

## Sprawozdanie 2

Zadanie 10. Liczbę naturalną  $C$  można przedstawić jako sumę parami różnych liczb naturalnych. Na przykład jeśli  $C = 6$ , to możemy  $C$  przedstawić na cztery sposoby:

$$1 + 2 + 3$$

$$1 + 5$$

$$2 + 4$$

$$6$$

a jeśli  $C = 10$ , to takimi podziałami są:

$$1 + 2 + 3 + 4$$

$$1 + 2 + 7$$

$$1 + 3 + 6$$

$$1 + 4 + 5$$

$$1 + 9$$

$$2 + 3 + 5$$

$$2 + 8$$

$$3 + 7$$

$$4 + 6$$

$$10$$

Skonstruuj algorytm wyczerpujący z nawrotami, generujący wszystkie podziały podanej liczby naturalnej  $C$ .

Adrian Rupala

14 maja 2018

# Spis treści

1	Teoria	1
2	Rozwiązanie	1

# 1. Teoria

Algorytm z nawrotami to algorytm wyszukiwania wszystkich lub kilku rozwiązań. Polega on na znajdowaniu wyniku metodą „prób i błędów”, wszelako z oznaczeniem niepowodzeń, dzięki czemu te same błędy nie są popełniane dwukrotnie.

Jeżeli problem pozwala na zastosowanie algorytmu wyszukiwania z nawrotami, to metoda ta może być znaczenie efektywniejsza niż wyszukiwanie wyczerpujące (zakładające przeszukiwanie wszystkich rozwiązań), ponieważ pojedynczy test może wyeliminować nie jedno, a wiele rozwiązań niedopuszczalnych.

Rekurencja to technika programowania, dzięki której funkcja, procedura lub podprogram jest w stanie w swoim ciele wywołać sama siebie. Pozwala ona łatwo wykonać wiele zadań, w których zachodzi potrzeba obliczenia wyników częściowych do obliczenia całości.

# 2. Rozwiązanie

Oto pseudokod przedstawiający rozwiązanie problemu przedstawionego w zadaniu.

---

```
bool znajdz_duplikat(int tablica[], int rozmiar_tablicy) {
    sort(tablica);
    for (int i = 1; i < rozmiar_tablicy - 1; i++) {
        if (tablica[i] == tablica[i + 1]){
            return true;
        }
    }
    return false;
}

void sprawdz_i_wypisz(int pozycja, int pozostalo) {
    if (pozostalo == 0) {
        for (int i = 1; i <= pozycja - 1; i++) {
            cout << tablica[i] << " + ";
        }
        cout << endl;
    } else {
        if(znajdzDuplikat(tablica, pozycja) == false){
            for (int k = tablica[pozycja - 1]; k <= pozostalo; k++) {
                tablica[pozycja] = k;
                sprawdz_i_wypisz(pozycja + 1, pozostalo - k);
            }
        }
    }
}

void wywołanie(int C) {
    tablica[0] = 1;
    sprawdz_i_wypisz(1, C);
}
```

---

Na samym początku przedstawiona została funkcja `znajdz_duplikat`. Sortuje ona tablicę, a następnie porównuje wszystkie istniejące elementy, aby zobaczyć czy występują powtórzone wartości. Zwraca ona logiczną wartość `true` jeśli występuje powtórzenie lub `false` jeśli porównywane wartości są różne.

Kolejna funkcja, nazwana `sprawdz_i_wypisz`, odpowiada za generowanie ciągów liczbowych i wypisywanie ich. Najpierw funkcja sprawdza ilość pozostałych wartości ciągu liczbowego. Jeżeli ten parametr ma wartość 0, zawartość ciągu zostaje wypisana. W przeciwnym wypadku sprawdzamy, czy pojedynczy element w tablicy wystąpił już wcześniej. Jeżeli nie znaleziono identycznego elementu, algorytm iteruje po kolejnych elementach ciągu liczbowego tak długo, jak ilość pozostałych elementów jest większa bądź równa zero. W momencie spełnienia warunku, funkcja wywoływana jest rekurencyjnie, gdzie parametr pozycji zostaje zwiększony o jeden, a ilość poszukiwanych liczb jest zmniejszana.

Ostatnia funkcja algorytmu odpowiada za przypisanie pierwszej wartości tablicy liczby 1, ponieważ jest to najmniejsza wartość, jaką może osiągnąć liczba w naszym rozkładzie oraz zostaje wywołana funkcja `sprawdz_i_wypisz` z parametrami: 1 jako pierwsza wartość pozycji, oraz liczbą, jaka ma zostać rozłożona.