# Zadanie 5 - dokumentacja końcowa

Michał Sokólski

## 1.Treść zadania

Chcemy zakupić zasilacze, do których podłączymy urządzenia elektryczne. Każdy zasilacz ma 3 wejścia. Dana jest maksymalna moc wspólna dla wszystkich zasilaczy.
Załóżmy, że mamy k (k >= 1) urządzeń, które należy podłączyć do zasilaczy. Każde z urządzeń ma swoją maksymalną moc. Suma mocy maksymalnych urządzeń nie może przekraczać mocy zasilacza.
Zaproponuj algorytm, który będzie minimalizował liczbę zasilaczy, które należy zakupić aby podłączyć wszystkie urządzenia.

Oceń złożoność czasową oraz pamięciową takiego algorytmu.

(errata)

po rozmowach proszę skupcie się na rozwiązaniu zadania dla liczby wejść zasilaczy = 2 a dla liczby 3 przedstawcie dyskusję czy istnieje heurystyka, która poprawi rozwiązanie typu brute force.
Pozdrawiam

## 2.Interpretacja treści

Możemy zinterpretować zadanie w następujący sposób. Niech $M$ będzie maksymalną mocą wspólną dla wszystkich zasilaczy. Zasilacze możemy zamodelować jako jednakowe “pudełka” o pojemności $M$. Urządzenia w takim razie możemy zamodelować jako przedmioty, które wkładamy do pudełek. Każdy taki przedmiot $a$ ma objętość równą swojej maksymalnej mocy - $a\_m$. Z ograniczenia w erracie wiadomo, że do każdego pudełka można włożyć maksymalnie dwa przedmioty oraz suma ich objętości musi być niewiększa niż $M$. Chcemy znaleźć algorytm który będzie minimalizował liczbę potrzebnych pudełek do zapakowania wszystkich przedmiotów. W ten sposób problem staje się wariacją bin-packing (problemu pakowania) z dodatkowymi ograniczeniami.

Warto zauważyć, że przedmioty są charakteryzowane w całości przez swoją objętość ($a\_m$), natomiast pudełka przez swoją pojemność ($M$). W takim razie problem można uprościć do matematycznie zwięzłej postaci:

Mamy dany zbiór N liczb naturalnych A = {a1, a2, … , aN} oraz M. Chcemy dokonać takiego podziału zbioru A, by każdy element Ai tej rodziny był zbiorem jedno lub dwuelementowym oraz suma elementów w Ai była niewiększa od M. Znaleźć moc takiej rodziny, która ma najmniej elementów ze wszystkich możliwych rodzin.

# 3.Opis algorytmu

Zadanie jest rozwiązywane przez prosty algorytm zachłanny.

## Dane wejściowe:

T - tablica elementów A (indeksowana od 0)
N - rozmiar T
M - ograniczenie z treści

## Algorytm:

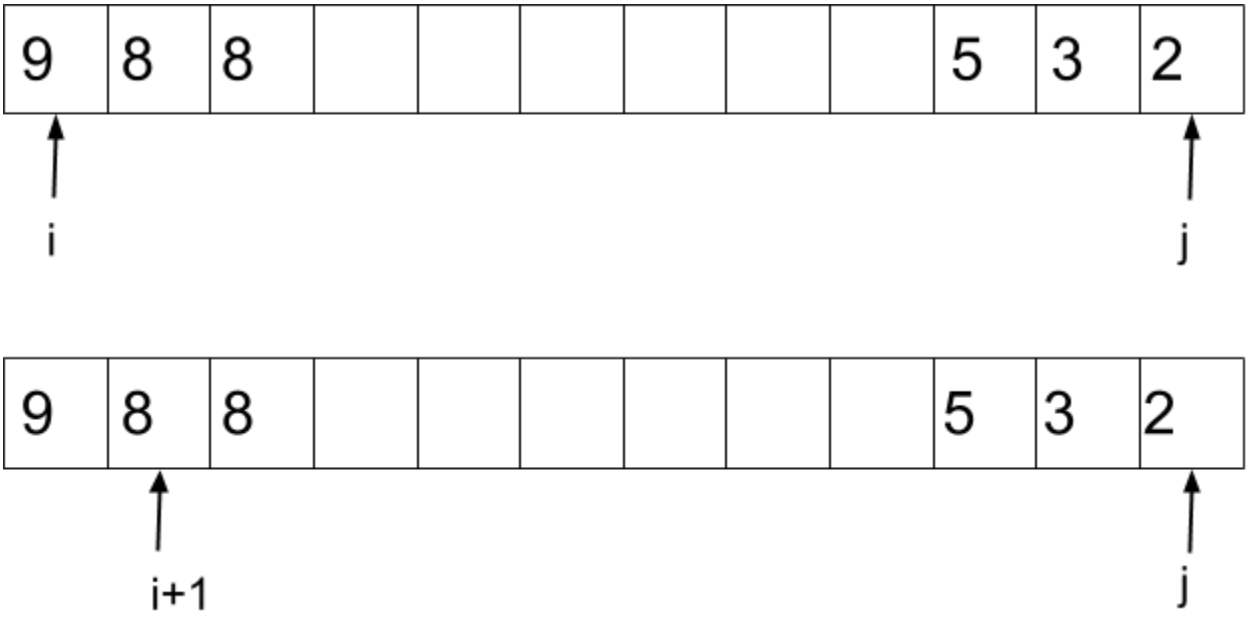
1. Posortuj T nierosnąco
2. i:=0, j:=N-1, Z:=0
3. Dopóki i<j:
    a. S := T[i] + T[j]
    b. Z:=Z+1
    c. Jeżeli S > M to i:=i+1 i przejdź do kroku 3.
    d. i:=i+1, j:=j-1 i przejdź do kroku 3.
4. Jeżeli i==j zwróć Z+1, w przeciwnym wypadku zwróć Z

## Idea algorytmu

Ogólna zasada działania algorytmu jest taka, że rozważamy kolejne pary elementów i sprawdzamy, czy jesteśmy w stanie włożyć je do jednego “pudełka”. Na koniec działania algorytmu zwracamy liczbę pudełek.

Rozważmy posortowaną tablicę T i M=10 i pewne indeksy i,j.

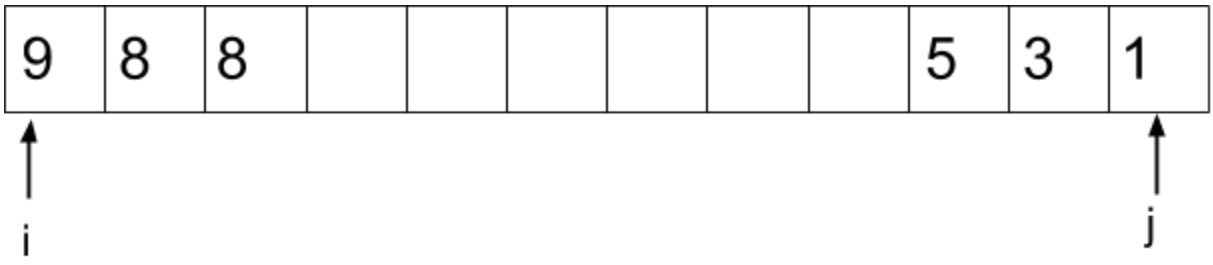
Przypadek 1

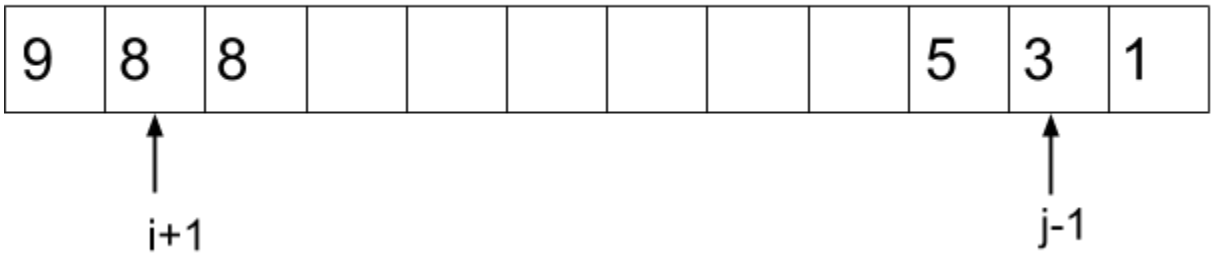



Na powyższej ilustracji widać sytuację, w której elementy T[i] oraz T[j] sumują się do 11 > M = 10. W takim razie zwiększamy indeks i. Musimy tak zrobić, bo wszystkie elementy o indeksach większych od j są większe lub równe T[j], w takim razie nie istnieje taki indeks k większy od j dla którego T[i] + T[k] <= M = 10. W takim razie T[i] musi być włożone samo do nowego “pudełka” (Z:=Z+1).

Przypadek 2

Na kolejnej ilustracji widać z kolei sytuację, w której T[i] + T[j] <=M=10. W takim razie możemy je razem włożyć do jednego “pudełka” (Z:=Z+1)




Przez to że przy każdym kroku indeksy i oraz j zbliżają się do siebie, w końcu rozważymy wszystkie elementy tablicy.

# 4. Analiza złożoności

W kroku 1. sortujemy T nierosnąco. Amortyzowana złożoność obliczeniowa sortowania tablicy liczb naturalnych to *O(nlogn)*. W kolejnych krokach mamy pętlę, przechodzącą przez całą tablicę. W każdym jej kroku, odległość między indeksami *i* oraz *j* się zmniejsza. W takim razie maksymalna ilość wykonanych iteracji przez pętlę wynosi *N-1*. Daje to czas linowy dla tej części algorytmu - *O(n)*. Złożoność pozostałych operacji w algorytmie jest stała. W takim razie całkowita złożoność obliczeniowa algorytmu wynosi: *O(nlogn) + O(n) = O(nlogn).*

Złożoność pamięciowa algorytmu wynosi O(N) bo musimy przechowywać N-elementową tablicę pamięci a sam algorytm działa w “miejscu”.

## 5. Problem dla zasilaczy o 3 wejściach

W przypadku gdy zasilacz ma 3 wejścia, problem jest analogiczny, ale do jednego “pudełka” mogą zmieścić się aż 3 przedmioty (pod warunkiem że się zmieszczą). W oczywisty sposób, rozwiązanie siłowe daje optymalne rozwiązanie, lecz ma dużą złożoność obliczeniową. Istnieją proste heurystyki, nie dające może zawsze optymalnego rozwiązania, lecz posiadające znacząco mniejszą złożoność obliczeniową. Jedną z takich heurystyk, jest first-fit: algorytm przetwarza przedmioty w dowolnej kolejności dodając je do pierwszego pudełka do którego się zmieści. Jeżeli przedmiot nie mieści się w żadnym pudełku, jest dodawany do nowego. Taki algorytm ma złożoność obliczeniową *O(n^2)*, lecz nie zawsze odnajduje rozwiązanie optymalne.

## 6. Problemy implementacyjne

Rozwiązanie zostało zaimplemenotwane w języku C++ i okazało się dość proste w implementacji. Wykorzystałem jedynie elementarne struktury danych i algorytmy z STD.

## 7. Wnioski

Zadanie okazało się stosunkowo proste zarówno pod względem algorytmicznym jak i programistycznym. Przypadek dla zasilaczy o 3 wejściach wydaje się być problemem NP-zupełnym.

*Michał Sokólski*