Size of The Subarray With Maximum Sum

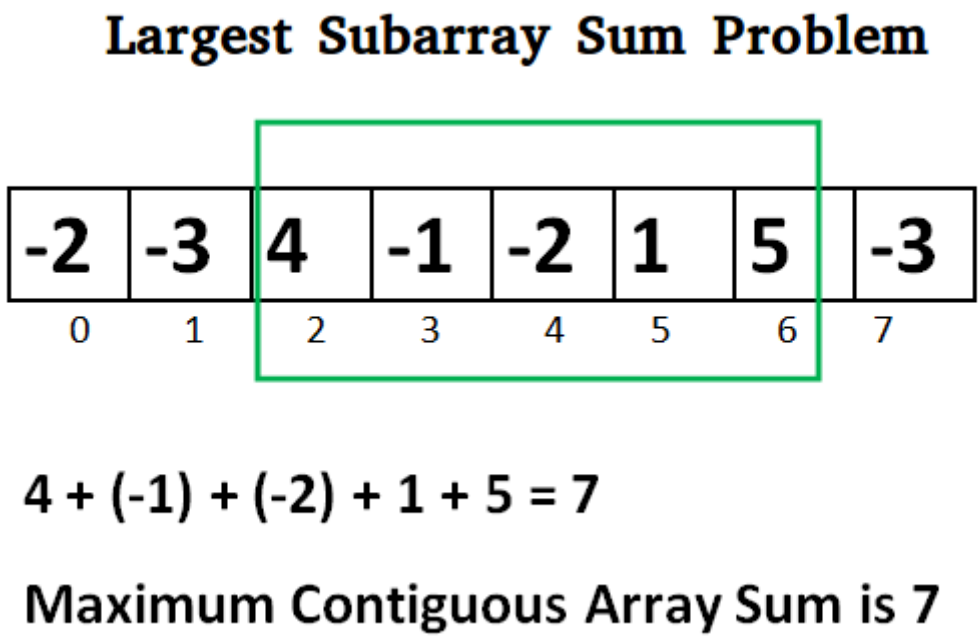
> 求元素和最大的子数组，子数组中的元素在位置上必须是连续的

O(n^2)

```
#include <bits/stdc++.h>

using namespace std;

int maxLenOfSubArr(vector<int> &arr, int &N) {
    vector< vector<int>> dp(N, vector<int>(N));
    dp[0][0] = arr[0];
    int maxSum = dp[0][0], maxLen = 1;
    for (int i = 0; i < N; ++i) {
        for (int j = i + 1; j < N; ++j) {
            dp[i][j] = dp[i][j - 1] + arr[j];
            if (dp[i][j] > maxSum) {
                maxSum = dp[i][j];
                maxLen = j - i;
            }
        }
    }
    return maxLen;
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N;
        scanf("%d", &N);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i)
            scanf("%d", &arr[i]);
        printf("%d", maxLenOfSubArr(arr, N));
    }
    return 0;

}
```

Largest Sum Contiguous Subarray

> Size of The Subarray With Maximum Sum问题是Largest Sum Contiguous Subarray的扩展，Largest Sum Contiguous Subarray解决下图之类的问题的高效方法



解法如下（Kadane's Algorithm），基本思路如下：

kadane算法利用了数学归纳法的思想，考虑数组第i+1个元素arr[i + 1]，数组到i+1位置的最大连续子序列和，分为两种情况

1. 包含arr[i + 1]，即maxSubArrSum = max(arr[i + 1]，以arr[i]结尾的最大连续子列和 + arr[i + 1])（因为要求是连续的，所以只有这两种情况max(arr[i + 1]，以arr[i]结尾的最大连续子列和 + arr[i + 1])）
2. 不含arr[i + 1]，这一步也可以通过上一步求出来

```
#include <bits/stdc++.h>

using namespace std;

int maxSubArraySum(vector<int> &arr, int &N) {
    int maxSoFar = arr[0], maxEndingHere = arr[0];
    for (int i = 1; i < N; ++i) {
        maxEndingHere = max(arr[i], maxEndingHere + arr[i]);
        maxSoFar = max(maxSoFar, maxEndingHere);
    }
    return maxSoFar;
}
```

```
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N;
        scanf("%d", &N);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i)
            scanf("%d", &arr[i]);
        printf("%d\n", maxSubArraySum(arr, N));
    }
    return 0;

}
```

Size of The Subarray With Maximum Sum

```
#include <bits/stdc++.h>

using namespace std;

int maxSubArraySum(vector<int> &arr, int &N) {
    int maxSoFar = arr[0], maxEndingHere = arr[0];
    int start = 0, end = 0, s = 0;
    for (int i = 1; i < N; ++i) {
        maxEndingHere += arr[i];
        if (maxSoFar < maxEndingHere) {
            maxSoFar = maxEndingHere;
            start = s;
            end = i ;
        }
        if (maxEndingHere < 0) {
            maxEndingHere = 0;
            s = i + 1;
        }
    }
    return end - start + 1;
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N;
        scanf("%d", &N);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i)
            scanf("%d", &arr[i]);
        printf("%d\n", maxSubArraySum(arr, N));
    }
    return 0;

}
```