Redactare şi comunicare ştiinţifică şi profesională
Laborator 7

Lect. dr. Adela Sasu

November 25, 2020

Exerciţiul 1: Redactaţi următorul text, respectând următoarele cerinţe:

- textul are o culoare definită folosind unul din modelele rgb, cmyk
- realizaţi figura într-un program la alegerea dumneavoastră sau puteţi folosi pachetul `TikZ`

3.1 Breviar teoretic

Fie G un graf orientat. G este un *arbore cu rădăcina* r, dacă există G un vârf r din care oricare alt vârf poate fi ajuns printr-un drum unic.

Definiția este valabilă şi pentru cazul unui graf neorientat, alegerea unei rădăcini fiind însă în acest caz arbitrară: orice arbore este un arbore cu rădăcină, iar rădăcina poate fi fixată în oricare vârf al său. Aceasta, deoarece dintr-un vârf oarecare se poate ajunge în oricare alt vârf printr-un drum unic.

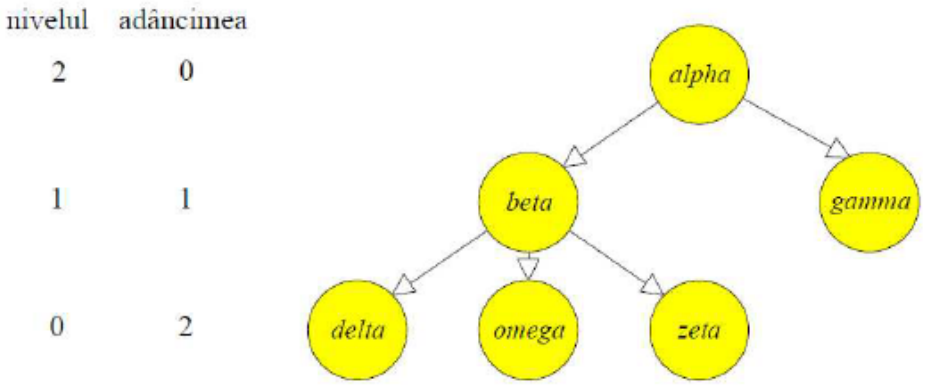

Figura 3.1: Un arbore cu rădăcină.

Când nu va fi pericol de confuzie, vom folosi termenul "arbore", în loc de termenul corect "arbore cu rădăcină". Cel mai intuitiv este să reprezentăm un arbore cu rădăcină, ca pe un arbore propriu-zis. În figura 3.1, vom spun că *beta* este *tată* lui *delta* şi fiul lui *alpha*, că *beta* şi *gamma* sunt *fraţi*, că *delta* este un *descendent* al lui *alpha*, iar *alpha* este un *ascendent* al lui *delta*.

Exerciţiul 2: Redactaţi următorul text, respectând următoarele cerinţe:

- algoritmul se redactează folosind pachetele `algorithm, algorithmicx, algpseudocode`
- realizaţi figura într-un program la alegerea dumneavoastră sau puteţi folosi pachetul `TikZ`

Presupunem, pentru simplificare, că vârfurile sunt numerotate, $V = \{1, 2, \ldots, n\}$, vârful 1 fiind sursa, şi că matricea L dă lungimea fiecărei muchii, cu $L[i, j] = +\infty$, dacă muchia (i, j) nu există. Soluția se va construi în tabloul $D[2 .. n]$. Algoritmul este:

function *Dijkstra*($L[1 .. n, 1 .. n]$)
  {iniţializare}
  $C \leftarrow \{2, 3, \ldots, n\}$ {$S = V \setminus C$ există doar implicit}
  for $i \leftarrow 2$ **to** n **do** $D[i] \leftarrow L[1, i]$
  {bucla greedy}
  repeat n–2 **times**
    $v \leftarrow$ vârful din C care minimizează $D[v]$
    $C \leftarrow C \setminus \{v\}$ {şi, implicit, $S \leftarrow S \cup \{v\}$}
    for fiecare $w \in C$ **do**
      $D[w] \leftarrow \min(D[w], D[v]+L[v, w])$
  return D

Pentru graful din Figura 6.5, paşii algoritmului sunt prezentaţi în Tabelul 6.3.

Observăm că D nu se schimbă dacă mai efectuăm o iteraţie pentru a-l scoate şi pe $\{2\}$ din C. De aceea, bucla greedy se repetă de doar n–2 ori.

Se poate demonstra următoarea proprietate:

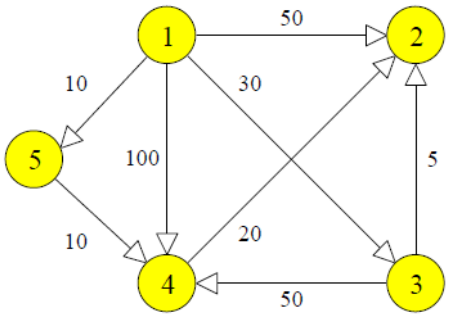

Figura 6.5 Un graf orientat.

Pasul	v	C	D
iniţializare	—	{2, 3, 4, 5}	[50, 30, 100, 10]
1	5	{2, 3, 4}	[50, 30, 20, 10]
2	4	{2, 3}	[40, 30, 20, 10]
3	3	{2}	[35, 30, 20, 10]

Tabelul 6.3 Algoritmul lui Dijkstra aplicat grafului din Figura 6.5.