

Zadanie 10. Podział liczby

Adrian Rupala

7 maja 2018

Treść zadania

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 .

Algorytm z nawrotami (backtracking) - algorytm wyszukiwania wszystkich lub kilku rozwiązań polegający 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 znacznie efektywniejsza niż wyszukiwanie wyczerpujące (zakładając przeszukiwanie wszystkich rozwiązań), ponieważ pojedynczy test może wyeliminować nie jedno a wiele rozwiązań niedopuszczalnych.

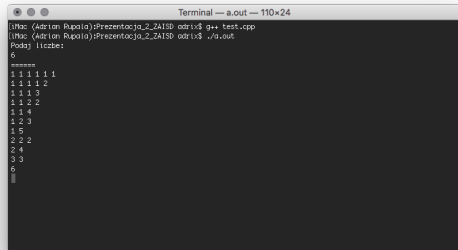
Rekurencja - technika programowania, dzięki której funkcja, procedura lub podprogram jest w stanie w swoim ciele wywołać sam siebie. Pozwala ona łatwo wykonać wiele zadań, w których potrzeba jest wyników cząstkowych do obliczenia całości.

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

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

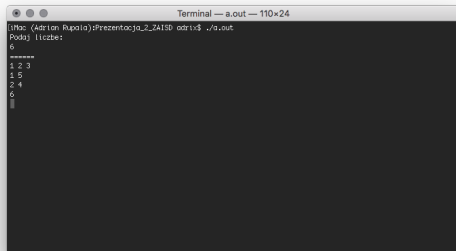
```
bool znajdz_duplikat(int lista[], int rozmiar_tablicy) {  
    sort(lista);  
    for (int i = 0; i < rozmiar_tablicy - 1; i++) {  
        if (lista[i] == lista[i + 1]){  
            return true;  
        }  
    }  
    return false;  
}
```

Wykonanie kodu



```
Terminal — a.out — 110x24
[Mac (Adrian Rupala):Prezentacja_2_ZA15D adrix$ g++ test.cpp
[Mac (Adrian Rupala):Prezentacja_2_ZA15D adrix$ ./a.out
Podaj liczbe:
6
=====
1 1 1 1 1 1
1 1 1 1 2
1 1 1 3
1 1 2 2
1 1 4
1 2 3
1 5
2 2 2
2 4
3 3
6
```

Rysunek 1: Wynik dla liczby 6 z powtórzeniami.



```
Terminal — a.out — 110x24
[Mac (Adrian Rupala):Prezentacja_2_ZA15D adrix$ ./a.out
Podaj liczbe:
6
=====
1 2 3
1 5
2 4
6
```

Rysunek 2: Wynik dla liczby 6 bez powtórzeń.

Wykonanie kodu

```
Terminal — a.out — 110x47
[Mac (Adrian Rupala):Prezentacja_2_ZAISD adrix$ ./a.out
Podaj liczbę:
10
=====
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 2
1 1 1 1 1 1 3
1 1 1 1 1 2 2
1 1 1 1 1 4
1 1 1 1 2 3
1 1 1 1 5
1 1 1 2 2 2
1 1 1 2 4
1 1 1 3 3
1 1 1 6
1 1 2 2 3
1 1 2 5
1 1 3 4
1 1 7
1 1 2 2 2 2
1 1 2 2 4
1 1 2 3 3
1 1 2 6
1 1 3 5
1 1 4 4
1 1 8
1 2 2 2 3
1 2 2 5
1 2 3 4
1 2 7
1 3 3 3
1 3 6
1 4 5
1 9
2 2 2 2 2
2 2 2 4
2 2 3 3
2 2 6
2 3 5
2 4 4
2 8
3 3 4
3 7
4 6
5 5
10
```

Rysunek 3: Wynik dla liczby 10 z

```
Terminal — a.out — 110x23
[Mac (Adrian Rupala):Prezentacja_2_ZAISD adrix$ ./a.out
Podaj liczbę:
10
=====
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
```

Rysunek 4: Wynik dla liczby 10 bez powtórzeń.

Dziękuję za uwagę!