Examples and Applications of Binary Search

CSCI2100 Tutorial 1
Hou Pong CHAN, Ken

Adapted from the slides of Tony Gong
In the Lectures

• Studies the binary search algorithm
• Solves the problem of determining if a particular value appears in a sorted list of integer or not
• Proves that the worst-case running time under the RAM model
  • \( O(\log_2 n) \)
Example 1

• Suppose we have the following sorted input set, where \( n = 8 \) and we are trying to find the value 13.
Example 1

• Initializing $L$ to be 1 and $R$ to $n$ (in this case 8)
Example 1

• Since $R > L$
• Proceed by computing $M$
Example 1

• Compare $v$ and the value indexed by $M$ (13 and 8)
• $v >$ the value indexed by $M$
• Means that the target is in the right half of the sorted sequence

```
  2  3  5  8  13  21  34  55 ...
```

```
  n  v  L  M  R ...

  8  13  1  4  8 ...
```
Example 1

- Look at the right half of the sorted sequence
- Set $L$ to be $M + 1$ (discard the left half)
- Recompute $M$
Example 1

• Compare \( v \) and the value indexed by \( M \) (13 and 21)
• \( v < \) the value indexed by \( M \)
• Means that the target is in the left half of the sorted sequence
Example 1

- Set $R$ to be $M - 1$ (discard the right half)
- $L$ and $R$ converged, $L, M, R = 5$
- Only a single value for us to check
- We find 13, return “yes”
The Sum of Two Integers Problem

• Problem Input:
  • A sequence of \( n \) positive integers in strictly increasing order in memory at the cells numbered from 1 up to \( n \)
  • The value \( n \) has been placed in Register 1
  • A positive integer \( v \) has been placed in Register 2

• Goal:
  • Determine whether if there exist two integers \( x \) and \( y \) (not necessarily distinct) in the sorted sequence such that \( x + y = v \)
Example

- A “yes”-input with $n = 12$, $v = 30$
Example

- A “no”-input with $n = 12$, $v = 29$

\[
\begin{array}{|c|c|}
\hline
n & v \\
\hline
12 & 29 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
2 & 3 & 5 & 7 & 11 & 13 \\
\hline
17 & 19 & 23 & 29 & 31 & 37 \\
\hline
\end{array}
\]

...
A First Attempt

• Naïve algorithm:
  • Enumerate all possible pairs in the sorted sequence
  • Check if they sum to $v$
  • There are $\binom{n}{2} = \frac{n(n-1)}{2}$ possible pairs
  • Time complexity: $O(n^2)$

• Can we do better than this?

• Hint: Take advantage of the fact that the given sequence is sorted!
Binary Search the Answer

• Find $x$ and $y$ such that $x + y = v$
• Rearrange into $y = v - x$
• Rephrase the problem
  • Whether if such a $y$ exists in the sequence for at least one of $x$ in the sequence
• Solution:
  • For each $x$ in the sequence:
    • Compute $y$ as $v - x$
    • Use binary search to see if $y$ exists in the sequence
The Repeated Binary Search Algorithm

• Pseudocode:

1. Let $n$ be register 1 and $v$ be register 2
2. register $i \leftarrow 1$, register $one \leftarrow 1$
3. while $i \leq n$
4. \hspace{1em} read into register $x$ the memory cell at address $i$
5. \hspace{1em} $y \leftarrow v - x$
6. \hspace{1em} if $BinarySearch(y) =$ “yes”
7. \hspace{2em} return “yes”
7. \hspace{1em} $i \leftarrow i + one$ (effectively increasing $i$ by 1)
8. return “no”
Time Complexity

• Worst case (when the output is “no”)
• This algorithm needs to run binary search $n$ times
• Time complexity of binary search: $O(\log_2 n)$
• Time complexity of this algorithm: $O(n \log_2 n)$

• Can we do even better?
An Even Better Algorithm

- Utilize the fact that the sequence is sorted
- Can find the solution by considering each term only once
- That means, time complexity is $O(n)$
An Even Better Algorithm

• Conceptually we have two pointers (orange)
• Begin by pointing at the start and the end resp.
• Summed the two numbers being pointed to
• We get 39, which is greater than the desired value, 30
An Even Better Algorithm

- \(2 + 37 > 30\) tell us that:
  - 37 will never appear in a valid solution, because 2 is the smallest term in the sequence and the sum is already greater than the desired value.
  - Should move the right pointer towards the left, since this will decrease the overall value of the sum.
An Even Better Algorithm

• $2 + 31 > 30$

• Move the right pointer towards the left
An Even Better Algorithm

- $2 + 29 > 30$
- Move the right pointer towards the left
An Even Better Algorithm

- $2 + 23 < 30$ tells us that
  - $2$ never appears in a solution because $23$ is the largest value that we have, and that the solution is still too small, so $2$ plus any other number in the sequence would also be too small
  - Should move the left pointer towards the right to increase our “estimate”
An Even Better Algorithm
An Even Better Algorithm

• Now, $7 + 23 = 30$
• We find $x$ and $y$ in the sequence such that $x + y = \nu$
• Return “yes”
An Even Better Algorithm

• Think of $x$ as the element being pointed to by the left pointer and $y$ as the element being pointed to by the right pointer:
  • If $x + y = v$, we are done
  • If $x + y > v$, we need to make the sum smaller, so move $y$ towards the left
  • If $x + y < v$, we need to make the sum bigger, so move $x$ towards the right
An Even Better Algorithm

• Algorithm:
  1. let \( n \) be register 1, and \( v \) be register 2
  2. register \( left \leftarrow 1 \), \( right \leftarrow n \)
  3. while \( left \leq right \)
  4. read into register \( x \) the memory cell at address \( left \)
  5. read into register \( y \) the memory cell at address \( right \)
  6. if \( x + y = v \) then
  7.     return “yes”
  8. else if \( x + y > v \) then
  9.     \( right \leftarrow right - 1 \)
 10. else
 11.     \( left \leftarrow left + 1 \)
 12. return “no”

• Time complexity: \( O(n) \)
Recap

• Review the binary search algorithm
• Look at a problem that can be solved by repeated application of binary search (although there exists a better algorithm)
• You are encouraged to run this algorithm on some input sets and convince yourself
• [https://leetcode.com/problems/two-sum/description/](https://leetcode.com/problems/two-sum/description/)