ASD - ćwiczenia XI

Podejście zachłanne

Zadania

1. W kolejce na poczcie, gdzie znajduje się tylko jedno okienko, stoi n klientów poindeksowanych liczbami od 1 do n. Klient o numerze i przyszedł „ze sprawą", której załatwienie wymaga $X[i]$ sekund, gdzie X jest n-elementową tablicą liczb naturalnych. Jeżeli klient i-ty zostanie poproszony do okienka w k-tej minucie to czas jego wyjścia z poczty wynosi dokładnie $k + X[i]$.

   (a) Zaproponuj funkcję opartą na podejściu zachłannym

   `real QUEUE(int X[], int n)`,

   która wyznaczy możliwie minimalny średni czas wyjścia dla wszystkich n klientów poczty.

   (b) Udowodnij, że zaproponowane przez Ciebie rozwiązanie zawsze generuje wyniki optymalne.

   (c) Zastąp tablicę X macierzą dodatnich liczb rzeczywistych Y rozmiaru $n \times 2$ i zdefiniuj czas wyjścia z poczty klienta i jako $\lceil Y[i][1] \cdot k + Y[i][2] \rceil$, gdzie k jest minutą, w której klient został poproszony do okienka. Zaproponuj nową wersję funkcji `QUEUE()`

   `real QUEUE(real Y[][], int n)`.

2. Dla podanego poniżej problemu podaj kolejno:

   - słowny opis strategii (metody) zachłannej rozwiązującej dane zadanie,
   - definicję operacji dominującej kształtującej złożoność rozwiązania,
   - możliwie dokładne oszacowanie kosztu pamięciowego i czasowego rozważanego podejścia,
   - przykład danych wejściowych, dla których wynik zastosowania zaproponowanej strategii zachłannej nie jest optymalny (oczywiście jeżeli takie dane istnieją, w przeciwnym przypadku uzasadnij fakt, że strategia jest optymalna dla omawianego problemu).

   Niech A i B będą zbiorami punktów na płaszczyźnie euklidesowej, spełniającymi warunek $|A| = |B| = n$. Naszym zadaniem jest połączenie wszystkich punktów ze zbioru A z punktami ze zbioru B w pary, tak aby łączna odległość między wszystkimi punktami w parach była jak najmniejsza.

3. Niech M będzie macierzą sąsiedztwa dla pewnego grafu skierowanego $G = (V, E)$. Zakładamy dalej, że wagii krawędzi grafu G są zgodne z poniższą tabelą Q:

waga krawędzi	liczba wystąpień wagii w macierzy M
0	8
1	3
2	6
3	7
4	1

- na podstawie tabeli Q zbuduj drzewo T dla prefiksowego kodu Huffmana (kodujemy wagii krawędzi na podstawie częstości ich występowania),
- korzystając z drzewa T odkoduj macierz sąsiedztwa M z wektora pionowego V (element wektora V odpowiada wierszowi macierzy M):

$$V = \begin{bmatrix} 1110011011 \\ 00110100101 \\ 11101000111 \\ 01000111001 \\ 11011100111 \end{bmatrix},$$

- narysuj graf reprezentowany przez macierz sąsiedztwa M (waga krawędzi równa 0 oznacz jej brak).

4. Dane są następujące kody dla alfabetu A, B, C, D, E:

KOD I	KOD II	KOD III
A - 0	A - 0	A - 000
B - 01	B - 100	B - 01
C - 1	C - 101	C - 10
D - 10	D - 110	D - 110
E - 11	E - 111	E - 111

(a) Narysuj drzewa kodowe dla każdego z trzech podanych kodów.

(b) Które z nich są kodami prefiksowymi?

(c) Które z powyższych kodów są kodami Huffmana dla pliku zawierającego:

- 15 symboli A,
- po 5 symboli B, C, D, E?

Zakładamy, że litery albo słowa o takiej samej częstości są porządkowane leksykograficznie.

(d) Czy KOD III może być kodem Huffmana dla jakiegoś pliku? Jeżeli tak, podaj częstości występowania symboli A, B, ..., E, jeżeli nie, podaj uzasadnienie.

Zadanie o najdłuższym łańcuchu słów (do domu)

Niech $W = \{W[1], W[2], \ldots, W[n]\}$ będzie zbiorem n niepustych i parami różnych słów, co najwyżej 20 literowych, nad alfabetem $\Sigma = \{$a, b, c, d,..., x, y, z$\}$. Dalej zakładamy, że mamy daną funkcję

$$TRANSFORM : W \times W \rightarrow \{TRUE, FALSE\},$$

która dla dwóch zadanych słów $W[i], W[j] \in W$ jest równa:

- $TRUE$, jeżeli słowo $W[i]$ można przekształcić do słowa $W[j]$, przez jedną z operacji:
  - dodanie pojedynczej litery, w dowolnym miejscu słowa $W[i]$,
  - usunięcie pojedynczej litery, z dowolnego miejsca słowa $W[i]$,
  - zamianę pojedynczej litery, w dowolnym miejscu słowa $W[i]$,
- $FALSE$, w przeciwnym przypadku.

Na przykład:

$$\begin{aligned} TRANSFORM\,(\text{las}, \text{pas}) &= TRUE, \\ TRANSFORM\,(\text{las}, \text{as}) &= TRUE, \\ TRANSFORM\,(\text{las}, \text{sas}) &= FALSE, \\ TRANSFORM\,(\text{las}, \text{plas}) &= TRUE. \end{aligned}$$

Następnie przyjmujemy, że zdefiniowano funkcję

$$COMPARE \; : \; W \times W \to \{0, 1, 2\},$$

która porównuje w porządku leksykograficznym dwa wejściowe słowa $W[i], W[j] \in W$ i jest równa:

- 0, jeżeli $W[i] = W[j]$,
- 1, jeżeli $W[i] < W[j]$,
- 2, jeżeli $W[i] > W[j]$.

Na przykład:

$$\begin{aligned} COMPARE\,(\text{las}, \text{pas}) &= 1, \\ COMPARE\,(\text{las}, \text{as}) &= 2, \\ COMPARE\,(\text{las}, \text{sas}) &= 1, \\ COMPARE\,(\text{as}, \text{as}) &= 0. \end{aligned}$$

Łańcuchem przekształceń słów $W[i] \to W[i+1] \to W[i+2] \to \ldots \to W[i+k]$ nazywamy dowolny ciąg słów należących do zbioru W, taki, że oba poniższe warunki są spełnione:

- $\forall j = 1, 2, \ldots, k \; : \; TRANSFORM\,(W[i+j-1], W[i+j]) = TRUE$,
- $\forall j = 1, 2, \ldots, k \; : \; COMPARE\,(W[i+j-1], W[i+j]) = 1$.

Zaprojektuj możliwie efektywną funkcję

```
int CHAIN(Word W[], int n),
```

która wyznaczy długość najdłuższego łańcucha przekształceń słów, jaki można zbudować przy pomocy słow ze zbioru W. Zakładamy, że W jest n elementową tablicą zmiennych typu `Word` o następującej specyfikacji:

```
typedef str_Word Word;

struct str_Word {
  char Letters[20];
};
```

a operacją dominującą jest ewaluacja predefiniowanych funkcji `TRANSFORM()` lub `COMPARE()`, których składnia wywołania jest następująca:

```
bool TRANSFORM(Word W1, Word W2),

int COMPARE(Word W1, Word W2).
```