AISD - Lista 6 Zadanie 6

11 maja 2020

1 Zadanie

Rozważmy stację przeładunkową o pojemności W. Na stację dla klientów zamawiamy taki sam towar, przychodzący w jednostkowych paczkach. Przechowywanie jednej paczki przez jeden dzień kosztuje c. Zamówienie dostawy dowolnej liczby paczek kosztuje P. Pokaż algorytm, który mając dany ciąg zapotrzebowań klientów (dla każdego dnia wiemy ile paczek zostanie od nas kupione) tak rozplanowuje nasz plan zamówień żeby zminimalizować całkowity koszt.

2 Rozwiązanie

2.1 Strategia

Aby rozwiązać problem optymalizacji zdefiniowałem funkcję $L(k, i)$ wyznaczającą minimalny koszt przechowania i zamówień od dnia k, przy i paczkach w magazynie (n - ilość dni):

$$L(k,i) = \begin{cases} k = n: & \begin{cases} A[n] > i: & P \\ A[n] \leq i: & 0 \end{cases} \\ \\ k < n: & \begin{cases} A[k] > i: & P + c \cdot i' + L(k+1, i') \text{ najmniejsze dla} \\ & i' \in [0, W - A[k]] \\ \\ A[k] \leq i: & \text{mniejsze z:} \begin{cases} P + c \cdot i' + L(k+1, i') \\ \text{ najmniejsze dla} \\ i' \in [i - A[k], W - A[k]] \\ \\ c \cdot (A[k] - i) + L(k+1, A[k] - i) \end{cases} \end{cases} \end{cases}$$

Ponieważ funkcja ta wymaga wyliczenia rekurencji, która bardzo szybko się bardzo rozgałęzia na wiele przypadków, złożoność wyliczenia jej wartości zaczynając od $k = 1$ jest ogromna.

Dlatego lepszym pomysłem, jest zaczęcie od końca, zapisanie wyników do dwuwymiarowej tabeli o rozmiarze $n \times W$, która reprezentuje najlepszy wynik dla każdego dnia w zależności od k i i. Następnie idąc w kierunku pierwszego dnia, korzystać z wyliczonych już wartości.

k				
L(1,0)	L(2,0)	...	L(n,0)	i
L(1,1)	...		L(n,1)	
...				
L(1,W)	...		L(n,W)	

Do zachowywania planu zamówień dostaw również potrzebujemy tablicy Z o rozmiarze $n \times W$ która będzie aktualizowana tak samo jak tablica L.

2.2 Algorytm

Z - tablica zamówień dostaw; Z' - tymczasowa tablica zamówień dostaw
A - tablica zamówień od klientów
L - tablica poprzednich wyników

```
k ← n − 1
while k ≥ 1 do                                        ▷ Zaczynamy od przedostatniego dnia
    i ← 0
    while i ≤ W do                                    ▷ Wyliczamy wszystkie możliwe i
        min ← MAX_INT
        if A[k] > i then                              ▷ W magazynie za mało
            min, i' ← find_min(0, W − A[k])           ▷ Ile zamówić?
            Z'[i] ← Z[i']
            Z'[i][k] ← i' − (i − A[k])                ▷ Dodaj nowe zamówienie
        else                                          ▷ W magazynie wystarczająco
            min, i' ← find_min(i − A[k], W − A[k])    ▷ Ile zamówić?
            Z'[i] ← Z[i']
            Z'[i][k] ← i' − (i − A[k])                ▷ Dodaj nowe zamówienie
            if c · (A[k] − i) + L[k + 1][A[k] − i] < min then   ▷ Nie zamawiać?
                min ← c · (A[k] − i) + L[k + 1][A[k] − i]
                Z'[i] ← Z[A[k] − i]                   ▷ Nie zamawiaj
            end if
        end if
        L[k][i] ← min                                 ▷ Zapisujemy najlepszą opcję do tablicy
        i ← i + 1
    end while
    Z ← Z'                                            ▷ Aktualizujemy tablicę zamówień dla każdego i
    k ← k − 1
end while
return Z                                              ▷ Zwróć tablicę zamówień
```

$find\_min(s, t, k)$:

```
min ← MAX_INT
i ← s
while s ≤ t do                                        ▷ Sprawdź wszystkie możliwe wyniki
    r ← P + c · s + L[k + 1][s]
    if r < min then                                   ▷ Nowa najniższa wartość
        min ← r
        i ← s
    end if
    s ← s + 1
end while
return min, i                                         ▷ Dla jakiego i będzie minimum i ile ono wynosi
```

2.3 Złożoność

Złożoność czasowa algorytmu wynosi $O(n \cdot W^2)$. Złożoność pamięciowa natomiast $O(n \cdot W)$.