Temă ~~Seminar~~ Seminar 3

Ex 6

$G = (V, E)$ neorientat conex

$V = \{1, 2, \dots, n\}$

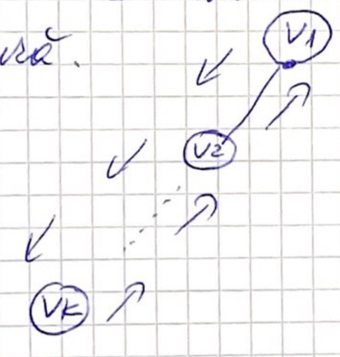
„<" relație de ordine pe V

DFS și BFS pornesc din nodul 1

Ce proprietăți trebuie să respecte G ca cele 2 parcurgeri să obțină același arbore parțial $T_{DFS} = T_{BFS}$?

Dacă G este graf neorientat conex ⇒ trebuie să conțină un ciclu de forma $C(v_1, v_2, \dots, v_k)$ cu $v_1 \to v_2 \to v_3 \to \dots \to v_k \to v_1$.

În parcurgerea DFS nodurile $v_1, \dots, v_k$ vor fi pe aceeași ramură. (1)



În schimb în parcurgerea BFS nodurile $v_1, \dots, v_k$ vor forma cel puțin 2 ramuri. (2)

Din 1 și 2 ⇒ $T_{BFS} \neq T_{DFS}$

Înseamnă că graful nu trebuie să conțină niciun ciclu. (3)

Dar (din cerință) G este neorientat conex. (4)

Din 3 și 4 ⇒ G este arbore.

Concluzie: Proprietatea pe care trb. să o respecte G este să fie arbore (să aibă $n-1$ muchii)

Ex 7

n intervale $[a_i, b_i]$ $1 \le i \le n$

Trb. să alegem o submulțime S de intervale a.î. $(\forall)$ 2 $\cap$ max 1 pct. iar suma lungimilor intervalelor să fie max.

1. Care sunt mulțimile de noduri și de muchii (V, E)? Este graf neorientat sau orientat?

2. Descrieți un algoritm eficient care găsește o rezolvare a problemei.

Rezolvare:

1. Construim graful G (V, E)

$V = \{[a_1, b_1], \ldots, [a_n, b_n]\} \mid b_1 \le b_2 \le \ldots \le b_n$

E este mulțimea muchiilor care ~~reprezintă~~ leagă intervalele care se ating la un capăt sau care sunt disjuncte.

Intervalele sunt ordonate după b_i; $i = \overline{1,n}$, ceea ce înseamnă că graful este orientat deoarece dacă avem de ex. nodurile [7, 13] și [14, 20] putem să mergem doar din $[7, 13] \longrightarrow [14, 20]$ nu și din $[14, 20] \longrightarrow [7, 13]$.

2. ~~Am~~ Un algoritm eficient este parcurgerea DFS. Putem parcurge nodurile în adâncime pentru a afla submulțimea de intervale cu maxim 1 pct de intersecție și sumă maximă. (respectiv, lungimea maximă a unui drum)

8. $G=(V,E)$ graf neorientat
$V=\{1,2,\ldots,n\}$
E este necunoscut
interogări $Q(i,j)$ către un oracol $= \begin{cases} 1, (\exists)\,[i,j] \\ 0, \text{altfel} \end{cases}$

Demonstrați că pentru orice algoritm A de determinare a comp. conexe $(\exists)$ G_A pentru care A trb să facă cel puțin $n(n-1)/2$ interogări.

Rezolvare:

Numărul maxim de muchii într-un graf neorientat cu n noduri este de $n(n-1)/2$ muchii.
Trb să verificăm dacă există muchie între toate nodurile deoarece se poate schimba numărul de componente de conexe.
Putem avea cel mai rău caz:

$\Rightarrow$ n componente conexe

Dar putem avea

$\Rightarrow$ $n-1$ componente conexe.

$\Rightarrow$ Trebuie să se facă cel puțin $n(n-1)/2$ interogări.

9. n puncte de interes (x_i, y_i) $1 \le i \le n$

Start $= (X_1, Y_1)$ vizitează fiecare alt punct de interes ca mai apoi să se întoarcă în locul de unde a plecat, parcurgând o distanță cât mai mică.

Distanța parcursă să fie de cel mult 2 ori mai mare decât distanța minimă. Descrieți un algoritm eficient care găsește un itinerariu corесp.

Rezolvare: Metoda Greedy.

Cât timp încă există puncte de interes nevizitate, turistul ~~le vizit~~ alege să le viziteze pe cele mai apropiate în ordinea distanței și distanța $<$ (distanța minimă) $* 2$.

Vom crea un vector bool viz[2, ..., n]

$$= \begin{cases} 1, & \text{vizitat} \\ 0, & \text{altfel} \end{cases}$$

La început toate punctele sunt 0 (nevizitate).

Vom crea un ~~vector de tip double~~ map <int, double> D care reține distanțele dintre $P(X_1, Y_1)$ și $P(X_i, Y_i)$ $i = \overline{2,n}$ și numărul de ordine al locației.

D[2, ..., n].   D[i] -> first = i; $i = \overline{2,n}$

D[i] -> second $= \sqrt{(x_i - x_1)^2 + (y_i - y_1)^2}$

Vom afla minimul dintre valorile map-ului.

Și apoi vom sorta map-ul după valoare.

Cât timp D -> second < 2 * min și viz[D-first] == 0 vom parcurge D și vom seta viz[D-first] = 1