Jakub Radoszewski	Marcin Pilipczuk, Jakub Radoszewski	Marcin Pilipczuk
Treść zadania	Opracowanie	Program

OI, Etap III, dzień pierwszy, 28.03.2007

Odważniki

Bajtocki Instytut Fizyki Doświadczalnej przy przeprowadzce do nowego budynku napotkał na nie lada problem logistyczny — kłopotliwe okazało się przeniesienie bogatej kolekcji precyzyjnych odważników.

Instytut ma do dyspozycji pewną liczbę kontenerów, każdy o ograniczonej wytrzymałości. Należy pomieścić w nich jak najwięcej odważników, gdyż pozostałe, niestety, będzie trzeba wyrzucić. Do każdego z kontenerów można włożyć dowolnie wiele odważników, ale nie wolno przekroczyć przy tym jego wytrzymałości. Kontener może również pozostać pusty.

Dowolne dwa odważniki w Instytucie mają ciekawą własność: masa jednego z nich jest zawsze wielokrotnością masy drugiego z nich. W szczególności, oba odważniki mogą mieć równe masy.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia wytrzymałości kontenerów i masy odważników,*
- *wyznaczy maksymalną liczbę odważników, jakie można pomieścić w kontenerach,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby całkowite n oraz m ($1 \leqslant n,m \leqslant 100\,000$), oddzielone pojedynczym odstępem i oznaczające odpowiednio: liczbę kontenerów i liczbę odważników. Drugi wiersz wejścia zawiera n liczb całkowitych w_i ($1 \leqslant w_i \leqslant 1\,000\,000\,000$ dla $1 \leqslant i \leqslant n$), pooddzielanych pojedynczymi odstępami i oznaczających wytrzymałości kontenerów, wyrażone w miligramach. W trzecim wierszu wejścia znajduje się m liczb całkowitych m_j ($1 \leqslant m_j \leqslant 1\,000\,000\,000$ dla $1 \leqslant j \leqslant m$), pooddzielanych pojedynczymi odstępami i oznaczających masy odważników, również wyrażone w miligramach.

Wyjście

Pierwszy i jedyny wierszy wyjścia powinien zawierać jedną liczbę całkowitą — największą liczbę odważników, jakie można porozmieszczać w kontenerach bez przekroczenia ich wytrzymałości.

Przykład

Dla danych wejściowych:

```
2 4
```

```
13 9
4 12 2 4
```

poprawnym wynikiem jest:

```
3
```

W kontenerach można umieścić dowolne trzy z czterech odważników, ale nie można umieścić wszystkich odważników.

Rozwiązanie

Wprowadzenie

Odważniki z Bajtockiego Instytutu Fizyki Doświadczalnej spełniają następującą zależność: gdy weźmiemy dowolne dwa z nich, to *zawsze masa jednego z nich będzie całkowitą wielokrotnością masy drugiego*. Ta prosta własność ma poważne konsekwencje — spróbujmy je odkryć.

Uporządkujmy wszystkie odważniki według mas w kolejności niemalejącej: $m_1 \leqslant m_2 \leqslant \cdots \leqslant m_m$. Wówczas dla dowolnych dwóch sąsiednich odważników zachodzi:

$$k_i \cdot m_i = m_{i+1}, \quad i = 1,2,\ldots,m-1,$$

gdzie k_i jest liczbą naturalną. Stąd

$$m_i = m_{i+1} \quad \text{albo} \quad 2 \cdot m_i \leqslant m_{i+1}.$$

To oznacza, że jeśli w ciągu mamy k różnych odważników $m_{i_1} < m_{i_2} < \ldots < m_{i_k}$, to

$$2^{j-1} \cdot m_{i_1} \;\leqslant\; m_{i_j},$$

dla $j = 1,2,\ldots,k$. Ponieważ z ograniczeń zadania wiemy, że masy wszystkich odważników mieszczą się w przedziale $[1,10^9]$, więc:

$$1 \leqslant m_{i_1} \quad \text{oraz} \quad 2^{k-1} \cdot m_{i_1} \leqslant m_{i_k} \leqslant 10^9.$$

Ponieważ $2^{30} = 1\,073\,741\,824 > 10^9 > 536\,870\,912 = 2^{29}$, więc nawet przyjmując minimalną masę pierwszego odważnika $m_{i_1} = 1$, mamy $k \leqslant 30$, co po prostu oznacza, że może być co najwyżej 30 różnych odważników!

Oznaczmy przez $M_1 < \cdots < M_k$ posortowane rosnąco wszystkie różne masy odważników, a przez $a_1,\ldots,a_k$ — liczby odważników o kolejnych masach. Aby wyznaczyć wartości M_i oraz a_i, wystarczy posortować zadany ciąg $m_1,\ldots,m_m$ i „zliczyć", ile razy występują w nim poszczególne wartości. Jeżeli zastosujemy dobry, szybki algorytm sortowania, na przykład sortowanie przez scalanie, to wykonamy te obliczenia w złożoności czasowej $O(m \log m)$.

Istnieje jednak bardziej pomysłowy i efektywniejszy sposób. Wyznaczmy przedziały (*kubełki*) ograniczone kolejnymi potęgami dwójki: $[2^j, 2^{j+1}-1]$ dla $j = 0,1,\ldots,29$. Zauważmy, że dwie liczby M_i, M_{i+1} nie mogą należeć do tego samego przedziału, bo jeśli $2^j \leqslant M_i < 2^{j+1}$, to $M_{i+1} \geqslant 2 \cdot M_i \geqslant 2 \cdot 2^j$. Wystarczy zatem podzielić odważniki na kubełki, a następnie wyznaczyć masę i liczbę odważników z każdego kubełka. Zakładając, że z jest największą masą odważnika, mamy $\log z$ kubełków (wiemy, że $\log z \leqslant 30$, ale być

może jest to wartość jeszcze mniejsza, jeśli na przykład maksymalna masa odważnika jest istotnie mniejsza od zadanego ograniczenia). Przydział odważnika do kubełka możemy zatem wykonać za pomocą wyszukiwania binarnego w czasie $O(\log\log z)$. Przejrzenie kubełków i zliczenie zawartych w nich odważników wymaga czasu $O(\log z+m)$. Ze względu na mniejsze znaczenie, składnik $O(\log z+m)$ możemy pominąć — widzimy więc, że wartości M_i oraz a_i umiemy wyznaczyć w czasie $O(m\log\log z)$.

Przedstawiony algorytm jest ciekawy wyłącznie z teoretycznego punktu widzenia, ponieważ w praktyce sortowanie przez scalanie działa podobnie szybko.

Wiedząc, z jakimi odważnikami mamy do czynienia, ich optymalne przyporządkowanie do kontenerów możemy wyznaczyć na dwa istotnie różne sposoby.

Rozwiązanie wzorcowe – wersja pierwsza

W dalszej części przez *rozwiązanie* będziemy rozumieć dowolne poprawne przyporządkowanie odważników do kontenerów, natomiast przez *rozwiązanie optymalne* — dowolne rozwiązanie, w którym liczba zapakowanych odważników jest maksymalna.

Pierwsze spostrzeżenie przypomina nam, że interesuje nas tylko liczba zapakowanych odważników, a nie ich sumaryczna masa:

Twierdzenie 1 *Jeśli w rozwiązaniu optymalnym pakujemy do kontenerów K odważników, to istnieje rozwiązanie optymalne, w którym pakujemy do kontenerów K* ***najlżejszych*** *odważników.*

Dowód Rozważmy dowolne rozwiązanie optymalne, w którym użyliśmy odważników $m_{i_1} \leqslant m_{i_2} \leqslant \cdots \leqslant m_{i_K}$, a pozostałe odważniki to $m_{j_1} \leqslant m_{j_2} \leqslant \cdots \leqslant m_{j_L}$, gdzie $L = m - i_K$. Zauważmy, że podmieniając w rozwiązaniu dowolny odważnik na lżejszy, nie zmieniamy poprawności rozwiązania (nie przeciążamy kontenera) ani jego optymalności (nie zmieniamy liczby odważników) — dopóki więc $m_{i_K} > m_{j_1}$, to możemy podmieniać te dwa odważniki (po każdej takiej podmianie może zajść potrzeba ponownego posortowania ciągów). Gdy dojdziemy do $m_{i_K} \leqslant m_{j_1}$, wiemy, że mamy rozwiązanie optymalne złożone z odważników o najmniejszych masach. ■

Wartość K wyznaczymy metodą prób. Wiemy, że rozwiązaniem jest liczba z przedziału $[0,m]$. Będziemy przeszukiwać ten przedział binarnie, testując kolejne, potencjalne wartości K:

- dla ustalonej wartości K wybierzemy zestaw K najlżejszych odważników i sprawdźmy, czy da się je zapakować do kontenerów;
- jeśli tak, to spróbujemy zwiększyć liczbę K;
- jeśli nie, to zmniejszamy liczbę K.

Postępując zgodnie z tym schematem, po $O(\log m)$ testach znajdziemy optymalną wartość K. Pozostaje tylko opisać sposób sprawdzenia, czy K określonych odważników można zapakować do kontenerów. Okazuje się, że problem ten można rozwiązać metodą zachłanną. Dla zestawu odważników $m_1 \leqslant m_2 \leqslant \cdots \leqslant m_K$ wystarczy:

- rozważać odważniki w kolejności *od najcięższych do najlżejszych*: $m_K, m_{K-1}, \ldots, m_1$,
- każdorazowo umieszczać rozważany odważnik *w dowolnym kontenerze, w którym się on jeszcze zmieści*.

W dowodzie poprawności takiego rozwiązania kluczowe jest następujące spostrzeżenie.

Twierdzenie 2 *Rozważmy rozwiązanie R, w którym najcięższy odważnik x ma masę m_K i znajduje się w j-tym kontenerze. Wówczas dla dowolnego $t \in \{1,\ldots,n\}$, jeżeli $m_K \leqslant w_t$ (czyli odważnik x mieści się w pustym, t-tym kontenerze), to R możemy przekształcić w rozwiązanie R_t, w którym odważnik x znajdzie się w t-tym kontenerze.*

Dowód Jeżeli łączna masa odważników, jakie w rozwiązaniu R zostały umieszczone w t-tym kontenerze, jest nie większa od m_K, to zamieniając miejscami te odważniki z odważnikiem x otrzymujemy szukane rozwiązanie R_t. Jeżeli w rozwiązaniu R łączna masa odważników w kontenerze t-tym jest większa niż m_K, to pokażemy, że można spośród nich wybrać taki zestaw, którego łączna masa jest równa dokładnie m_K. Wówczas zamieniając miejscami odważnik x i wybrany zestaw, także otrzymamy szukane rozwiązanie R_t.

Rozważmy więc kontener, w którym łączna masa odważników przekracza m_K. Dla wygody oznaczmy masy tych odważników $m'_1 \geqslant m'_2 \geqslant \cdots \geqslant m'_s$. Wybierzmy zestaw odważników $m'_1, m'_2, \cdots, m'_j$, dobierając kolejne elementy od najcięższych do najlżejszych, tak długo, jak długo masa zestawu nie przekracza m_K, tzn.

$$m'_1 + \cdots + m'_j \leqslant m_K < m'_1 + \cdots + m'_{j+1}.$$

Okazuje się, że w ten sposób zawsze otrzymamy zestaw o masie równej dokładnie m_K! Uzasadnimy ten fakt, pokazując przez indukcję, że w każdym kroku konstrukcji zestawu masa każdego z pozostałych odważników dzieli masę, jakiej brakuje już utworzonemu zestawowi do m_K, tzn.

$$m'_k \quad | \quad m_K - (m'_1 + \cdots + m'_i), \tag{1}$$

dla $0 \leqslant i \leqslant j$ oraz każdego $k = i+1, i+2, \ldots, s$. Z własności tej wynika, że do zestawu zawsze bierzemy odważnik o masie nie większej niż różnica m_K i masy zestawu. Stąd mamy pewność, że w j-tym kroku masa konstruowanego zestawu osiągnie dokładnie m_K.

Przejdźmy do zapowiedzianego dowodu indukcyjnego własności (1).

Baza indukcji. Dla pustego zestawu pozostała do uzupełnienia masa wynosi m_K. Dla każdego z dostępnych odważników $m'_k \leqslant m_K$ oraz (co wynika ze specyficznych właściwości zestawu odważników) $m'_k \mid m_K$.

Krok indukcyjny. Załóżmy, że wybraliśmy już odważniki $m'_1 \geqslant m'_2 \geqslant \cdots \geqslant m'_i$ i pozostało nam do wypełnienia $y = m_K - (m'_1 + m'_2 + \cdots + m'_i)$ miligramów. Zauważmy, że $m'_{i+1} \mid y$, ponieważ:

- z założenia indukcyjnego $m'_i \mid m_K - (m'_1 + m'_2 + \cdots + m'_{i-1}) = y + m'_i$, czyli także $m'_i \mid y$;
- wiemy także, że $m'_{i+1} \mid m'_i$, a więc $m'_{i+1} \mid y$.

Zauważmy wreszcie, że dla każdego $l > i+1$ zachodzi $m'_l \mid m'_{i+1}$, a zatem $m'_l \mid y$, co kończy dowód.

■

Korzystając z udowodnionego twierdzenia, możemy wykazać poprawność algorytmu zachłannego przydziału odważników do kontenerów. Przypomnijmy, że mamy K najlżejszych odważników $m_1, m_2, \ldots, m_K$ i sprawdzamy, czy zmieszczą się one w kontenerach, przydzielając je zgodnie z zasadą:

- rozważamy odważniki w kolejności *od najcięższych do najlżejszych*: $m_K, m_{K-1}, \ldots, m_2, m_1$;
- każdorazowo umieszczamy rozważany odważnik *w dowolnym kontenerze, w którym się on jeszcze zmieści*.

Jeśli istnieje upakowanie R tych odważników do kontenerów, to możemy je podmienić na dowolne rozwiązanie, w którym najcięższy odważnik jest w dowolnym kontenerze, w którym się mieści — wybierzmy kontener t, do którego przydzieli go algorytm zachłanny. Teraz możemy zmodyfikować problem, zmniejszając wytrzymałość kontenera t o masę m_K oraz usuwając odważnik m_K z zestawu. Ze zredukowanym problemem postępujemy analogicznie — odważnik o największej masie m_{K-1}, który gdzieś w rozwiązaniu R został przydzielony, przesuwamy do kontenera wybranego w rozwiązaniu zachłannym. Kontynuując tę przebudowę wyjściowego rozwiązania, dochodzimy do wniosku, że *jeśli istnieje rozwiązanie dla rozważanych odważników, to istnieje dla nich rozwiązanie zachłanne*. Wniosek przeciwny: *jeśli nie istnieje rozwiązanie, to nie istnieje także rozwiązanie zachłanne*, jest oczywisty.

Przekonawszy się o poprawności przedstawionego algorytmu, możemy zastanowić się nad jego efektywną implementacją. Przede wszystkim zauważmy, że zamiast zastanawiać się nad wyborem i umieszczać dany odważnik w *dowolnym* kontenerze, możemy go włożyć w szczególności do kontenera o *największej wytrzymałości*. Potrzebujemy więc struktury danych do przechowywania kontenerów, dla której będziemy w stanie wykonywać operacje:

- pobranie kontenera o największej wytrzymałości,
- wstawienie kontenera (o zmodyfikowanej wytrzymałości).

Do tego celu wygodnie jest użyć kolejkę priorytetową, czyli na przykład kopiec binarny (opis można znaleźć w [19]), który pozwala każdą z powyższych operacji wykonać w złożoności czasowej $O(\log n)$.

Możemy już zaprezentować kompletny algorytm:

```
1: Posortuj(m_1, m_2, ..., m_m);
2: Wyznacz(M_i, a_i);
3: { Wyszukiwanie binarne wartości K. }
4: d := 0; g := m;
5: while d < g do
6: begin
7:    K := ⌊(d+g)/2⌋
8:    if Zapakuj(K) then d := K;
```

```
    else g := K − 1;
end
return d;
```

```
function Zapakuj(K);
begin
  { Próbujemy zapakować K najlżejszych odważników. }
  S := Kopiec(w_1, ..., w_n);
  for i := K downto 1 do
  begin
    t :=Weź_Największy(S);
    if w_t < m_i then
      return false;
    Wstaw(S,t,w_t − m_i);
  end
  return true;
end
```

Złożoność. Przyjrzyjmy się jeszcze złożoności czasowej powyższego algorytmu. Każde wywołanie funkcji *Zapakuj* jest wykonywane w złożoności czasowej $O(n)$ (koszt konstrukcji n-elementowego kopca binarnego) plus $O(m \log n)$ ($K \leqslant m$ operacji wstawiania/usuwania na kopcu n-elementowym). Funkcję *Zapakuj* wywołujemy $O(\log m)$ razy z racji binarnego wyszukiwania wartości K w przedziale $[0,m]$. Cały algorytm ma zatem złożoność $O((n+m\log n)\log m)$, niezależnie od tego, na ile efektywną metodę wstępnego sortowania odważników wybierzemy. Algorytm ten został zaimplementowany w plikach `odw3.cpp`, `odw8.pas` (własna implementacja kopca) oraz `odw2.cpp` (kopiec z biblioteki STL).

Dalsze usprawnienia

Poniżej przedstawimy, jak zrealizować efektywniej zaprezentowane rozwiązanie. Uzyskana poprawa czasu działania jest jednak praktycznie „niemierzalna", więc na zawodach obie wersje rozwiązania uzyskiwały taką samą ocenę — maksymalną. Opisane w niniejszej sekcji usprawnienia są więc głównie interesujące z teoretycznego punktu widzenia i można je potraktować jako ciekawostkę.

Na początek skoncentrujmy się na ulepszeniu implementacji funkcji *Zapakuj*. Zauważmy, że w skonstruowanym algorytmie w żadnym miejscu nie skorzystaliśmy z faktu, że liczba różnych mas odważników jest bardzo mała, rzędu $O(\log z)$ (przypomnijmy, że przez z oznaczyliśmy maksymalną masę odważnika M_k). Spróbujmy zastąpić kopiec bardziej efektywną w tym przypadku strukturą. Dla każdej wartości M_i ($i \in \{1,\ldots,k\}$) utwórzmy listę L_i tych kontenerów, których wytrzymałości są zawarte w przedziale $[M_i, M_{i+1}-1]$ (dla wygody przyjmujemy, że $M_{k+1} = \infty$). Będziemy teraz rozważać odważniki w kolejności niemalejących mas $M_k,\ldots,M_1$. Dla każdego M_i wykonujemy następujące czynności:

1. jeżeli $L_{i+1} \neq \emptyset$, to scalamy listy L_i oraz L_{i+1};

2. dla każdego spośród a_i odważników o masie M_i:
   (a) bierzemy dowolny kontener $w_j \in L_i$ — jeżeli takiego kontenera nie ma, to kończymy działanie z informacją, że odważników nie da się zapakować;
   (b) w przeciwnym razie wstawiamy odważnik do kontenera w_j oraz
   (c) zmieniamy wytrzymałość kontenera na $w_j - M_i$ i umieszczamy go na właściwej liście (listę odnajdujemy, na przykład, za pomocą wyszukiwania binarnego).

Zauważmy, że w tym algorytmie wykorzystujemy fakt, że odważnik możemy umieścić w *dowolnym* dostatecznie wytrzymałym kontenerze, a niekoniecznie tylko w *najbardziej wytrzymałym*.

Przeanalizujmy koszty czasowe poszczególnych kroków opisanego algorytmu. Znalezienie za pomocą wyszukiwania binarnego listy, na której powinien zostać umieszczony rozważany kontener, to $O(\log\log z)$, gdyż liczba list jest rzędu $O(\log z)$. Początkowe rozmieszczenie kontenerów na listach można więc wykonać w złożoności czasowej $O(n\log\log z)$, wykorzystując wielokrotnie wyszukiwanie binarne. Połączenie list w punkcie 1. można wykonać w czasie stałym, a w ciągu całego algorytmu takich połączeń wykonamy najwyżej $\log z$, stąd sumaryczny czas tego kroku to $O(\log z)$. Operacje opisane w punktach 2a-2c wykonujemy dla każdego odważnika, czyli łącznie $K \leqslant m$ razy, przy czym operacje opisane w punktach 2a oraz 2b mają złożoność czasową $O(1)$, a operacja z punktu 2c ma złożoność $O(\log\log z)$. Razem daje to złożoność $O(m(1+\log\log z))$. Cała funkcja *Zapakuj* ma więc złożoność równą $O((n+m)\log\log z)$. Ulepszenie wydaje się być nikłe, ale pozbyliśmy się z rozwiązania kopca, który może nie być wszystkim znany. Rozwiązanie wykorzystujące zmodyfikowaną wersję funkcji `Zapakuj` zostało zaimplementowane w plikach `odw4.cpp` oraz `odw9.pas`.

Dalsze usprawnienia możemy uzyskać, zamieniając binarne wyszukiwanie wartości K na wyszukiwanie liniowe! Może się to wydawać w pierwszej chwili dziwne, gdyż zamiana $\log m$ testów na m testów nie wydaje się być dobrym pomysłem na poprawę efektywności. Oszczędność jednak wyniknie z możliwości bazowania na rozmieszczeniu odważników z poprzedniego testu przy konstrukcji rozmieszczenia w kolejnym teście. Aby zauważyć zależność pomiędzy takimi rozmieszczeniami, załóżmy, że mamy posortowane odważniki $m_m \geqslant m_{m-1} \geqslant \cdots \geqslant m_1$ i rozpocznijmy od testu dla $K = m$. Wówczas możliwe są dwie sytuacje.

- Udaje się nam zapakować wszystkie odważniki i możemy stwierdzić, że wynikiem jest $K = m$.
- Pakujemy odważniki $m_m, m_{m-1}, \ldots, m_{i+1}$, a dla m_i nie ma już miejsca. Wówczas przechodzimy do testu dla $K = m-1$.

Istotą ulepszenia jest takie przejście do testu dla $K = m-1$, przy którym korzystamy z upakowania odważników dla $K = m$ — robimy to następująco:

- usuwamy odważnik m_m z kontenera;
- odważniki $m_{m-1}, \ldots, m_{i+1}$ pozostawiamy na dotychczasowych miejscach;
- odważnik m_i umieszczamy w kontenerze, w którym poprzednio był m_m;
- kontynuujemy rozmieszczanie kolejnych odważników: $m_{i-1}, m_{i-2}, \ldots$.

W ten sposób, kosztem modyfikacji wytrzymałości jednego kontenera przechodzimy od jednego testowanego rozmieszczania do kolejnego. Przypomnijmy, że w poprzednim rozwiązaniu konieczne było rozmieszczenie wszystkich odważników od początku. Pseudokod ostatecznej wersji rozwiązania, w której zastosowaliśmy oba ulepszenia, prezentujemy poniżej. Zakładamy przy tym dla uproszczenia, że najcięższy odważnik mieści się w jakimś kontenerze (wszystkie odważniki niemieszczące się w żadnym kontenerze możemy odrzucić na samym początku).

```
1:  Posortuj(m_i);
2:  Wyznacz(M_i,a_i);
3:  for i := 1 to m do umieszczony[i] := nigdzie;
4:  K := m; { Parametr, który będziemy zmniejszać przy kolejnych porażkach. }
5:  for i := m downto 1 do
6:  begin
7:    if Da_Się_Umieścić(m_i) then
8:      t :=Weź_Kontener_Nie_Mniejszy_Od(m_i);
9:    else
10:   begin
11:     t := umieszczony[K];
12:     umieszczony[K] := nigdzie;
13:     w_t = w_t + m_K;
14:     K := K - 1;
15:   end
16:   umieszczony[i] := t;
17:   w_t = w_t - m_i;
18: end
19: return K;
```

Dokładna implementacja algorytmu zależy od wyboru struktury, w której będziemy przechowywać kontenery. Zastosujemy do tego celu opisane wcześniej „kubełki” — tablicę $\log z$ list kontenerów o wytrzymałościach z przedziału $[M_i, M_{i+1}-1]$. Inicjalizacja struktury wymaga czasu $O(n \log\log z)$, a każda z instrukcji dostępu do kontenera (wyszukanie, wstawienie i usunięcie odważnika) jest wykonywana w złożoności $O(\log\log z)$. Daje to łączną złożoność czasową rozwiązania $O((n+m)\log\log z)$ w przypadku zastosowania efektywnego algorytmu wstępnego sortowania mas odważników. Szacowanie złożoności jest w tym przypadku nieco sztuczne, gdyż samo wczytanie wejścia wymaga wykonania około $(n+m)\log z$ operacji (wszak wejście jest w praktyce wczytywane znak po znaku, a długość zapisu dziesiętnego liczby naturalnej z to $\lfloor \log_{10} z \rfloor !$). Implementacje tego rozwiązania znajdują się w plikach `odw5.cpp` oraz `odw10.pas`.

Rozwiązanie wzorcowe – wersja druga

W pierwszej wersji rozwiązania wzorcowego odważniki umieszczaliśmy w kontenerach w kolejności od najcięższych do najlżejszych. Okazuje się, że można również poszukiwać rozwiązania, analizując odważniki począwszy od najlżejszych. Jednak w tym przypadku tak proste kryterium zachłanne, jak w pierwszej wersji rozwiązania, nie jest już

poprawne. Umieszczając za każdym razem najlżejszy odważnik w najmniej (lub najbardziej) wytrzymałym z pozostałych kontenerów, nie otrzymamy poprawnego algorytmu (zachęcamy Czytelnika do znalezienia przykładu, który pozwala to wykazać). Musimy więc zastosować nieco subtelniejszą regułę.

W dalszej części dla uproszczenia przyjmiemy założenie, że najlżejszy odważnik w zestawie ma masę 1 ($M_1 = 1$). Zauważmy, że bez straty ogólności możemy przekształcić rozważany przypadek w spełniający to ograniczenie. Jeżeli w danym zestawie mamy $M_1 > 1$, to możemy zarówno masy wszystkich odważników, jak i wytrzymałości kontenerów podzielić (całkowitoliczbowo) przez M_1. Dzielenie w przypadku odważników będzie dokładne (i tak masy wszystkich odważników są podzielne przez M_1). W przypadku kontenera, jeżeli reszta z dzielenia jego wytrzymałości przez M_1 jest niezerowa, to i tak tej reszty nie dałoby się wykorzystać przy pakowaniu, więc możemy o nią zmniejszyć kontener.

Przy założeniu, że $M_1 = 1$, wprowadźmy następującą definicję.

Definicja 1 *Rozkładem* liczby naturalnej x przy układzie mas odważników $M_1, \ldots, M_k$ nazywamy taki ciąg $r_1, \ldots, r_k$, że $0 \leqslant r_i < \frac{M_{i+1}}{M_i}$ dla $i = 1, \ldots, k-1$ oraz $x = \sum_{i=1}^{n} r_i M_i$.

Przykład 1 Przyjrzyjmy się rozkładowi liczb z przykładu w treści zadania. Mamy tam do czynienia z ciągiem mas $M = (2, 4, 12)$, co po podzieleniu przez $M_1 = 2$ daje $M' = (1, 2, 6)$. Wytrzymałości kontenerów równe 13 i 9 zostają przekształcone do wartości 6 i 4. Rozkładami kilku przykładowych liczb przy tym układzie odważników są:

- dla 5 ciąg $1, 2, 0$,
- dla 7 ciąg $1, 0, 1$,
- dla 21 ciąg $1, 1, 3$.

■

Wprowadzone pojęcie rozkładu jest analogiczne do zapisu liczby w określonym systemie pozycyjnym (np. binarnym, dziesiątkowym), gdyż układ $M_1, \ldots, M_k$ jest uogólnieniem układów postaci $b^0, b^1, \ldots, b^{k-1}$ dla $b \geqslant 2$ całkowitego. Analogicznie jak w przypadku układów pozycyjnych, można pokazać, że dla liczby x istnieje dokładnie jeden rozkład. Co więcej, można go skonstruować „tradycyjnie": pomniejszając wartość x o M_k aż do momentu, kiedy $x < M_k$, następnie pomniejszając wartość x o M_{k-1} itd. Udowodnienie opisanych własności rozkładów pozostawiamy Czytelnikowi jako łatwe ćwiczenie.

Podstawą drugiej wersji rozwiązania wzorcowego jest fakt, że możemy bez zmniejszenia wyniku zamienić każdy kontener na zestaw kontenerów o wytrzymałościach odpowiadających jego rozkładowi w układzie $M_1, \ldots, M_k$. Spostrzeżenie to formalizujemy następująco:

Twierdzenie 3 *Niech dany będzie kontener o wytrzymałości w, do którego zapakowaliśmy odważniki o masach $\mu_1 \leqslant \mu_2 \leqslant \ldots \leqslant \mu_s$. Niech dalej $r_1, \ldots, r_k$ będzie rozkładem w. Wówczas te same odważniki można zapakować do zestawu kontenerów $(r_1 \times M_1, \ldots, r_k \times M_k)$, czyli złożonego z: r_1 kontenerów o wytrzymałości M_1, …, r_k kontenerów o wytrzymałości M_k.*

Dowód Pokażemy, jak rozmieścić odważniki $\mu_1, \mu_2, \ldots, \mu_s$ w zestawie kontenerów $(r_1 \times M_1, \ldots, r_k \times M_k)$.

Rozdrobnieniem układu kontenerów R nazwiemy układ kontenerów powstały przez podzielenie niektórych kontenerów z R na mniejsze. Sformułujmy dwa przydatne dalej spostrzeżenia:

- Jeśli zestaw odważników da się rozmieścić w rozdrobnieniu układu kontenerów R, to da się także rozmieścić w układzie kontenerów R. Stąd wynika, że w trakcie rozmieszczania odważników w układzie kontenerów $(r_1 \times M_1, \ldots, r_k \times M_k)$ możemy rozdrabniać ten układ — jeśli ostatecznie rozmieścimy w nim odważniki, to daje się je także rozmieścić w układzie początkowym.

- Jeśli $r_1, \ldots, r_j$ jest rozkładem liczby w w układzie $M_1, \ldots, M_j$, to $\sum_{i=1}^{j-1} r_i \cdot M_i < M_j$. Tę zależność można wywnioskować, na przykład, z zachłannej metody konstrukcji rozkładu liczby względem $M_1, \ldots, M_j$.

Odważniki będziemy rozmieszczać w kontenerach w kolejności od najcięższego do najlżejszego: $\mu_s, \mu_{s-1}, \ldots, \mu_1$. W każdym kroku będziemy modyfikować początkowy zestaw kontenerów tak, by po umieszczeniu odważnika μ_i o masie M_j mieć zestaw pustych kontenerów $(r_1 \times M_1, \ldots, r_j \times M_j)$, gdzie $r_1, \ldots, r_j$ jest rozkładem $w - (\mu_s + \cdots + \mu_i)$ względem $M_1, \ldots, M_j$.

Stan zerowy Na początku rzeczywiście mamy układ kontenerów $(r_1 \times M_1, \ldots, r_k \times M_k)$, gdzie $r_1, \ldots, r_k$ jest rozkładem w względem $M_1, \ldots, M_k$.

Kolejny krok Niech M_ℓ będzie masą kolejnego rozważanego odważnika μ_t $(t = s, s-1, \ldots, 1)$. Jeśli $\ell < j$, to rozdrobnijmy wszystkie kontenery o wytrzymałości większej niż M_ℓ na kontenery o wytrzymałości M_ℓ — uzyskujemy nowy zestaw kontenerów $(r_1 \times M_1, \ldots, r_\ell \times M_\ell)$, gdzie — jak łatwo zauważyć — $r_1, \ldots, r_\ell$ jest rozkładem $w - (\mu_s + \cdots + \mu_{t+1})$ względem $M_1 \ldots, M_\ell$. Ponieważ μ_t mieści się w tym zestawie, więc $M_\ell \leqslant \sum_{p=1}^{\ell} r_p \cdot M_p$. To z kolei oznacza, że $r_\ell \geqslant 1$, więc w zestawie mamy kontener, w którym zmieści się μ_t. Umieszczamy w nim ten odważnik i usuwamy z zestawu zapełniony kontener (czyli zmniejszamy o jeden r_ℓ).

Przed przejściem do rozważania kolejnego odważnika mamy więc zachowany niezmiennik: zestaw pustych kontenerów to układ $(r_1 \times M_1 \ldots, r_\ell \times M_\ell)$, gdzie $r_1, \ldots, r_\ell$ jest rozkładem względem $M_1, \ldots, M_\ell$ pozostałej wytrzymałości zestawu $w - (\mu_s + \cdots \mu_t)$.

Stan końcowy Niezmiennik, który zachowujemy w trakcie rozmieszczania odważników, gwarantuje, że uda się nam umieścić wszystkie odważniki w rozdrobnieniu układu początkowego, a więc można je także umieścić w układzie początkowym, co należało pokazać.

■

Na mocy pokazanego twierdzenia, zamiast rozważać pojemniki o wytrzymałościach w_i, możemy dla każdego w_i znaleźć rozkład i rozważać problem upakowania odważników do kontenerów o wytrzymałościach należących do zbioru $\{M_1, \ldots, M_k\}$. Wynikowy zestaw kontenerów może być teraz bardzo duży, niemniej jednak rodzajów kontenerów będzie tylko $O(\log z)$. Okazuje się, że dla tak otrzymanego zestawu kontenerów poprawne jest rozwiązanie zachłanne, w którym umieszczamy odważniki w kolejności od najlżejszych do najcięższych, każdorazowo wkładając dany odważnik do kontenera o najmniejszej możliwej wytrzymałości. Po umieszczeniu odważnika w kontenerze, możemy — na mocy Twierdzenia 3 — pozostałą wytrzymałość kontenera rozłożyć na kontenery o rozmiarach

należących do zbioru $\{M_1,\ldots,M_k\}$, dzięki czemu w żadnym momencie nie będziemy operować na innych wytrzymałościach kontenerów niż wymienione.

Zanim udowodnimy poprawność tego rozwiązania, zapiszmy je w pseudokodzie:

```
 1: Posortuj(m_i);
 2: Wyznacz(M_i,a_i);
 3: { Konstruujemy rozdrobnienie początkowego zestawu kontenerów }
 4: { na kontenery o wytrzymałościach M_1,...,M_k. }
 5: for i := 1 to k do kontenery[i] := 0;
 6: for i := 1 to n do Rozłóż(w_i);
 7: { Rozmieszczamy odważniki od najlżejszych do najcięższych. }
 8: przetworzone := 0;
 9: for i := 1 to k do { Rozważamy odważniki o masie M_i. }
10:    for j := 1 to a_i do
11:    begin
12:       gdzie := i;
13:       while gdzie ⩽ k and kontenery[gdzie] = 0 do
14:          gdzie := gdzie + 1;
15:       if gdzie > k then
16:          { Nie da się umieścić odważnika — kończymy. }
17:          return przetworzone;
18:       else
19:          przetworzone := przetworzone + 1;
20:       kontenery[gdzie] := kontenery[gdzie] - 1;
21:       Rozłóż(M_gdzie - M_i);
22:    end
23: { Wszystkie odważniki udało się rozmieścić. }
24: return m;
25:
26: procedure Rozłóż(x);
27: begin
28:    { Rozkład wykonujemy za pomocą algorytmu zachłannego. }
29:    for i := k downto 1 do
30:    begin
31:       kontenery[i] := kontenery[i] + ⌊x/M_i⌋;
32:       x := x - M_i · ⌊x/M_i⌋;
33:    end
34: end
```

Złożoność. Rozkład w_i możemy znaleźć w złożoności czasowej $O(\log z)$ za pomocą opisanego powyżej algorytmu zachłannego, co daje koszt rozdrobnienia kontenerów równy $O(n\log z)$. Złożoność czasowa głównej pętli powyższego algorytmu to, jak łatwo widać, $O(m\log z)$, gdyż dla każdego odważnika co najwyżej raz przeglądamy listę wszystkich wytrzymałości kontenerów oraz rozdrabniamy co najwyżej jeden kontener. Stąd cały algorytm ma złożoność $O((n+m)\log z)$, jeżeli założyć, że sortowanie odważników

wykonujemy bardziej efektywną metodą. Algorytm jest więc podobnie efektywny jak pierwsze rozwiązanie wzorcowe.

Poprawność. Zastanówmy się na koniec nad poprawnością tego algorytmu. Z Twierdzenia 1 wynika, że możemy umieszczać odważniki, począwszy od najlżejszych. Pozostaje wyjaśnić, dlaczego możemy rozważany odważnik za każdym razem umieszczać w najmniejszym z pozostałych kontenerów.

Porównajmy w tym celu działanie naszego algorytmu A i pewnego algorytmu optymalnego O, który także przydziela odważniki do układu rozdrobnionych kontenerów. Oznaczmy przez x masę najmniejszego odważnika, który został różnie rozmieszczony w tych dwóch rozwiązaniach. Powiedzmy, że:

- w algorytmie A włożyliśmy x do kontenera K_1,
- w algorytmie O włożyliśmy x do kontenera K_2 (o wytrzymałości $w_{K2} > w_{K1}$; ponadto zachodzi $w_{K1} \mid w_{K2}$).

Po umieszczeniu w kontenerze odważnika x w algorytmie A rozkładamy kontener o wytrzymałości $w_{K1} - x$, a w algorytmie O rozkładamy kontener o wytrzymałości $w_{K2} - x = (w_{K2} - w_{K1}) + (w_{K1} - x)$. Porównajmy zestawy kontenerów w obu rozwiązaniach po przeprowadzonej operacji.

- W algorytmie A mamy pusty cały kontener o wytrzymałości w_{K2} i kontenery $L_1, \ldots, L_t$ powstałe z rozdrobnienia kontenera o wytrzymałości $w_{K1} - x$.
- W algorytmie O pozostał pusty kontener o wytrzymałości w_{K1}, a rozdrobniliśmy kontener o wytrzymałości $(w_{K2} - w_{K1}) + (w_{K1} - x)$. Ponieważ pierwszy składnik tej sumy jest podzielny przez w_{K1}, to oznacza, że w wyniku rozkładu dostaliśmy kontenery $N_1, \ldots, N_s$ oraz $L_1, \ldots, L_t$, gdzie kontenery z pierwszej grupy mają wytrzymałości podzielne przez w_{K1}, a kontenery z drugiej grupy są identyczne jak otrzymane z rozkładu K_1 w algorytmie A.

Przedstawione porównanie wykazuje, że układ kontenerów w algorytmie optymalnym jest rozdrobnieniem układu kontenerów z algorytmu A, stąd każde rozmieszczenie uzyskane za pomocą algorytmu optymalnego zostanie także znalezione za pomocą algorytmu A, co dowodzi optymalności algorytmu A.

Implementacje przedstawionego w tym rozdziale rozwiązania wzorcowego znajdują się w plikach `odw.cpp` i `odw7.pas`.

Testy

Zadanie było sprawdzane na 10 zestawach testów. W poniższej tabelce n oznacza liczbę kontenerów, m to liczba odważników, a K to liczba odważników rozmieszczonych w kontenerach w optymalnym rozwiązaniu.

W opisach testów przyjęto następujące nazewnictwo:

- *test poprawnościowy:* mały test, w którym do pewnego kontenera trzeba włożyć zarówno duże, jak i małe odważniki;

- *test małych odważników:* mały test, w którym jest więcej małych odważników; odważniki trzeba dobrze rozdzielić między dwa kontenery, tak by zmieściły się najcięższe z nich;
- *test potęg dwójki:* duży losowy test, w którym masy odważników są potęgami dwójki;
- *test grupowania:* test losowy, w którym liczba kontenerów jest zbliżona do połowy liczby odważników, a suma wytrzymałości kontenerów jest równa dokładnie sumie mas odważników; aby uzyskać optymalne rozmieszczenie, trzeba umieszczać w kontenerach zarówno małe, jak i duże odważniki; w testach „grupowania 2" masy odważników są potęgami 2, natomiast w testach „grupowania 3-2" są one elementami ciągu $1, 3, 6, 18, 36, \ldots$;
- *test całkowitego rozmieszczenia:* duży test losowy, w którym masy odważników są potęgami dwójki; w kontenerach daje się rozmieścić wszystkie odważniki;
- *test jednostkowych mas:* duży test losowy, w którym połowa odważników ma masy 1, a masy pozostałych są potęgami 2 („jednostkowe masy 2") albo elementami ciągu $1, 3, 6, 18, 36, \ldots$ („jednostkowe masy 3-2"); w rozwiązaniu optymalnym do każdego kontenera trzeba włożyć odważnik o masie równej 1.

Nazwa	n	m	K	Opis
odw1a.in	3	9	8	test poprawnościowy
odw1b.in	4	6	5	test poprawnościowy
odw1c.in	5	4	0	test z zerowym wynikiem
odw1d.in	2	8	7	test małych odważników
odw2a.in	3	9	8	test poprawnościowy
odw2b.in	3	7	6	test poprawnościowy
odw2c.in	2	5	3	test na przypadek brzegowy
odw2d.in	3	8	7	test małych odważników
odw3a.in	100000	100000	96636	test potęg dwójki
odw3b.in	50047	100000	99999	test grupowania 3-2
odw4a.in	100000	100000	96663	test potęg dwójki
odw4b.in	50117	100000	99999	test grupowania 2
odw4c.in	50000	100000	100000	test jednostkowych mas 3-2
odw5a.in	100000	100000	96634	test potęg dwójki
odw5b.in	49843	100000	99999	test grupowania 3-2
odw6a.in	100000	100000	96693	test potęg dwójki
odw6b.in	49986	100000	99999	test grupowania 2
odw7a.in	100000	100000	96699	test potęg dwójki

Nazwa	n	m	K	Opis
odw7b.in	50064	100000	99999	test grupowania 2
odw8a.in	100000	100000	96646	test potęg dwójki
odw8b.in	49982	100000	99999	test grupowania 2
odw9a.in	100000	100000	96696	test potęg dwójki
odw9b.in	50001	100000	99999	test grupowania 3-2
odw9c.in	100000	100000	100000	test całkowitego rozmieszczenia
odw9d.in	50000	100000	100000	test jednostkowych mas 2
odw10a.in	100000	100000	96726	test potęg dwójki
odw10b.in	50004	100000	99999	test grupowania 3-2
odw10c.in	100000	100000	100000	test całkowitego rozmieszczenia
odw10d.in	50000	100000	100000	test jednostkowych mas 2

Rozwiązania zawodników o wykładniczej złożoności czasowej względem n oraz m przechodziły pierwsze dwa testy. Proste rozwiązania zachłanne z błędnym kryterium (na przykład rozwiązanie umieszczające odważniki od najlżejszych w najmniejszych czy też największych możliwych kontenerach) nie przechodziły żadnych testów. Wszystkie wersje rozwiązania wzorcowego przechodziły wszystkie testy.