Sprawozdanie Programowanie dynamiczne – Wyznaczanie optymalnej wielkości partii produkcyjnej

Jakub Ochman grupa 3. AiR

Zadanie 1

Implementacja metody programowania dynamicznego dla zagadnienia wyznaczania optymalnej wielkości partii produkcyjnej:

In [15]:
```python
import sys

def optimal_production_plan(q, g, h, Ymin, Ymax, y0, y_end):
    # q     - lista zapotrzebowania na produkt w kolejnych miesiącach
    # g     - lista kosztów produkcji dla każdej wielkości produkcji
    # h     - lista kosztów magazynowania dla każdego stanu magazynu
    # Ymin  - minimalny dopuszczalny stan magazynu
    # Ymax  - maksymalny dopuszczalny stan magazynu
    # y0    - stan magazynu na początku planu (przed pierwszym miesiącem)
    # y_end - wymagany stan magazynu po ostatnim miesiącu

    n = len(q)  # liczba miesięcy w planie

    # dp[i] to słownik: klucz = stan magazynu po i-tym miesiącu,
    # wartość = minimalny koszt dojścia do tego stanu
    dp = [dict() for _ in range(n + 1)]

    # decisions[i] to słownik decyzji dla miesiąca i:
    # klucz = stan magazynu po i-tym miesiącu,
    # wartość = (stan magazynu przed miesiącem, wielkość produkcji)
    decisions = [dict() for _ in range(n)]

    # koszt na początku dla stanu magazynu y0
    dp[0][y0] = 0

    # przejście przez kolejne miesiące
    for i in range(n):
        # iteracja po możliwych stanach magazynu przed miesiącem i
        for y_prev in dp[i]:
            # sprawdzenie wszystkich dopuszczalnych wielkości produkcji
            for x in range(0, Ymax + 1):
                # obliczenie stanu magazynu po produkcji i zużyciu zapotrzebowania
                y_next = y_prev + x - q[i]

                # uwzględnienie tylko stanów magazynu mieszczących się w dopuszczalnym zakresie
                if Ymin <= y_next <= Ymax:
                    # obliczenie kosztu produkcji i magazynowania
                    cost = g[x] + h[y_next]

                    # obliczenie całkowitego kosztu dojścia do nowego stanu
                    total_cost = dp[i][y_prev] + cost

                    # aktualizacja minimalnego kosztu i decyzji, jeśli znaleziono lepsze rozwiązanie
                    if y_next not in dp[i + 1] or total_cost < dp[i + 1][y_next]:
                        dp[i + 1][y_next] = total_cost
                        decisions[i][y_next] = (y_prev, x)

    # sprawdzenie, czy możliwe jest osiągnięcie wymaganego stanu końcowego magazynu
    if y_end not in dp[n]:
        raise ValueError("Nie można osiągnąć końcowego stanu magazynu.")

    # odtworzenie optymalnej strategii produkcji od końca
    strategy = [0] * n
    y = y_end
    for i in range(n - 1, -1, -1):
        y_prev, x = decisions[i][y]
        strategy[i] = x
        y = y_prev

    min_cost = dp[n][y_end]

    # dp       - tablica słowników z minimalnymi kosztami dojścia do danego stanu w każdym miesiącu
    # decisions- tablica słowników z decyzjami produkcyjnymi prowadzącymi do danego stanu
    # min_cost - minimalny całkowity koszt realizacji planu produkcji i magazynowania
    # strategy - lista optymalnych wielkości produkcji dla kolejnych miesięcy
    return dp, decisions, min_cost, strategy
```

Poniżej przedstawiono funkcje pomocnicze wykorzystane do wypisania macierzy i wyników funkcji:

In [16]:
```python
def print_cost_matrix(dp, Ymin, Ymax):
    print("Macierz kosztów:")
    for i, d in enumerate(dp):
        row = []
        for y in range(Ymin, Ymax + 1):
            if y in d:
                row.append(f"{d[y]:4}")
            else:
                row.append("   -")
        print(" ".join(row))
```

```
def print_production_matrix(decisions, Ymin, Ymax):
    print("Macierz decyzji:")
    for i, d in enumerate(decisions):
        row = []
        for y in range(Ymin, Ymax + 1):
            if y in d:
                row.append(f"{d[y][1]:4}")
            else:
                row.append("   -")
        print(" ".join(row))

def print_results(dp, decisions, min_cost, strategy, Ymin, Ymax):
    print_cost_matrix(dp, Ymin, Ymax)
    print()
    print_production_matrix(decisions, Ymin, Ymax)
    print("\nCałkowity koszt:", min_cost)
    print("Optymalna ścieżka produkcji:", strategy)
```

Zadanie 2

Zadanie obliczeniowe. Obliczenia wykonane dla n=4, n=6, n=12. Przeprowadzono obliczenia dla przykładowych zestawów danych, wyznaczając ksozty minimalne dla każdego możliwego stanu magazynu na kolejnych etapach planu. Otrzymana macierz decyzji pokazuje, jakie działania (produkcja i magazynowanie) prowadzą do optymalnych kosztów w danym stanie. Na końcu wyznaczono optymalną strategię produkcji oraz podano minimalny łączny koszt realizacji planu.

In [17]:
```
# n = 4
q1 = [3, 2, 4, 3]
g1 = [i * 2 for i in range(13)]
h1 = [i for i in range(13)]
Ymin1, Ymax1 = 0, 12
y01, y_end1 = 4, 3
print("n = 4")
dp1, dec1, cost1, strat1 = optimal_production_plan(q1, g1, h1, Ymin1, Ymax1, y01, y_end1)
print_results(dp1, dec1, cost1, strat1, Ymin1, Ymax1)

# n = 5
q2 = [2, 3, 1, 4, 2]
g2 = [i * 2 for i in range(13)]
h2 = [i for i in range(13)]
Ymin2, Ymax2 = 0, 12
y02, y_end2 = 5, 2
print("\nn = 5")
dp2, dec2, cost2, strat2 = optimal_production_plan(q2, g2, h2, Ymin2, Ymax2, y02, y_end2)
print_results(dp2, dec2, cost2, strat2, Ymin2, Ymax2)

# n = 12
q3 = [3, 2, 3, 2, 4, 3, 2, 2, 3, 3, 2, 4]
g3 = [i * 2 for i in range(13)]
h3 = [i for i in range(13)]
Ymin3, Ymax3 = 0, 12
y03, y_end3 = 6, 4
print("\nn = 12")
dp3, dec3, cost3, strat3 = optimal_production_plan(q3, g3, h3, Ymin3, Ymax3, y03, y_end3)
print_results(dp3, dec3, cost3, strat3, Ymin3, Ymax3)
```

```
n = 4
Macierz kosztów:
   -    -    -    -    0    -    -    -    -    -    -    -    -
   -    1    4    7   10   13   16   19   22   25   28   31   34
   3    6    9   12   15   18   21   24   27   30   33   36   40
  11   14   17   20   23   26   29   32   35   39   43   47   51
  17   20   23   26   29   32   35   38   41   44   48   52   56

Macierz decyzji:
   -    0    1    2    3    4    5    6    7    8    9   10   11
   1    2    3    4    5    6    7    8    9   10   11   12   12
   4    5    6    7    8    9   10   11   12   12   12   12   12
   3    4    5    6    7    8    9   10   11   12   12   12   12

Całkowity koszt: 26
Optymalna ścieżka produkcji: [0, 1, 4, 6]

n = 5
Macierz kosztów:
   -    -    -    -    -    0    -    -    -    -    -    -    -
   -    -    -    3    6    9   12   15   18   21   24   27   30
   3    6    9   12   15   18   21   24   27   30   33   36   39
   5    8   11   14   17   20   23   26   29   32   35   38   42
  13   16   19   22   25   28   31   34   37   41   45   49   53
  17   20   23   26   29   32   35   38   41   44   47   51   55

Macierz decyzji:
   -    -    -    0    1    2    3    4    5    6    7    8    9
   0    1    2    3    4    5    6    7    8    9   10   11   12
   1    2    3    4    5    6    7    8    9   10   11   12   12
   4    5    6    7    8    9   10   11   12   12   12   12   12
   2    3    4    5    6    7    8    9   10   11   12   12   12

Całkowity koszt: 23
Optymalna ścieżka produkcji: [0, 0, 1, 4, 4]

n = 12
Macierz kosztów:
   -    -    -    -    -    -    0    -    -    -    -    -    -
   -    -    -    3    6    9   12   15   18   21   24   27   30
   -    4    7   10   13   16   19   22   25   28   31   34   37
   8   11   14   17   20   23   26   29   32   35   38   42   46
  12   15   18   21   24   27   30   33   36   39   42   46   50
  20   23   26   29   32   35   38   41   44   48   52   56   60
  26   29   32   35   38   41   44   47   50   53   57   61   65
  30   33   36   39   42   45   48   51   54   57   60   64   68
  34   37   40   43   46   49   52   55   58   61   64   68   72
  40   43   46   49   52   55   58   61   64   67   71   75   79
  46   49   52   55   58   61   64   67   70   73   77   81   85
  50   53   56   59   62   65   68   71   74   77   80   84   88
  58   61   64   67   70   73   76   79   82   86   90   94   98

Macierz decyzji:
   -    -    -    0    1    2    3    4    5    6    7    8    9
   -    0    1    2    3    4    5    6    7    8    9   10   11
   2    3    4    5    6    7    8    9   10   11   12   12   12
   2    3    4    5    6    7    8    9   10   11   12   12   12
   4    5    6    7    8    9   10   11   12   12   12   12   12
   3    4    5    6    7    8    9   10   11   12   12   12   12
   2    3    4    5    6    7    8    9   10   11   12   12   12
   2    3    4    5    6    7    8    9   10   11   12   12   12
   3    4    5    6    7    8    9   10   11   12   12   12   12
   3    4    5    6    7    8    9   10   11   12   12   12   12
   2    3    4    5    6    7    8    9   10   11   12   12   12
   4    5    6    7    8    9   10   11   12   12   12   12   12

Całkowity koszt: 70
Optymalna ścieżka produkcji: [0, 0, 2, 2, 4, 3, 2, 2, 3, 3, 2, 8]
```

Zadanie 3

Model można rozszerzyć o koszty uruchomienia produkcji, które pojawiają się tylko, gdy produkcja w danym miesiącu jest większa niż zero, co lepiej odzwierciedla realne koszty rozruchu linii. Można także wprowadzić ograniczenia na maksymalną produkcję w danym miesiącu, co uwzględnia ograniczoną wydajność maszyn lub dostępność zasobów. W modelu można uwzględnić zmienne koszty produkcji i magazynowania w czasie, by odzwierciedlić np. sezonowość cen energii lub kosztów przechowywania.

Złożoność obliczeniowa algorytmu wynosi $O(n \cdot (Ymax - Ymin)^2)$, ponieważ dla każdego z n miesięcy i każdego możliwego stanu magazynowego rozważane są wszystkie możliwe decyzje produkcyjne.

Źródła:

- Na podstawie materiałów z zajęć oraz wykładu
- Na podstawie opisu programowania dynamicznego: https://en.wikipedia.org/wiki/Dynamic_programming

---

Środowisko:

Jupiter Notebook w Visual Studio Code z rozszerzeniem Jupiter, Python