Ethan Wong

SYOPS

I fand binary trees particularly interesting this week so I decrded to do a bit more research about them. My first question was:

Q: How are binary trees used in the real world?

A: Binary trees have many different forms and therefore have a variety of applications. One example that corresponded with my computer science class was something called "Huffman Coding". This is a technique used frequently in computer science to compress large files. Binary search trees are also utilized in a lot of search applications for data structures, such as maps and sets. A final example of where binary trees shine is the A* algorithm. This is a famous algorithm that AI applications use to find the effcient, short paths between points.

While researching for my first question, I became curious how binary search is implemented and how exactly it relates to binary trees.

Q: What is binary search and how does the algorithm for it look?

A: Binary search is essentially a simple way to find an object within a set. This is accomplished by sorting the set and repeatedly dividing it in half. If the object is less than the middle element of the set, we know the object wouldn't be in the latter half so we discard it. If the object is greater than the middle element of the set, we know that the object wouldn't be in the first half so we discard it. This is repeated until the object is found. This idea is commonly implemented with recursion.

Sample Binary Search Algorithm:

```
def Bsearch(lst, L, R, x):
  if R >= L:
    middle = L + (R - L) // 2
    if lst[middle] == x:
      return middle
    elif array[middle] > x:
      return bSearch(array, middle - 1, R, x)
    else:
      return bSearch(array, middle + 1, R, x)
  else:
    return "Item not in array"
```