PROGRAMOWANIE ZACHŁANNE

Programowanie zachłanne skierowane jest na rozwiązywanie **zadań optymalizacyjnych**, czyli takich, w których poszukuje się najlepszego (najgorszego) rozwiązania. Polega ono na dokonywaniu zawsze takiego wyboru, który w danej chwili jest maksymalny (minimalny). Program poszukuje więc lokalnego maksimum (minimum), co nie zawsze prowadzi do optymalnego rozwiązania w sensie ogólnym, ale w wielu przypadkach jest wystarczające. Algorytm zachłanny w każdej iteracji wykonuje określone operacje, zmierzające do zamierzonego celu. Metody programowania, które opierają się zdobywaniu i wykorzystywaniu doświadczenia w trakcie rozwiązywania problemów na każdym kolejnym etapie nazywamy **heurestycznymi**. Technikę tę stosuje się wtedy, gdy maksymalizowana (minimalizowana) jest pewna wartość. W każdej kolejnej iteracji algorytm realizuje określone operacje pod kątem optymalizacji na danym etapie.

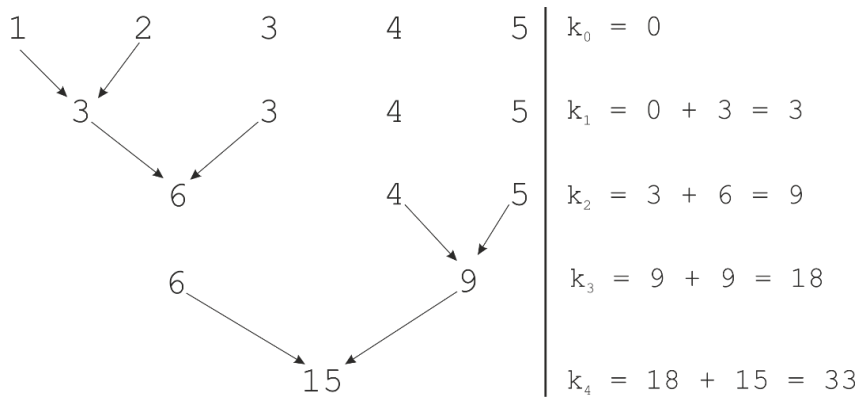
Minimalizacja kosztów sklejania par liczb

Dobrym zastosowaniem programowania zachłannego jest minimalizacja kosztów sklejania ze sobą par liczb. Załóżmy, że mamy dany ciąg n nieujemnych $k_1, k_2, k_3, k_4, k_5,$. Sklejenie dwóch liczb polega na wybraniu dwóch liczb z podanego ciągu i zastąpienie ich sumą tych liczb. Celem jest sklejenie wszystkich liczb i uzyskanie jednej wartości w taki sposób, by całkowity koszt sklejania był minimalny, przy czym kosztem jednego sklejenia nazywamy sumę wartości sklejanych liczb. Można to zapisać w pseudokodzie następująco:

```
koszt ← 0
      dopóki istnieją co najmniej dwie liczby do sklejenia wykonuj
            zastąp aktualnie dwie najmniejsze liczby ich sumą
            koszt ← koszt + suma
```

Postępując zachłannie, w każdym kolejnym kroku wybieramy sklejenie o najmniejszym w danym momencie koszcie sklejenia, co prowadzi do zadowalającego wyniku – z uwagą, że można znaleźć taki ciąg liczb, dla którego algorytm nie wyznaczy minimalnego rozwiązania.

Przykład
Dany jest ciąg liczb: $1, 2, 3, 4, 5$. Wyznacz koszt sklejenia tych liczb.



Spróbujmy wyznaczyć koszt sklejania tych samych liczb w inny sposób, np. od lewej strony do prawej.

Ciąg liczb	Sklejane liczby	Koszt sklejenia pary	Koszt skumulowany
$\{1, 2, 3, 4, 5\}$	–	–	0
$\{1, 2, 3, 4, 5\}$	1 i 2	3	$0 + 3 = 3$
$\{3, 3, 4, 5\}$	3 i 3	6	$3 + 6 = 9$
$\{6, 4, 5\}$	6 i 4	10	$9 + 10 = 19$
$\{10, 5\}$	10 i 5	15	$19 + 15 = 34$
$\{15\}$	–	–	34

Jak widać, koszt sklejania nieznacznie powiększył się i wynosi 34. Dobranie innych par liczb zakończy się jeszcze większym kosztem. Łatwo domyślić się, że aby ustalić maksymalny koszt sklejania, należy dobierać w pary największe elementy ciągu.

Ciąg liczb	Sklejane liczby	Koszt sklejenia	Koszt skumulowany
$\{1, 2, 3, 4, 5\}$	–	–	0
$\{1, 2, 3, 4, 5\}$	5 i 4	9	$0 + 9 = 9$
$\{1, 2, 3, 9\}$	9 i 3	12	$9 + 12 = 21$
$\{1, 2, 12\}$	12 i 2	14	$21 + 14 = 35$
$\{1, 14\}$	14 i 1	15	$35 + 15 = 50$
$\{15\}$	–	–	50

Zatem warianty kosztów sklejania wahają się w tym przykładzie liczbowym od 33 do 50. Złożoność przedstawionego algorytmu wynika z dwukrotnego szukania minimum (lub maksimum) w każdym kroku. W każdej iteracji liczba wyrazów maleje o jeden, a zatem liczba porównań wynosi:

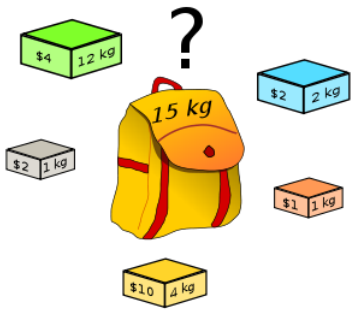
$$(n-1) + (n-2) + \cdots + 1 + (n-2) + (n-3) + \cdots + 1 = \frac{n(n-1)}{2} + \frac{(n-1)(n-2)}{2} = n^2 - 2n + 1 =$$

$$= O(n^2)$$

Złożoność algorytmu można poprawić poprzez wykorzystanie metody „dziel i zwyciężaj” do znajdowania pierwszego i drugiego wyrazu minimalnego (lub maksymalnego).

Problem plecakowy

Znanym w informatyce przykładem zastosowania programowania zachłannego jest tzw. **problem plecakowy** (ang. *knapsack problem*). Polega on na zapakowaniu plecaka (bagażnika samochodu, luku bagażowego w samolocie itp.) – o ograniczonej pojemności – przedmiotami o określonej wadze i wartości tak, aby całkowita wartość spakowanych przedmiotów była jak największa. Metoda zachłanna nie gwarantuje optymalnego rozwiązania, ale otrzymany wynik z pewnością będzie korzystny.



Problem plecakowy występuje w dwóch odmianach. Jeśli mamy do dyspozycji nieograniczoną liczbę każdego dostępnego przedmiotu, to mówimy o **ogólnym problemie plecakowym**. Jeśli zaś mamy do dyspozycji co najwyżej po jednej sztuce każdego przedmiotu, wówczas mówimy o **decyzyjnym problemie plecakowym**.

Rozwiązując ogólny problem plecakowy można przyjąć strategię wyboru przedmiotów o jak największej wartości, ale mieszczących się w plecaku lub strategię wyboru rzeczy, które ważą jak najmniej albo uwzględnić obie strategie jednocześnie, przy czym na ogół ostatnie rozwiązanie jest najkorzystniejsze. Wyboru dokonujemy zatem na podstawie stosunku wartości W do wagi C każdego z n przedmiotów

$$\frac{W_i}{C_i}$$

W pierwszej kolejności wybieramy jak największe ilorazy. Zakładamy więc, że przedmioty uporządkowane są w kolejności odpowiadającej temu kryterium

$$\frac{W_1}{C_1} \geq \frac{W_2}{C_2} \geq \cdots \geq \frac{W_n}{C_n}$$

Weźmy pod uwagę następujący przykład liczbowy (zakładając, że pojemność plecaka wynosi np. 11 jednostek).

i (numer przedmiotu)	1	2	3	4	5
$W[i]$ (wartość przedmiotu o numerze i)	8	3	1	2	1
$C[i]$ (waga przedmiotu o numerze i)	4	2	1	3	7

Pakowanie plecaka rozpoczynamy od przedmiotu o numerze 1, ponieważ ma najkorzystniejszy stosunek $\frac{W_i}{C_i}$. Do plecaka zmieszczą dwa takie przedmioty o łącznej wartości 16. Do dyspozycji mamy więc jeszcze 3 jednostki pojemności plecaka. Drugim w kolejności przedmiotem jest numer 2, a do plecaka zmieści się tylko jeden taki przedmiot. W tym momencie w plecaku mamy przedmioty o łącznej wartości 19 i pozostała nam jeszcze jedna jednostka pojemności. Do 100% zapakowania plecaka idealnie nadaje się więc przedmiot nr 3 o wartości 1. W pełni zapakowany plecak będzie więc miał wartość 20 i jest to najlepsze rozwiązanie.

Ćwiczenia do wykonania na zajęciach

C1.
Algorytmem zachłannym możemy też obsłużyć problem wydawania reszty. Załóżmy, że mamy do dyspozycji nieograniczony zbiór monet o określonych nominałach i mamy wydać resztę używając jak najmniej tych monet. Przykładowo, jeśli dysponujemy zbiorem monet o nominałach $\{20, 10, 5, 2, 1\}$ i mamy wydać resztę 37 zł, to pojedynczo wydajemy monety 20, 10, 5 i 2 zł, aby osiągnąć cel w postaci minimalnego zestawu monet.

Opis słowny algorytmu:
Użytkownik podaje dane do tablicy nominałów oraz żądaną resztę, przy czym tablica nominałów powinna zawierać wyłącznie liczby 1, 2 i 5 oraz ich dziesiętne wielokrotności, a także powinna być posortowana w kolejności malejącej. Algorytm wydawania reszty zaczyna swoje działanie od sprawdzenia, czy wśród nominałów jest wartość mniejsza lub równa reszcie R. Jeśli odnaleziony nominał N równy jest R, to spełniony jest warunek stopu i algorytm kończy się. Jeśli nie, to ustalamy największą możliwą liczbę monet L nominału mniejszego od R dzieląc całkowicie R przez N. Następnie od reszty R odejmujemy iloczyn $L \cdot N$ i szukamy dalej wśród kolejnych nominałów aż wypłacenia całej reszty.

Na podstawie opisu słownego algorytmu napisz w języku C# (Python, Java, C++) program, który ustala sposób wydania reszty przy użyciu najmniejszej liczby monet. Zadbaj o czytelny i przyjazny interfejs użytkownika.

C2.
Liczbami półpierwszymi nazywamy liczby naturalne, które są iloczynem dwóch liczb pierwszych. Przykładowo liczbami taki są: 34, bo $34 = 2 \cdot 17$, a także $841 = 29 \cdot 29$ itd.

W pliku `liczby.txt` znajduje się 500 liczb naturalnych, z których każda ma co najwyżej 6 cyfr. Napisz program, który wyodrębni wszystkie liczby półpierwsze z pliku `liczby.txt`. Program powinien zapisać wyodrębnione liczby do pliku `liczby_wynik.txt`.

C3.
Napisz program, który w zadanym przez użytkownika przedziale $< 2,\ 1000 >$ liczb naturalnych wyszukuje wszystkie liczby względnie pierwsze (ang. *relatively prime integers, coprime*) względem zadanej przez użytkownika liczby p. Zadbaj o czytelny i zrozumiały interfejs użytkownika.

Uwaga:
Liczba naturalna m jest względnie pierwsza z liczbą naturalną n wtedy i tylko wtedy, gdy $NWD(m, n) = 1$. Definicja ta oznacza, iż liczby n i m nie posiadają wspólnych podzielników za wyjątkiem liczby 1. Przykładem takiej pary liczb jest np. 6 i 35.

Zadania domowe

Z1.
Przeanalizuj podane wyżej informacje teoretyczne i na ich podstawie napisz w języku C# (Python, Java, C++) program, który wyznacza minimalny i maksymalny koszt sklejenia ciągu liczb podanych przez użytkownika w sposób iteracyjny i /lub rekurencyjny (mile widziany). Dodatkowo program powinien wyświetlać liczbę operacji poświęconych na wyszukiwanie elementów do sklejenia oraz liczbę sklejeń. Zadbaj o czytelny i przyjazny interfejs użytkownika.

Z2.
Na polu stoją dwa zbiorniki o pojemności 1000 litrów każdy, przy czym początkowo pierwszy z nich jest pełny, a drugi pusty. Masz do dyspozycji trzy czerpaki do wody o pojemnościach 2, 3 i 11 litrów. Napisz program, który określi najmniejszą łączną liczbę użycia czerpaków do nalania wody z pierwszego zbiornika do drugiego z dokładnością do jednego litra w zakresie od 1 do 1000 litrów. Program powinien pokazać również liczbę użyć każdego czerpaka. Zadbaj o czytelny i zrozumiały interfejs użytkownika.

Uwaga:
Czerpaki mogą być użyte nie tylko do nalewania wody ze zbiornika pierwszego do drugiego, ale też z powrotem, czyli do przelania nadmiaru wody z drugiego zbiornika do pierwszego. W razie zaistnienia dwóch konkurencyjnych rozwiązań (o takiej samej łącznej liczbie użycia czerpaków) nalewanie wody do drugiego zbiornika ma pierwszeństwo przed przelewaniem nadmiaru z powrotem do pierwszego zbiornika.

Dla przykładu, w celu nalania do drugiego zbiornika 9 litrów wody możemy użyć trzykrotnie czerpaka 3-litrowego, ale warunki zadania spełnia jednokrotne użycie czerpaka 11-litrowego do nalania wody i jednokrotne użycie czerpaka 2-litrowego do przelania nadmiaru wody z powrotem do pierwszego zbiornika. Natomiast do przelania 7 litrów wody należy użyć dwukrotnie czerpaka 2-litrowego i jednokrotnie czerpaka 3-litrowego, mimo że do tego samego celu prowadzi jednokrotne użycie czerpaka 11-litrowego do nalania wody i dwukrotne użycie czerpaka 2-litrowego do przelania nadmiaru wody z powrotem.

Z3.
Jeszcze niedawno PIN karty bankomatowej weryfikowany był przez moduł HSM (ang. *Hardware Security Model*). Oprócz kodu PIN i numeru karty moduł ten potrzebował tablicy konwersji, tj. 16-elementowej tablicy cyfr od 0 do 9, która była unikalna dla każdego banku. Schemat działania modułu HSM jest następujący:

- szyfrowany jest numer karty, w wyniku czego otrzymuje się liczbę zapisaną szesnastkowo,
- z otrzymanej liczby pozostawia się jedynie cztery pierwsze cyfry szesnastkowe,
- pozostawione cyfry zamienia się na dziesiętne z wykorzystaniem tablicy konwersji – cyfrze szesnastkowej przypisuje się odpowiednik dziesiętny, który znajduje się w komórce tablicy o równym jej indeksie.

Tak otrzymany numer powinien być zgodny z kodem PIN.

Przykład
Bankowa tablica konwersji $T = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5)$.

Zaszyfrowany numer karty $N = A8B3\ 1022\ E223\ 1022$.

Pozostawiamy pierwsze cztery cyfry, tj $A8B3$ i zamieniamy je na cyfry dziesiętne.

Znak	**A**	**8**	**B**	**3**
Indeks tablicy T	**10**	**8**	**11**	**3**
Wartość	**9**	**7**	**0**	**2**

Zatem wprowadzany kod PIN powinien zawierać kolejno cyfry 9702.

Na podstawie powyższych rozważań oraz znajomości zaszyfrowanego numeru karty napisz w języku C# (Python, Java, C++) program, który oblicza kod PIN, przy czym numer karty podaje użytkownik.

Z4.
Metoda szyfrowania płotowego polega na tym, że znaki tekstu jawnego zapisuje się w taki sposób, aby tworzyły kształt przypominający płot. Tekst zaszyfrowany (szyfrogram) otrzymuje się odczytując kolejne wiersze tak utworzonej konstrukcji. Klucz tego algorytmu to wysokość płotu, czyli liczba wierszy.

Zaszyfrujmy dla przykładu wyraz `KRYPTOANALIZA` stosując płot o wysokości 3 wierszy. Po wprowadzeniu tekstu jawnego do tablicy uzyskujemy następujący efekt:

```
K           T           A           A
  R       P   O       N   L       Z
    Y           A           I
```

Odczytując szyfrogram wierszami uzyskujemy wynik: `KTAARPONLZYAI`.

Na podstawie powyższych informacji napisz program, który szyfruje i deszyfruje tekst podany przez użytkownika metodą płotową, przy czym użytkownik podaje wysokość płotu $2 \leq h \leq 5$ (domyślnie płot powinien być ustawiony na $h = 3$).

Termin przysłania rozwiązanych zadań mija w dniu 1**2 maja 2023 r.** (o północy). Zasady nazewnictwa plików są takie same, jak dotychczas.