**Part-1: Code**

Click on the link to view: [Code\_Part1.docx](https://livecoloradotech-my.sharepoint.com/personal/bavitha_battipati_student_ctuonline_edu/_layouts/15/Doc.aspx?sourcedoc=%7B64448789-6ADB-4B3C-B0D1-E0F489CB0CAC%7D&file=Document%2015.docx&action=default&mobileredirect=true)

### **Part 2: Understanding Time Complexity**

#### **Time Complexity of Ternary Search**

The ternary search algorithm works by dividing the playlist into three equal sections at each step. If the song you're looking for isn't found at the two pivot points (often called mid1 and mid2), the search continues in only one of those three sections. Because the size of the search space is reduced to one-third with each iteration, the total number of steps required to complete the search depends on how many times you can divide the playlist by three.

In simpler terms, the time complexity of ternary search grows logarithmically in relation to the size of the playlist, but with a base of 3. This is written as **O(log₃(n))**, where **n** is the size of the playlist.

As the playlist grows larger, the algorithm will take a few more steps to find the song. However, the search remains efficient because the size of the section being searched shrinks significantly with each step.

#### **Comparing Ternary Search and Binary Search**

Ternary search is similar to binary search, but instead of dividing the playlist into two sections at each step, it divides it into three. This means that binary search reduces the search space more quickly than ternary search. The time complexity of binary search is **O(log₂(n))**, indicating that the number of steps required depends on how many times you can divide the playlist by two.

**Here’s the key difference:**

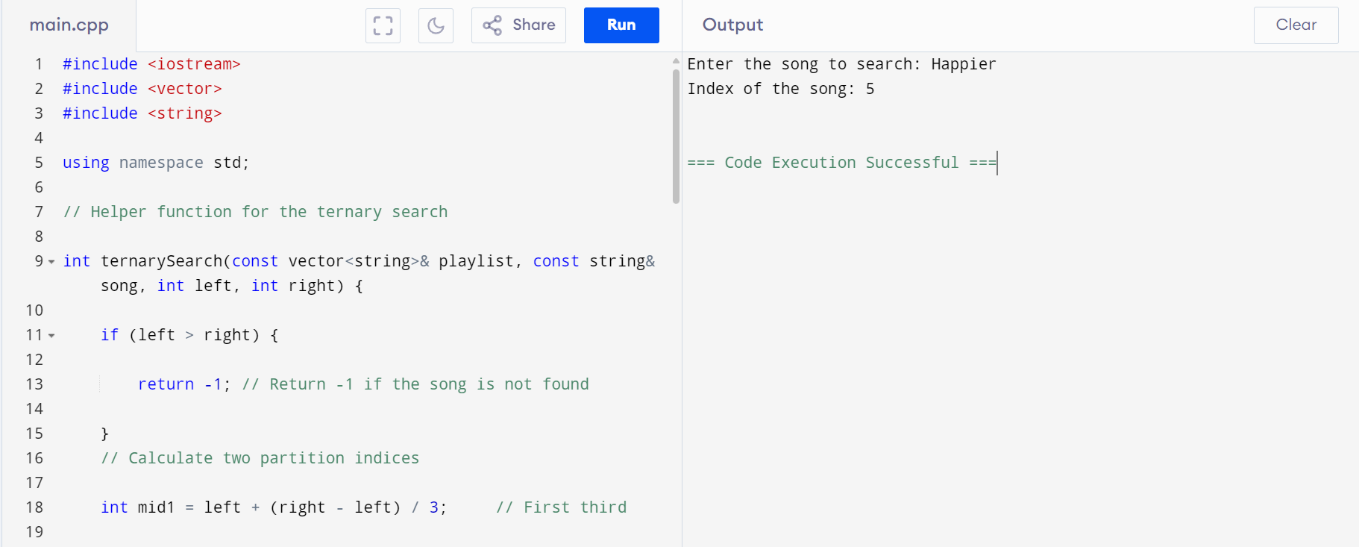
• Ternary search (O(log3(n))) takes more steps because dividing by 3 reduces the size of the playlist less drastically than dividing by 2.

• Binary search (O(log2(n))) is slightly faster for this reason.

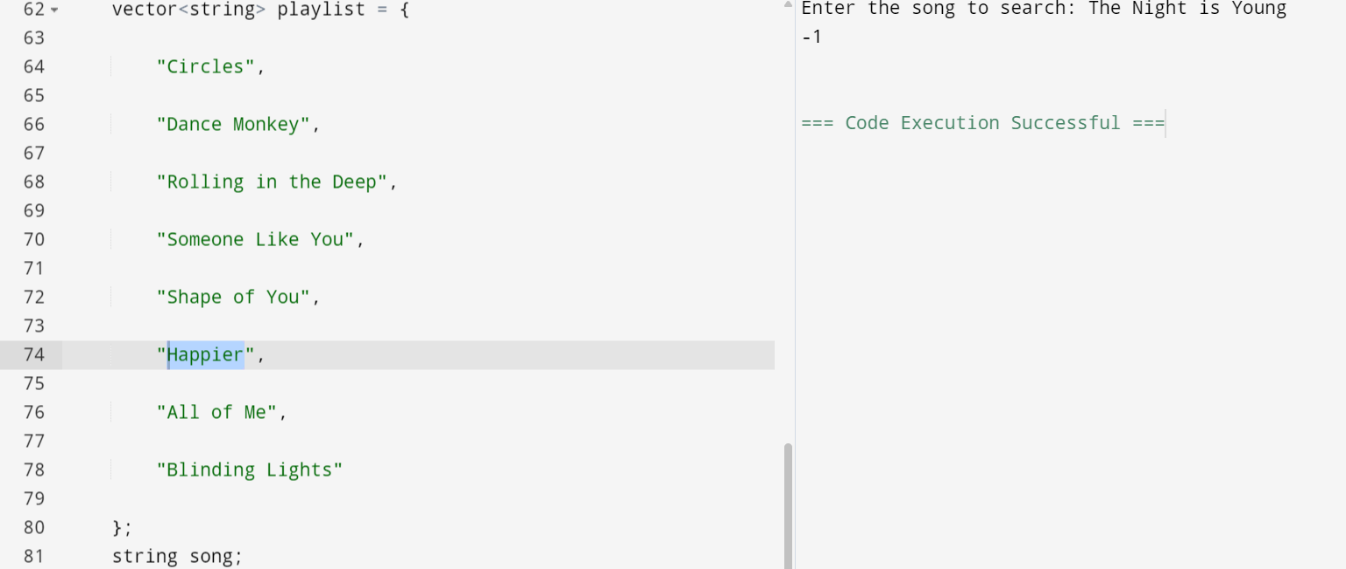
However, the difference in speed is negligible for smaller playlists. For very large playlists, binary search is generally preferred because it requires fewer steps overall.

**Why Use Ternary Search?**

Even though binary search is faster in most cases, ternary search demonstrates an interesting variation of divide-and-conquer. It shows how dividing the problem into more parts can still lead to efficient solutions. In specific cases, such as parallel processing or distributed systems, ternary search may have advantages.

**Output:** 

**Output when entered a song that doesn’t exist in the playlist:**



**Online IDE used to run code:** [Online C++ Compiler - Programiz](https://www.programiz.com/cpp-programming/online-compiler/)

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

Knuth, D. E. (1997). The Art of Computer Programming: Sorting and Searching (Vol. 3). Addison-Wesley.

GeeksforGeeks. (n.d.). Ternary Search. Retrieved from [https://www.geeksforgeeks.org](https://www.geeksforgeeks.org/ternary-search/)/ternary-search/

Stroustrup, B. (2013). The C++ Programming Language (4th ed.). Addison-Wesley.