Sortare topologică

Enunţul problemei de sortare topologică este următorul: fie o listă de n obiecte, şi o listă de preferinţe. O preferinţă este exprimată în felul următor: obiectul a este preferat obiectului b. Se cere să se ordoneze cele n obiecte, în ordinea descrescătoare a preferinţelor, conform preferinţelor exprimate în lista de preferinţe.

O reprezentare convenabilă a spaţiului acestei probleme presupune construirea unui graf care să aibă în vârfurile sale cele n obiecte. Vom trasa un arc în graf, de la varful v la varful w dacă obiectul asociat vârfului w este preferat obiectului asociat vârfului v Astfel, un obiect care este cel mai preferat în lista de obiecte va avea asociat un nod în graf care nu are succesori, adică în care doar intră arcuri. Pornind de la această observaţie, putem construi algoritmul pentru sortare topologică

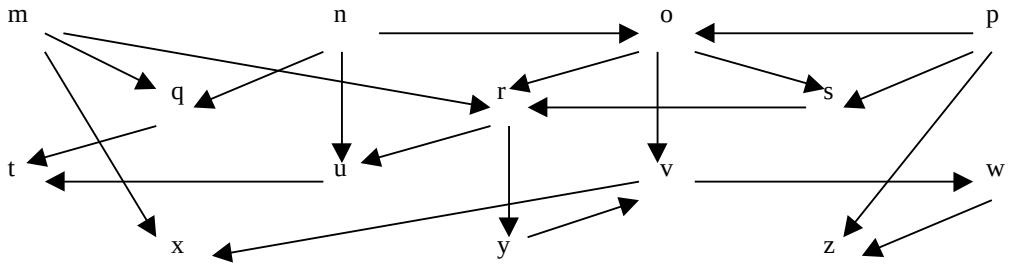
```
SortareTopologică(ListaObiecte o, ListăPreferinţe l)
       Construieste graful asociat listei obiectelor şi listei preferintelor
       While (graful nu este vid)
              Determină un nod care nu are succesori în graf
              Afisează (prelucrează nodul)
              Elimină nodul din graf, impreună cu arcele asociate acestui nod
       Endwhile
End
```

În cazul în care la o iteratie, în ciclul while, nu putem determina un nod fără succesori, problema dată nu are soluţie. Un exemplu de problemă fără soluţie este: sunt date 3 obiecte, a, b, c şi lista de preferinţe: a este preferat lui b, b este preferat lui c şi c este preferat lui a.

Exemplu: se dau obiectele: m, n, o, p, q, r, s, t, u, v, w, x, y, z şi lista de preferinte:

q este preferat lui m,	q este preferat lui n,	o este preferat lui n,	o este preferat lui p
r este preferat lui m,	r este preferat lui o,	r este preferat lui s,	s este preferat lui o
s este preferat lui p,	t este preferat lui q,	t este preferat lui u,	u este preferat lui n
u este preferat lui r,	v este preferat lui o,	w este preferat lui v,	x este preferat lui m
x este preferat lui v,	y este preferat lui r,	v este preferat lui y,	z este preferat lui p
z este preferat lui w			

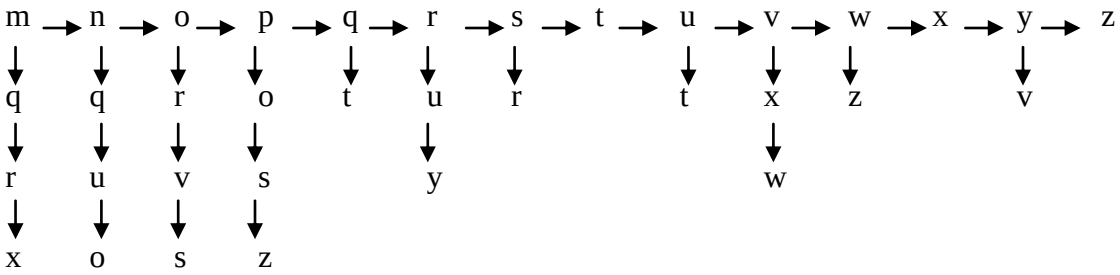
Figura de mai jos reprezintă graful asociat listei de preferinţe de mai sus.



Pornind de la acest graf, putem obţine următoarea sortare topologică a obiectelor:
z x w v y t u r s o p q n m

Pentru rezolvarea acestei probleme, este necesară o reprezentare convenabilă a grafului asociat problemei. De interes sunt nodurile fără succesori, deci cel mai convenabil putem reprezenta acest graf ca o listă de noduri, fiecare nod memorând o listă a succesorilor săi.

Astfel, graful de mai sus se reprezintă:



Pe prima linie avem o listă cu toate nodurile grafului. Pe coloane avem reprezentate liste cu succesorii unui nod. Deci, pentru reprezentarea grafului sunt necesare 2 tipuri de liste. Graful de mai sus poate fi construit în faza de citire a listei de preferinţe. Astfel, când se citeşte o pereche de obiecte, reprezentând o preferinţă, se verifică dacă obiectele citite sunt în lista de obiecte; în caz negativ, se adaugă obiectele la lista de obiecte; apoi se adaugă nodul mai preferat în vârful listei de succesori ai nodului mai puţin preferat. Implementările cu liste simplu înlănţuite sunt suficiente pentru rezolvarea problemei.

Mai jos, este descris algoritmul de rezolvare a acestei probleme.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct succCell {
	struct nodeCell *node;
	struct succCell *next;
} succList;

typedef struct nodeCell {
	char nodeDesc[30];
	struct succCell *sList;
	struct nodeCell *next;
} nodeList;

/* cauta nodul in lista. daca nu exista, il adauga la lista */
nodeList* searchNode(nodeList *nl, char* node) {
	nodeList *p, *q;

	p = nl->next;
	while(p && strcmp(p->nodeDesc, node)!=0) p = p->next;
	if (p!=0) return p;
```

```
        q = (nodeList*)malloc(sizeof(nodeList));
        q->next = nl->next;
        q->sList = 0;
        strcpy(q->nodeDesc, node);
        nl->next = q;
        return q;
}

/* adauga arcul node1 -> node2 in graf */
void addLink(nodeList *nl, char *node1, char *node2) {
        nodeList *p, *q;
        succList *r;

        /* identifica locatia primului nod */
        p = searchNode(nl, node1);
        /* identifica locatia nodului 2 */
        q = searchNode(nl, node2);

        /* adauga arc in lista de succesori a nodului p */
        r = (succList*)malloc(sizeof(succList));
        r->node = q;
        r->next = p->sList;
        p->sList = r;
}

/* sterge un nod fara succesori din graf, returneaza valoarea stearsa */
int delete(nodeList *nl, char* value) {
        nodeList **p, *p1, *p2;
        succList **r, *q;

        p = &(nl->next);
        while(*p && ((*p)->sList!=0)) p = &((*p)->next);
        if (*p==0) return 0; /* nu s-a gasit element de sters */

        p1 = *p;
        *p = p1->next;
        strcpy(value, p1->nodeDesc);

        /* itereaza pe lista, si sterge din fiecare sublista elementul p1, daca e
gasit */
        p2 = nl->next;
        while (p2) {
                r = &(p2->sList);
                while (*r && ((*r)->node!=p1)) r = &(*r)->next;
                if (*r != 0) {
                        q = *r;
                        *r = q->next;
                        free(q);
                }
                p2 = p2->next;
        }
        return 1;
}

void main() {
        char sir[2][30], s[30];
        int i = 0;
        nodeList nl;
        nl.next = 0;

        strcpy(sir[0], ""); strcpy(sir[1], ""); strcpy(s, "");
        scanf("%s", sir[i]); i = !i;
```

```
	while (strcmp(sir[!i],".")!=0) {
		if (i==0) {
			addLink(&nl, sir[1], sir[0]);
		}
		scanf("%s", sir[i]); i = !i;
	}

	printf("sortare topologica:\n");
	while(delete(&nl, s)!=0)
		printf("%s ", s);
	if (nl.next!=0)
		printf("\nnu s-a putut gasi o sortare topologica\n");
}
```