Programowanie dynamiczne – Wyznaczanie optymalnej wielkości partii produkcyjnej

In [1]:
```
import numpy as np
```

In [20]:
```
def production_partion(months, cost, storage_cost, storage, delivery):
    result = np.zeros((storage + 1, 2*months+1))

    f_col = 2
    x_col = 1

    while f_col < 2*months + 1:
        for i in range(storage+1):
            minimal = np.inf
            product_count = 0

            for x in range(max(0, delivery-i), min(storage + delivery - i, 6)):
                if x+i-delivery < 0 or x+i-delivery > storage + 1:
                    value = storage_cost[x] + cost(x) * (i + x - delivery)
                else:
                    value = storage_cost[x] + cost(x) * (i + x - delivery) + result[x+i-

                if value < minimal:
                    minimal = value
                    product_count = x

            result[i][f_col] = minimal # Wartosc funkcji
            result[i][x_col] = product_count # Pojemnosc
        f_col += 2
        x_col += 2

    print("Macierz decyzji optymalnych: \n", result)
    # months_list = [0 for _ in range(months)]
    # i = 0
    # column = 2*months
    # arg = np.argmin(result[:,column])
    # currentCost = result[arg][column]
```

In [21]:
```
production_partion(months=6, cost=lambda x: 2, storage_cost=[0, 15, 18, 19, 20, 24], del
```

```
Macierz decyzji optymalnych:
 [[  0.   3.  19.   3.  38.   4.  52.   3.  71.   3.  90.   4. 104.]
 [  0.   2.  18.   5.  30.   5.  49.   5.  68.   5.  82.   5. 101.]
 [  0.   1.  15.   4.  26.   4.  45.   4.  64.   4.  78.   4.  97.]
 [  0.   0.   0.   0.  19.   0.  38.   0.  52.   0.  71.   0.  90.]
 [  0.   0.   2.   0.  20.   0.  32.   0.  51.   0.  70.   0.  84.]]
```

Zlozonosc obliczeniowa algorytmu

$O(storage^2 \cdot months)$