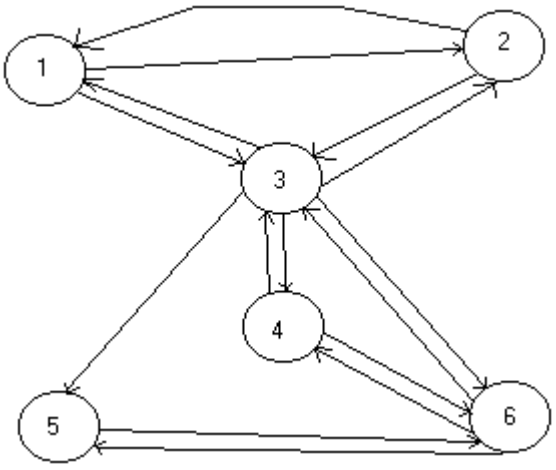
BLIA-MAS Laborator 03

Cuprins:
a. Structuri in LISP (grafuri)
b. DFS in LISP
c. Exemple

Structuri in LISP (grafuri) – DFS, BFS in LISP – Exemple

In LISP grafurile se reprezinta sub forma unor liste. Avem mai multe modalitati de a reprezenta un graf, toate folosind structura de lista.

Fie graful de mai jos:



Reprezentarea acestuia in LISP folosind liste ar fi urmatoarea:

((nod (lista noduri_succesoare))*)

Astfel:

```
(   (1 (2 3))
    (2 (1 3))
    (3 (1 2 4 5 6))
    (4 (3 6))
    (5 (3 6))
    (6 (3 4 5))
)
```

Pentru a memora informatii referitoare la fiecare nod putem ori sa folosim o lista ajutatoare:

```
(       (nod1 (lista succesori nod1))
        (nod2 (lista succesori nod2))
        .....
        (nodn (lista succesori nodn))
)

(       (nod1 (lista atribute nod1))
        (nod2 (lista atribute nod2))
        .....
        (nodn (lista atribute nodn))
)
```

sau putem folosi o singura lista pentru reprezentarea grafului respectiv:

```
(       (nod1 ((lista atribute nod1) (lista succesori nod1)))
        (nod2 ((lista atribute nod2) (lista succesori nod2)))
        .....
        (nodn ((lista atribute nodn) (lista succesori nodn)))
)
```

sau perechi de cate trei (cheie nod, lista atribute nod, lista succesori nod:

```
(       (nod1 (lista atribute nod1) (lista succesori nod1))
        (nod2 (lista atribute nod2) (lista succesori nod2))
        .....
        (nodn (lista atribute nodn) (lista succesori nodn))
)
```

Algoritmii DFS si BFS au fost studiati in anul 3, astfel incat nu vom mai insista asupra lor, deoarece se considera cunoscuti.

Vom reprezenta in LISP graful din imaginea de mai sus, folosind a doua modalitate prezentata:

```
(
        (1 ((atribute_nod_1)   (2 3)))
        (2 ((atribute_nod_2)   (1 3)))
        (3 ((atribute_nod_3)   (1 2 4 5 6)))
        (4 ((atribute_nod_4)   (3 6)))
        (5 ((atribute_nod_5)   (3 6)))
        (6 ((atribute_nod_6)   (3 4 5)))
)
```

In lista de atribute asociata fiecarui nod am considerat urmatoarele atribute (in ordine): (timp_de_start, timp_de_terminare, culoare, parinte)

Astfel initial vom avea:

```
(
        (1 ((0 0 0 0) (2 3)))
        (2 ((0 0 0 0) (1 3)))
        (3 ((0 0 0 0) (1 2 4 5 6)))
        (4 ((0 0 0 0) (3 6)))
        (5 ((0 0 0 0) (3 6)))
        (6 ((0 0 0 0) (3 4 5)))
       )
    )
```

Vom considera o variabila ajutatoare, numarul de noduri pe care il are graful, care va fi retinut in prima locatie a grafului. Astfel graful nostru va fi reprezentat in LISP astfel:

```
( numar_de_noduri_din_graf
      (     (nod1 ((lista atribute nod1) (lista succesori nod1)))
            (nod2 ((lista atribute nod2) (lista succesori nod2)))
            .....................................................................
            (nodn ((lista atribute nodn) (lista succesori nodn)))
      )
)
```

Varianta in LISP pentru graful nostru:

```
(6
    (
     (1 ((0 0 0 0) (2 3)))
        (2 ((0 0 0 0) (1 3)))
        (3 ((0 0 0 0) (1 2 4 5 6)))
        (4 ((0 0 0 0) (3 6)))
        (5 ((0 0 0 0) (3 6)))
        (6 ((0 0 0 0) (3 4 5)))
       )
)
```

In cele ce urmeaza se vor prezentaimplementarea in LISP a grafului, a unor functii ajutatoare pentru prelucrare graf, precum si a metodei de parcurgere graf DFS.

```
(setq timp 0)

(defun get_NrNod(nod)
 (car nod)
)
```

```
(defun set_NrNod(nod x)
 (setf (car nod) x)
)

(defun get_Informatie(nod)
 (caadr nod)
)

(defun set_Informatie(nod x)
 (setf (caadr nod) x)
)

(defun get_debut(nod)
 (car (get_Informatie nod))
)

(defun set_debut(nod x)
 (setf (car (get_Informatie nod)) x)
)

(defun get_iesire(nod)
 (cadr (get_Informatie nod))
)

(defun set_iesire(nod x)
 (setf (cadr (get_Informatie nod)) x)
)

(defun get_culoare(nod)
 (caddr (get_informatie nod))
)

(defun set_culoare(nod x)
 (setf (caddr (get_Informatie nod)) x)
)

(defun get_parinte(nod)
 (cadddr (get_Informatie nod))
)

(defun set_parinte(nod x)
 (setf (cadddr (get_Informatie nod)) x)
)

(defun get_succesori(nod)
```

```
 (cadadr nod)
)

(defun get_noduri(G)
 (cadr G)
)

(defun Coloreaza_Alb(Lista_Noduri)
 (cond
  ((null Lista_Noduri) nil)
  (t
     (setq nod (car Lista_Noduri) )
     (set_culoare nod 'alb)
     (Coloreaza_Alb (cdr Lista_Noduri))

  )
 )
)

(defun Apeluri_DfsNod(Lista_Noduri G)
 (cond
  ((null Lista_Noduri) nil)
  (t
    (setq urm (car Lista_Noduri))
    (cond
     ((eq (get_culoare urm) 'alb)
       (set_parinte urm nil)
       (Dfs_Nod urm G)
     )
    )
    (Apeluri_DfsNod (cdr Lista_Noduri) G)
  )
 )
)


(defun Dfs_Alg(G)
  (setq timp timp)
  (setq Temp (cadr G)) ;setam Temp = lista de noduri a grafului
  (Coloreaza_Alb Temp);coloram in alb toate nodurile

  (Apeluri_DfsNod Temp G)

)

(defun noduri_adiacente(nod G)
```

```
 (setq succesori (get_succesori nod))
 (dolist (xnr succesori)
   (dolist (xnod (get_noduri G))
    (cond
     ((= (get_NrNod xnod) xnr)
       (cond
        ( (eq (get_culoare xnod) 'alb)
              (set_parinte xnod (get_NrNod nod))
              (Dfs_Nod xnod G)
           )
       )
     )
    )
   )
 )
)

(defun Dfs_Nod(Nod G)
 (set_culoare nod 'gri)
 (set_debut nod timp)
 (setq timp (+ timp 1))

 (noduri_adiacente Nod G)
 (set_culoare Nod 'negru)
 (set_iesire Nod timp)
 (setq timp (+ timp 1))
)
```

Pentru a testa functionarea acestui program:

```
(setq G '(6
          (
              (1 ((0 0 0 0) (2 3)))
              (2 ((0 0 0 0) ()))
              (3 ((0 0 0 0) (4 5 6)))
              (4 ((0 0 0 0) (5)))
              (5 ((0 0 0 0) ()))
              (6 ((0 0 0 0) ()))
            )
           )
)

(setq Temp (cadr G))
(print Temp)
```

```
(setq radacina (car Temp))
(print radacina)

(print (get_Informatie radacina))


(print (get_debut radacina))

(print (get_iesire radacina))

(print (get_culoare radacina))

(print (get_parinte radacina))

(set_debut radacina 8)

(print radacina)

(print (get_succesori radacina))



(Dfs_Alg G)
(print G)
```

Tema de laborator:

Implementarea in LISP a BSF-ului folosind metodelel rpezentate mai sus.