In [18]:

```python
import random
import time
import tracemalloc
import matplotlib.pyplot as plt
from random import sample
```

In [19]:

```python
def interpolationSearch(A, x):

    # search space is `A[left…right]`
    (left, right) = (0, len(A) - 1)

    while A[right] != A[left] and A[left] <= x <= A[right]:

        # estimate mid
        pos = left + (x - A[left]) * (right - left) // (A[right] - A[left])

        # key is found
        if x == A[pos]:
            return pos

        # discard all elements in the right search space,
        # including the middle element
        elif x < A[pos]:
            right = pos - 1

        # discard all elements in the left search space,
        # including the middle element
        else:
            left = pos + 1

    # if the key is found
    if x == A[left]:
        return left

    # `x` doesn't exist in the list
    return -1
#--------------------------------------------------------------------
def jumpSearch( arr , x , n ):

    # Finding block size to be jumped
    step = math.sqrt(n)

    # Finding the block where element is
    # present (if it is present)
    prev = 0
    while arr[int(min(step, n)-1)] < x:
        prev = step
        step += math.sqrt(n)
        if prev >= n:
            return -1

    # Doing a linear search for x in
    # block beginning with prev.
    while arr[int(prev)] < x:
        prev += 1

        # If we reached next block or end
        # of array, element is not present.
        if prev == min(step, n):
            return -1

    # If element is found
    if arr[int(prev)] == x:
        return prev

    return -1

#------------------------------------------------------
# For best case
def measure_best(seq,number):
    start = time.time()
    tracemalloc.start()
    subset = sample(seq, number)
    interpolationSearch(subset,subset[number//2])
    end = time.time()
    runtimeInter = end - start
    current, peakI = tracemalloc.get_traced_memory()
    usageInter = current / 10**6
    print("Using interpolation search: ")
    print(f"Best case memory usage is: {usageInter} MB; Peak was {peakI / 10**6} MB")
    print(f"Best case runtime is: {runtimeInter} sec")
    start = time.time()
    tracemalloc.start()
    subset = sample(seq, number)
    interpolationSearch(subset,subset[number//2])
    end = time.time()
    runtimeJump = end - start
    current, peakJ = tracemalloc.get_traced_memory()
    usageJump = current / 10**6
    print("Using jump search: ")
```

```python
    print(f"Best case memory usage is: {usageJump} MB; Peak was {peakJ / 10**6} MB")
    print(f"Best case runtime is: {runtimeJump} sec")
    return runtimeInter, peakI, runtimeJump, peakJ

# For average case
def measure_average(seq,number):
    start = time.time()
    tracemalloc.start()
    subset = sample(seq, number)
    interpolationSearch(subset,subset[50])
    end = time.time()
    runtimeInter = end - start
    current, peakI = tracemalloc.get_traced_memory()
    usageInter = current / 10**6
    print("Using interpolation search: ")
    print(f"Average case memory usage is: {usageInter} MB; Peak was {peakI / 10**6} MB")
    print(f"Average case runtime is: {runtimeInter} sec")
    start = time.time()
    tracemalloc.start()
    subset = sample(seq, number)
    interpolationSearch(subset,subset[50])
    end = time.time()
    runtimeJump = end - start
    current, peakJ = tracemalloc.get_traced_memory()
    usageJump = current / 10**6
    print("Using jump search: ")
    print(f"Average case memory usage is: {usageJump} MB; Peak was {peakJ / 10**6} MB")
    print(f"Average case runtime is: {runtimeJump} sec")
    return runtimeInter, peakI, runtimeJump, peakJ

# For worst case
def measure_worst(seq,number):
    start = time.time()
    tracemalloc.start()
    subset = sample(seq, number)
    interpolationSearch(subset,subset[number-1])
    end = time.time()
    runtimeInter = end - start
    current, peakI = tracemalloc.get_traced_memory()
    usageInter = current / 10**6
    print("Using interpolation search: ")
    print(f"Worst case memory usage is: {usageInter} MB; Peak was {peakI / 10**6} MB")
    print(f"Worst case runtime is: {runtimeInter} sec")
    start = time.time()
    tracemalloc.start()
    subset = sample(seq, number)
    interpolationSearch(subset,subset[number-1])
    end = time.time()
    runtimeJump = end - start
    current, peakJ = tracemalloc.get_traced_memory()
    usageJump = current / 10**6
    print("Using jump search: ")
    print(f"Worst case memory usage is: {usageJump} MB; Peak was {peakJ / 10**6} MB")
    print(f"Worst case runtime is: {runtimeJump} sec")
    return runtimeInter, peakI, runtimeJump, peakJ
```

In [20]:

```
random.seed(1)
sequence = [i for i in range(100000)]
seqRange = [100,500,1000,5000,10000,50000,100000]
bestCase = []
averageCase = []
worstCase = []
for i in range(7):
     print(f"{i+1}: for {seqRange[i]} items\n --------------------")
     bestCase.append(measure_best(sequence,seqRange[i]))
     averageCase.append(measure_average(sequence,seqRange[i]))
     worstCase.append(measure_worst(sequence,seqRange[i]))
print(bestCase)
print(averageCase)
print(worstCase)
```

```
1: for 100 items
 --------------------
Using interpolation search:
Best case memory usage is: 28.836752 MB; Peak was 31.152103 MB
Best case runtime is: 0.0009930133819580078 sec
Using jump search:
Best case memory usage is: 28.838383 MB; Peak was 31.152103 MB
Best case runtime is: 0.0 sec
Using interpolation search:
Average case memory usage is: 28.8399 MB; Peak was 31.152103 MB
Average case runtime is: 0.0 sec
Using jump search:
Average case memory usage is: 28.841493 MB; Peak was 31.152103 MB
Average case runtime is: 0.000995635986328125 sec
Using interpolation search:
Worst case memory usage is: 28.843033 MB; Peak was 31.152103 MB
Worst case runtime is: 0.0 sec
Using jump search:
Worst case memory usage is: 28.844288 MB; Peak was 31.152103 MB
Worst case runtime is: 0.0009984970092773438 sec
2: for 500 items
 --------------------
Using interpolation search:
Best case memory usage is: 28.849544 MB; Peak was 31.152103 MB
Best case runtime is: 0.000997304916381836 sec
```

```
Using jump search:
Best case memory usage is: 28.850168 MB; Peak was 31.152103 MB
Best case runtime is: 0.001978158950805664 sec
Using interpolation search:
Average case memory usage is: 28.85167 MB; Peak was 31.152103 MB
Average case runtime is: 0.001999378204345703 sec
Using jump search:
Average case memory usage is: 28.853281 MB; Peak was 31.152103 MB
Average case runtime is: 0.0010073184967041016 sec
Using interpolation search:
Worst case memory usage is: 28.85479 MB; Peak was 31.152103 MB
Worst case runtime is: 0.001987457275390625 sec
Using jump search:
Worst case memory usage is: 28.856397 MB; Peak was 31.152103 MB
Worst case runtime is: 0.0009970664978027344 sec
3: for 1000 items
 --------------------
Using interpolation search:
Best case memory usage is: 28.862422 MB; Peak was 31.152103 MB
Best case runtime is: 0.004983901977539O625 sec
Using jump search:
Best case memory usage is: 28.863853 MB; Peak was 31.152103 MB
Best case runtime is: 0.003405570983886718 8 sec
Using interpolation search:
Average case memory usage is: 28.867941 MB; Peak was 31.152103 MB
Average case runtime is: 0.0029871463775634766 sec
Using jump search:
Average case memory usage is: 28.869554 MB; Peak was 31.152103 MB
Average case runtime is: 0.001993417739868164 sec
Using interpolation search:
Worst case memory usage is: 28.871062 MB; Peak was 31.152103 MB
Worst case runtime is: 0.001993417739868164 sec
Using jump search:
Worst case memory usage is: 28.850508 MB; Peak was 31.152103 MB
Worst case runtime is: 0.0029900074005126953 sec
4: for 5000 items
 --------------------
Using interpolation search:
Best case memory usage is: 28.884707 MB; Peak was 31.152103 MB
Best case runtime is: 0.015951871871948242 sec
Using jump search:
Best case memory usage is: 28.882489 MB; Peak was 31.152103 MB
Best case runtime is: 0.01597452163696289 sec
Using interpolation search:
Average case memory usage is: 28.884222 MB; Peak was 31.152103 MB
Average case runtime is: 0.015938758850097656 sec
Using jump search:
Average case memory usage is: 28.884904 MB; Peak was 31.152103 MB
Average case runtime is: 0.015949726104736328 sec
Using interpolation search:
Worst case memory usage is: 28.884806 MB; Peak was 31.152103 MB
Worst case runtime is: 0.015523910522460938 sec
```

```
Using jump search:
Worst case memory usage is: 28.883918 MB; Peak was 31.152103 MB
Worst case runtime is: 0.015080928802490234 sec
5: for 10000 items
 --------------------
Using interpolation search:
Best case memory usage is: 28.924775 MB; Peak was 31.152103 MB
Best case runtime is: 0.029845476150512695 sec
Using jump search:
Best case memory usage is: 28.925173 MB; Peak was 31.152103 MB
Best case runtime is: 0.02890300750732422 sec
Using interpolation search:
Average case memory usage is: 28.926602 MB; Peak was 31.152103 MB
Average case runtime is: 0.03055882453918457 sec
Using jump search:
Average case memory usage is: 28.926203 MB; Peak was 31.152103 MB
Average case runtime is: 0.032889366149902344 sec
Using interpolation search:
Worst case memory usage is: 28.924202 MB; Peak was 31.152103 MB
Worst case runtime is: 0.031864166259765625 sec
Using jump search:
Worst case memory usage is: 28.919154 MB; Peak was 31.152103 MB
Worst case runtime is: 0.03891253471374512 sec
6: for 50000 items
 --------------------
Using interpolation search:
Best case memory usage is: 29.238504 MB; Peak was 31.152103 MB
Best case runtime is: 0.23654603958129883 sec
Using jump search:
Best case memory usage is: 29.239397 MB; Peak was 31.152103 MB
Best case runtime is: 0.23236584663391113 sec
Using interpolation search:
Average case memory usage is: 29.239064 MB; Peak was 31.152103 MB
Average case runtime is: 0.25104331970214844 sec
Using jump search:
Average case memory usage is: 29.239843 MB; Peak was 31.152103 MB
Average case runtime is: 0.24285507202148438 sec
Using interpolation search:
Worst case memory usage is: 29.239504 MB; Peak was 31.152103 MB
Worst case runtime is: 0.2372276782989502 sec
Using jump search:
Worst case memory usage is: 29.24034 MB; Peak was 31.152103 MB
Worst case runtime is: 0.22856807708740234 sec
7: for 100000 items
 --------------------
Using interpolation search:
Best case memory usage is: 29.640184 MB; Peak was 31.152103 MB
Best case runtime is: 0.47336554527282715 sec
Using jump search:
Best case memory usage is: 29.640208 MB; Peak was 31.244547 MB
Best case runtime is: 0.5106029510498047 sec
Using interpolation search:
```

```
Average case memory usage is: 29.640488 MB; Peak was 31.244547 MB
Average case runtime is: 0.49272704124450684 sec
Using jump search:
Average case memory usage is: 29.640872 MB; Peak was 31.244874 MB
Average case runtime is: 0.4965023994445801 sec
Using interpolation search:
Worst case memory usage is: 29.640872 MB; Peak was 31.244874 MB
Worst case runtime is: 0.5332720279693604 sec
Using jump search:
Worst case memory usage is: 29.641038 MB; Peak was 31.245264 MB
Worst case runtime is: 0.4961988925933838 sec
[(0.0009930133819580078, 31152103, 0.0, 31152103), (0.000997304916381836, 31152103, 0.001978158950805664, 3115210
3), (0.004983901977539062, 31152103, 0.003405570983886718 8, 31152103), (0.01595187187194824 2, 31152103, 0.0159745
2163696289, 31152103), (0.02984547615051269 5, 31152103, 0.02890300750732422, 31152103), (0.23654603958129883, 3115
2103, 0.23236584663391113, 31152103), (0.47336554527282715, 31152103, 0.5106029510498047, 31244547)]
[(0.0, 31152103, 0.000995635986328125, 31152103), (0.001999378204345703, 31152103, 0.001007318496704101 6, 3115210
3), (0.0029871463775634766, 31152103, 0.001993417739868164, 31152103), (0.015938758850097656, 31152103, 0.01594972
6104736328, 31152103), (0.03055882453918457, 31152103, 0.03288936614990234 4, 31152103), (0.25104331970214844, 3115
2103, 0.24285507202148438, 31152103), (0.49272704124450684, 31244547, 0.4965023994445801, 31244874)]
[(0.0, 31152103, 0.0009984970092773438, 31152103), (0.001987457275390625, 31152103, 0.0009970664978027344, 3115210
3), (0.001993417739868164, 31152103, 0.0029900074005126953, 31152103), (0.015523910522460938, 31152103, 0.01508092
8802490234, 31152103), (0.031864166259765625, 31152103, 0.03891253471374512, 31152103), (0.2372276782989502, 31152
```

In [21]:

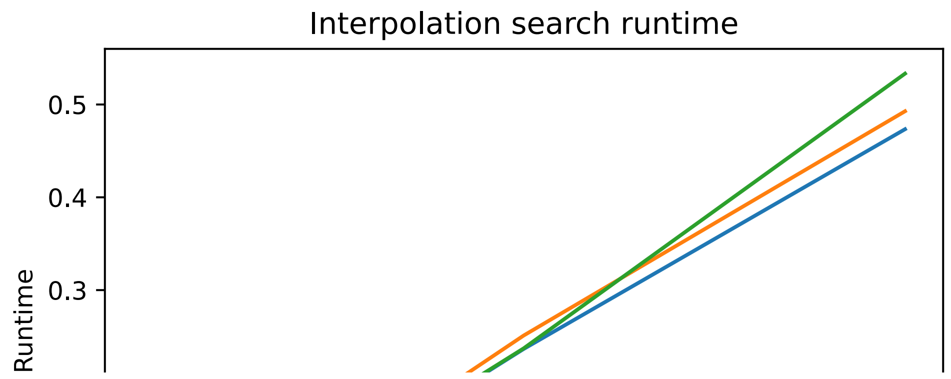
```python
best_inter_runtime, best_inter_usage, best_jump_runtime, best_jump_usage = map(list, zip(*bestCase))
average_inter_runtime, average_inter_usage, average_jump_runtime, average_jump_usage = map(list, zip(*averageCase))
worst_inter_runtime, worst_inter_usage, worst_jump_runtime, worst_jump_usage = map(list, zip(*worstCase))
```

In [24]:

```python
fig, ax = plt.subplots()
ax.plot(seqRange, best_inter_runtime, label='Best')
ax.plot(seqRange, average_inter_runtime, label='Average')
ax.plot(seqRange, worst_inter_runtime, label='Worst')
ax.set_xlabel('Items')
ax.set_ylabel('Runtime')
ax.set_title("Interpolation search runtime")
```

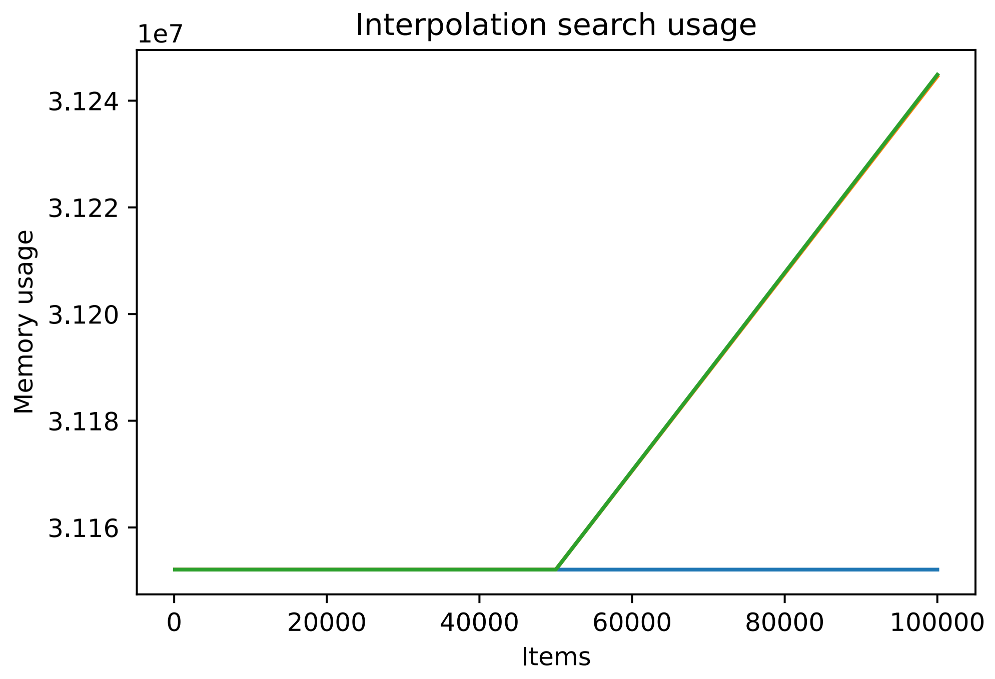
Out[24]:

```
Text(0.5, 1.0, 'Interpolation search runtime')
```



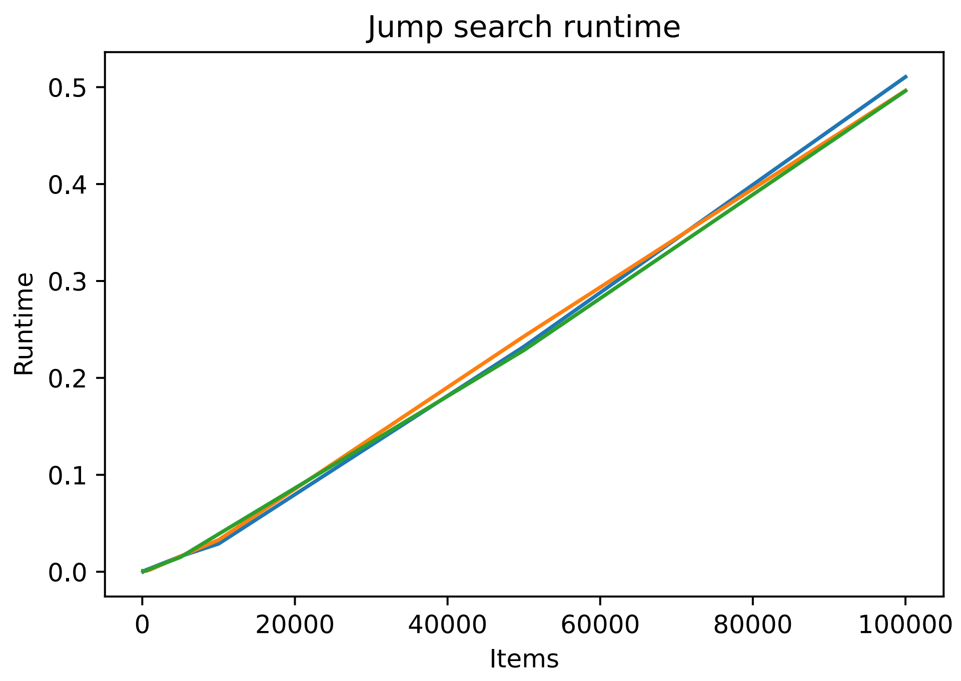
In [25]:
```
fig, ax = plt.subplots()
ax.plot(seqRange, best_inter_usage, label='Best')
ax.plot(seqRange, average_inter_usage, label='Average')
ax.plot(seqRange, worst_inter_usage, label='Worst')
ax.set_xlabel('Items')
ax.set_ylabel('Memory usage')
ax.set_title("Interpolation search usage")
```

Out[25]:
```
Text(0.5, 1.0, 'Interpolation search usage')
```



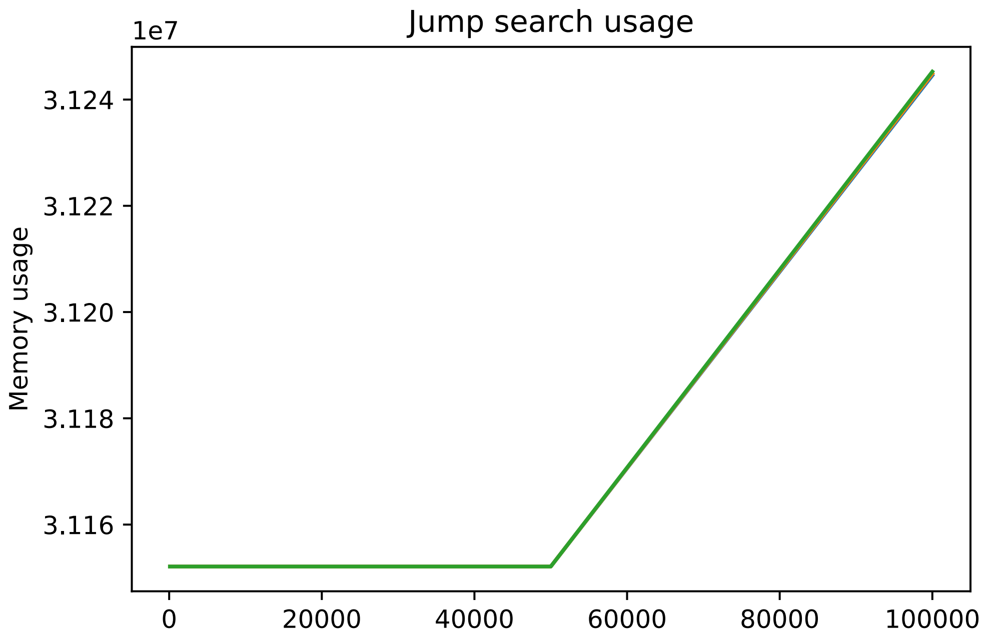
In [28]:
```python
fig, ax = plt.subplots()
ax.plot(seqRange, best_jump_runtime, label='Best')
ax.plot(seqRange, average_jump_runtime, label='Average')
ax.plot(seqRange, worst_jump_runtime, label='Worst')
ax.set_xlabel('Items')
ax.set_ylabel('Runtime')
ax.set_title("Jump search runtime")
```

Out[28]:
```
Text(0.5, 1.0, 'Jump search runtime')
```



In [29]:
```python
fig, ax = plt.subplots()
ax.plot(seqRange, best_jump_usage, label='Best')
ax.plot(seqRange, average_jump_usage, label='Average')
ax.plot(seqRange, worst_jump_usage, label='Worst')
ax.set_xlabel('Items')
ax.set_ylabel('Memory usage')
ax.set_title("Jump search usage")
```

Out[29]:
```
Text(0.5, 1.0, 'Jump search usage')
```


Jump search usage
1e7
Memory usage
3.124
3.122
3.120
3.118
3.116
0
20000
40000
60000
80000
100000

In [ ]: