Gąsienica

- **złożoność obliczeniowa:** O(2n)
- algorytm przeszukiwania ciągów w celu znalezienia danych podciągów o pewnych właściwościach
- wyróżnia się dwa typy/sposoby użycia algorytmu gąsienicy

1. zliczenie ilość podciągów o pewnych własnościach

PROBLEM

Dany jest ciąg liczb naturalnych o długości **n** i liczba **k**. Oblicz liczbę podciągów, takich że suma ich wszystkich elementów jest równa dokładnie **k**.

IMPLEMENTACJA

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

using namespace std;

int n, k;
int a[1000007]{};

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    cin >> n >> k;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }

    int x = 1, y = 1;
    int suma = a[1];
    int wynik = 0;

    while(y <= n){
        if(suma < k){
            y++;
            suma += a[y];
        } else if(suma > k){
            suma -= a[x];
            x++;
        } else{
            wynik++;
```

```
            suma = suma - a[x] + a[y + 1];
            x++;
            y++;
        }
    }
    cout << wynik;
}
```

2. znalezienie największego/najmniejszego podciągu o pewnych właściowściach

PROBLEM

Dany jest ciąg liczb naturalnych o długości **n** i liczba **k**. Oblicz długość najdłuższego podciągu, którego suma wszystkich elementów jest nie większa niż **k**.

IMPLEMENTACJA

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

using namespace std;

int n, k;
int a[1000007]{};

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    cin >> n >> k;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }

    int wynik = 0;
    int sum = 0;
    for(int l = 0, p = 0; l < n; l++){
        while(sum <= k){
            wynik = max(wynik, p - l);
            if(p == n) break;
            sum += a[p++];
        }
        sum -= a[l];
    }
    cout << wynik;
}
```