)$

$c = 1$

$n_0 = 0$

$f(n) \leq g(n)$

Order from asymptotically smallest to largest:
**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 \( f(n) \leq c \cdot g(n) \)

**Examples :**
- \( n \)
- \( n^3 \)
- \( \log n \)
- \( 2^n \)
- \( 7n^2 + \frac{n^3}{3} \)
- \( 1 \)
- \( 1 + \log n \)
- \( n \log n \)
- \( n + \log n \)

To show \( \log n = O(1 + \log n) \), we need to find constants \( c, n_0 \)

\[ f(n) \leq c \cdot g(n) \quad \forall n \geq n_0 \quad c=1, \quad n_0=1 \]

\[ \log n \leq 1 + \log n \quad \checkmark \quad \text{done} \]
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 \( n = O(n^3) \)

\[
\begin{align*}
7n^2 + \frac{n^3}{3} & = \left(7n^2 + \frac{n^3}{3}\right) \\
& \leq 1 \cdot n^3 \\
& \leq \frac{2}{3} n^3 \\
& \leq \frac{2}{3} n \\
& \leq n \\
\text{take } n_0 = 11
\end{align*}
\]
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 \( n^3 = O(7n^2 + n^3/3) \)
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 \( n^3 = O(n^3/3-7n^2) \)

\[
\begin{align*}
\text{need to find} & \quad c, n_0 \\
\text{c= 4} & \quad \text{need to show:} \\
n^3 & \leq 4 \cdot \left( \frac{n^3}{3} - 7n^2 \right) = \frac{4}{3} n^3 - 28n^2 \\
\downarrow & \\
28n^2 & \leq \frac{1}{3} n^3 \\
28 \cdot \frac{3}{7} & \leq n \\
\text{take this to be} n_0 & \\
\text{then } \forall n \geq n_0: \quad n^3 & \leq 4 \cdot \left( \frac{n^3}{3} - 7n^2 \right)
\end{align*}
\]
Running Time

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

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

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

change of base formula: \[ \log_a n = \frac{\log_b n}{\log_b a} \]

need to find \( c, n_0 \): want to show: \( \log n \leq c \cdot \log_{10} n \)

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

take \( c = \log_2 10 \) \( \checkmark \)
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 \(2^n\)

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

\[ \text{is } 3^n = O(2^n) \text{? No!} \]

Proof by contradiction: suppose \(3^n = O(2^n)\) then we have \(c, n_0\) constants s.t.

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

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

Since \(\frac{3}{2} > 1\), exponential \(\left(\frac{3}{2}\right)^n \to \infty\) as \(n \to \infty\)

Cannot be \(\leq\) constant
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:

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

Same ordering as before but backwards

(each function should be \Omega (next function in the ordering))
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

\( \text{takes constant steps} \)
\( \implies \text{cn steps} \)
\( O(n) \) steps

Also linear:

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

\( O(n) \) + \( O(n) \) = \( O(n) \)
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. \( \text{count} = 0 \)
2. \( \text{for } i = 1 \text{ to } n : \)
3. \( \text{if } x_i y_i \leq x_{i+1} y_{i+1} \) \( \text{intersect } p \) then \( \text{count}++ \)
4. \( \text{return IN} \)
5. \( \text{if } \text{count} \text{ even return } \text{OUT} \)

Idea: \( \text{call} \) draw a line from \( A \) to \( \infty \)
\( \text{count } \# \text{ crossings with the line} \)
\( \text{if even } \Rightarrow A \text{ is out} \)
\( \text{else } A \text{ is in} \)

Careful: \( \text{special cases when line goes through a point or an edge of the polygon} \)

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.

Pseudo code:

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

\( O(n) \)
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. \hspace{1em} for \( j = 1 \) to \( n \) do
3. \hspace{2em} something

\[
\{ \begin{align*}
0(n^2) \\
0(n) + 0(n) &= O(n^2)
\end{align*} \]
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.

Idea:

1) start w. leftmost point and a pretend edge going up
2) compute segments from last point to each other point
3) find segment w. smallest angle w. last CH edge (pretend)
4) this segment will be next edge of the CH
5) repeat until get back to start (leftmost)
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 point (polar $\rightarrow$ according to angles w.r.t. a central pt)
2. Start left-most & the next pt (pts 1&2)
3. Go clockwise through the points
4. w. next pt:
   - add to the temp conv hull
5. While the angle w. the last pt is left:
   - last pt
6. Remove B from tch (the pt before last)
7. return tch
A survey of common running times

Cubic

```plaintext
for i = 1 to \( n \) do
  for j = 1 to \( n \) do
    for k = 1 to \( n \) something
    \( O(n^3) \)
  this is not really cubic (though it is \( O(n^3) \))
  not a tight bound
  this is between \( O(n\log n) \) and \( O(n^2) \)
  \( \text{best in here} \)
  \( \text{in here (cut out right)} \)
What about:

for i = 1 to \( \sqrt{n} \) do:
  for j = 1 to \( \sqrt{n} \) do:
    for k = 1 to \( \sqrt{n} \) do:
       something
\( \sqrt{n} \) possible for i
\( \sqrt{n} \) possible for j
\( \sqrt{n} \) possible for k
\( \Rightarrow \# \text{steps} \) \( O(\sqrt{n} \cdot \sqrt{n} \cdot \sqrt{n}) \)
\( = O(n^{3/2}) \)
```
A survey of common running times

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

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

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