**מציאת תת-מערך עם סכום מקסימלי (קצר ביותר)**

בהינתן מערך של מסמפרים. יש למצוא רצף התאים שסכום שלהם גדול ביותר במערך.

The **maximum subarray problem** is the task of finding the contiguous subarray within a one-dimensional array of numbers which has the **largest sum**.

For example, for the sequence of values

−2, 1, −3, **4, −1, 2, 1**, −5, 4

the contiguous subarray with the largest sum is

**4, −1, 2, 1,**

with sum **6**.

**Sum = 6**

**Problem's extension:**

Find not only **max** **sum** but also **first & last index** and **length** of subarray with the max sum

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **index** | 0 | 1 | 2 | **3** | **4** | **5** | **6** | 7 | 8 |
| **array** | -2 | 1 | -3 | **4** | **-1** | **2** | **1** | -5 | 4 |

**Sum = 6**

**First index = 3**

**Last index = 6**

**Length = 4**

**An extreme cases:**

1. All elements of the array are negative

**arr[] = {-2, -8, -1, -5, -2}**

**The maximum sum is equal to the maximum element of the array:**

**sum = max(-2, -8, -1, -5, -2) = -1**

1. All elements of the array are positive

**arr[] = {2, 8, 1, 5, 2}**

**The maximum sum is equal to the sum of all array elements:**

**sum = 2 + 8 + 1 + 5 + 2 = 18**

**Algorithms:**

**Greedy algorithm doesn't work.**

1. **Exhaustive Search**
   1. **Algorithm with complexity O(n³) - full search**

**arr[] = {1, 10, -15, 3, -10}**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **index** | | | | |  | **sum** |
| **0** | **1** | **2** | **3** | **4** |  |  |
| **1** |  |  |  |  | **1** |
| **1** | **10** |  |  |  | **11** |
| **1** | **10** | **-15** |  |  | **-4** |
| **1** | **10** | **-15** | **3** |  | **-1** |
| **1** | **10** | **-15** | **3** | **-10** | **-11** |
|  | **10** |  |  |  | **10** |
|  | **10** | **-15** |  |  | **-5** |
|  | **10** | **-15** | **3** |  | **-2** |
|  | **10** | **-15** | **3** | **-10** | **-12** |
|  |  | **-15** |  |  | **-15** |
|  |  | **-15** | **3** |  | **-12** |
|  |  | **-15** | **3** | **-10** | **-22** |
|  |  |  | **3** |  | **3** |
|  |  |  | **3** | **-10** | **-7** |
|  |  |  |  | **-10** | **-10** |

**int** max = arr[0];

System.*out*.println("arr : " + Arrays.*toString*(arr));

**for** (**int** i = 0; i < arr.length; i++){

**for** (**int** j = i; j < arr.length; j++){

**int** sum = 0;

**for** (**int** k = i; k <= j; k++)

sum = sum + arr[k];

max = **Math.*max***(max, sum);

System.out.println("i = " + i + " j = " + j + " sum = " + sum + " max = " + max);

}

}

System.*out*.println("max = " + max + "\n");

**arr: [1, 10, -15, 3, -10]**

**max = 11**

arr : [1, 10, -15, 3, -10]

i = 0 j = 0 sum = 1 max = 1

**i = 0 j = 1 sum = 11 max = 11**

i = 0 j = 2 sum = -4 max = 11

i = 0 j = 3 sum = -1 max = 11

i = 0 j = 4 sum = -11 max = 11

i = 1 j = 1 sum = 10 max = 11

i = 1 j = 2 sum = -5 max = 11

i = 1 j = 3 sum = -2 max = 11

i = 1 j = 4 sum = -12 max = 11

i = 2 j = 2 sum = -15 max = 11

i = 2 j = 3 sum = -12 max = 11

i = 2 j = 4 sum = -22 max = 11

i = 3 j = 3 sum = 3 max = 11

i = 3 j = 4 sum = -7 max = 11

i = 4 j = 4 sum = -10 max = 11

max = 11

**Complete solution:**

Find **max** **sum,** **first index, last index,** **length** of subarray with the max sum.

**int** max = arr[0];

**int firstIndex = 0;**

**int lastIndex = 0;**

**int length = 0;**

System.*out*.println("arr : " + Arrays.*toString*(arr));

**for** (**int** i = 0; i < arr.length; i++){

**for** (**int** j = i; j < arr.length; j++){

**int** sum = 0;

**for** (**int** k = i; k <= j; k++)

sum = sum + arr[k];

max = Math.*max*(max, sum);

**if (sum == max){**

**firstIndex = i;**

**lastIndex = j;**

**}**

}

}

**length = lastIndex + 1 - firstIndex;**

System.*out*.println("max = " + max + " firstIndex = " + firstIndex +

" lastIndex = " + lastIndex + " length = " + length + "\n");

}

* 1. **Algorithm with complexity O(n²) - full search**

**complexity O(n³)**

**for** (**int** i = 0; i < arr.length; i++){

**for** (**int** j = i; j < arr.length; j++){

**int** sum = 0;

**for** (**int** k = i; k <= j; k++){

sum = sum + arr[k];

}

max = **Math.*max***(max, sum);

}

}

**---------------------------------------------------------------------------**

arr: [1, 10, -15, 3, -10]

**System.*out*.println sum**

i = 0 j = 0 k = 0 sum = 1 sum = arr[0]

i = 0 j = 1 k = 0 sum = 1 sum = arr[0]

i = 0 j = 1 k = 1 sum = 11 sum = arr[0] + arr[1]

i = 0 j = 2 k = 0 sum = 1 sum = arr[0]

i = 0 j = 2 k = 1 sum = 11 sum = arr[0] + arr[1]

i = 0 j = 2 k = 2 sum = -4 sum = arr[0] + arr[1] + arr[2]

i = 0 j = 3 k = 0 sum = 1 sum = arr[0]

i = 0 j = 3 k = 1 sum = 11 sum = arr[0] + arr[1]

i = 0 j = 3 k = 2 sum = -4 sum = arr[0] + arr[1] + arr[2]

i = 0 j = 3 k = 3 sum = -1 sum = arr[0] + arr[1] + arr[2] + arr[3]

i = 0 j = 4 k = 0 sum = 1 sum = arr[0]

i = 0 j = 4 k = 1 sum = 11 sum = arr[0] + arr[1]

i = 0 j = 4 k = 2 sum = -4 sum = arr[0] + arr[1] + arr[2]

i = 0 j = 4 k = 3 sum = -1 sum = arr[0] + arr[1] + arr[2] + arr[3]

i = 0 j = 4 k = 4 sum = -11 sum = arr[0] + arr[1] + arr[2] + arr[3] + arr[4]

---------------------------------------------------------

i = 1 j = 1 k = 1 sum = 10 sum = arr[1]

i = 1 j = 2 k = 1 sum = 10 sum = arr[1]

i = 1 j = 2 k = 2 sum = -5 sum = arr[1] + arr[2]

i = 1 j = 3 k = 1 sum = 10 sum = arr[1]

i = 1 j = 3 k = 2 sum = -5 sum = arr[1] + arr[2]

i = 1 j = 3 k = 3 sum = -2 sum = arr[1] + arr[2] + arr[3]

i = 1 j = 4 k = 1 sum = 10 sum = arr[1]

i = 1 j = 4 k = 2 sum = -5 sum = arr[1] + arr[2]

i = 1 j = 4 k = 3 sum = -2 sum = arr[1] + arr[2] + arr[3]

i = 1 j = 4 k = 4 sum = -12 sum = arr[1] + arr[2] + arr[3] + arr[4]

---------------------------------------------------------

i = 2 j = 2 k = 2 sum = -15 sum = arr[2]

i = 2 j = 3 k = 2 sum = -15 sum = arr[2]

i = 2 j = 3 k = 3 sum = -12 sum = arr[2] + arr[3]

i = 2 j = 4 k = 2 sum = -15 sum = arr[2]

i = 2 j = 4 k = 3 sum = -12 sum = arr[2] + arr[3]

i = 2 j = 4 k = 4 sum = -22 sum = arr[2] + arr[3] + arr[4]

---------------------------------------------------------

i = 3 j = 3 k = 3 sum = 3 sum = arr[3]

i = 3 j = 4 k = 3 sum = 3 sum = arr[3]

i = 3 j = 4 k = 4 sum = -7 sum = arr[3] + arr[4]

---------------------------------------------------------

i = 4 j = 4 k = 4 sum = -10 sum = arr[3]

**max = 11**

**complexity O() →complexity O()**

**for** (**int** i = 0; i < arr.length; i++){

**for** (**int** j = i; j < arr.length; j++){

**int** sum = 0;

**for** (**int** k = i; k <= j; k++){

sum = sum + arr[j];

}

max = **Math.*max***(max, sum);

}

}

**---------------------------------------------------------------------------**

**int** max = arr[0];

**for** (**int** i = 0; i < arr.length; i++){

**int** sum = 0;

**for** (**int** j = i; j < arr.length; j++){

sum = sum + arr[j];

**// sum - the sum of all items x[i] + ... + x[j]**

max = Math.*max*(max, sum);

**System.*out*.println("i = " + i + " j = " + j + " sum = " + sum + " max = " + max);**

}

}

System.*out*.println("max = " + max);

}

Result:

**arr : [1, 10, -15, 3, -10]**

**max = 11**

1. **Dynamic programming using matrix**
   1. **The first option (top down) - complexity O()**

**int** arr[] = {1, 10, -15, 3, -10};

**int** mat[][]:

**1**, 0, 0, 0, 0

0, **10**, 0, 0, 0

0, 0, **-15**, 0, 0

0, 0, 0, **3**, 0

0, 0, 0, 0, **-10**

**The build of the matrix:**

**for** (**int** i = 0; i < n; i++)

**for** (**int** j = i+1; j < n; j++)

**mat[i][j] = mat[i][j-1] + mat[j][j]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 | 11 | -4 | -1 | -11 | 0 |
|  | 10 |  |  |  | 1 |
|  |  | -15 |  |  | 2 |
|  |  |  | 3 |  | 3 |
|  |  |  |  | -10 | 4 |

mat[i][j] = mat[i][j-1] + mat[j][j]

**i=0**

**j=i+1=1**

mat[0][1] = mat[0][0] + mat[1][1] = 1 + 10 = 11

**j=2**

mat[0][2] = mat[0][1] + mat[2][2] = 11 - 15 = -4

**j=3**

mat[0][3] = mat[0][2] + mat[3][3] = -4 + 3 = -1

**j=4**

mat[0][4] = mat[0][3] + mat[4][4] = -1 - 10 = -11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 | 11 | -4 | -1 | -11 | 0 |
|  | 10 | -5 | -2 | -12 | 1 |
|  |  | -15 |  |  | 2 |
|  |  |  | 3 |  | 3 |
|  |  |  |  | -10 | 4 |

mat[i][j] = mat[i][j-1] + mat[j][j]

**i=1**

**j=i+1=2**

mat[1][2] = mat[1][1] + mat[2][2] = 10 - 15 = -5

**j=3**

mat[1][3] = mat[1][2] + mat[3][3] = -5 + 3 = -2

**j=4**

mat[1][4] = mat[1][3] + mat[4][4] = -2 - 10 = -12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 | 11 | -4 | -1 | -11 | 0 |
|  | 10 | -5 | -2 | -12 | 1 |
|  |  | -15 | -12 | -22 | 2 |
|  |  |  | 3 |  | 3 |
|  |  |  |  | -10 | 4 |

mat[i][j] = mat[i][j-1] + mat[j][j]

**i=2**

**j=i+1=3**

mat[2][3] = mat[2][2] + mat[3][3] = -15 + 3 = -12

**j=4**

mat[2][4] = mat[2][3] + mat[4][4] = -12 - 10 = -22

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 | **11** | -4 | -1 | -11 | 0 |
|  | 10 | -5 | -2 | -12 | 1 |
|  |  | -15 | -12 | -22 | 2 |
|  |  |  | 3 | -7 | 3 |
|  |  |  |  | -10 | 4 |

mat[i][j] = mat[i][j-1] + mat[j][j]

**i=3**

**j=i+1=3**

mat[3][4] = mat[3][3] + mat[4][4] = 3 - 10 = -7

**sum = MAX(mat[i][j]) = 11**

* 1. **The second option (bottom up) - complexity O()**

**int** arr[] = {1, 10, -15, 3, -10};

**int** [][]mat:

**1**, 0, 0, 0, 0

0, **10**, 0, 0, 0

0, 0, **-15**, 0, 0

0, 0, 0, **3**, 0

0, 0, 0, 0, **-10**

**The build of the matrix:**

**for** (**int** i = n-1; i >= 0; i--)

**for** (**int** j = i+1; j < n; j++)

**mat[i][j] = mat[i+1][j] + mat[i][i]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 |  |  |  |  | 0 |
|  | 10 |  |  |  | 1 |
|  |  | -15 |  |  | 2 |
|  |  |  | 3 | -7 | 3 |
|  |  |  |  | -10 | 4 |

**mat[i][j] = mat[i+1][j] + mat[i][i]**

**i=n-1=4-1=3**

**j=i+1=3+1=4**

mat[3][4] = mat[4][4] + mat[3][3] = -10 + 3 = -7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 |  |  |  |  | 0 |
|  | 10 |  |  |  | 1 |
|  |  | -15 | -12 | -22 | 2 |
|  |  |  | 3 | -7 | 3 |
|  |  |  |  | -10 | 4 |

**mat[i][j] = mat[i+1][j] + mat[i][i]**

**i=2**

**j=i+1=3**

mat[2][3] = mat[3][3] + mat[2][2] = 3 - 15 = -12

**j=4**

mat[2][4] = mat[3][4] + mat[2][2] = -7 - 15 = -22

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 |  |  |  |  | 0 |
|  | 10 | -5 | -2 | -12 | 1 |
|  |  | -15 | -12 | -22 | 2 |
|  |  |  | 3 | -7 | 3 |
|  |  |  |  | -10 | 4 |

**mat[i][j] = mat[i+1][j] + mat[i][i]**

**i=1**

**j=i+1=2**

mat[1][2] = mat[2][2] + mat[1][1] = -15 +10 = -5

**j=3**

mat[1][3] = mat[2][3] + mat[1][1] = -12 +10 = -2

**j=4**

mat[1][4] = mat[2][4] + mat[1][1] = -22 +10 = -12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |  |
| 1 | **11** | -4 | -1 | -11 | 0 |
|  | 10 | -5 | -2 | -12 | 1 |
|  |  | -15 | -12 | -22 | 2 |
|  |  |  | 3 | -7 | 3 |
|  |  |  |  | -10 | 4 |

**mat[i][j] = mat[i+1][j] + mat[i][i]**

**i=0**

**j=i+1=1**

mat[0][1] = mat[1][1] + mat[0][0] = 10 +1 = 11

**j=2**

mat[0][2] = mat[1][2] + mat[0][0] = -5 +1 = -4

**j=3**

mat[0][3] = mat[1][3] + mat[0][0] = -2 +1 = -1

**j=4**

mat[0][4] = mat[1][4] + mat[0][0] = -12 +1 = -11

**sum = MAX(mat[i][j]) = 11**

**Complete solution:**

Find **max** **sum,** **first index, last index,** **length** of subarray with the max sum.

**//O(N^2) - bottom up: mat[i][j] = mat[i+1][j] + arr[i]**

**int** n = arr.length;

**int** [][]mat = **new** **int**[n][n];

**int** sumMax=a[0], beginMax=0, endMax=0, length=0;

**for** (**int** i=0; i<n; i++){

mat[i][i] = arr [i];

**if** (sumMax < arr [i]){

sumMax = arr [i];

beginMax = endMax = i;

}

}

**for** (**int** i=n-1; i>=0; i--){

**for** (**int** j=i+1; j<n; j++){

mat[i][j] = mat[i+1][j] + arr [i];

**if** (sumMax < mat[i][j]){

sumMax = mat[i][j];

beginMax = i;

endMax = j;

}

}

}

length = endMax +1 - beginMax;

System.***out***.println("sumMax = "+sumMax+", interval: ["+beginMax + ","+endMax+"], length: "+length);