- ## Opis problemu:

Programowanie dynamiczne – technika projektowania algorytmów polegająca na podziale problemu względem kilku parametrów i optymalnym rozwiązaniu wszystkich jego pod problemów. Wartości obliczone dla pod problemów zapamiętujemy w tablicy z której ostatecznie obliczamy wartość dla głównego problemu. Liczba istotnie różnych pod problemów jest wielomianowa. Programowanie dynamiczne jest jedną z bardziej skutecznych metod rozwiązywania problemów NP-trudnych.

Problem PARTITION – Czy można podzielić zbiór wartości na dwie części tak, aby sumy każdego z osobna były sobie równe.
Dla przykładu: $S = \{3,2,1,2,1,1\}$
Jesteśmy w stanie podzielić zbiór S na dwa zbiory, gdzie suma każdego z nich będzie równa 5: $S_1 = \{1,1,1,2\},\ S_2 = \{2,3\}$
Warto dodać że rozwiązanie nie jest unikalne, ponieważ możemy znaleźć inne podziały: $S_1 = \{1,1,3\},\ S_2 = \{1,2,2\}$
Powyższy problem można rozwiązać za pomocą programowania dynamicznego.

- ## Opis algorytmu:

Tworzymy dwu wymiarową tablice tab[i][j]. Numery wierszy (1,2,…,n) odpowiadają pod zbiorą z kolei kolumny są liczbami naturalnymi. Jeżeli istnieje podzbiór {a1,….,ai} dla którego suma rozmiarów jest równa j, tab[i][j] = 1 w przeciwnym wypadku tab[i][j] = 0. Iterujemy tablicę wstawiający w poszczególne komórki true albo false (0,1) korzystając z warunku:

$$Dla\ i = 1\ wykonaj: tab(1,j) = 1 <=> j = 0\ lub\ j = s(a1)$$

$$Dla\ i = 1 < i \le n, 0 \le j \le \frac{1}{2}\sum_{a \epsilon A} s(a) = L\ \ wykonaj:$$

$$tab(i,j) = 1 <=> tab(i-1,j) = 1\ \ lub$$

$$[s[ai] \le j\ i\ tab\big(i-1, j-s(ai)\big) = 1]$$

$$Jeżeli\ tab[n,L] = 1\ to\ zbiór\ da\ się\ podzielić\ w\ ten\ sposób$$

Tablica elementów t(i, j) dla przykładowego problemu:

$i \setminus j$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T	T	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*
2	T	T	*F*	*F*	*F*	*F*	*F*	*F*	*F*	T	T	*F*	*F*	*F*
3	T	T	*F*	*F*	*F*	T	T	*F*	*F*	T	T	*F*	*F*	*F*
4	T	T	*F*	T	T	T	T	*F*	T	T	T	*F*	T	T
5	T	T	*F*	T	T	T	T	*F*	T	T	T	T	T	T

Fragment kodu:

Funkcja wypełniająca tablicę pomocniczą według warunków algorytmu.

```
for(int j=0; j<sumaA; j++)
    if((j==0)||j==tabA[0]) tab[0][j] = 1;

for(int i=1; i<length; i++)
{
    for(int j=0; j<=L; j++)
    {
        if((tab[i-1][j]==1)) tab[i][j] = 1;
        if((tabA[i]<=j)&&(tab[i-1][(j-tabA[i])]==1)) tab[i][j] = 1;
    }
}
```

Przykładowe wywołanie programu:

```
Problem PARTITION
Podaj liczebnosc zbioru glownego: 5
Czy chcesz zainicjowac rozmiary elementor recznie (1), losowo (2)?: 1
a1: 1
a2: 9
a3: 5
a4: 3
a5: 8
L= 13
T T F F F F F F F F F F
T T F F F F F F T T F F
T T F F F T T F F T T F F
T T F T T T T F T T T F T
T T F T T T T F T T T T T
Podzial jest mozliwy
```

```
Problem PARTITION
Podaj liczebnosc zbioru glownego: 4
Czy chcesz zainicjowac rozmiary elementor recznie (1), losowo (2)?: 1
a1: 1
a2: 2
a3: 3
a4: 4
L= 5
T T F F F
T T T T F
T T T T T
T T T T T
Podzial jest mozliwy
```

- **Wnioski:**

  Program korzystając z programowania dynamicznego poprawnie rozwiązuje problem PARTITION. Spełnia on kryteria zadania (maksymalna liczebność zbioru = 10, maksymalna suma rozmiarów elementów = 55). Wynik oraz tablicę pomocniczą wyświetla w konsoli Windows oraz zapisuje do pliku output.txt.

Źródła:

[1]. From Wikipedia - Partition problem.

[2]. dr hab. inż. Zbigniew Kokosiński - Wykład 10: Programowanie dynamiczne