Week 3 Tutorial

CSE Dept, CUHK
The Predecessor Search Problem

Problem Input

- An array $A$ of $n$ integers in ascending order
- A search value $q$

Goal:

Find the predecessor of $q$ in $A$.

Remark: the predecessor of $q$ is the largest element in $A$ that is smaller than or equal to $q$. 
Example

1. If $q = 23$, the predecessor is 21.
2. If $q = 21$, the predecessor is also 21.
3. If $q = 1$, no predecessor.

\[
\begin{array}{cccccccc}
2 & 3 & 5 & 8 & 13 & 21 & 34 & 55 \\
\end{array}
\]

$A$
Binomial Search

- If $A$ contains $q$, binary search will find $q$ directly.
- If $A$ does not contain $q$, the predecessor of $q$ can be easily inferred from where the algorithm terminates.

<table>
<thead>
<tr>
<th>2</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>13</th>
<th>21</th>
<th>34</th>
<th>55</th>
</tr>
</thead>
</table>

$A$
The Two-Sum Problem

**Input**

- A array of $n$ integers in ascending order.
- An integer $v$.

**Goal:**

Determine whether $A$ contains two different integers $x$ and $y$ such that $x + y = v$. 

CSCI2100 2021 Fall, CUHK
Example

- If $v = 30$, answer “yes”.
- If $v = 25$, answer “no”.

| 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 |
Solution

Use binary search as a building brick.

**Key idea:** For each $x$ in the array, look for $v - x$ with binary search.
Analysis

This algorithm performs at most \( n \) binary searches.

Cost of the algorithm: \( O(n \log n) \)

Can you do even better?
Try to solve this problem in \( O(n) \) time (not covered in this tutorial).
Recall the definition of $f(n) = O(g(n))$:

$$f(n) = O(g(n)),$$ if there exist two positive constants $c_1$ and $c_2$ such that $f(n) \leq c_1 \cdot g(n)$ holds for all $n \geq c_2$.

Another approach is to compute $\lim_{n \to \infty} \frac{f(n)}{g(n)}$ and decide as follows:

- $f(n) = O(g(n))$, if the limit is bounded by an constant;
- $f(n) \neq O(g(n))$, if the limit is $\infty$.

Note: there is a third possibility for the limit, where the approach will fail. We will discuss this at the end of the tutorial.
Exercise 1

Let $f(n) = 10n + 5$ and $g(n) = n^2$. Prove $f(n) = O(g(n))$. 
Exercise 1

Let \( f(n) = 10n + 5 \) and \( g(n) = n^2 \). Prove \( f(n) = O(g(n)) \).

Method 1: Constant finding

1. Fix \( c_1 \)
2. Solve for \( c_2 \)
3. If a \( c_2 \) cannot be found, go back to Step 1 and try a different \( c_1 \).
Exercise 1

Let $f(n) = 10n + 5$ and $g(n) = n^2$. Prove $f(n) = O(g(n))$ (try $c_1 = 5$)

\[
\begin{align*}
  f(n) &\leq c_1 \cdot g(n) \\
  \Leftrightarrow &\quad 10n + 5 \leq c_1 \cdot n^2 \\
  \Leftrightarrow &\quad 5(2n + 1) \leq 5 \cdot n^2 \\
  \Leftrightarrow &\quad 2n + 1 \leq n^2 \\
  \Leftrightarrow &\quad 2 \leq (n - 1)^2 \\
  \Leftrightarrow &\quad 3 \leq n
\end{align*}
\]

Hence, it suffices to set $c_2 = 3$. 
Exercise 1

Let $f(n) = 10n + 5$ and $g(n) = n^2$. Prove $f(n) = O(g(n))$.

**Method 2: Limit**

$$\lim_{n \to \infty} \frac{10n + 5}{n^2} = \lim_{n \to \infty} \frac{10 + 5/n}{n} = 0.$$ 

Hence, $f(n) = O(g(n))$. 
Exercise 2

Let $f(n) = 10n + 5$ and $g(n) = n^2$. Prove $g(n) \neq O(f(n))$.

Method 1: Constant finding (prove by contradiction)

Suppose that $g(n) = O(f(n))$, i.e., there are constants $c_1, c_2$ such that, for all $n \geq c_2$, we have

\[
    n^2 \leq c_1 \cdot (10n + 5) \\
    \Rightarrow \quad n^2 \leq c_1 \cdot 20n \\
    \Leftrightarrow \quad n \leq 20c_1
\]

which cannot hold for all $n \geq c_2$, regardless of $c_2$. This gives a contradiction.
Exercise 2

Let $f(n) = 10n + 5$ and $g(n) = n^2$. Prove $g(n) \neq O(f(n))$.

Method 2: Limit

$$\lim_{{n \to \infty}} \frac{n^2}{10n + 5} = \infty.$$ 

Hence, $g(n) \neq O(f(n))$. 
In some rare scenarios, the limit approach may fail. We will see an example next.
Consider \( f(n) = 2^n \). Define \( g(n) \) as:

- \( g(n) = 2^n \) if \( n \) is even;
- \( g(n) = 2^{n-1} \) otherwise.

Since \( f(n) \leq 2g(n) \) holds for all \( n \geq 1 \), it holds that \( f(n) = O(g(n)) \).

However, \( \lim_{n \to \infty} \frac{f(n)}{g(n)} \) does not exist, because it keeps jumping between 1 and 2 as \( n \) increases!