
Zadanie 1

(a) Zaimplementuj algorytmy rozwiązujące podane poniżej problemy (zastosuj programowanie dynamiczne bądź strategię zachłanną).
(b) Komunikacja programu z użytkownikiem powinna się odbywać za pomocą plików o podanych niżej formatach.

Problem 1 (Modyfikacja koloru kwiatu tulibajtu)

W instytucie genetyki roślin w Bajtlandzie prowadzone są badania nad nowymi odmianami tulibajtu z wykorzystaniem technik inżynierii genetycznej. Botanicy zmierzają w kierunku pozyskania wielu interesujących barw tego pięknego kwiatu. Wśród setek barwników roślinnych wyróżnia się trzy główne grupy związków chemicznych. Są to karotenoidy, betalainy oraz flawonoidy. Ostatnia z tych grup ma największy wpływ na ostateczny kolor kwiatów. Do flawonoidów należy wiele klas związków, takich jak antocyjaniny (odpowiedzialne za kolor pomarańczowy, czerwony, purpurowy, fioletowy i niebieski), aurony, chalkony (żółty), flawony i flawonole (postrzegane jako bezbarwne lub bladożółte). Spośród znanych barwników roślinnych wybrano podstawowe i oznaczono je kolejnymi znakami alfabetu systemu 16-kowego, tj. 0, 1, ..., 9, A, B, ..., F. Ostatnio prowadzone badania skupiają się na odpowiednim klasyfikowaniu i krzyżowaniu materiału genetycznego różnych odmian tulibajtu. Materiał genetyczny każdej odmiany jest charakteryzowany za pomocą specjalnie zdefiniowanego ciągu oznaczeń substancji barwiących, które mają największy wpływ na barwę kwiatu tejże odmiany. Np. tulibajt biały ma **charakterystykę** DDAF6AB34ADF, natomiast tulibajt purpurowy BD16A436BAF. Genetycy zauważyli, że najbardziej intensywne barwy kwiatów uzyskuje się krzyżując takie odmiany, dla których **subkod genetyczny** jest opisany za pomocą co najmniej 6 substancji (być może tych samych ale zapisanych na różnych pozycjach). Subkod genetyczny wyznaczony dla dwóch odmian stanowi uporządkowany (najdłuższy z możliwych) ciąg znaków, które w niezmienionej kolejności występują zarówno w charakterystyce jednej jak i drugiej odmiany. Np. subkodem tulibajtu białego i purpurowego jest D6A4AF, ale również DA6BAF jak i D6A3AF. Dwie odmiany mogą mieć kilka subkodów, ale każdy z nich ma tyle samo znaków. Jeśli subkod dwóch odmian ma co najmniej 6 znaków, skrzyżowanie pary da intensywną barwę kwiatu. Botanicy wyznaczyli charakterystyki wielu odmian tulibajtu i chcą sprawdzić, które z nich warto krzyżować. Zwrócili się z zadaniem do informatyków.

Przykład

Po skrzyżowaniu dwóch odmian o charakterystykach odpowiednio
DDAF**6A**B3**4A**D**F** i B**D**1**6A4**36B**AF**
uzyskamy odmianę z subkodem genetycznym
D6A4AF albo D6A3AF albo D6ABAF albo DA6BAF.

Dane:

- W pierwszym wierszu pliku in0301.txt znajduje się liczba n typu int oznaczająca ilość par odmian tulibajtu, dla których należy policzyć długość subkodu.
- W kolejnych 2n liniach umieszczone są charakterystyki odmian tulibajtu. Dla każdej kolejnej pary charakterystyk należy wyznaczyć długość subkodu.

Wyjście:

- W każdej i-tej (i=1, …, n) linii pliku out0301.txt wypisz: liczbę reprezentującą długość subkodu wyznaczonego dla i-tej pary charakterystyk oraz (po spacji) jeden z subkodów odpowiadających i-tej parze charakterystyk.

Przykład

In0301.txt
3 // n
45A64D5F5B342
456345AC6AB7345
879A7D680A968
67A896BD789
567456D6E74F56
352D4E5CA73B56
Out0301.txt
8 45645B34
6 7A8968
6 545756

Problem 2 (Weekend Modnisi) (patrz [SDK])

Koniec najbliższego tygodnia Modnisia zamierza spędzić na Wyspach Kanaryjskich. Przejrzała całą szafę i oczywiście nie ma w co się ubrać.
(1) Tylko jedna turkusowa sukienka od 9Fashion, (2) tylko jedna sukienka od G-Star, …, (101) tylko jedna sukienka (z dużym dekoltem) od Desigual, …, (211) tylko jedna spódniczka od Levi's, …, (489) tylko jedna spódniczka od Saint Tropez, …, (1968) tylko jeden T-shirt od b.young, …, (3895) tylko jeden T-shirt od Tommy Hilfiger, …, (6894) tylko jedna para niebieskich butów od Nike Sportswear, …, (8694) tylko jedna para czarnych butów od Dr. Martens, …. I jak tu ładnie się ubrać ☹?
Dodatkowo, okazało się, że podróżując LOT'em Modnisia może zabrać (po dodatkowej opłacie) bagaż o wadze co najwyżej W kg (W∈N, W∈[5, 30]). Mocno przesadzone przepisy LOT'u na pewno nie zepsują Modnisi weekendu. Zdecydowała, że weźmie tylko najpotrzebniejsze ubrania, tzn. tak spakuje walizkę, aby jej koszt był możliwie najwyższy, przy uwzględnieniu bezsensownie zaniżonego przez LOT maksymalnego udźwigu bagażu. Ponieważ weekend już blisko, Modnisi potrzebna jest pomoc w wyselekcjonowaniu ubrań, które ze sobą zabierze na wycieczkę. Problem jest następujący:
Mamy n ubrań o wartościach (cenach) p_i i wagach w_i (i=1, ..., n). Ubrania są ponumerowane (każde ubranko po jednej sztuce). Należy znaleźć $\max\{\sum_{i=1}^{n} p_i x_i\}$ przy warunkach: $\sum_{i=1}^{n} w_i x_i \le W$; $x_i \in \{0, 1\}$; $p_i, w_i, W \in N_+ \cup \{0\}$, i=1,...,n.

Dane:

- W pierwszej linii pliku In0302.txt znajdują się dwie (oddzielone znakiem spacji) dodatnie liczby całkowite: n (ilość możliwych przedmiotów) i W (udźwig walizki),
- W kolejnych n liniach znajdują się pary liczb reprezentujące wartość (cenę) ubrania p_i i jego wagę w_i (i=1, ..., n).

Wyjście:

- W kolejnych liniach wypisz możliwe rozwiązania powyższego problemu. Liczby reprezentujące numery wybranych ubrań oddziel pojedynczymi spacjami.

Przykład

In0302.txt
4 6 // n W
2 1 // p_1 w_1
3 2 // p_2 w_2
3 4 // p_3 w_3
4 5 // p_4 w_4
Out0302.txt
1 4 // pierwsze rozwiązanie
2 3 // drugie rozwiązanie

Problem 3 (Kruskal) (patrz [SDK], [CLR], [RN])

Zaimplementuj algorytm Kruskala wyznaczania minimalnego drzewa rozpinającego.
Za strukturę danych do reprezentacji grafu przyjmij tablicę list incydencji. Do sortowania zastosuj metodę o złożoności O(mlgm) (m – liczba krawędzi grafu).

Dane:

- Graf G=<V,E> skończony, spójny, niezorientowany z wagami.
- Pierwsza linia pliku In0303.txt zawiera liczbę $n \in N_+$ oznaczającą ilość wierzchołków grafu.
- W następnych n liniach znajduje się opis kolejnych list incydencji (zapis „2 3, 8 4, 9 8" w drugiej linii oznacza, że wierzchołek 1 jest połączony z wierzchołkami 2, 8 i 9 pojedynczymi krawędziami o wagach odpowiednio 3, 4 i 8).

Wyjście:

- W pierwszej linii pliku Out0303.txt wypisz krawędzie (i ich wagi) tworzące minimalne drzewo rozpinające.
- W drugiej linii pliku zapisz liczbę naturalną będąca sumą wag krawędzi minimalnego drzewa rozpinającego.

Przykład
In0303.txt
9 17
2 3, 8 4, 9 8
1 3, 3 6, 4 5, 9 6
2 6, 4 4, 9 1
2 5, 3 4, 5 3, 9 2
4 3 6 1 9 5
5 1 7 5 9 2
6 5 8 4 9 2
1 4 7 4 9 2
1 8 2 6 3 1 4 2 5 5 6 2 7 2 8 2
jedno z rozwiązań:
Out0303.txt
3 9 [1], 5 6 [1], 4 9 [2], 6 9 [2], 7 9 [2], 8 9 [2], 1 2 [3], 1 8 [4],
17

Wskazówka:
Idea algorytmu Kruskala:
(1) Posortuj krawędzie grafu G=<V,E> ze względu na wartości wag (zapamiętaj je w tablicy E)
(2) Utwórz początkowy podział P zbioru V jako rodzinę jednoelementowych zbiorów $\{v_i\}$ ($(i=1,...,n)$, gdzie $v_i \in V$) reprezentujący las początkowy ($G_i=<V_i,E_i>$, $V_i=\{v_i\}$, $E_i=\emptyset$, $i=1,...,n$),
(3) $F=\emptyset$ - zainicjalizuj zbiór krawędzi minimalnego drzewa rozpinającego,
(4) wybierz spośród nieanalizowanych krawędzi tablicy E krawędź o najmniejszej wadze,
(5) jeśli wybrana krawędź łączy dwa wierzchołki z różnych zbiorów podziału,
(5a) dodaj ją do zbioru krawędzi tworzonego minimalnego drzewa rozpinającego F,
(5b) połącz podzbiory zawierające wierzchołki dodanej krawędzi,
(6) powtarzaj kroki (4), (5) dopóki zbiór F nie będzie zawierał n-1 krawędzi.

```
typedef struct Edge{
      int weight;
      int beg, end;
      }*PEdge;

void Kruskal (int n, int m, Edge E[], PEdge F[]) {
PEdge e; int p, q, j=0;
HeapSort(m, E);                          //sortowanie w porządku niemalejącym ze względu na wagi
for (int i=1; i<=n;i++) X[i]=i;          //początkowy podział zbioru wierzchołków,
while(card(F)<n-1) {
      e = min(E); p = Find(X, e->beg);
      q = Find(X, e->end);
      if(!Equal(p, q)) {
            Union(X, p, q);
            F[++j]=e; }}}
```

Problem 4 (Jarnika-Prima) (patrz [DA])
Zaimplementuj algorytm Jarnika-Prima (zgodnie z algorytmem podanym w notatkach) wyznaczania minimalnego drzewa rozpinającego.
Za strukturę danych do reprezentacji grafu przyjmij tablicę list incydencji. Do sortowania zastosuj metodę o złożoności O(mlgm) (m – liczba krawędzi grafu).

Dane:

- Graf G=<V, E> skończony, spójny, niezorientowany z wagami.
- Pierwsza linia pliku In0304.txt zawiera liczbę $n \in N_+$ oznaczającą ilość wierzchołków.
- W następnych n liniach znajduje się opis kolejnych list incydencji (zapis „2 1, 5 2" w drugiej linii oznacza, że wierzchołek 1 jest połączony z wierzchołkami 2 i 5 pojedynczymi krawędziami o wagach odpowiednio 1 i 2).

Wyjście:

- W pierwszej linii pliku Out0304.txt wypisz krawędzie (i ich wagi) tworzące minimalne drzewo rozpinające.
- W drugiej linii pliku zapisz liczbę naturalną będąca sumą wag krawędzi minimalnego drzewa rozpinającego.

Przykład
In0304.txt
13
2 1 5 2
1 1 3 1 13 2
2 1 4 2 11 3 12 1 13 2
3 2 5 3 11 2
1 2 4 3 6 1 8 2 11 1
5 1 7 4 11 4
6 4 8 3 12 3
5 2 7 3 10 1 11 1
11 1
8 1 11 2
3 3 4 2 5 1 6 4 8 1 9 1 10 2
3 1 7 3
2 2 3 2
Out0303.txt
1 2 [1], 2 3 [1], 3 12 [1], 1 5 [2], 5 6 [1], 5 11 [1], 8 11 [1], 8 10 [1], 9 11 [1], 3 4 [2], 2 13 [2], 7 8 [3]
17

Problem 5 (Algorytm kompresji i dekompresji Huffmana)
Zaimplementuj algorytm kompresji i dekompresji metodą Huffmana (zgodnie z algorytmem podanym w notatkach). Po skompresowaniu danych umieść zawartość skompresowanego pliku w oddzielnym pliku Huff.txt i poddaj go dekompresji.

Dane:

- W pierwszej linii pliku In0305.txt znajduje się tekst, który ma zostać poddany kompresji.

Wyjście:

- W pierwszej linii pliku Out0305.txt przedstaw tablicę częstości wystąpień poszczególnych znaków w tekście.
- W drugiej linii pliku przedstaw uporządkowaną niemalejąco listę drzew jednowęzłowych.

- W trzeciej linii pliku Out0305.txt wypisz liście drzewa Huffmana w porządku od lewej strony do prawej.
- W czwartej linii pliku Out0305.txt wypisz kody Huffmana.
- Po linii oddzielającej, złożonej ze znaków „/”, umieść zawartość skompresowanego pliku (tablicę częstości i zakodowany tekst).

Przykład
In0305.txt
ALA MA KOTA
Out0305.txt
A 4, L 1, _ 2, M 1, K 1 O 1 T 1
K 0.0909090909, L 0.0909090909, M 0.0909090909, O 0.0909090909, T 0.0909090909, _ 0.1818181818, A 0.3636363636
K 0.0909090909, L 0.0909090909, M 0.0909090909, O 0.0909090909, T 0.0909090909, _ 0.1818181818, A 0.3636363636
K=000, L=001, M=010, O=011, T=100, _=101, A=11,
//
A-4 L-1 _-2 M-1 K-1 O-1 T-1
≟WCÿ

Zestawy

Zestaw	01	02	03	04	05	06
Zad1	1	1	1	2	2	2
Zad2	3	4	5	3	4	5
