Which factor below is most important in choosing which algorithm to use?

A) Simplest
B) Easiest to implement/code
C) Takes least time
D) Use up less space (memory)
E) Gives a more precise answer
Are you my student?

Problem: Is a student in a class?

Input:
1. unsorted list of students in class (L)
2. particular student (S)

Algorithm:
1. Go one by one checking each student
2. If match, YES
3. If reach end of L and no matches, NO

Worst case running time as a function of the size of L?

A) Constant - O(1)
B) Logarithmic - O(log n)
C) Linear - O(n)
D) Quadratic - O(n^2)
E) Exponential O(c^n)
What is this kind of search called?

A) Big-Oh search
B) Key search
C) Traveling Salesman search
D) Binary search
E) Linear search
Are you my student?

**Problem:** Is a student in a class?

**Input:**
1. sorted list of students in class (L)
2. particular student (S)

**Algorithm:**
1. Start in middle of L
2. If match, YES
3. Throw away half of L and check again in the middle of the remaining part of L.
4. If nobody left, NO

Worst case running time as a function of the size of L?

A) Constant - O(1)
B) Logarithmic - O(log n)
C) Linear - O(n)
D) Quadratic - O(n^2)
E) Exponential O(c^n)
What is this kind of search called?

A) Big-Oh search
B) Key search
C) Traveling Salesman search
D) Binary search
E) Linear search
Problem: Is a student in a class?

Input:
1. list of students in class (L) given to you in advance and you have unlimited storage/memory.

Algorithm:
Can we do better then logarithmic?

Worst case running time as a function of the size of L?

A) Constant - O(1)
B) Logarithmic - O(log n)
C) Linear - O(n)
D) Quadratic - O(n^2)
E) Exponential - O(c^n)
Yes, We Can! (do better then logarithmic)

<table>
<thead>
<tr>
<th>In-class?</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>No</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>StuID</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In-class?</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>StuID</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>
No Free Lunch Theorem

- Computer Science cultural reference
- “TANSTAAFL”
- “There Ain't No Such Thing As A Free Lunch”
  - Can't get something for nothing.
  - Have to make tradeoffs.
- What was tradeoff in last algorithm?
**Are you my twin?**

**Problem:** Do any two students in a class share a birthday?

**Input:** unsorted list of birthdays of students in class (L)

**Algorithm idea:**
- Compare each student's birthday to that of all the other students
  - If match => YES
  - If checked last student in L and no matches => NO

**Worst case running time as a function of the size of L?**

A) Constant - O(1)  
B) Logarithmic - O(log n)  
C) Linear - O(n)  
D) Quadratic - O(n^2)  
E) Exponential O(c^n)
Graphical Representation: