Universitatea Politehnica din București
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare

SORTAREA TOPOLOGICĂ A GRAFURILOR ORIENTATE

Introducere

- Sortarea topologică este un algoritm care poate fi modelat cu ajutorul grafurilor
- Sortarea topologică este utilă în situația în care anumite elemente sau evenimente trebuie dispuse într-o anumită ordine

Obiectivul algoritmului

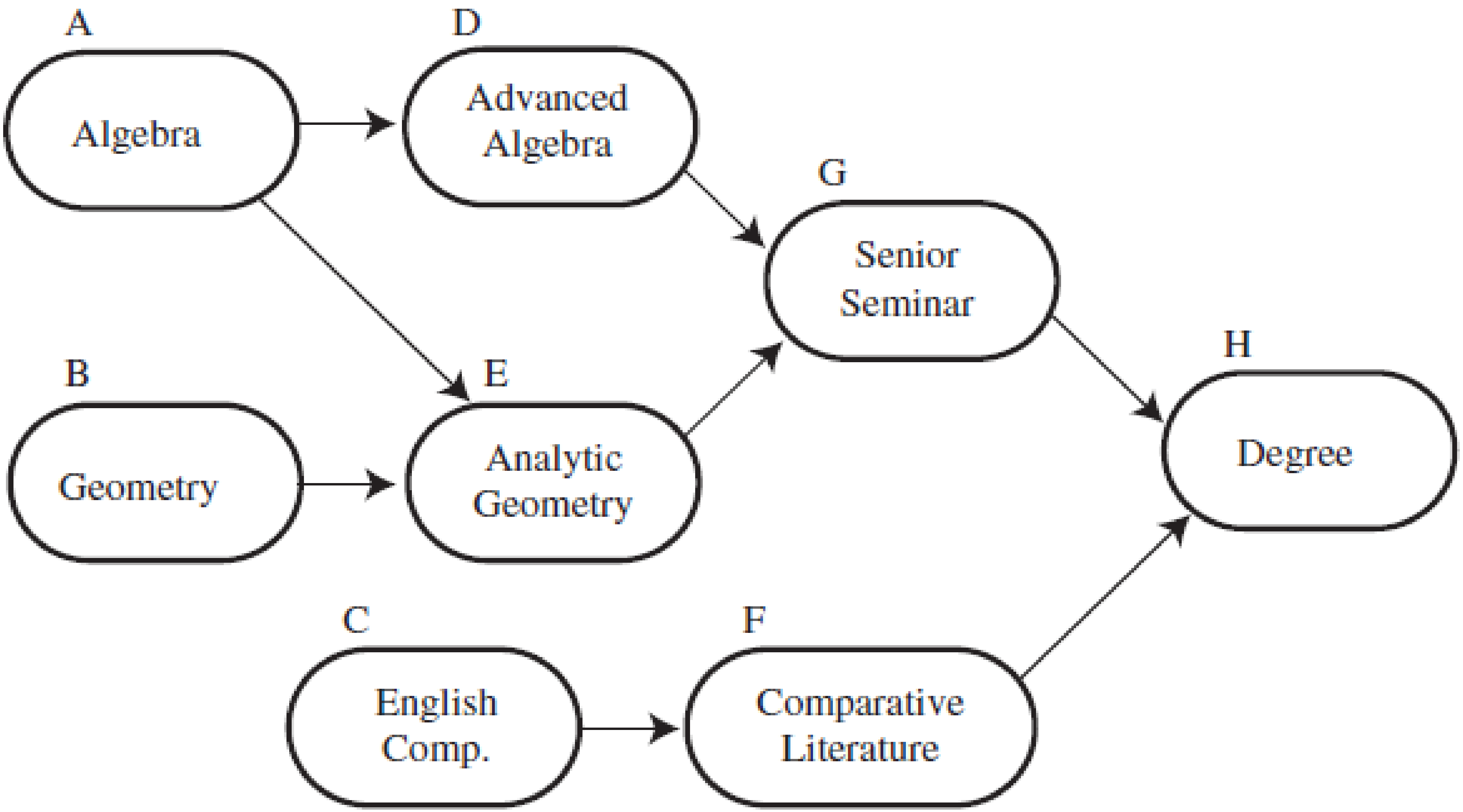
- Ordonarea elementelor într-o succesiune liniară, astfel încât fiecare element să fie precedat în această succesiune de elementele care îl condiţionează
- Elementele pot fi privite ca noduri într-un graf orientat, iar relaţiile de condiţionare ca arce în acest graf

Observații

- Sortarea topologică a nodurilor unui graf orientat **nu este posibilă** dacă graful conţine cel puţin un **ciclu**
- Dacă nu există niciun element fără condiţionări, atunci sortarea nu poate începe
- Uneori este posibilă numai o sortare topologică parţială, pentru o parte din noduri

Condiționarea cursurilor

- Frecventarea unor cursuri este condiționată de absolvirea anterioară a altora
- Alegerea unui anumit pachet de cursuri constituie o premisă pentru obținerea diplomei într-o anumită specialitate


A
Algebra
D
Advanced Algebra
G
Senior Seminar
H
Degree
B
Geometry
E
Analytic Geometry
C
English Comp.
F
Comparative Literature

Modelare cu grafuri orientate

- În graful care modelează relațiile de condiționare, arcele trebuie să aibă o **orientare**
- În acest caz, graful este **orientat**
- Într-un graf orientat, ne putem deplasa într-un singur sens, de-a lungul unui arc

Sortarea topologică

- Se alcătuiește o listă a tuturor cursurilor necesare pentru obținerea unei diplome
- Se aranjează cursurile în ordinea în care acestea pot fi frecventate
- Obținerea diplomei reprezintă ultimul punct de pe listă, care poate arăta astfel:
- BAEDGCFH

Observații

- Aranjat în acest mod, graful este **sortat topologic**
- Toate cursurile care trebuie absolvite înaintea altui curs dat îl vor precede pe acesta în listă
- Există mai multe posibilități de a ordona cursurile, satisfăcând restricțiile de condiționare

Observații

- O altă sortare topologică este:
- CFBAEDGH
- Sortarea topologică poate modela și alte situații, cum ar fi planificarea unor activități
- Modelarea planificării unor activități cu ajutorul grafurilor se numește **analiză a drumului critic**

3 soluții pentru sortarea topologică

- Determinarea unei soluţii de sortare topologică se poate face în 3 moduri:
- **1**) Începând cu elementele fără predecesori (necondiţionate) şi continuând cu elementele care depind de acestea

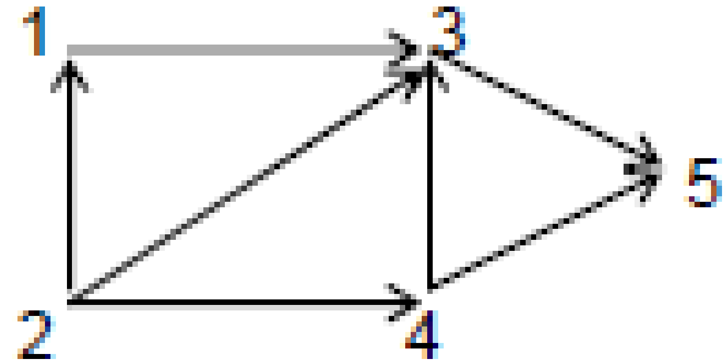
3 soluții pentru sortarea topologică

- **2**) Începând cu elementele fără succesori (finale) şi mergând către predecesori, din aproape în aproape
- **3**) Algoritmul de explorare în adâncime a unui graf orientat, completat cu afişarea nodului din care începe explorarea, după ce s-au explorat toate celelalte noduri

Observaţii

- Aceste metode pot folosi diferite structuri de date pentru reprezentarea relaţiilor dintre elemente
- În cazul **1**) trebuie să putem găsi uşor predecesorii unui element
- În cazul **2**) trebuie să putem găsi uşor succesorii unui element

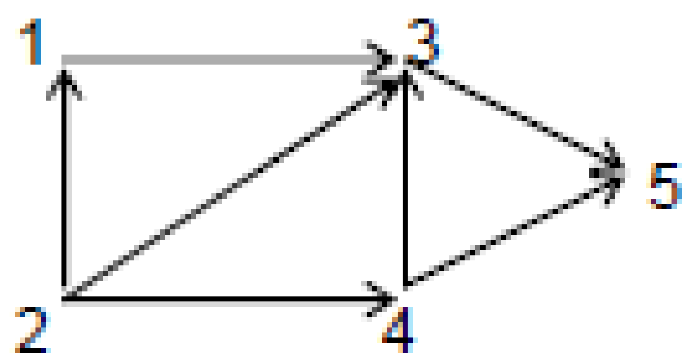
Exemplu



- **a** = {1, 2, 3, 4, 5}
- Există relaţii de condiţionare de forma:
- a[i] << a[j], exprimate astfel:
- a[i] **condiţionează** pe a[j], sau
- a[j] **este condiţionat** de a[i]
- a[i] este un **predecesor** al lui a[j]
- a[j] este un **succesor** al lui a[i]
- Un element poate avea oricâţi succesori şi predecesori

Observații

- Mulţimea **a**, supusă unor relaţii de condiţionare, poate fi vazută ca un graf orientat, având ca noduri elementele a[i]
- Un **arc** de la a[i] la a[j] arată că a[i] **condiţionează** pe a[j]
- 2 << 1   1 << 3   2 << 3   2 << 4   4 << 3   3 << 5   4 << 5
- Două soluţii posibile:
  - 2, 1, 4, 3, 5
  - 2, 4, 1, 3, 5



1) Sortare topologică cu liste de predecesori

repetă
caută un nod nemarcat şi fără predecesori
dacă s-a găsit ***atunci***
afişează nod şi marchează nod
şterge nod marcat din graf
până când nu mai sunt noduri fără predecesori
dacă rămân noduri nemarcate ***atunci***
nu este posibilă sortarea topologică

Se obține sortarea topologică: 2, 1, 4, 3, 5

Funcţia de sortare topologică

```
//funcţia determină numărul de condiţionări ale nodului v
int nrcond (Graf g, int v ) {
  int j, cond = 0;              // cond = număr de condiţionări
  for (j = 1; j <= g.n; j++)
    if (arc(g, j, v))
        cond++;
  return cond;
}
```

Funcţia de sortare topologică

```
// funcţia de sortare topologică şi afişarea rezultatului
void topsort (Graf g) {
  int i, j, n = g.n, ns, gasit, sortat[50] = {0};
  ns = 0;                 // noduri sortate şi afişate
  do {
    gasit = 0;
     // caută un nod nesortat, fără condiţionări
    for (i = 1; i <= n && ! gasit; i++)
```

Funcţia de sortare topologică

```
  if ( ! sortat[i] && nrcond(g, i) == 0) {  // i fără condiţionări
              găsit = 1;
              sortat[i] = 1;  ns++;  // noduri sortate
              printf("%d ", i);        // afişează nod găsit
              delNod(g, i);            // elimină nodul i din graf
      }
  } while (găsit);
  if (ns != n)  printf("\n Sortare topologica imposibila !");
}
```

2) Sortare topologică cu liste de succesori

repetă
  caută un nod fără succesori
  pune nod găsit în stivă şi marchează ca sortat
  elimină nod marcat din graf
până când nu mai există noduri fără succesori
dacă nu mai sunt noduri nemarcate ***atunci***
 repetă
  scoate nod din stivă şi afişează nod
 până când stiva e goală

La extragerea din stivă se obține: 2, 4, 1, 3, 5

3) Sortare topologică folosind explorarea în adâncime

- Algoritmul de sortare topologică, derivat din explorarea DFS, se bazează pe faptul că explorarea în adâncime vizitează toţi succesorii unui nod
- Explorarea DFS va fi repetată, până când se vizitează toate nodurile din graf

Observații

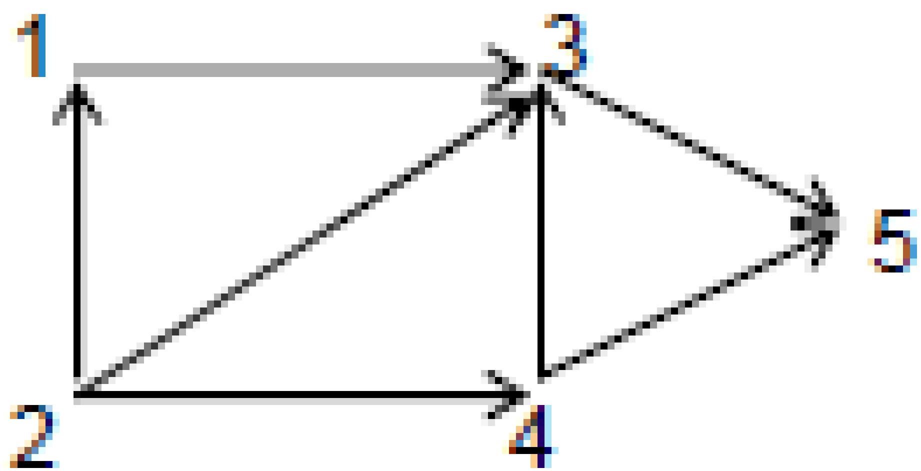
- Funcţia **ts** este derivată din funcţia **dfs**, în care s-a înlocuit afişarea cu punerea într-o stivă a nodului cu care s-a început explorarea, după ce s-au memorat în stivă succesorii săi
- În final se scoate de pe stivă şi se afişează tot ce a pus pe stivă funcţia **ts**
- Implementarea următoare realizează sortarea topologică, ca o variantă de explorare în adâncime a unui graf **g**, folosind o stivă **s** pentru memorarea nodurilor

```
Stack s;              // stiva se foloseşte în două funcţii
// sortare topologică pornind dintr-un nod v
void ts (Graf g, int v) {
  vazut[v] = 1;
  for (int w = 1; w <= g.n; w++)
    if (arc(g, v, w) && ! vazut[w])
      ts(g, w);
  push(s, v);
}
```

```
// sortare topologică a grafului g
int main () {
  int i, j, n; Graf g;
  readG(g); n = g.n;
  for (j = 1; j <= n; j++)
    vazut[j] = 0;
  initSt(s);
  for (i = 1; i <= n; i++)
    if (vazut[i] == 0)
       ts(g, i);
  while( ! emptySt(s)) {     // scoate din stivă şi afişează
     pop(s, i);
     printf("%d ", i);
  }
}
```

Exemplu

- Secvenţa de apeluri şi evoluţia stivei pentru graful:
- 2-1,1-3, 2-3, 2-4, 4-3, 3-5, 4-5



Apel	**Stiva**	**Din**
ts(1)		main()
ts(3)		ts(1)
ts(5)		ts(3)
push(5)	5	ts(5)
push(3)	5, 3	ts(3)
push(1)	5, 3, 1	ts(1)
ts(2)		main()
ts(4)		ts(2)
push(4)	5, 3, 1, 4	ts(4)
push(2)	5, 3, 1, 4, 2	ts(2)