Python

```python
arr = [12, 3, 5, 7, 19]
k = 2
def select(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k]
    medians = [sorted(arr[i:i + 5])[len(arr[i:i + 5]) // 2] for i in rang
    pivot = select(medians, len(medians) // 2)
    lows = [x for x in arr if x < pivot]
    highs = [x for x in arr if x > pivot]
    pivots = [x for x in arr if x == pivot]
    if k < len(lows):
        return select(lows, k)
    elif k < len(lows) + len(pivots):
        return pivot
    else:
        return select(highs, k - len(lows) - len(pivots))
print(select(arr, k - 1))
```

Input Goes Here...

Copy

> Run

> Run+URL (Generates URL as well)

Time(sec) : 0.012 Memory(MB) : 6.171875

Output:

Copy

```
5
```

Python

```python
def partition(arr, pivot):
    low = [x for x in arr if x < pivot]
    high = [x for x in arr if x > pivot]
    pivots = [x for x in arr if x == pivot]
    return low, pivots, high
def find_median(arr):
    arr.sort()
    return arr[len(arr) // 2]
def median_of_medians(arr, k):
    if len(arr) <= 5:
        arr.sort()
        return arr[k-1]
    sublists = [arr[i:i+5] for i in range(0, len(arr), 5)]
    medians = [find_median(sublist) for sublist in sublists]
    pivot = median_of_medians(medians, len(medians) // 2 + 1)
    low, pivots, high = partition(arr, pivot)
    if k <= len(low):
        return median_of_medians(low, k)
    elif k <= len(low) + len(pivots):
        return pivots[0]
    else:
        return median_of_medians(high, k - len(low) - len(pivots))
arr1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
k1 = 6
print(partition(arr1,k1))
```

Input Goes Here..

Copy

> Run

> Run+URL (Generates URL as well)

Time(sec) : 0.009

Memory(MB) : 6.265625

Output:

Copy

```
([1, 2, 3, 4, 5], [6], [7, 8, 9, 10])
```

Python

```python
def karatsuba(x, y):
    if x < 10 or y < 10:
        return x * y
    n = max(len(str(x)), len(str(y)))
    m = n // 2
    high_x, low_x = divmod(x, 10**m)
    high_y, low_y = divmod(y, 10**m)
    P1 = karatsuba(high_x, high_y)
    P2 = karatsuba(low_x, low_y)
    P3 = karatsuba(high_x + low_x, high_y + low_y)
    return P1 * 10**(2*m) + (P3 - P1 - P2) * 10**m + P2
x = 1234
y = 5678
z = karatsuba(x, y)
print(z)

```

Input Goes Here..

Copy

> Run

> Run+URL (Generates URL as well)

Time(sec) : 0.005

Memory(MB) : 6.19140625

Output:

Copy

```
7006652
```

Python

```python
def strassen_2x2(A, B):
    a, b = A[0][0], A[0][1]
    c, d = A[1][0], A[1][1]
    e, f = B[0][0], B[0][1]
    g, h = B[1][0], B[1][1]
    P1 = a * (f - h)
    P2 = (a + b) * h
    P3 = (c + d) * e
    P4 = d * (g - e)
    P5 = (a + d) * (e + h)
    P6 = (b - d) * (g + h)
    P7 = (a - c) * (e + f)
    C11 = P5 + P4 - P2 + P6
    C12 = P1 + P2
    C21 = P3 + P4
    C22 = P1 + P5 - P3 - P7
    C = [[C11, C12], [C21, C22]]
    return C
A = [[1, 7], [4, 2]]
B = [[6, 8], [3, 5]]
C = strassen_2x2(A, B)
print("Resulting matrix C:")
for row in C:
    print(row)

```

Input Goes Here..

Copy

> Run

> Run+URL (Generates URL as well)

Time(sec) : 0.008 Memory(MB) : 6.25

Output:

Copy

```
Resulting matrix C:
[27, 43]
[30, 42]
```

Python

```python
from itertools import combinations
def subset_max_sum(l, r, s):
    largest_subset = None
    for m in range(1, len(l) + 1):
        for sets in combinations(l, m):
            if sum(sets) == s:
                if largest_subset is None or len(sets) > len(largest_subs
                    largest_subset = sets
    for p in range(1, len(r) + 1):
        for subset in combinations(r, p):
            if sum(subset) == s:
                if largest_subset is None or len(subset) > len(largest_su
                    largest_subset = subset
    return largest_subset
a = [1, 1, 1, 2, 3, 4, 1, 6]
s = 5
n = len(a)
l = a[:n//2]
r = a[n//2:]
print(subset_max_sum(l, r, s))

```

Input Goes Here..

Copy

> Run

> Run+URL (Generates URL as well)

Time(sec) : 0.011

Memory(MB) : 6.3203125

Output:

Copy

```
(1, 1, 1, 2)
```

Python Online Compiler

IDE

Sign In

C
C++
C++14
C#
Java
Perl
PHP
Python
Python 3
Scala
Swift
Rust
Golang
R
Node JS
HTML & JS

Python

```python
from itertools import combinations
a = [1, 2, 3,1, 4, 5, 6,2, 7,3]
s = 10
n = len(a)
l = a[:n//2]
r = a[n//2:]
for m in range(len(l) + 1):
    for sets in combinations(l, m):
        if sum(sets) == s:
            print("Subsets from left:", sets)
for p in range(len(r) + 1):
    for subset in combinations(r, p):
        if sum(subset) == s:
            print("Subsets from right:", subset)
```

Input Goes Here..

Copy

> Run

> Run+URL (Generates URL as well)

Time(sec) : 0.011

Memory(MB) : 6.25390625

Output:

Copy

```
('Subsets from left:', (1, 2, 3, 4))
('Subsets from left:', (2, 3, 1, 4))
('Subsets from right:', (7, 3))
('Subsets from right:', (5, 2, 3))
```