... albo psikus! (rozwiązanie)

Autor zadania: **Bartosz Kostka**
Opracowanie: **Alicja Kluczek, Marek Skiba**
Opis rozwiązania: **Alicja Kluczek**

Olimpiada Informatyczna Juniorów

Nazwijmy *przedziałem* dowolny spójny fragment domków. W zadaniu musimy zliczyć liczbę takich przedziałów, że suma cukierków ze wszystkich domków na przedziale jest parzysta.

Na początek rozważmy dość powolne rozwiązanie – dla każdej pary (L, R) stanowiącej początek i koniec przedziału obliczmy, jaka jest w nim suma cukierków. Jeśli będzie parzysta, to taką parę wliczymy do wyniku.

psi_powolne.cpp

```cpp
// W zmiennej wynik trzymamy informację ile znaleźliśmy par (L, R) o parzystej sumie
// Uwaga!  Par (L, R) może być więcej, niż mieści typ int
long long wynik = 0;
for (int L = 0; L < N; L++) {
    for (int R = L; R < N; R++) {
        // Badamy, czy suma na przedziale [L, R] jest parzysta
        long long suma = 0;
        for (int i = L; i <= R; i++) {
            suma += cukierki[i];
        }
        if (suma % 2 == 0) {
            wynik++;
        }
    }
}
```

psi_powolne.py

```python
# W zmiennej wynik trzymamy informację ile znaleźliśmy par (L, R) o parzystej sumie
wynik = 0
for L in range(N):
    for R in range(i, N):
        # Badamy, czy suma na przedziale [L, R] jest parzysta
        suma = 0
        for i in range(L, R+1):
            suma += cukierki[i]
        if suma % 2 == 0:
            wynik += 1
```

Złożoność tego rozwiązania to $O(N^3)$. Choć powolne, posłuży jako podstawa rozwiązania wzorcowego.

Spróbujmy teraz je przyspieszyć. Powtarzającą się operacją jest zliczanie sumy na przedziale – często przechodzimy po elementach, które w przeszłości już dodawaliśmy. W szczególności, jeśli mamy ustalony początek przedziału L, to niepotrzebnie sumujemy wielokrotnie te same wartości, gdy R zwiększa się o 1. Skorzystajmy z faktu, że znając sumę elementów na przedziale $[L, R]$, potrafimy obliczyć sumę na przedziale $[L, R+1]$ w czasie stałym – dodając do tej sumy element o indeksie $R+1$.

psi_z_optymalizacja.cpp

```cpp
long long wynik = 0;
for (int L = 0; L < N; L++) {
    // Tym razem spamiętujemy sumę wcześniejszych elementów
    long long suma = 0;
    for (int R = L; R < N; R++) {
        // Dodajemy tylko nowy element do wcześniej policzonej sumy
        suma += cukierki[R];
        if (suma % 2 == 0) {
            wynik++;
        }
    }
}
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

psi_z_optymalizacja.py

```
wynik = 0
for L in range(N):
    # Tym razem spamiętujemy sumę wcześniejszych elementów
    suma = 0
    for R in range(i, N):
        # Dodajemy tylko nowy element do wcześniej policzonej sumy
        suma += cukierki[R]
        if suma % 2 == 0:
            wynik += 1
```

Z tą optymalizacją nasze rozwiązanie ma złożoność $O(N^2)$ i otrzymuje 40 punktów.

Do poprawienia rozwiązania tak, żeby zdobyło 100 punktów, musimy poczynić jeszcze jedną obserwację. Niech $suma_R$ to suma wszystkich cukierków od domku numer 1 aż do R-tego – tak zwana *suma prefiksowa*. Zauważmy jednak, że suma przedziału $[L, R]$ to nic innego, jak różnica $suma_R - suma_{L-1}$ (dla wygody przyjmiemy, że dla $L = 1$ ta wartość to 0). Co więcej, wartości $suma_i$ dla wszystkich $i = 1, 2, \ldots N$ możemy wyliczyć tylko raz, na samym początku algorytmu, w czasie $O(N)$ – wtedy będziemy mogli policzyć sumę na dowolnym przedziale $[L, R]$ w czasie stałym.

Kiedy zatem suma na przedziale $[L, R]$ jest parzysta? Zdarzy się to tylko w dwóch przypadkach: $suma_{L-1}$ i $suma_R$ muszą być albo naraz nieparzyste, albo naraz parzyste. W takim razie dla każdego R wystarczy tylko pamiętać, ile było wcześniej wartości $suma$ o takiej samej parzystości, czyli ile jest możliwych $L < R$ takich, że $suma_{L-1}$ daje tę samą parzystość co $suma_R$. Krótko mówiąc, musimy tylko pamiętać, ile spośród wartości $suma$ było dotychczas parzystych, a ile nieparzystych.

W ten sposób złożoność poprawia się z $O(N^2)$ do $O(N)$, co jest konieczne do uzyskania 100 punktów.

psi_wzorcowe.cpp

```
#include <iostream>
using namespace std;

int main() {
    int N;
    cin >> N;
    int cukierki[N];
    for (int i = 0; i < N; i++) {
        cin >> cukierki[i];
    }

    long long wynik = 0;
    int parz = 1; // Ile jest takich domków, ze suma cukierków ze wszystkich domków
                  // na lewo od niego jest parzysta (w szczególności traktujemy
                  // zero domków jako sume parzysta)
    int nieparz = 0; // Podobnie tylko suma nieparzysta

    long long suma = 0; // Suma cukierków we wszystkich rozważonych dotąd domkach

    for (int i = 0; i < N; i++) {
        suma += cukierki[i];
        if (suma % 2 == 0) { // Jeśli suma wszystkich cukierków dotąd jest parzysta
            wynik += parz; // Dodajemy do wyniku wszystkie wystąpienia sum parzystych
            parz++; // Wystąpila kolejna suma parzysta, aktualizujemy licznik
        } else { // Jeśli suma wszystkich cukierków dotąd jest nieparzysta, analogicznie
            wynik += nieparz;
            nieparz++;
        }
    }

    cout << wynik << "\n";
}
```

psi_wzorcowe.py

```
def main():
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
    N = int(input())
    s = input()
    cukierki = list(map(int, s.split()))

    parz = 1 # Ile jest takich domków, ze suma cukierków ze wszystkich domków
            # na lewo od niego jest parzysta (w szczególnosci traktujemy
            # zero domków jako sumę parzystą)
    nieparz = 0 # Podobnie tylko suma nieparzysta

    suma = 0 # Suma cukierków we wszystkich rozważonych dotąd domkach
    wynik = 0

    for cukierek in cukierki:
        suma += cukierek
        if suma % 2 == 0: # Jeśli suma wszystkich cukierków dotąd jest parzysta
            wynik += parz # Dodajemy do wyniku wszystkie wystąpienia sum parzystych
            parz += 1 # Wystąpiła kolejna suma parzysta, aktualizujemy licznik
        else: # Jesli suma wszystkich cukierkow dotąd jest nieparzysta, analogicznie
            wynik += nieparz
            nieparz += 1

    print(wynik)

main()
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".