Traversarea grafurilor. Aplicaţii

Analiza algoritmilor

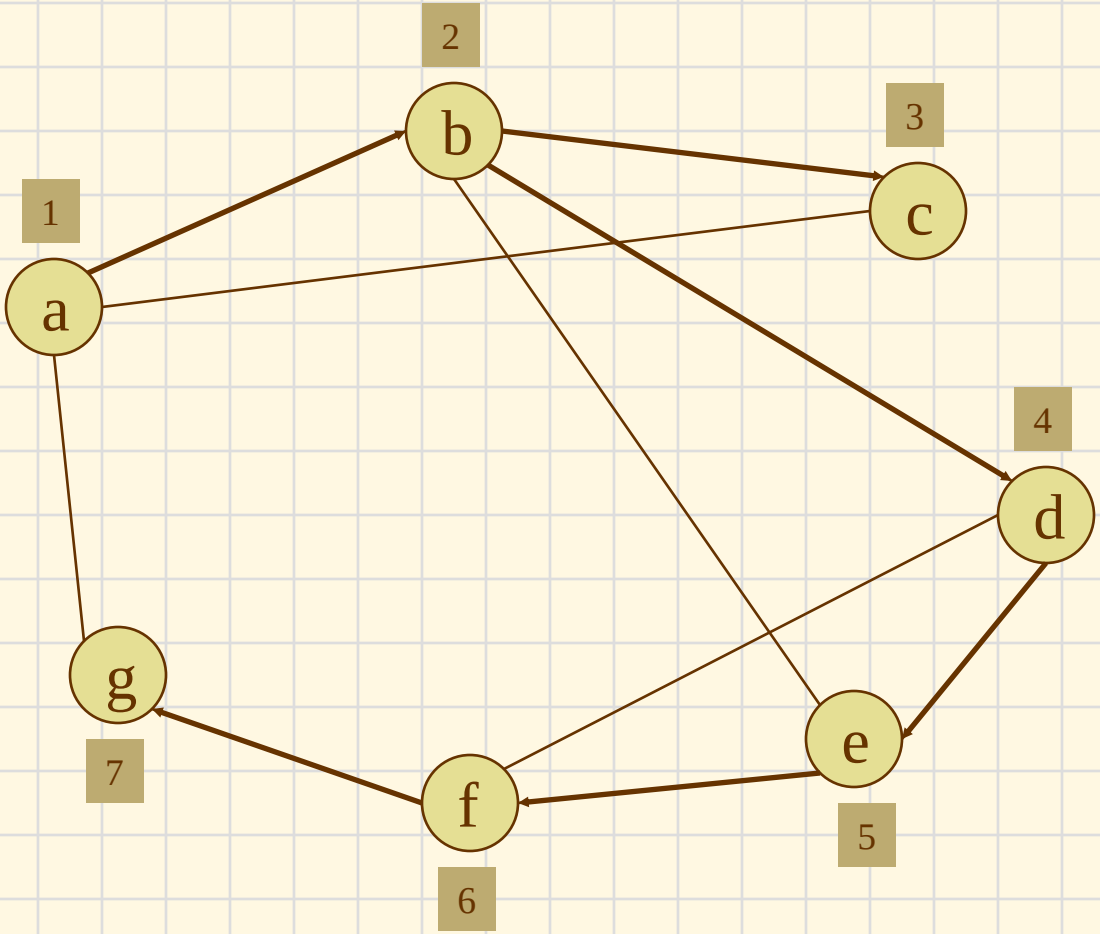
-laborator-

conf. dr. ing. Ciprian-Bogdan Chirila

Cuprins

- Traversarea grafurilor
  - Traversarea grafurilor în adâncime
  - Traversarea grafurilor prin cuprindere
- Aplicaţii ale traversării grafurilor
  - Determinarea arborilor de acoperire
  - Determinarea componentelor conexe
  - Puncte de articulaţie şi componte biconexe
- Aplicaţie
  - L’ami de mon ami c’est mon ami

Traversarea grafurilor în adâncime (depth-first search)



TDA graf

```
typedef struct
{
    int nrNoduri;
    char tabChei[NRMAXNODURI];
    int tabVizitat[NRMAXNODURI];
    char matAdiacenta[NRMAXNODURI][NRMAXNODURI];
} Graf;
```

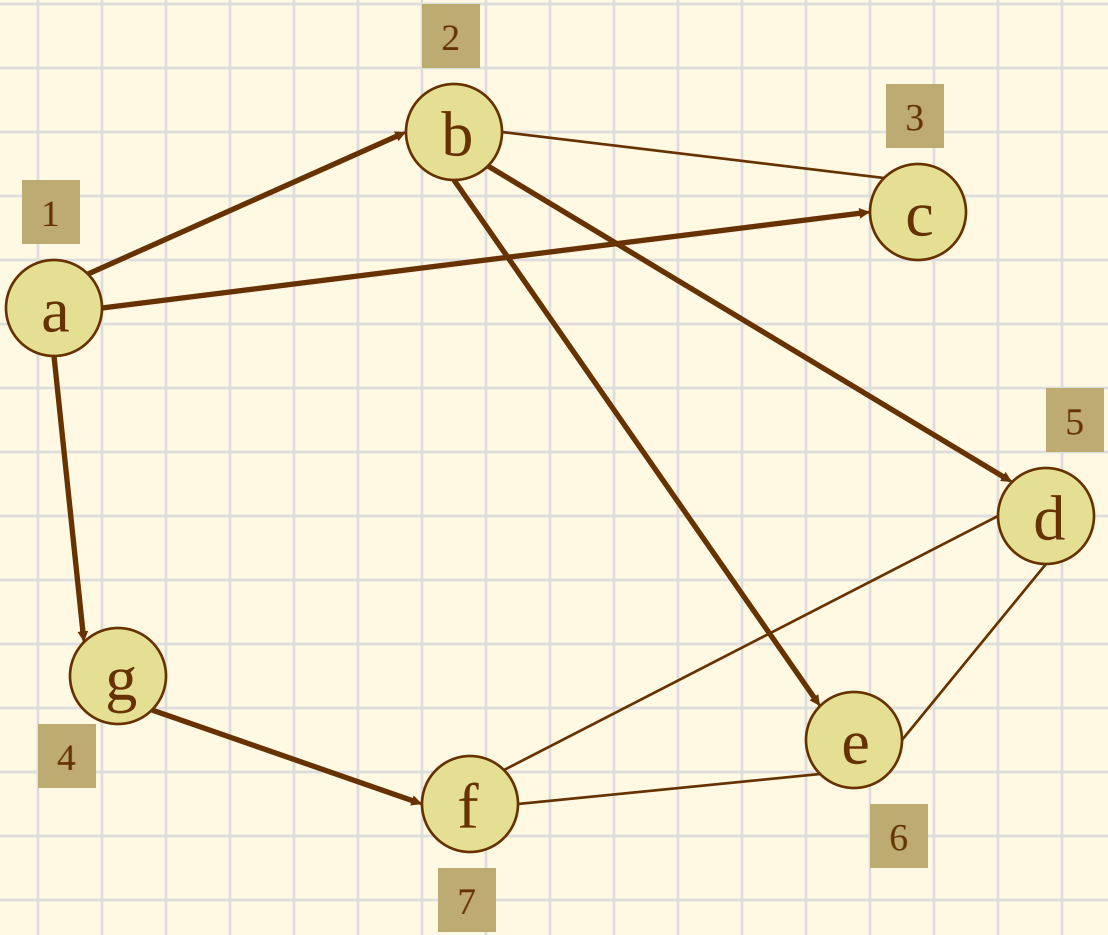
Traversare în adâncime

```
void traverseazaInAdancime(Graf g)
{
    int i;
    printf("Traversare in adancime:\n");
    for (i=0;i<g.nrNoduri;i++)
    {
        if (g.tabVizitat[i]==0)
        {
            traverseazaInAdancime(g,i);
            printf("\n");
        }
    }
    printf("\n");
}
```

Traversare în adâncime

```
void traverseazaInAdancime(Graf &g,int indexNod)
{
    int i;
    if (!g.tabVizitat[indexNod])
    {
        printf("%c ",g.tabChei[indexNod]);
        g.tabVizitat[indexNod]=1;
        for (i=0;i<g.nrNoduri;i++)
        {
            if (g.matAdiacenta[indexNod][i])
            {
                traverseazaInAdancime(g,i);
            }
        }
    }
}
```

Traversarea grafurilor prin cuprindere (breadth-first search)



Traversare prin cuprindere

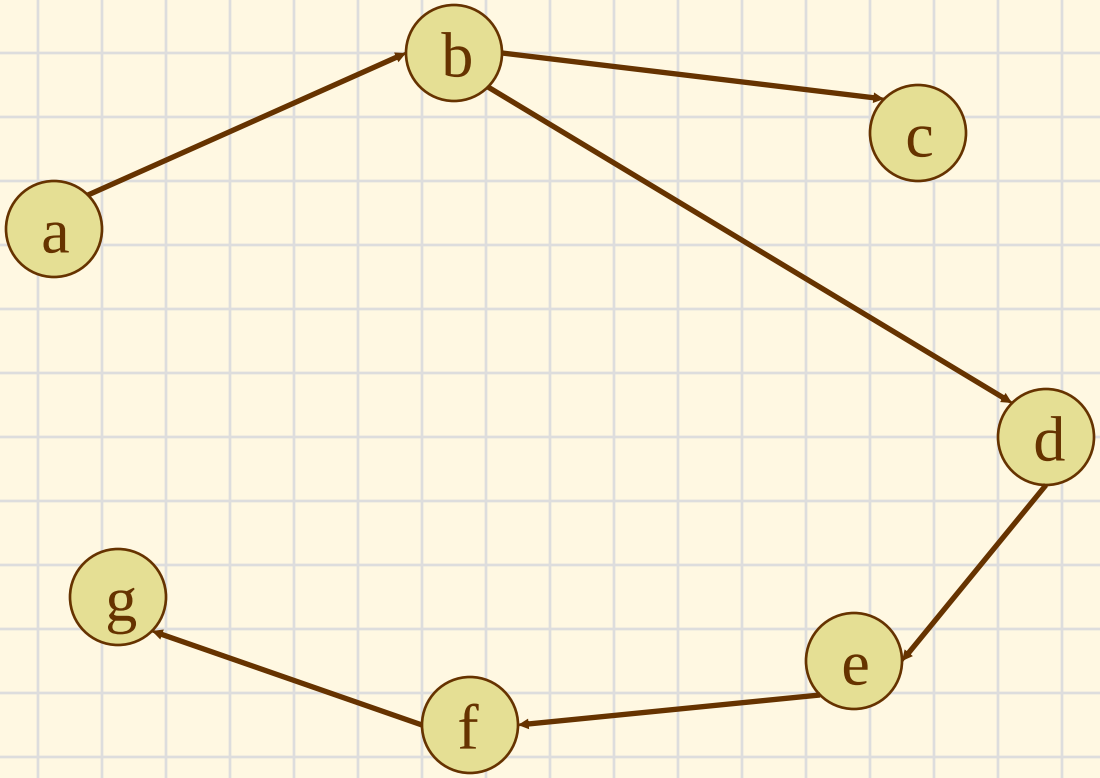
```
void traverseazaPrinCuprindere(Graf g)
{
    int i;
    printf("Traversare prin cuprindere:\n");
    for (i=0;i<g.nrNoduri;i++)
    {
        if (g.tabVizitat[i]==0)
        {
            traverseazaPrinCuprindere(g,i);
            printf("\n");
        }
    }
    printf("\n");
}
```

Traversare prin cuprindere

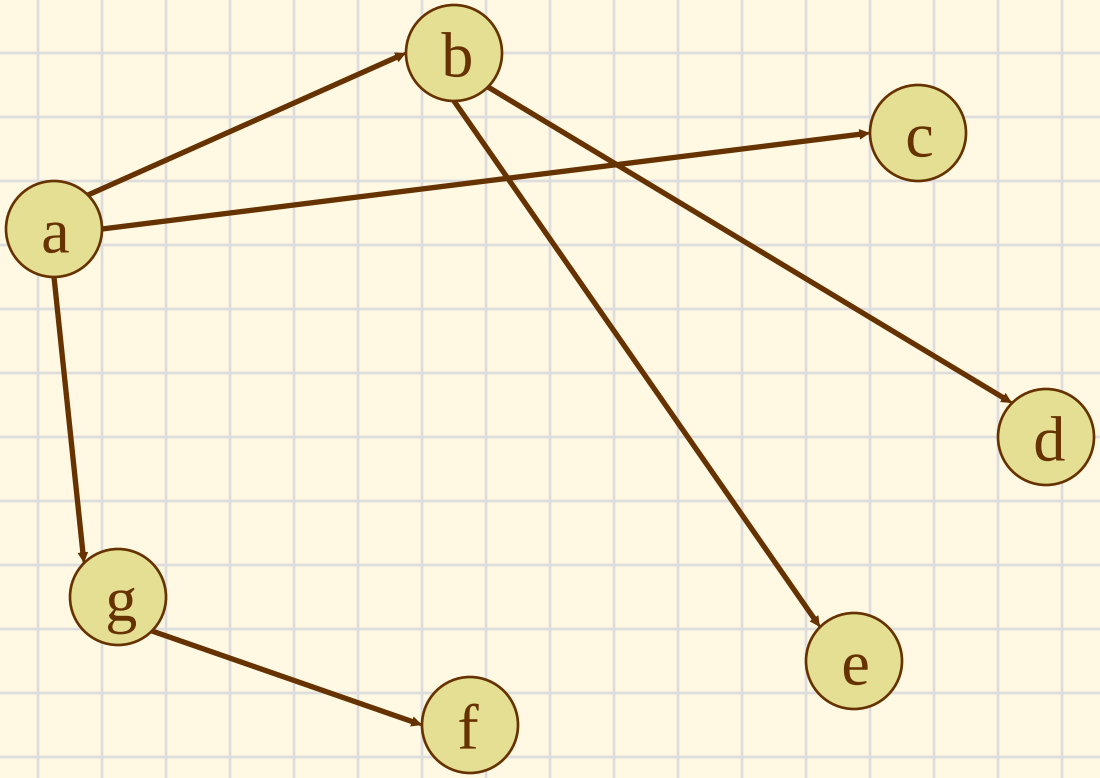
```
void traverseazaPrinCuprindere(Graf &g,int indexNod)
{
    int i;
    Coada coada;
    initializare(coada);

    adauga(coada,indexNod);
    g.tabVizitat[indexNod]=1;
    do
    {
        indexNod=scoate(coada);
        printf("%c ",g.tabChei[indexNod]);
        for (i=0;i<g.nrNoduri;i++)
        {
            if (g.matAdiacenta[indexNod][i] && !(g.tabVizitat[i]))
            {
                g.tabVizitat[i]=1;
                adauga(coada,i);
            }
        }
    }
    while (!vida(coada));
}
```

Determinarea arborilor de acoperire (traversare în adâncime)



Determinarea arborilor de acoperire (traversare prin cuprindere)

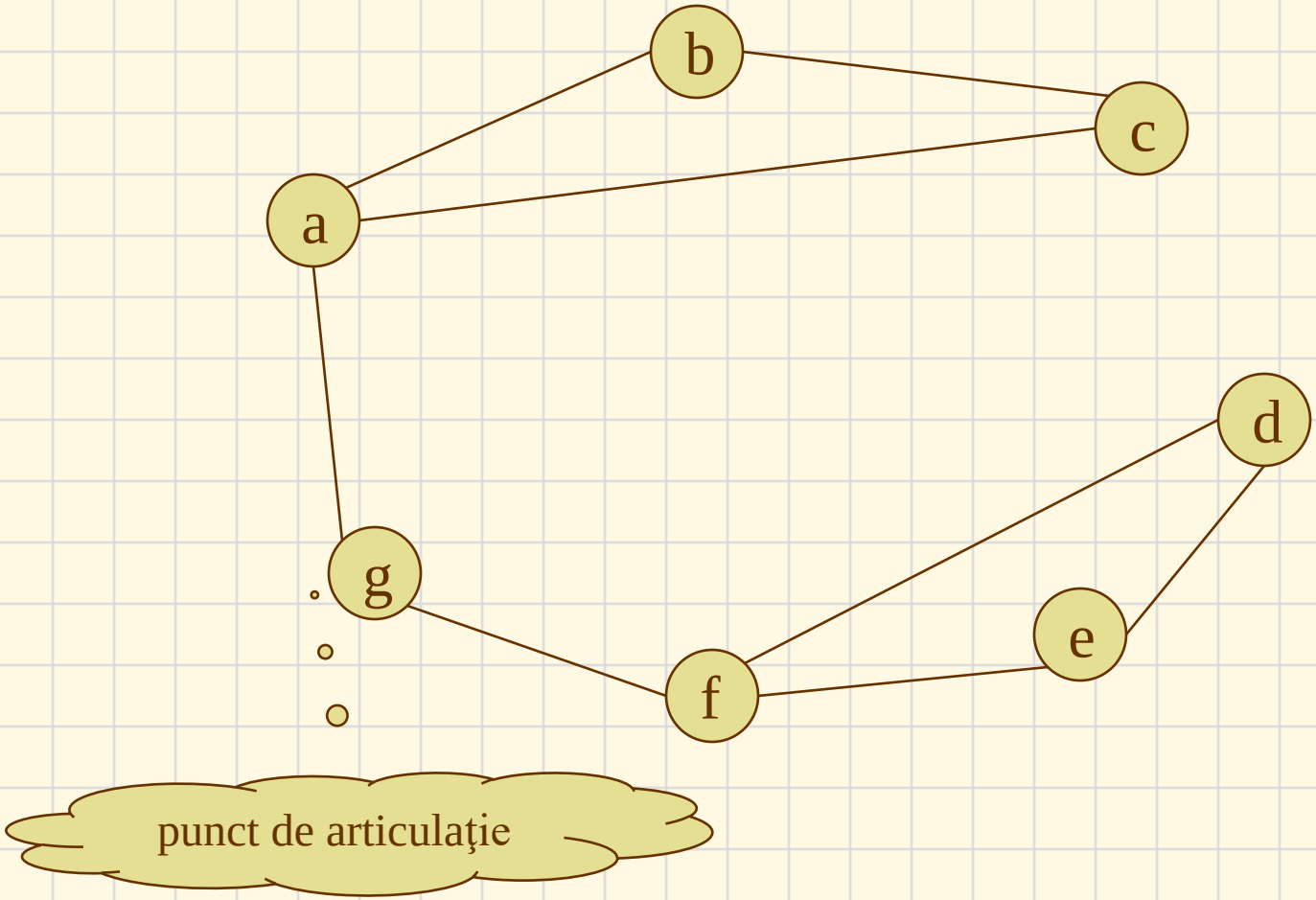


Determinarea componentelor conexe

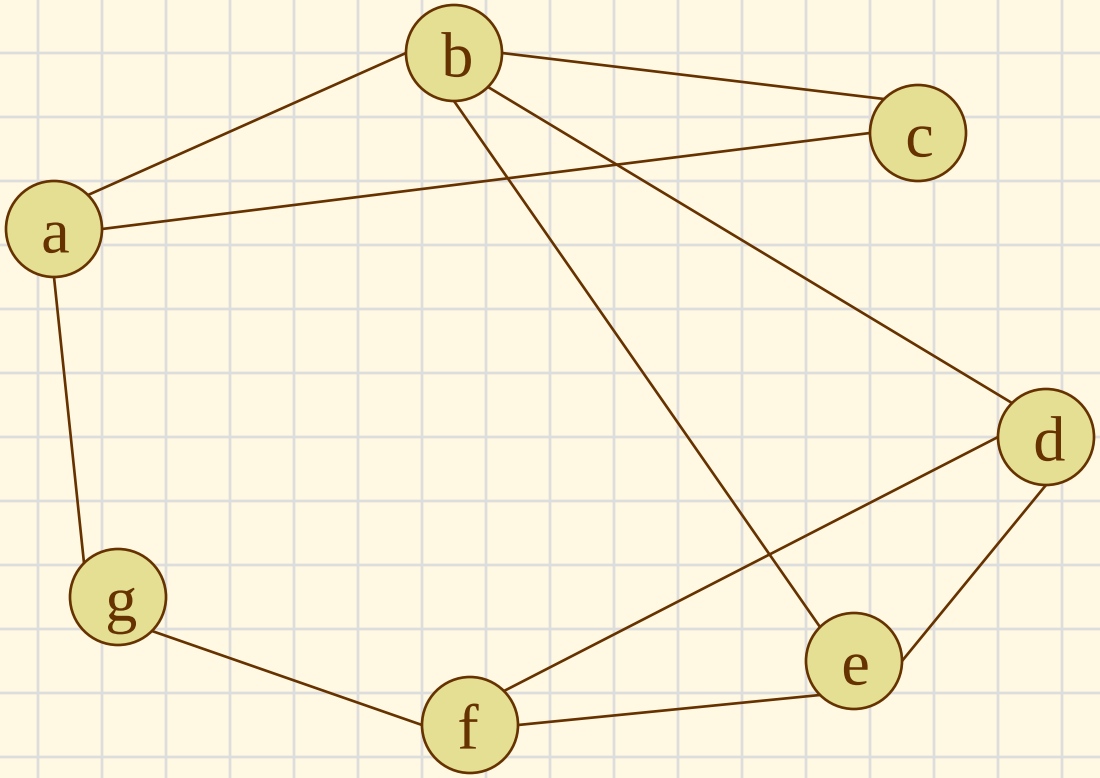
```
void traverseazaInAdancime(Graf g)
{
    int i;
    printf("Traversare in adancime:\n");
    for (i=0;i<g.nrNoduri;i++)
    {
        if (g.tabVizitat[i]==0)
        {
            traverseazaInAdancime(g,i);
            printf("\n");
        }
    }
    printf("\n");
}
```

cod de delimitare între două componente conexe

Puncte de articulaţie şi componente biconexe



Puncte de articulaţie şi componente biconexe (2)



componentă biconexă

Aplicaţie

L’ ami de mon ami c’est mon ami.