In [25]:
```
import pandas as pd
import math
import time
data=pd.read_csv("data500.csv")
data
arr=data["numbers"]
arr
def insertion(arr):
    size=len(arr)
    if size<=1:
        return
    for i in range(1,size):
        key=arr[i]
        j=i-1
        while j>=0 and key<arr[j]:
            arr[j+1]=arr[j]
            j-=1
        arr[j+1]=key
start=time.perf_counter()
insertion(arr)
# DAC("data500.csv")
end=time.perf_counter()
timetaken=end-start
timetaken
```

Out[25]: 8.278960600495338

In [5]:
```
data.head(1)
```

Out[5]:

	numbers
0	4

In [6]:
```
data.tail(1)
```

Out[6]:

	numbers
499	998

In [2]:
```
import pandas as pd
import math
import time
def mergesort(arr,l,r):
    if l<r:
        mid=int((l+r)/2)
        mergesort(arr,l,mid)
        mergesort(arr,mid+1,r)
        merge(arr,l,mid,r)
def merge(arr,l,mid,r):
    n1=mid-l+1
    n2=r-mid
    L=[]
    R=[]
    for i in range(n1):
        L.append(arr[l+i])
        i+=1
    for j in range(n2):
        R.append(arr[mid+j+1])
        j+=1
    i=j=0
    k=l
    while i<n1 and j<n2:
        if L[i]<R[j]:
            arr[k]=L[i]
            i+=1
        else:
            arr[k]=R[j]
            j+=1
        k+=1
    while i<n1:
        arr[k]=L[i]
        i+=1
        k+=1
    while j<n2:
        arr[k]=R[j]
        j+=1
        k+=1
data=pd.read_csv("data500.csv")
arr=data["numbers"]
start=time.perf_counter()
mergesort(arr,0,len(arr)-1)
end=time.perf_counter()
timetaken=end-start
timetaken
```

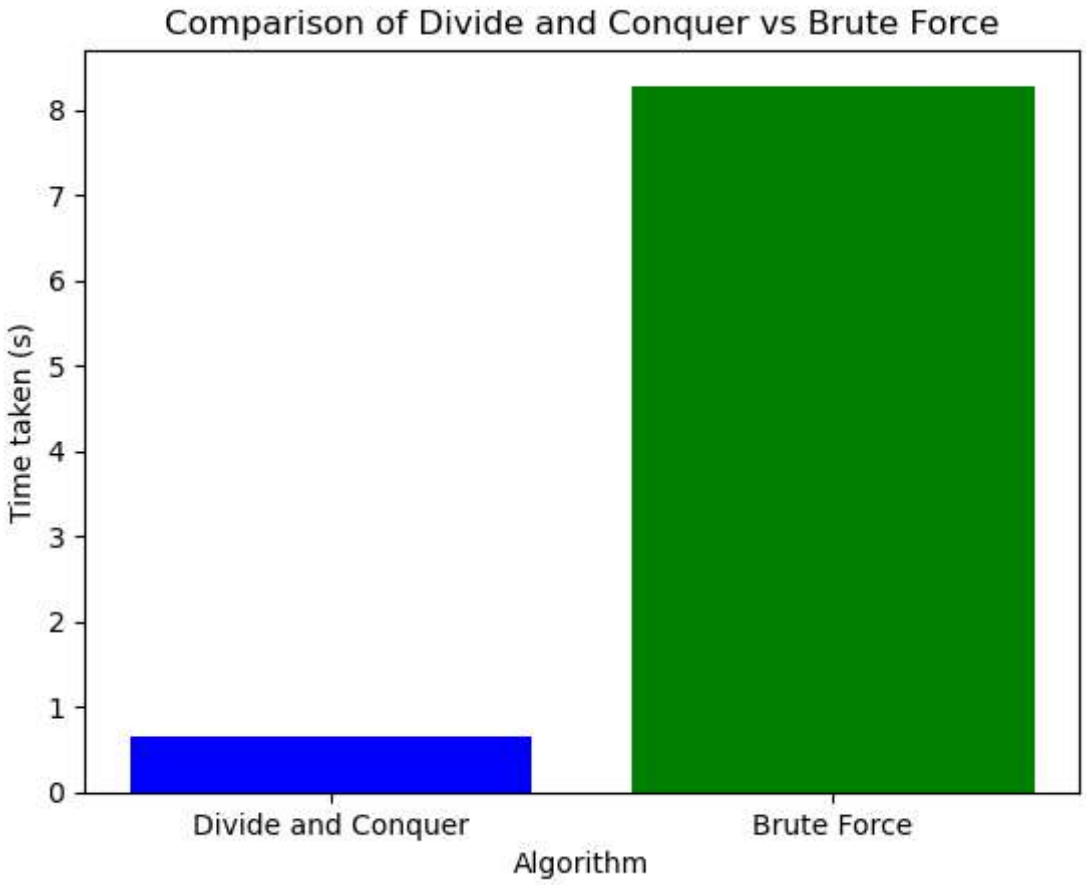
Out[2]: 0.6565837000000556

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt

labels = ["Divide and Conquer", "Brute Force"]
times = [0.6565837000000556,8.278960600495338]

# Define custom colors for each bar
colors = ["blue", "green"]

plt.bar(labels, times, color=colors)
plt.xlabel("Algorithm")
plt.ylabel("Time taken (s)")
plt.title("Comparison of Divide and Conquer vs Brute Force")
plt.show()
```



In [3]:
```
# Statement-2: In a school there is a class photograph. The class topper (say
John) needs to
# stand in the middle and the other students need to stand height wise. Consid
er that there are N
# students in the class.
def sort_height(J,N,h):
    h.append(J)
    h.sort()
    size=len(h)
    ind=h.index(J)
    count_l=0
    count_r=0
    for i in range(0,ind):
        count_l+=1
    for j in range(ind+1,size):
        count_r+=1
    if count_l==0:
        i=0
        while count_l!=count_r:
            h.insert(i,h[ind]-(i+1))
            i+=1
            count_l+=1
        return h
    elif count_r==0:
        i=0
        while count_l!=count_r:
            h.insert(ind+(i+1),h[ind]+(i+1))
            i+=1
            count_r+=1
        return h
    elif count_l<count_r:
        while count_l!=count_r:
            h.pop()
            count_r-=1
        return h
    elif count_l>count_r:
        while count_l!=count_r:
            a=h[len(h)-1]+1
            h.append(a)
            count_r+=1
        return h
    else:
        return h

J=5
N=1
h=[6]
sort_height(J,N,h)
print(h)
```

```
[4, 5, 6]
```

In [ ]:

In [ ]: