TDDI16 – Föreläsning 7

Fler grafalgoritmer

Filip Strömbäck

LINKÖPING UNIVERSITY

Planering

Vecka	Fö	Lab
36	Komplexitet, Linjära strukturer	----
37	Träd, AVL-träd	1---
38	Hashning	12--
39	Grafer och kortaste vägen	12--
40	Fler grafalgoritmer	-23-
41	Sortering	--3-
42	Mer sortering, beräkningsbarhet	--34
43	Tentaförberedelse	---4

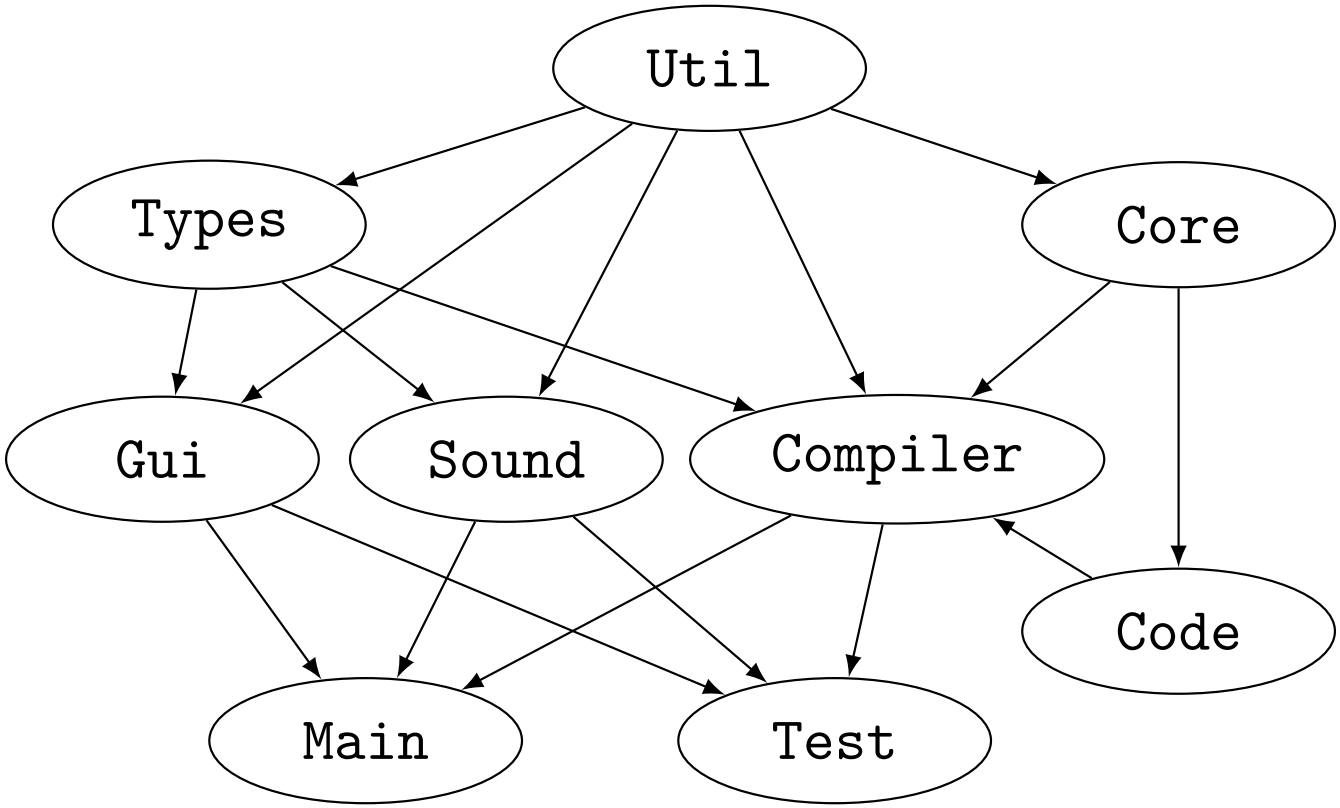
LINKÖPING UNIVERSITY

Problem

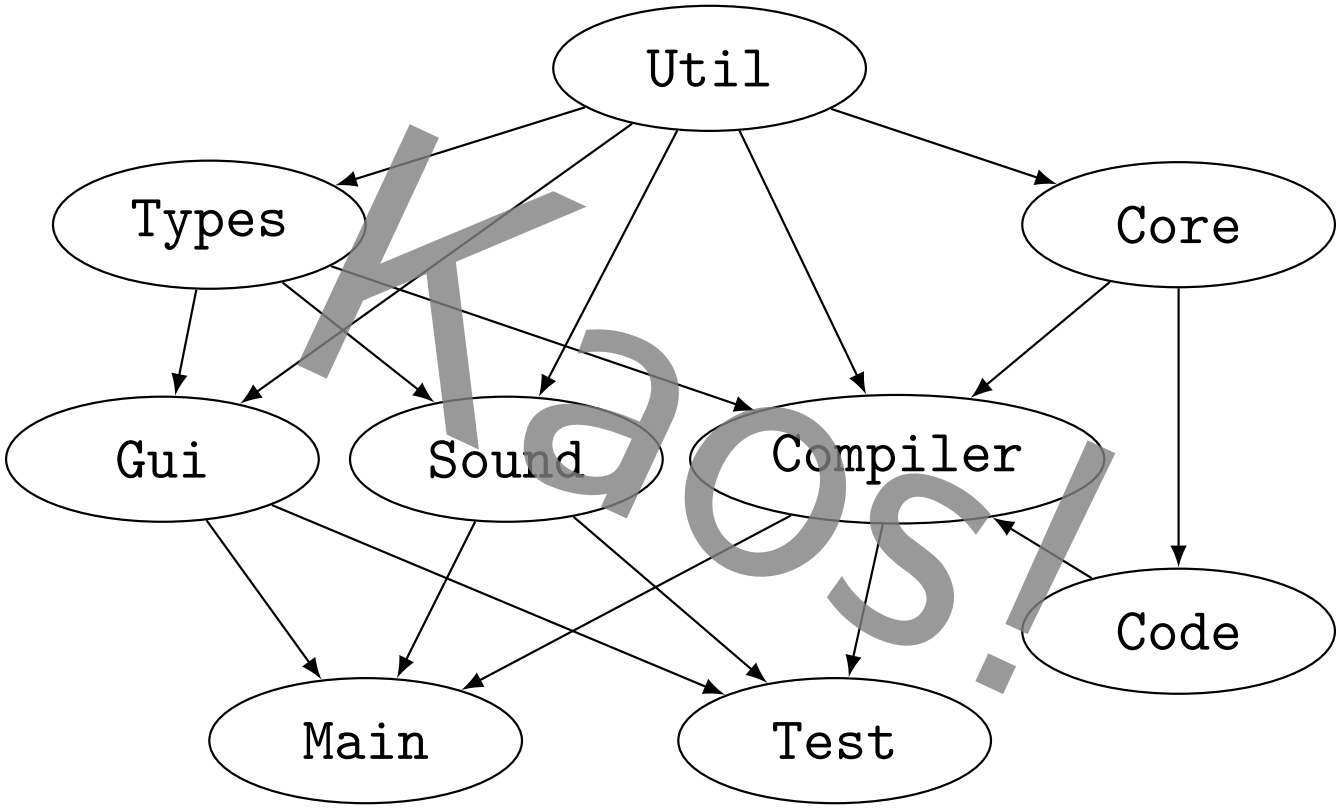
Du ska kompilera ett stort mjukvaruprojekt som består av ett antal delprojekt. Dessa projekt beror på varandra på olika sätt, oftast på ett sådant sätt att det som produceras av ett projekt används för att bygga andra projekt.

I och med att projektet har vuxit sig så stort har du blivit trött på att hålla koll på alla beroenden själv, så du vill bygga ett program som håller koll på beroendena åt dig, och bygger projekten i rätt ordning utifrån det.

Problem



Problem



lakttagelser:

Det här är ju en graf!

Vi vill hitta en sekvens av noder så att:

- Om det finns en båge från a till b så ska a vara före b
- Alla noder finns med exakt en gång

$\Rightarrow$ Grafen får alltså inte innehålla cykler

$\Rightarrow$ Finns ofta flera sekvenser

Kallas **topologisk sortering**

Topologisk sortering

Det finns två vanliga algoritmer:

Kahn's algoritm: $\mathcal{O}(|E| + |V|)$

Undersök vilka noder som inte har beroenden, välj sådana successivt.

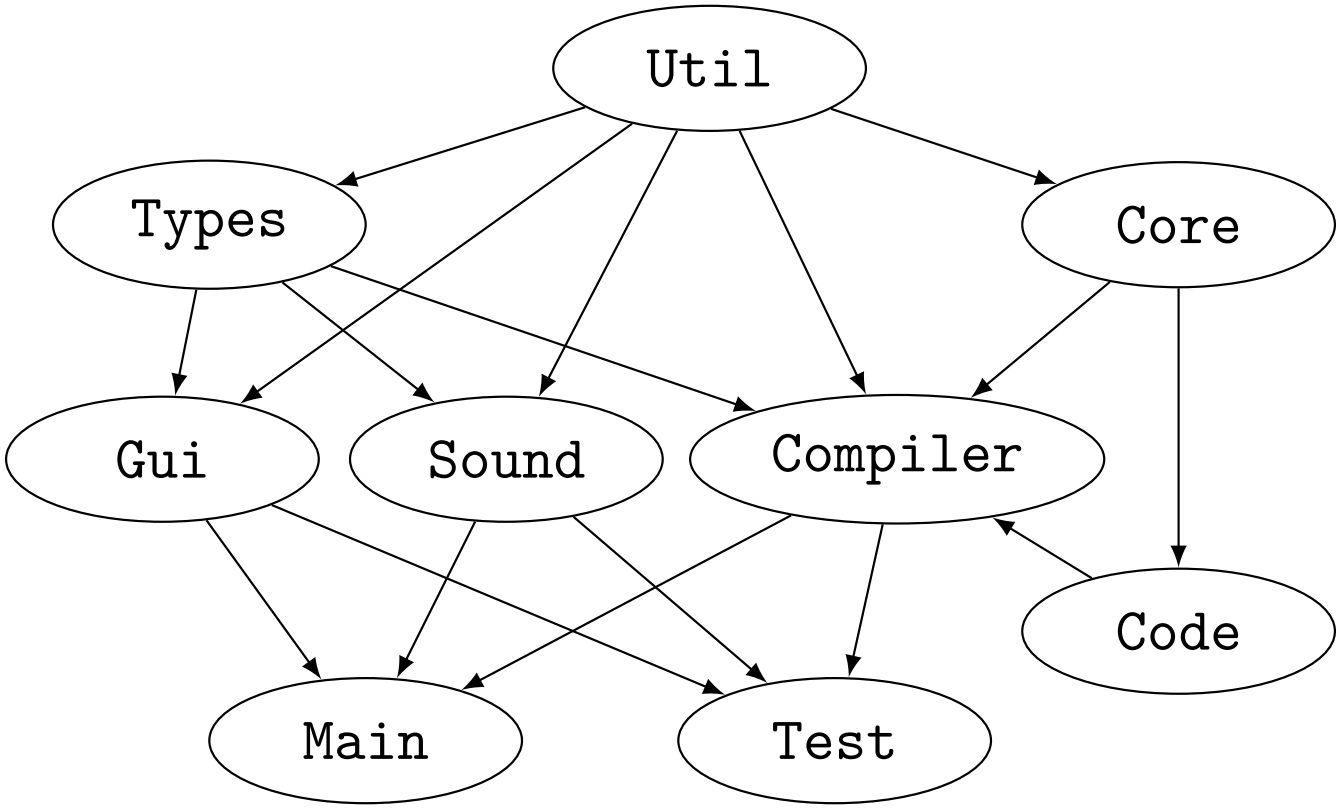
DFS: $\mathcal{O}(|E| + |V|)$

Utnyttja en DFS för att hitta en ordning baklänges (liknande postorder för trädtraversering).

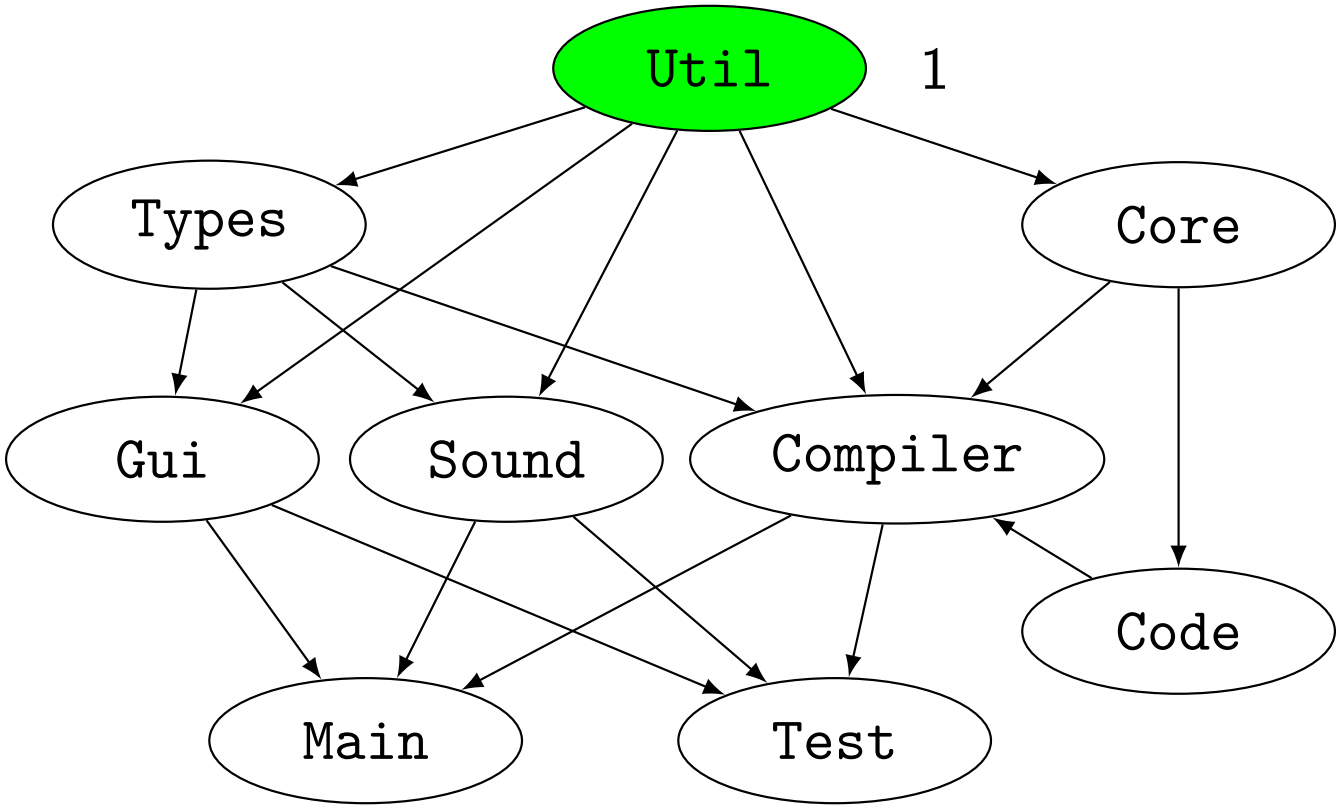
Topologisk sortering – Kahn's algoritm

1. Hitta alla noder utan inkommande bågar (*indegree* $= 0$), läg dem på en kö, Q
2. Välj den första noden i kön, lägg den sist i resultatsekvensen, R
3. För varje utåtgående båge: minska nodens grad med 1 och lägg på Q om den blev 0
4. Repetera från steg 2 tills Q är tom

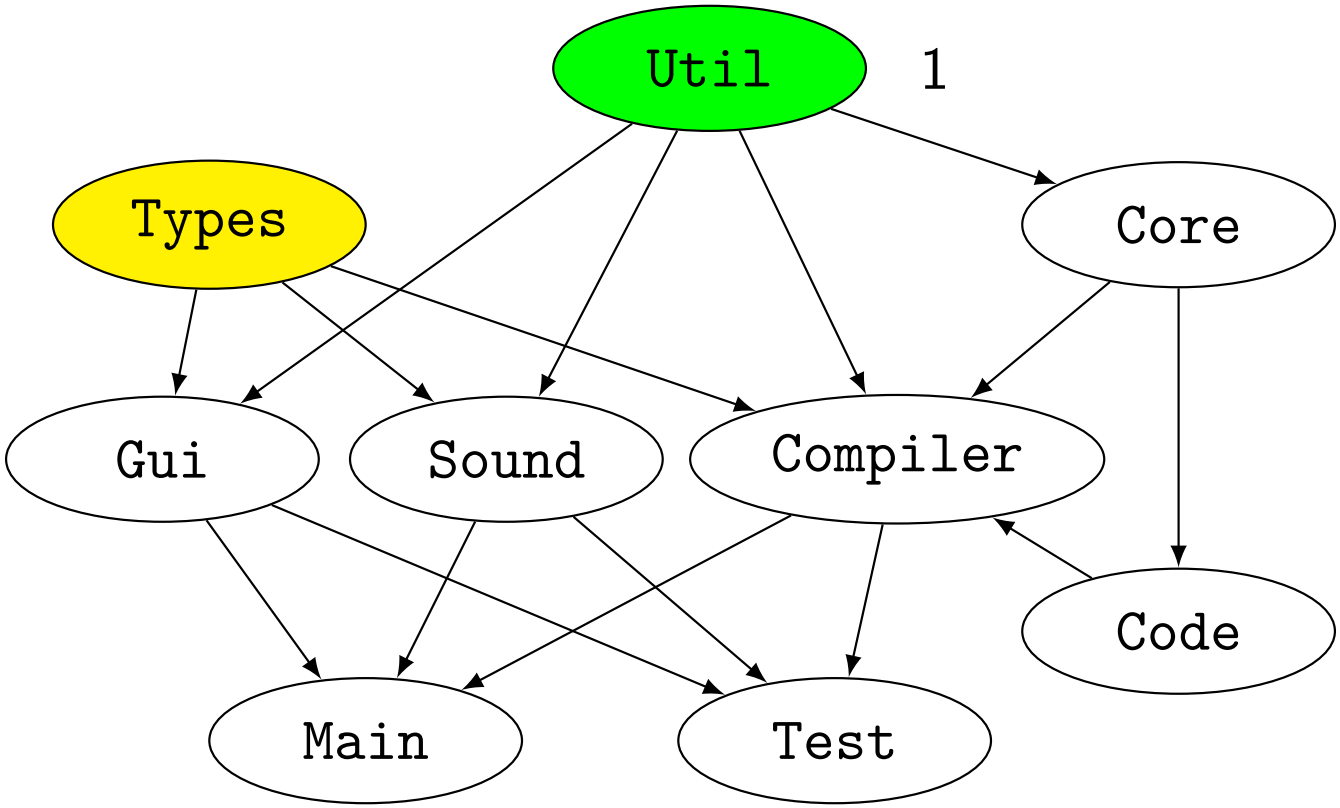
Topologisk sortering – Kahn's algoritm



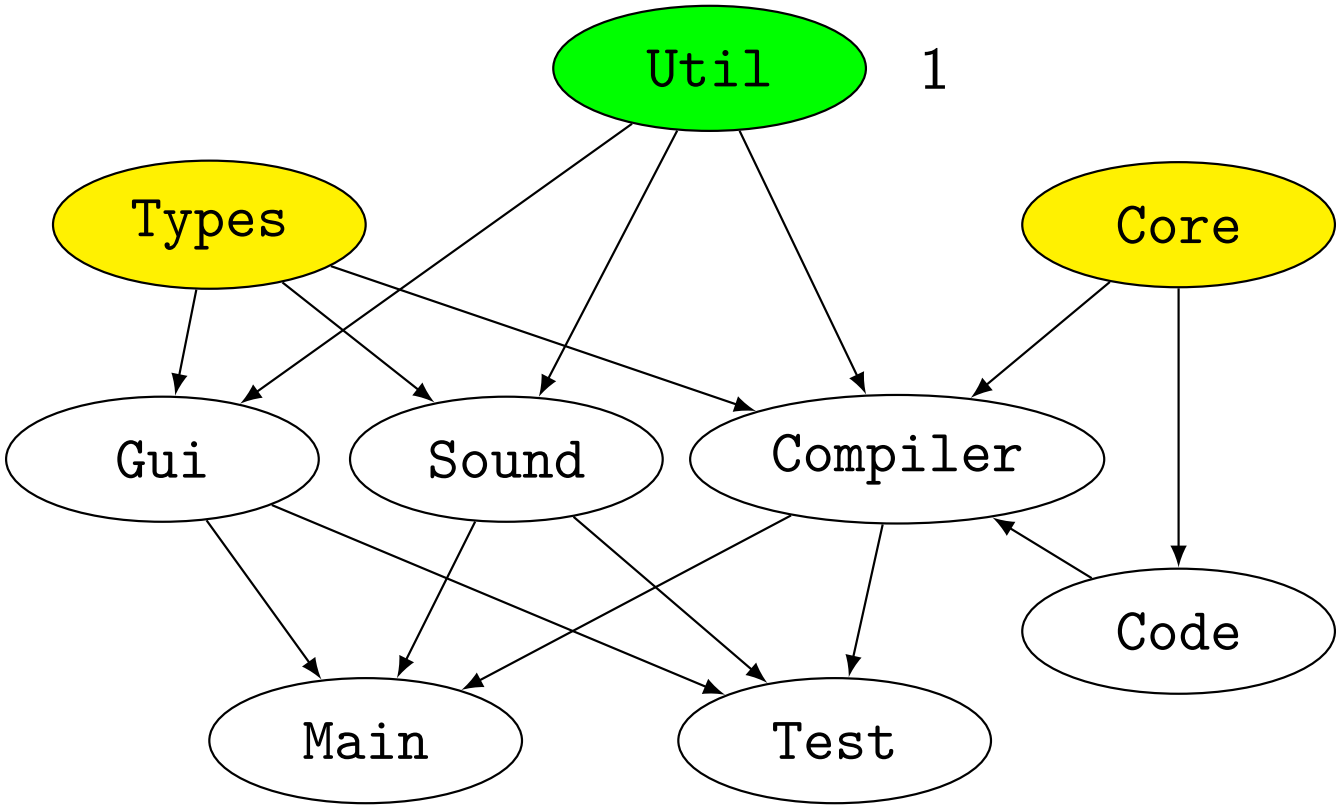
Topologisk sortering – Kahn's algoritm



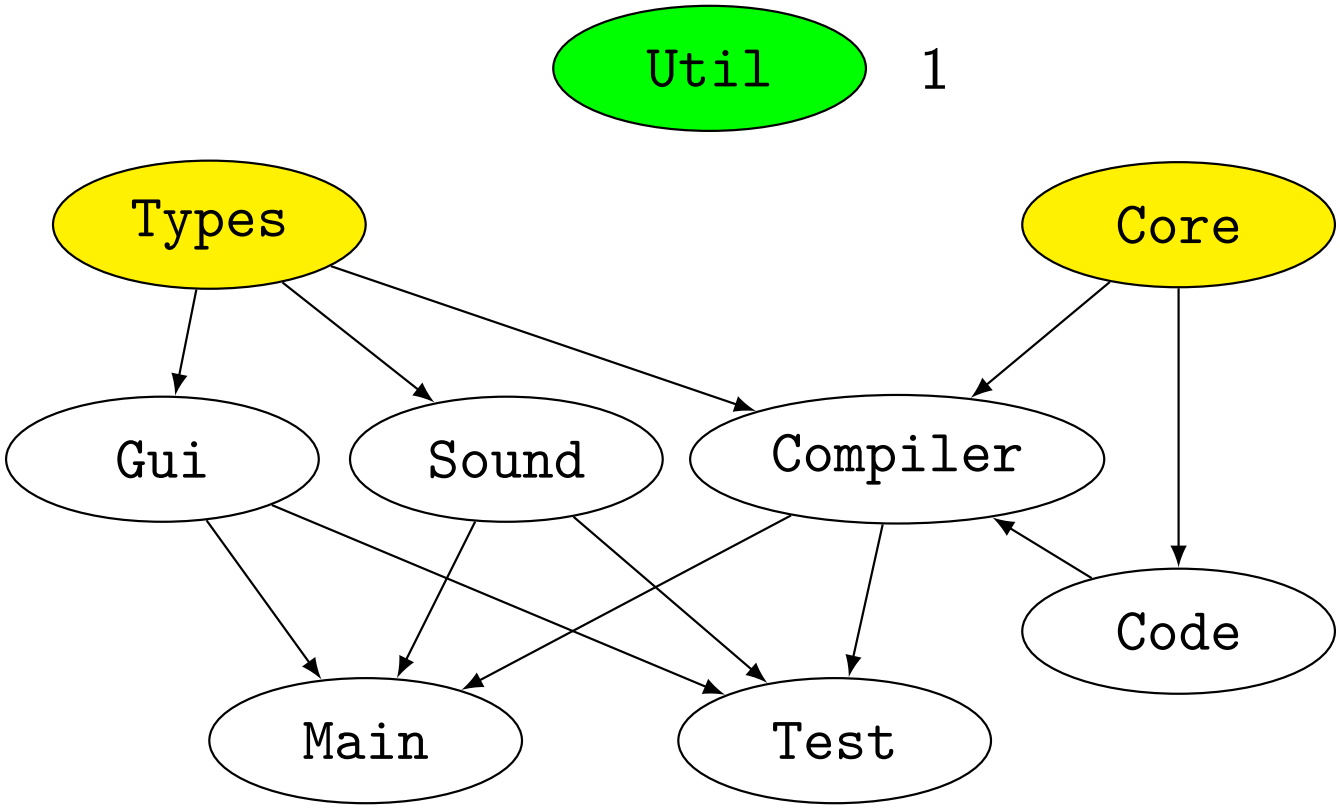
Topologisk sortering – Kahn's algoritm



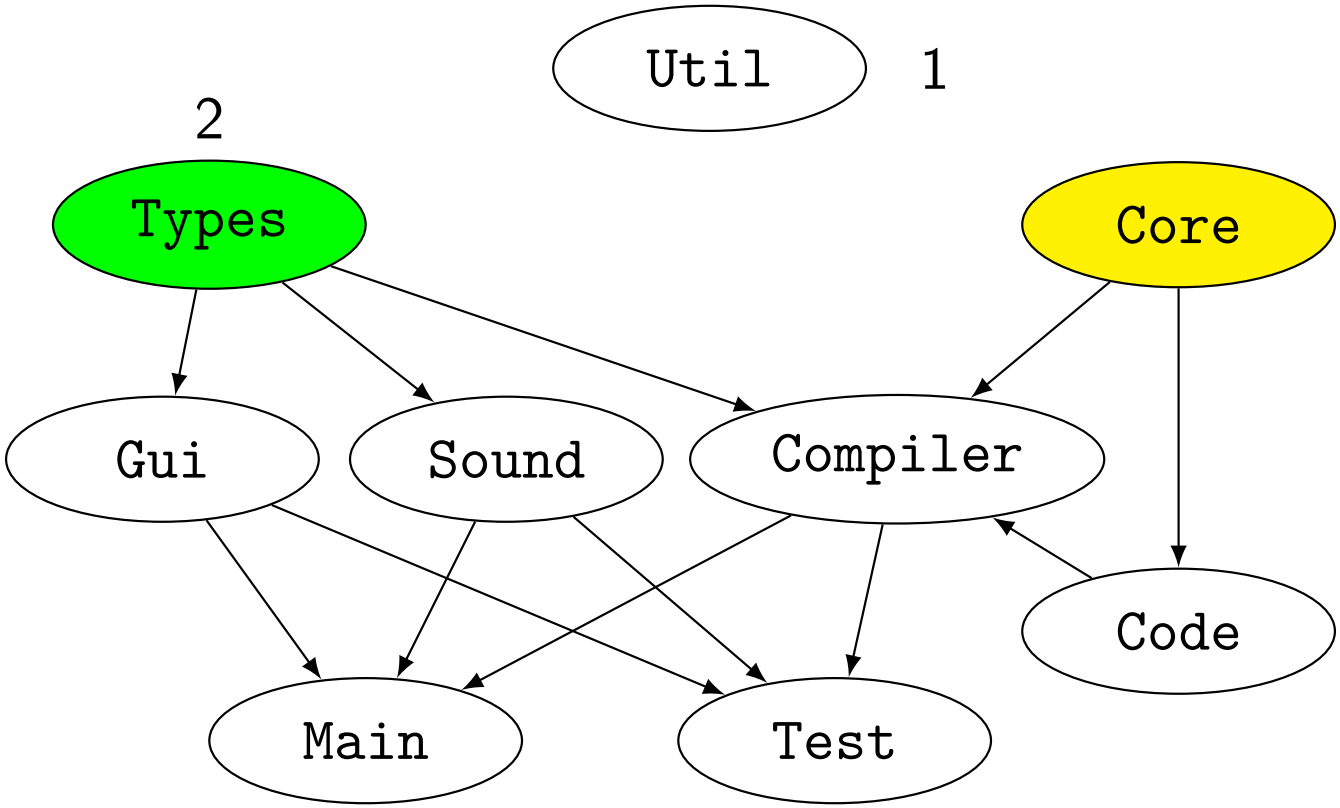
Topologisk sortering – Kahn's algoritm



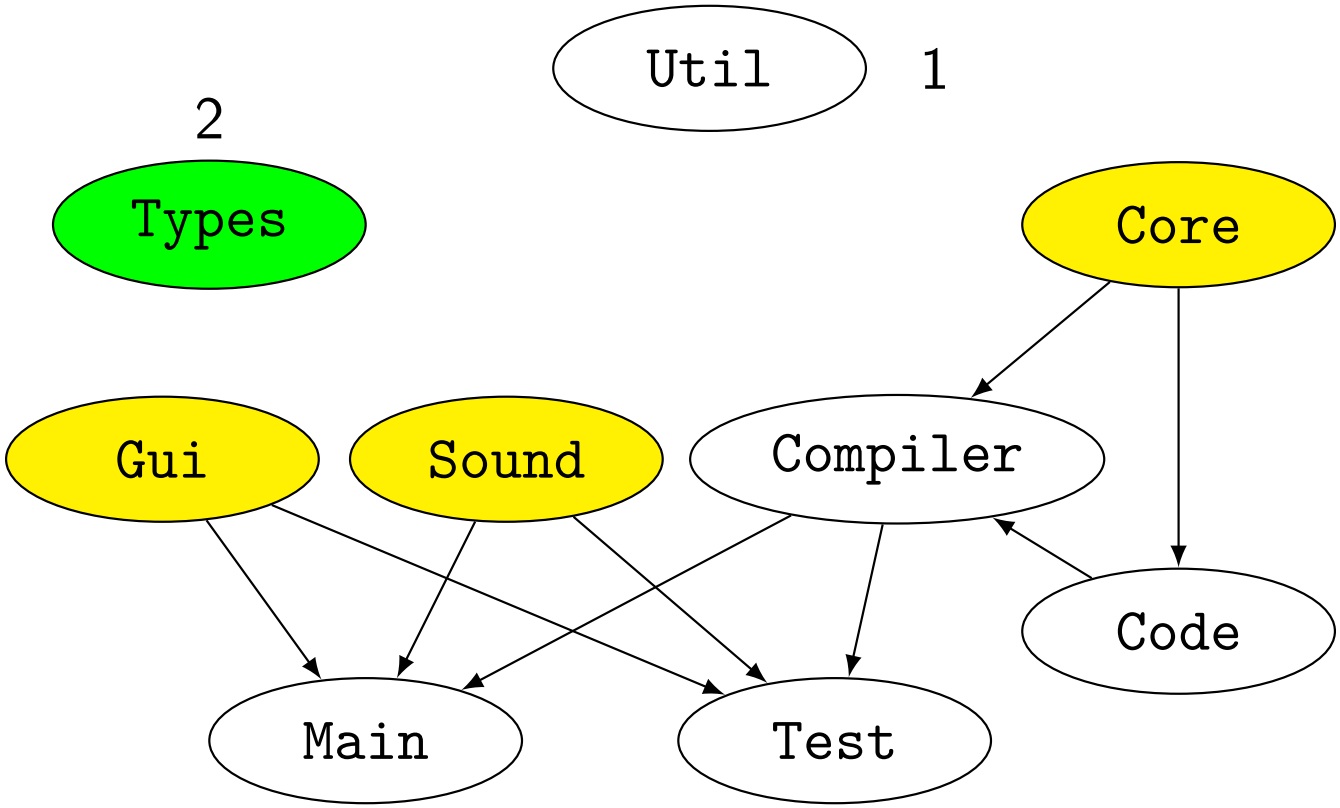
Topologisk sortering – Kahn's algoritm



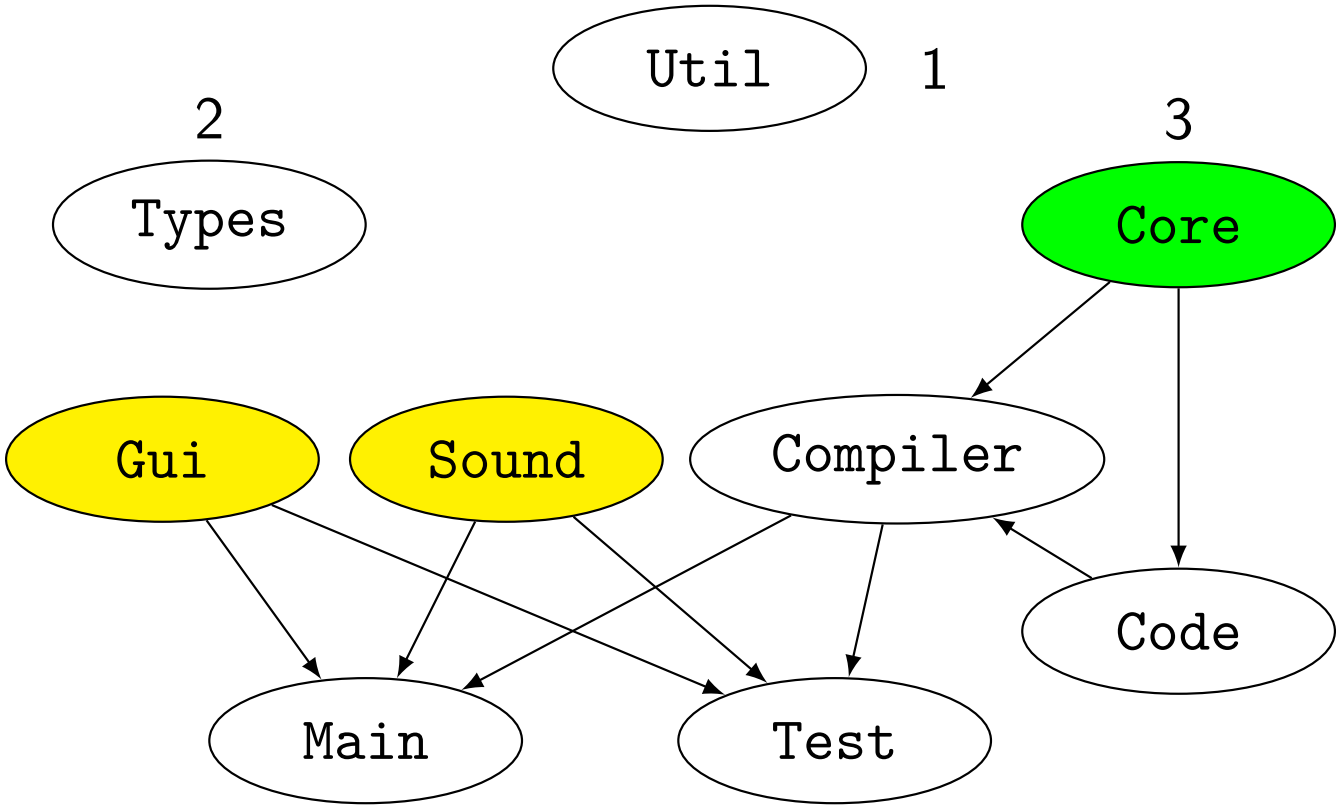
Topologisk sortering – Kahn's algoritm



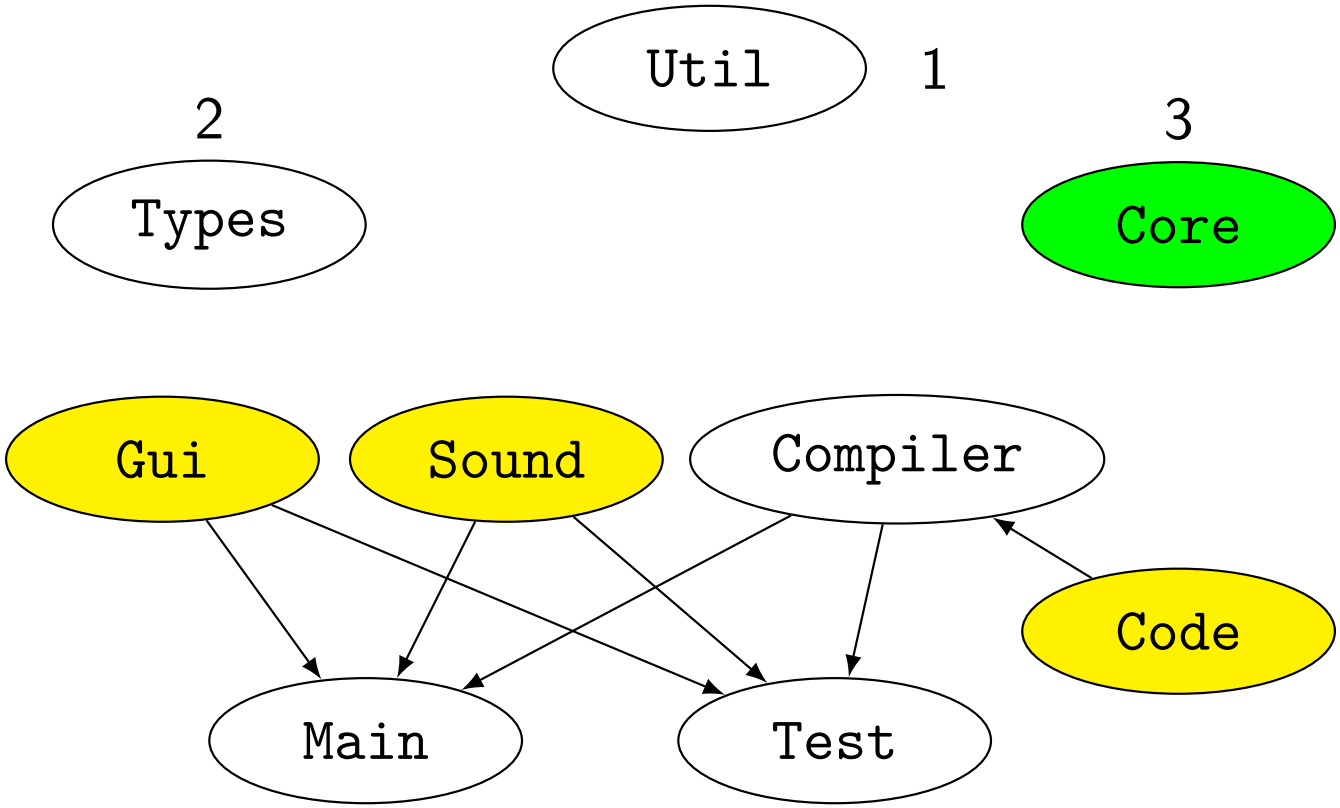
Topologisk sortering – Kahn's algoritm



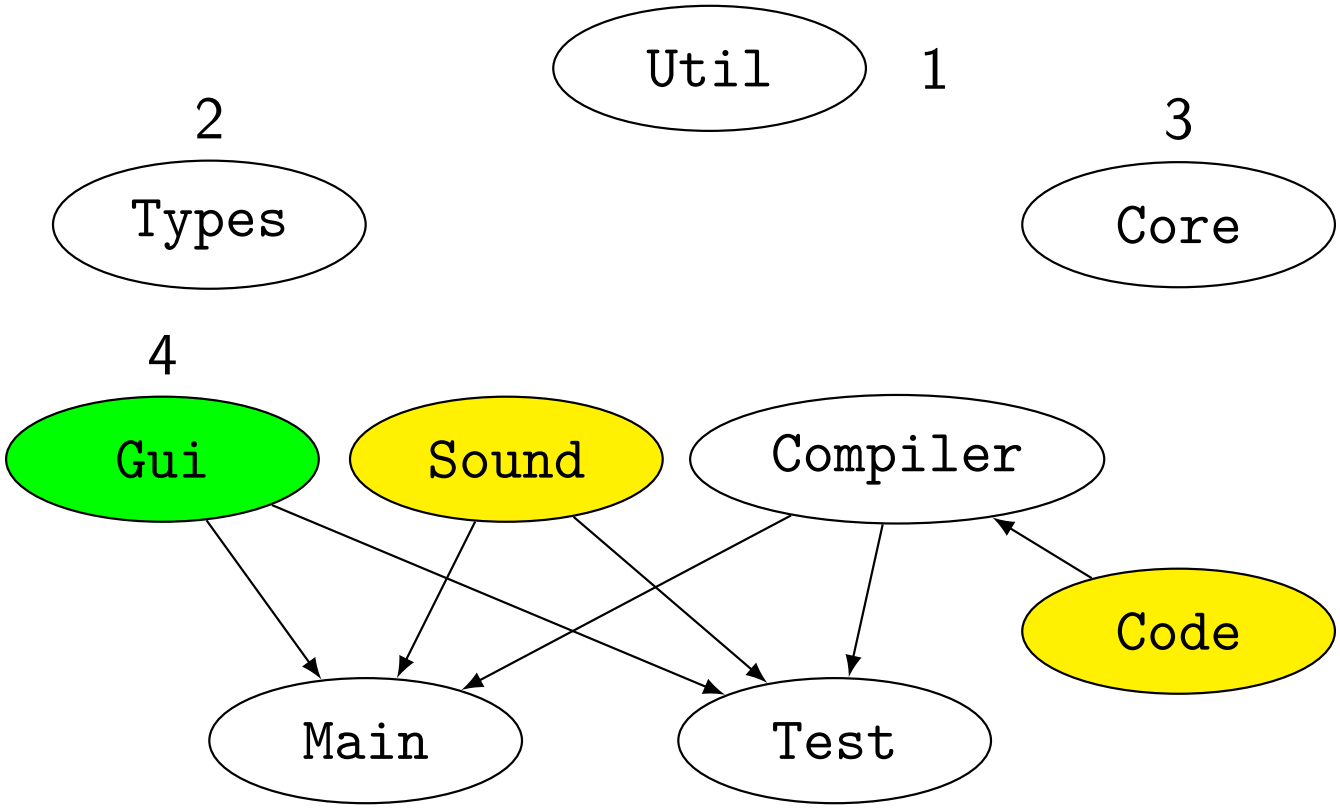
Topologisk sortering – Kahn's algoritm



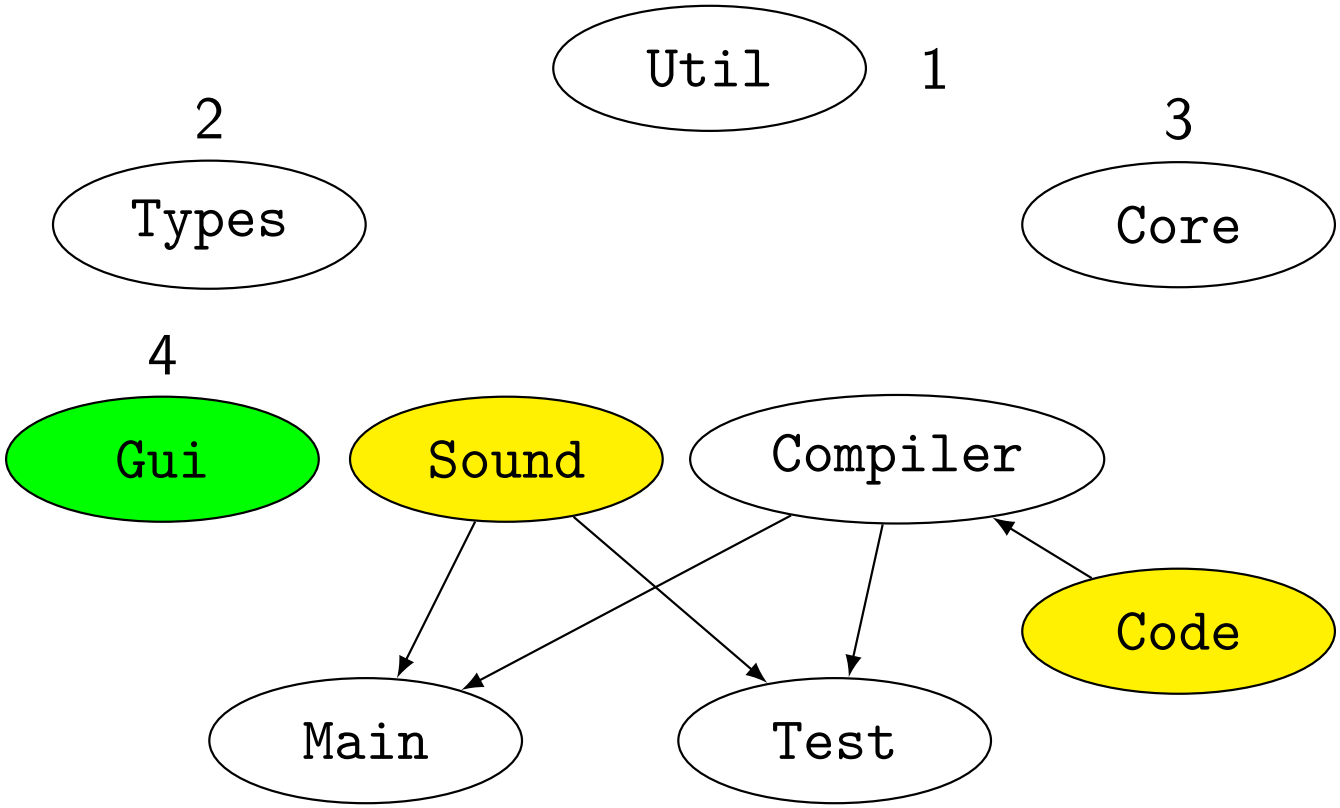
Topologisk sortering – Kahn's algoritm



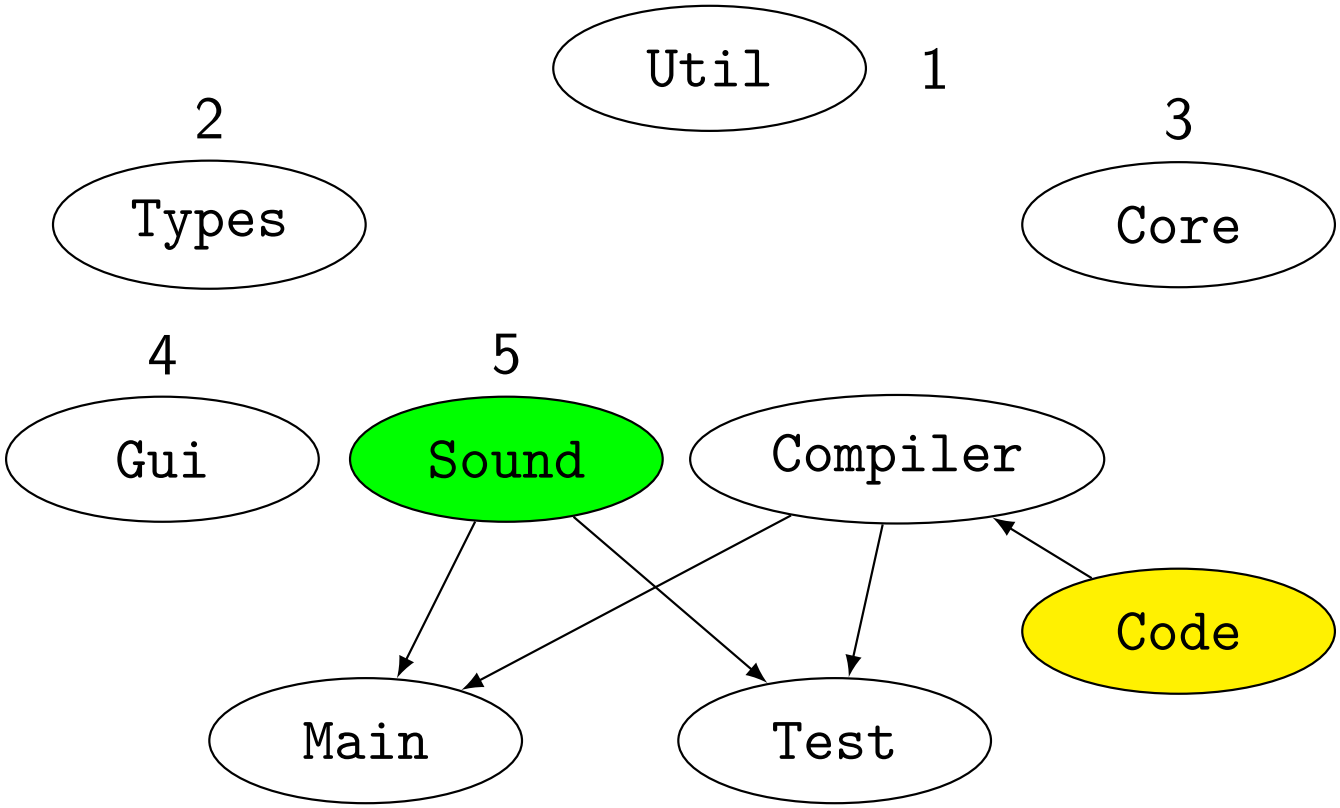
Topologisk sortering – Kahn's algoritm



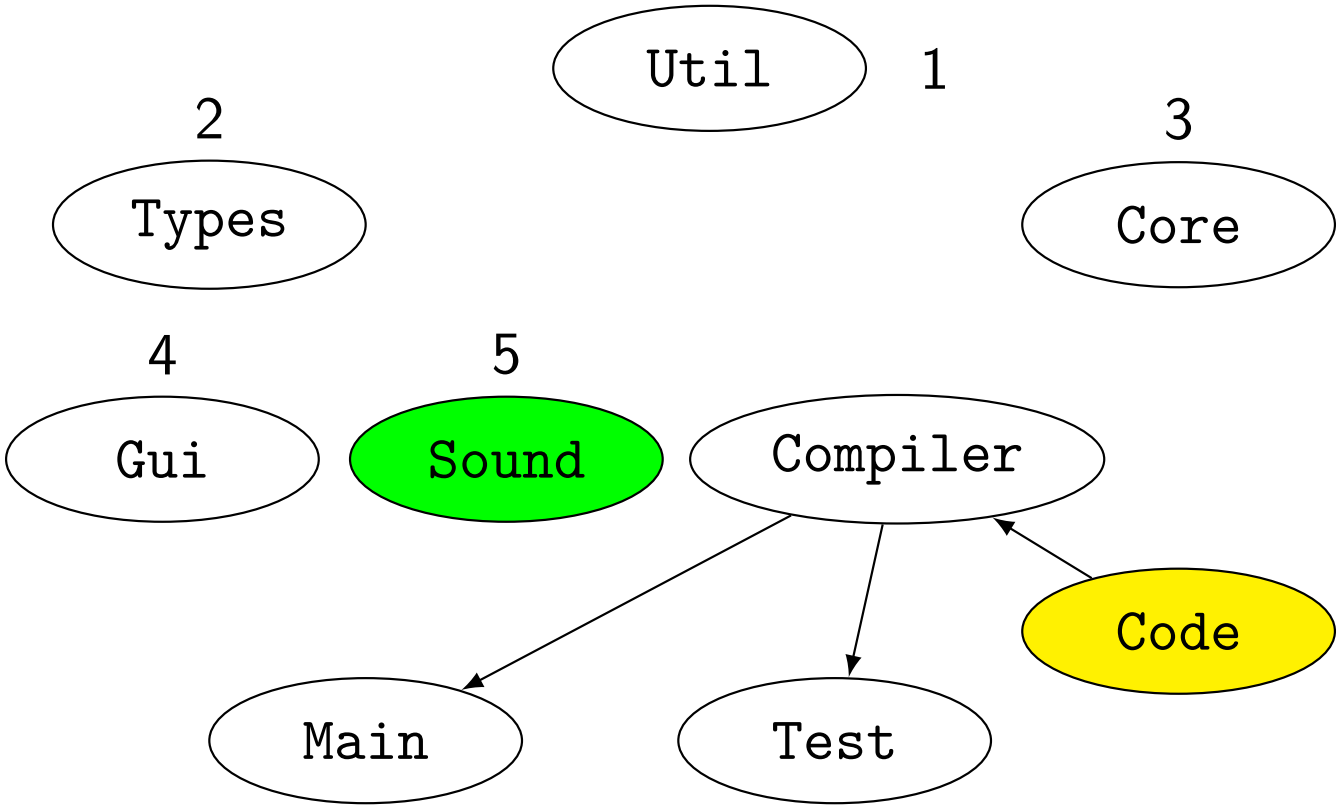
Topologisk sortering – Kahn's algoritm



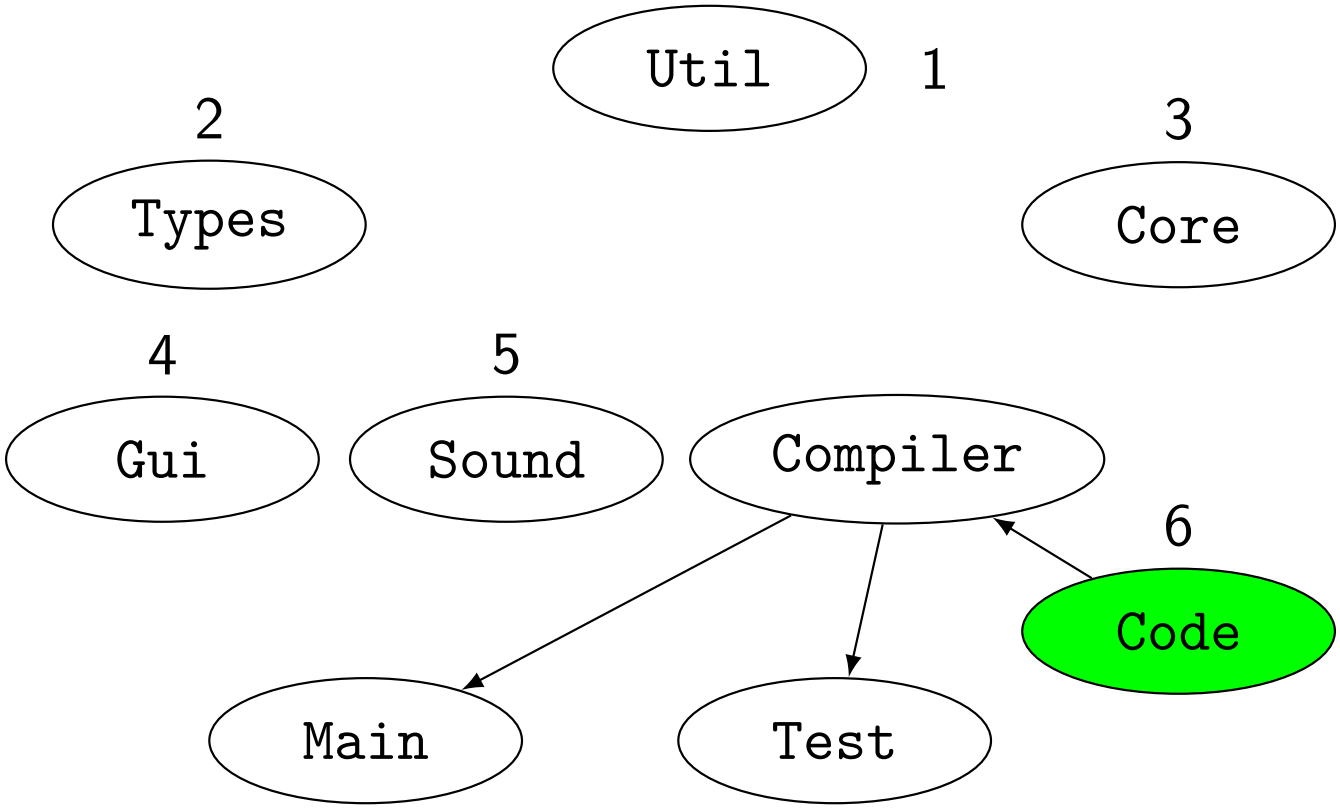
Topologisk sortering – Kahn's algoritm



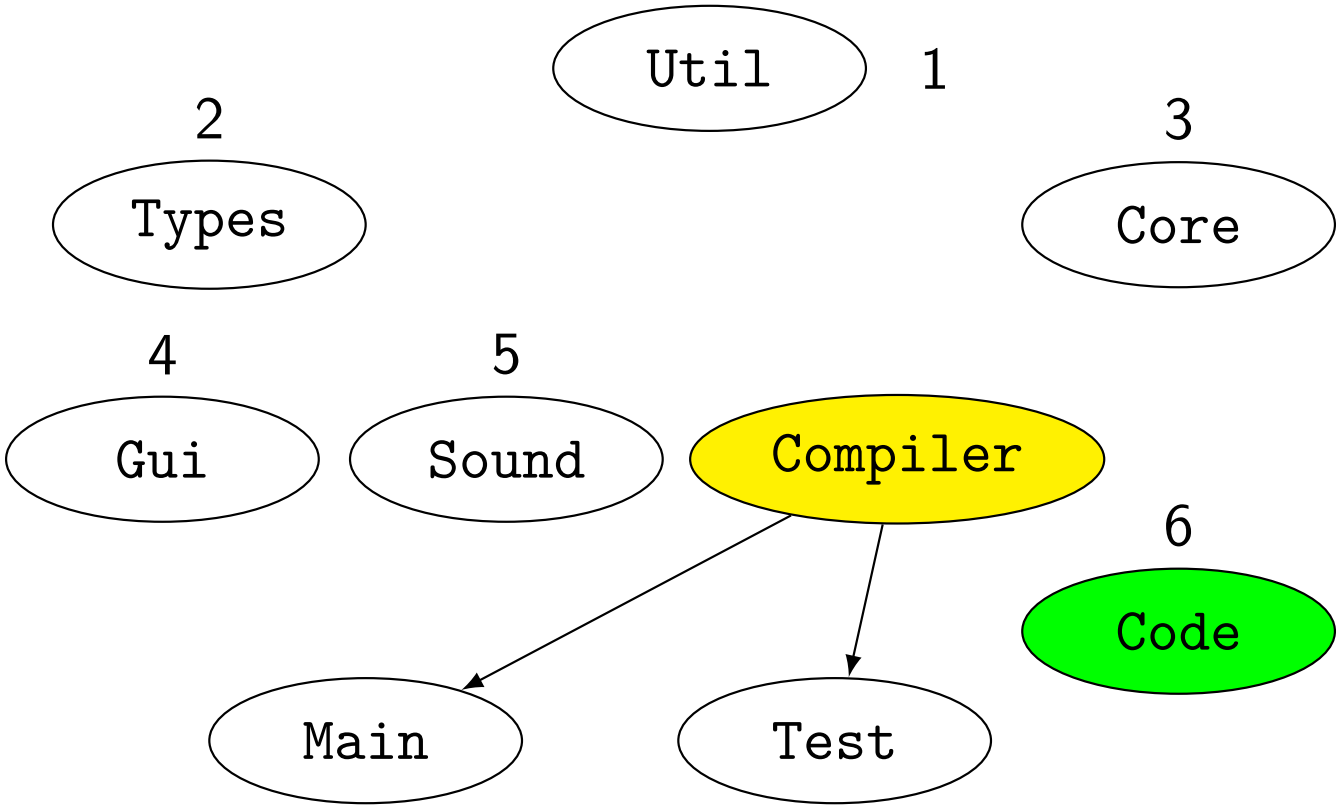
Topologisk sortering – Kahn's algoritm



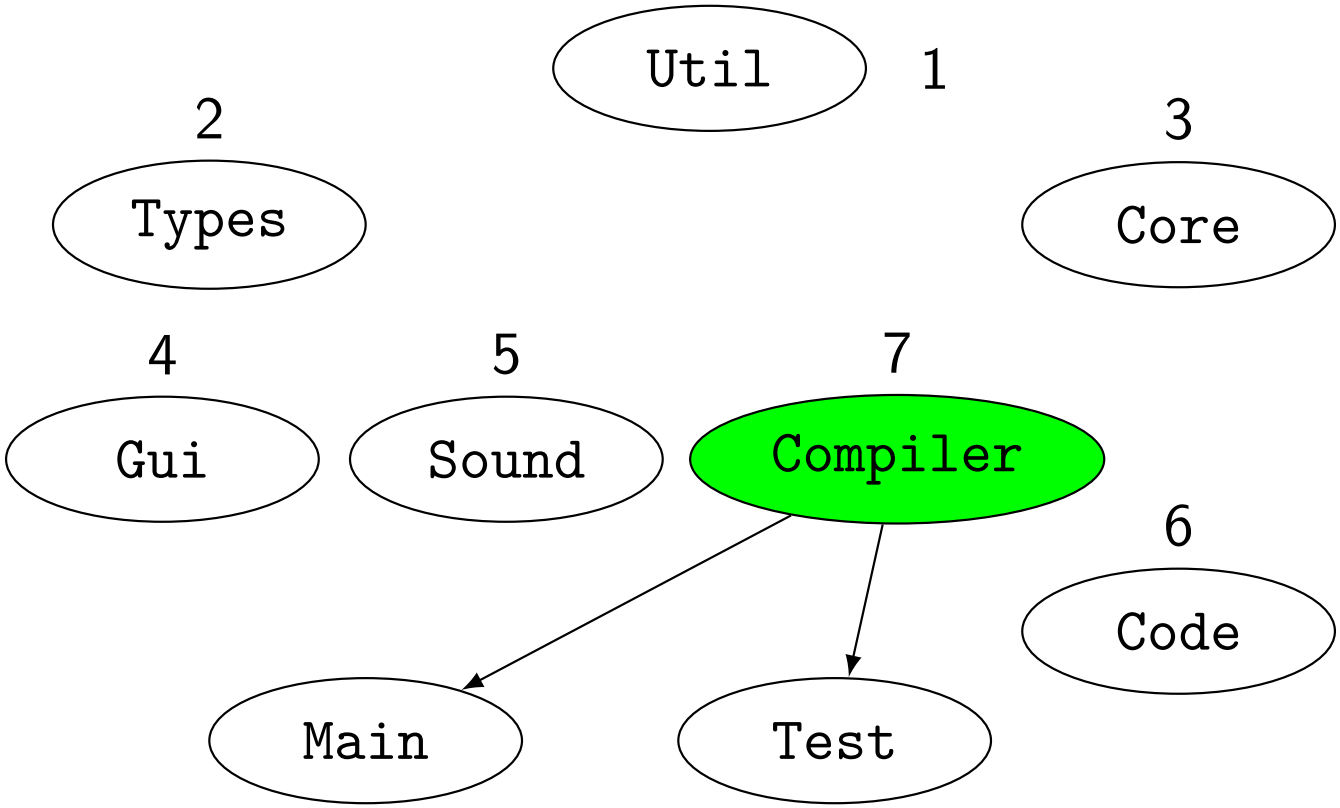
Topologisk sortering – Kahn's algoritm



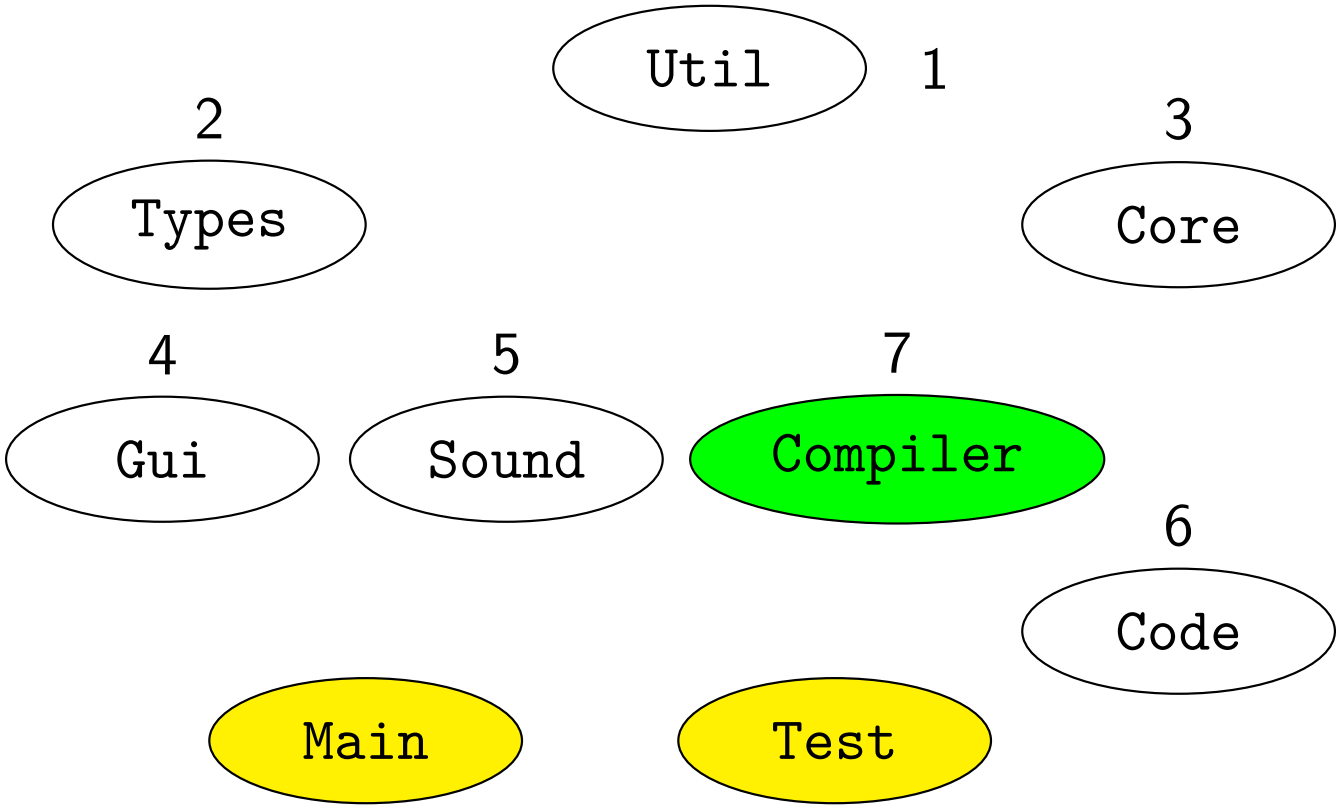
Topologisk sortering – Kahn's algoritm



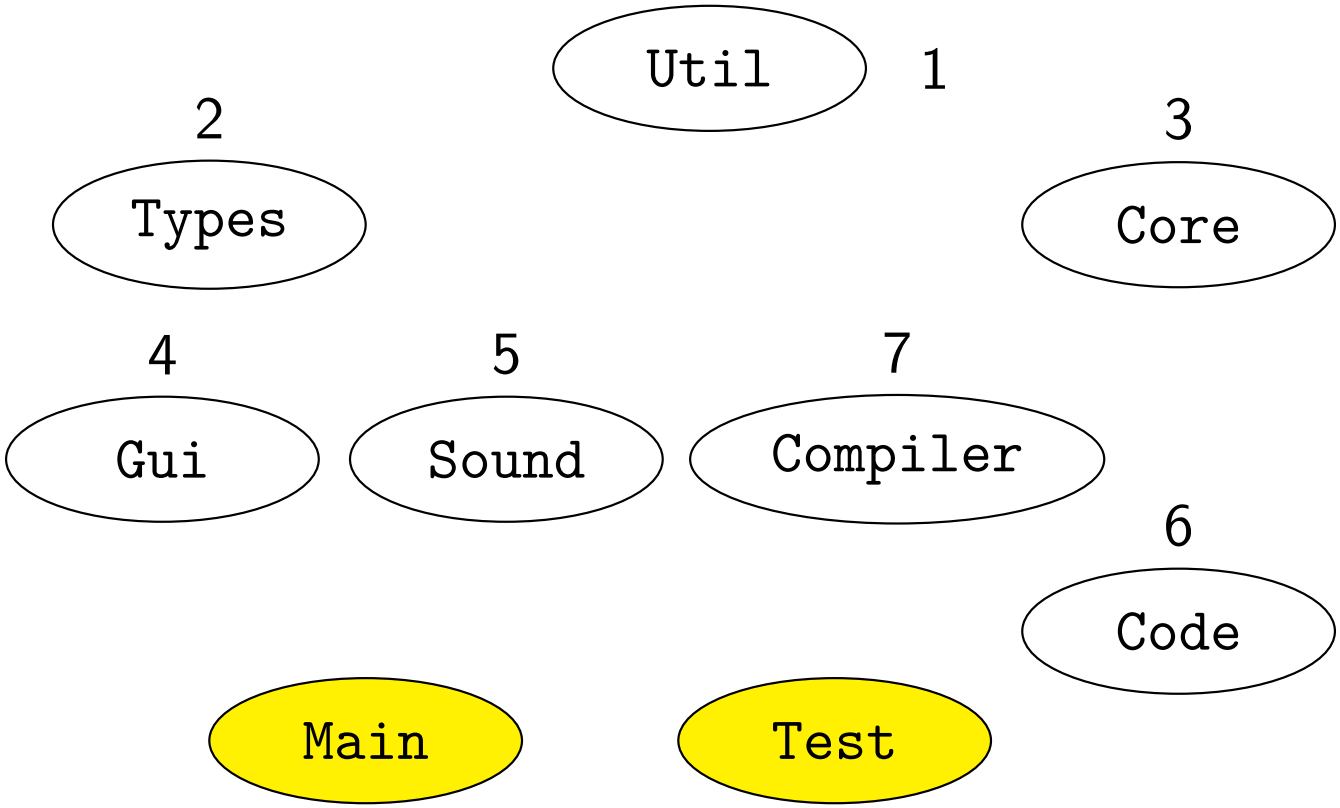
Topologisk sortering – Kahn's algoritm



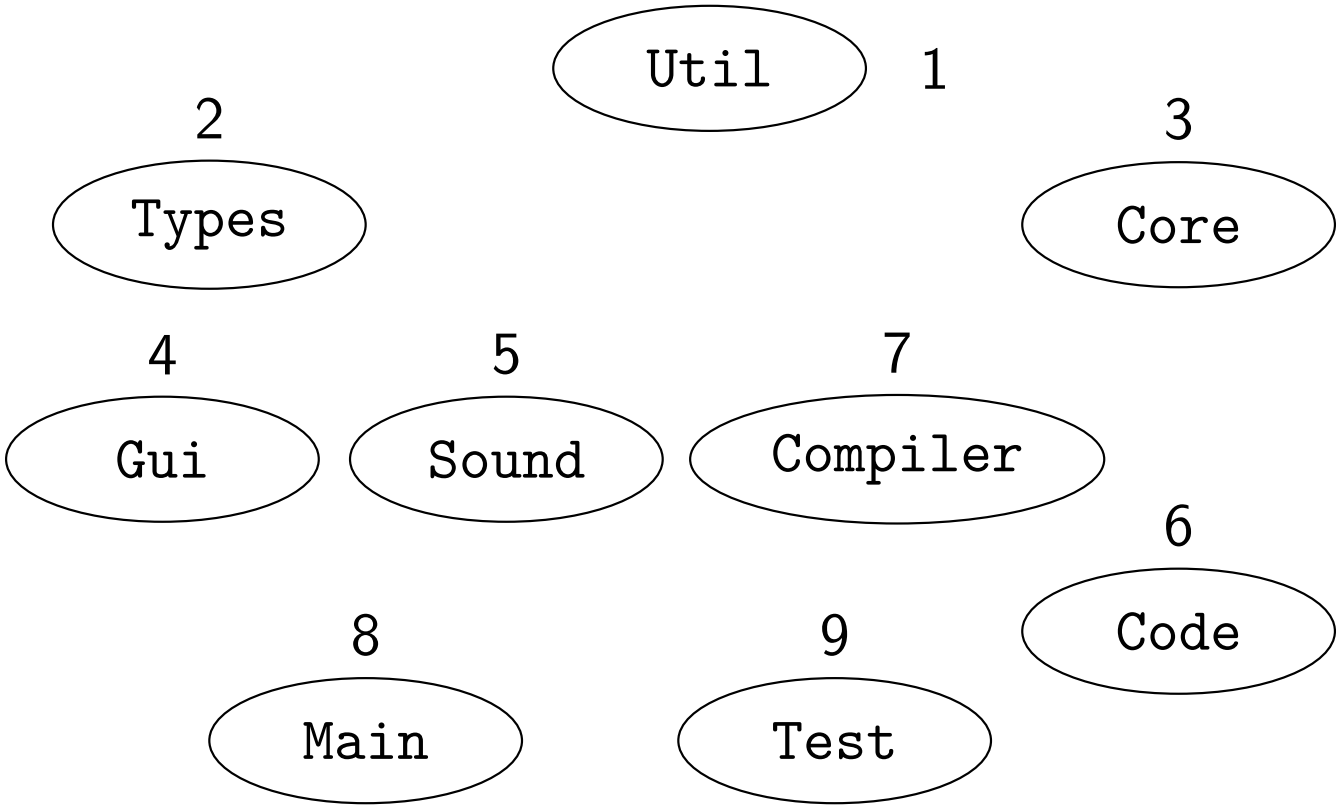
Topologisk sortering – Kahn's algoritm



Topologisk sortering – Kahn's algoritm



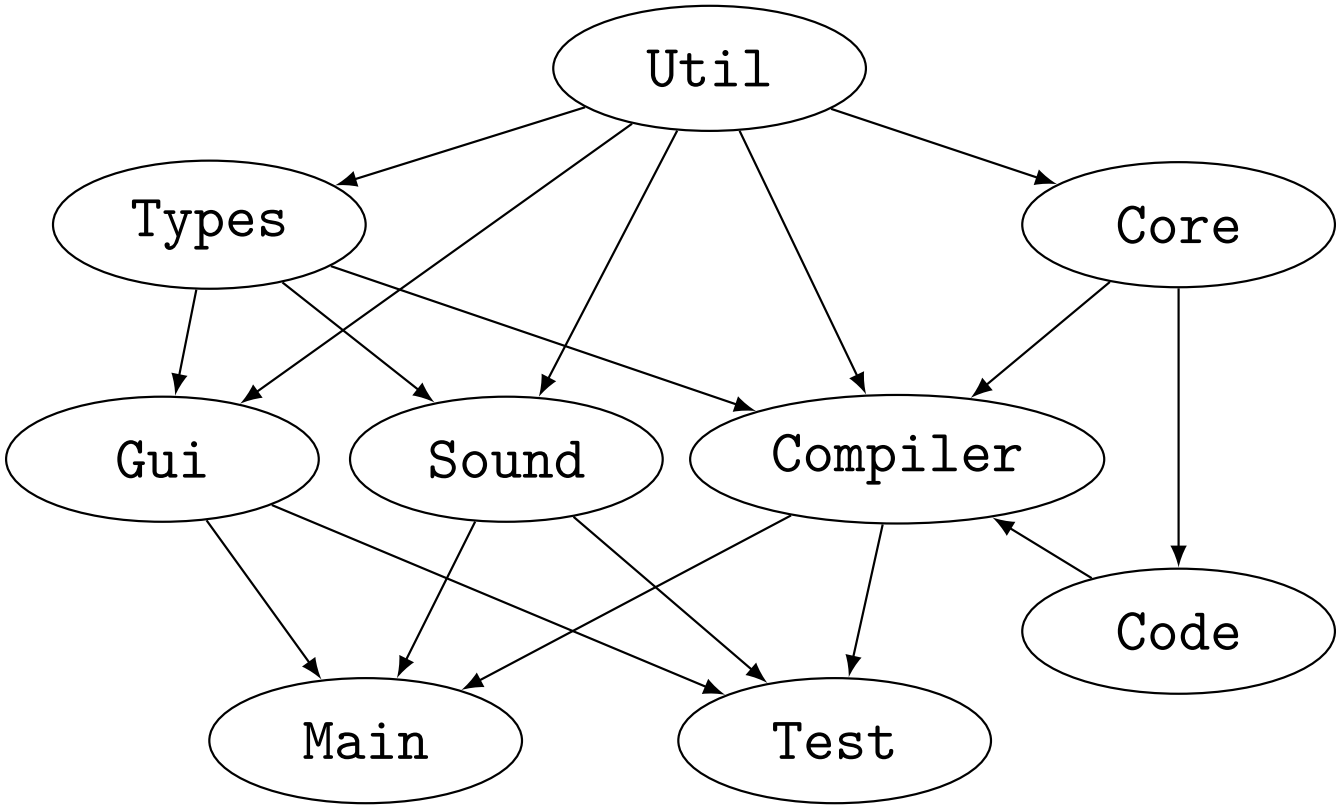
Topologisk sortering – Kahn's algoritm



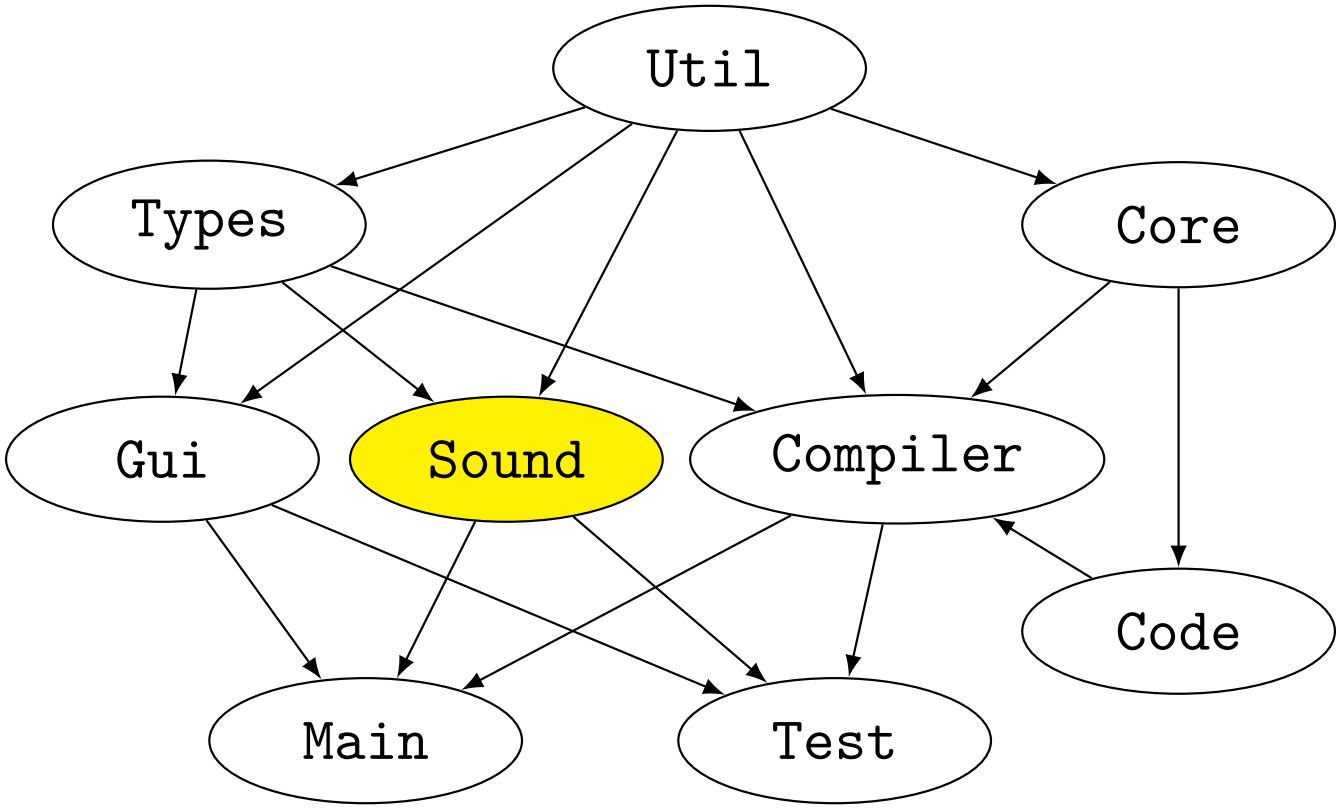
Topologisk sortering – DFS

```
// Anropas för alla noder i grafen, i någon ordning.
void visit(list<Node *> &out, Node *n) {
  if (n->permanent_mark) return;
  if (n->temporary_mark) throw "impossible";
  n->temporary_mark = true;
  for (Node *m : n->edges())
    visit(out, m);
  n->temporary_mark = false;
  n->permanent_mark = true;
  out.push_front(n);
}
```

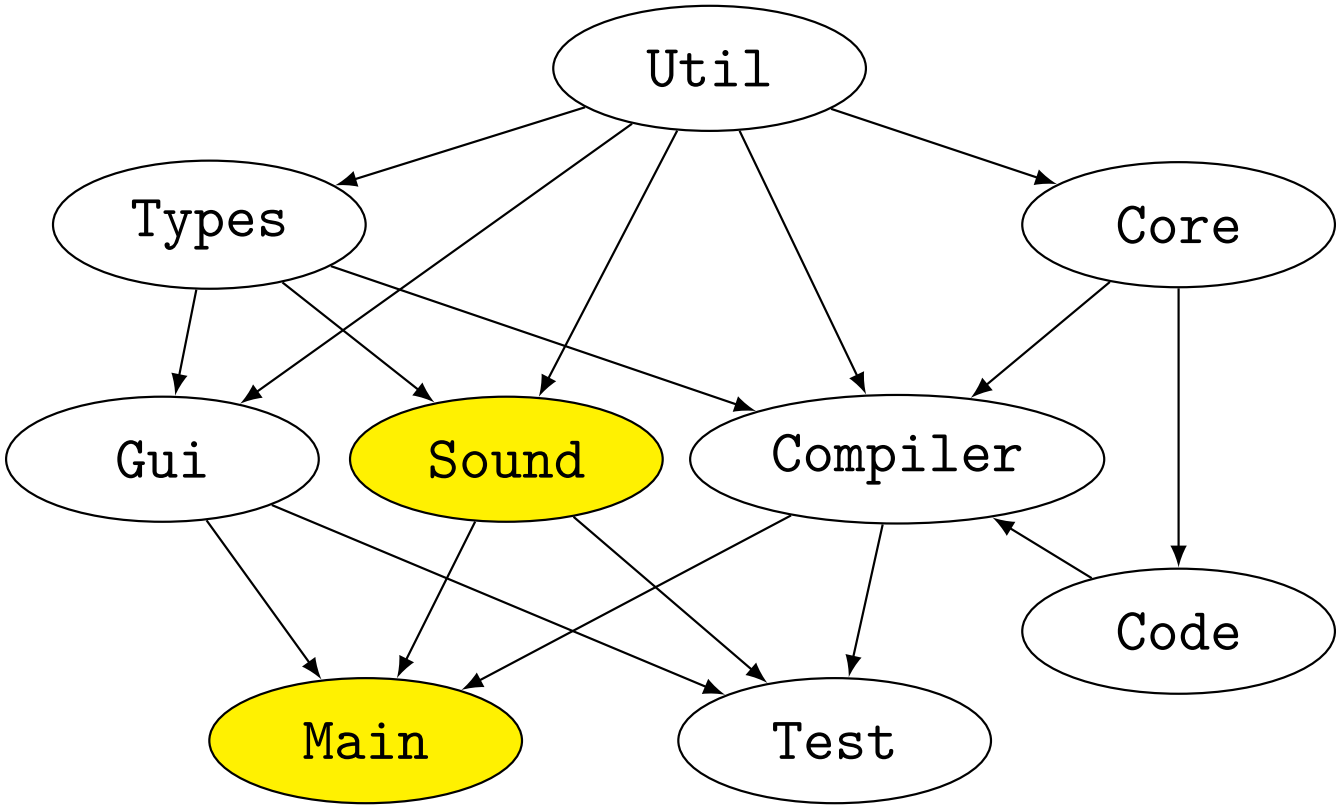
Topologisk sortering – DFS



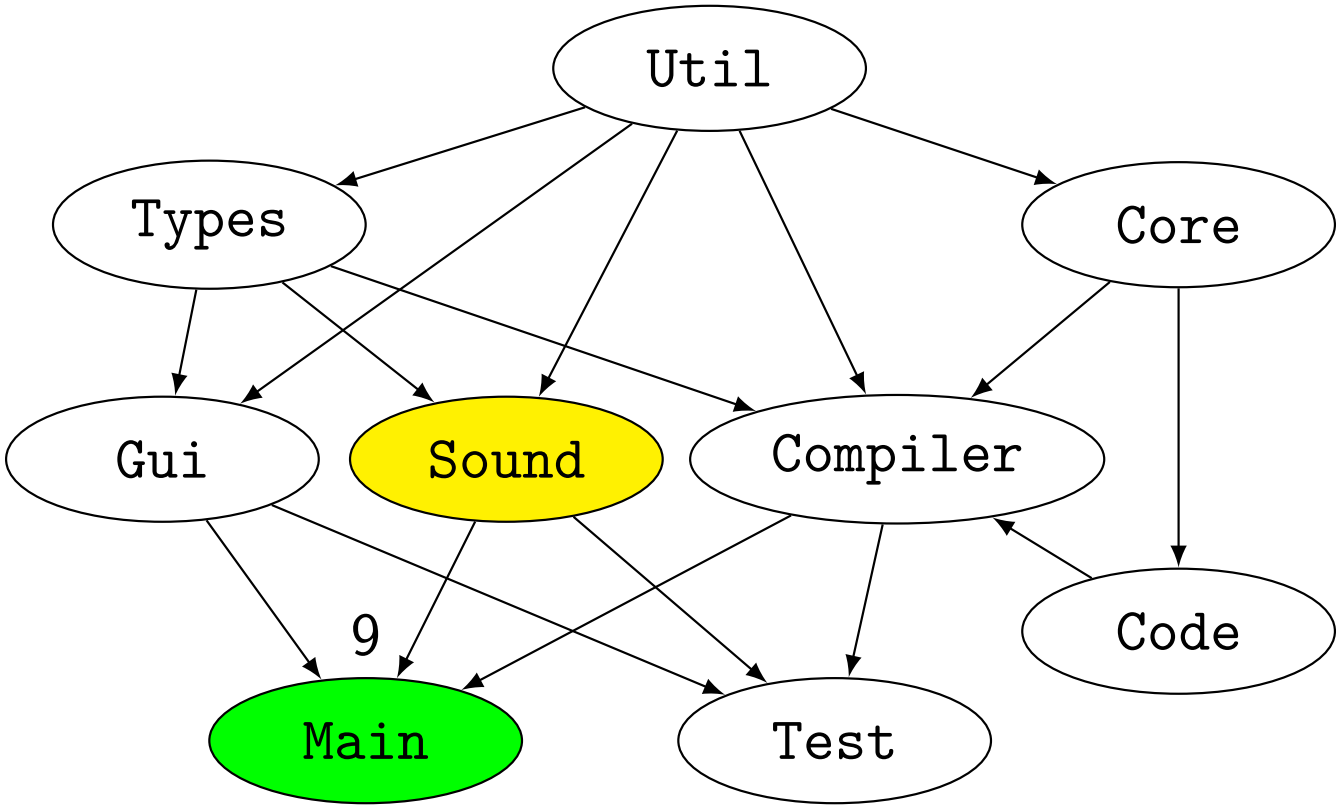
Topologisk sortering – DFS



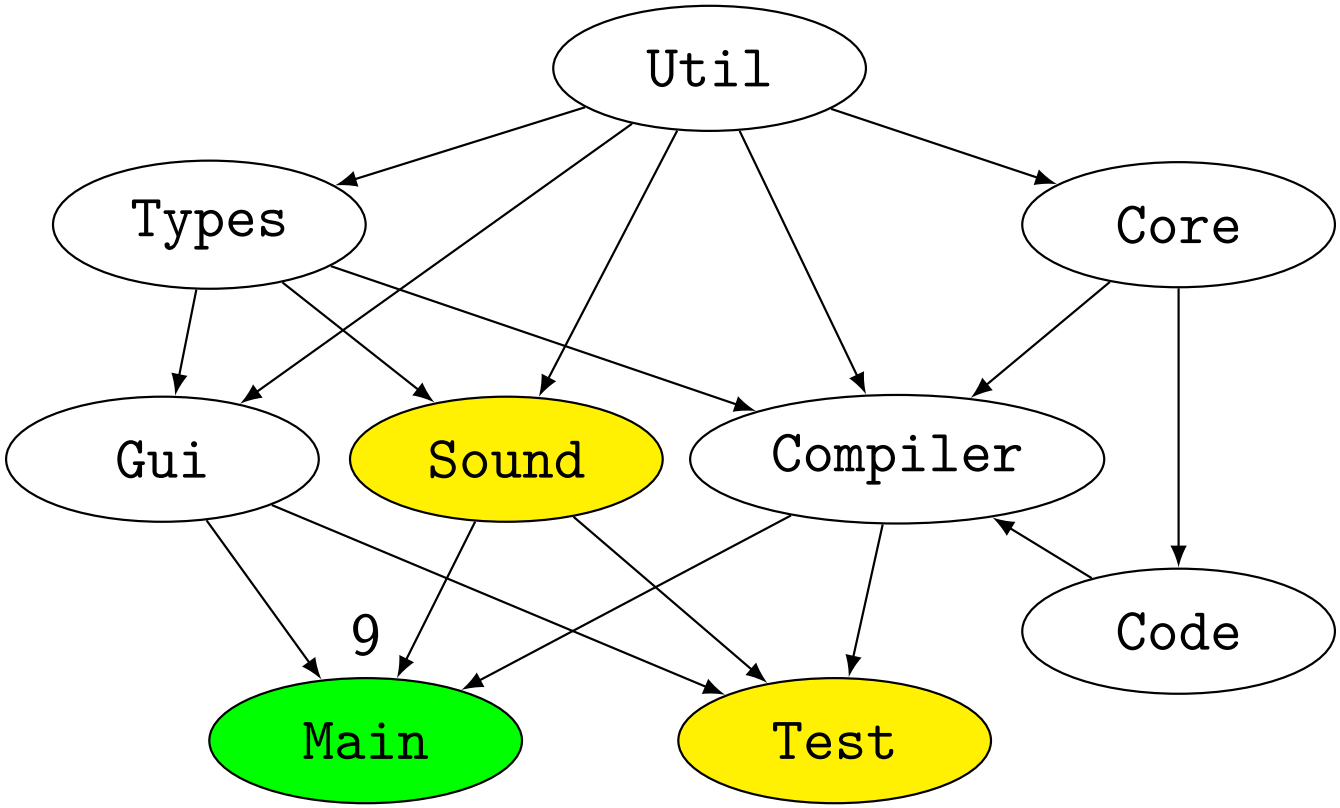
Topologisk sortering – DFS



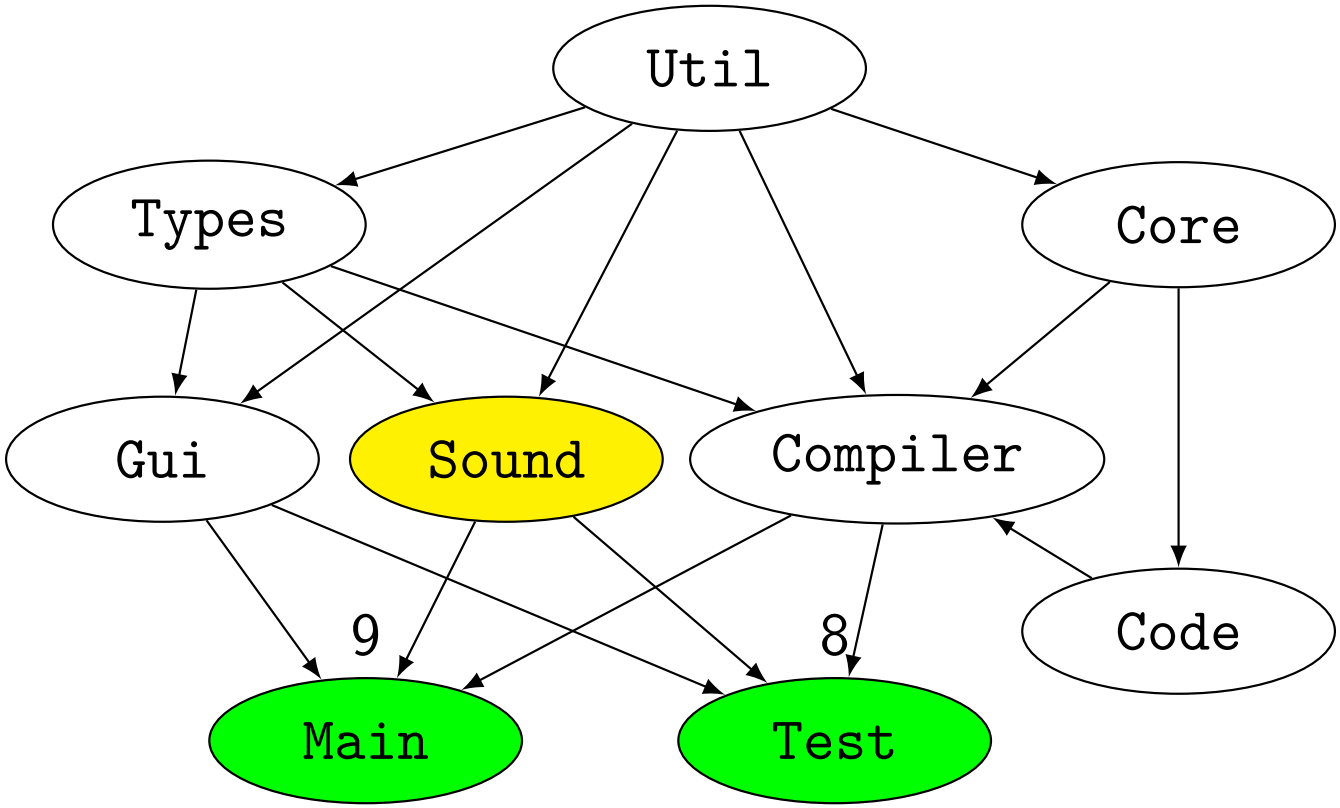
Topologisk sortering – DFS



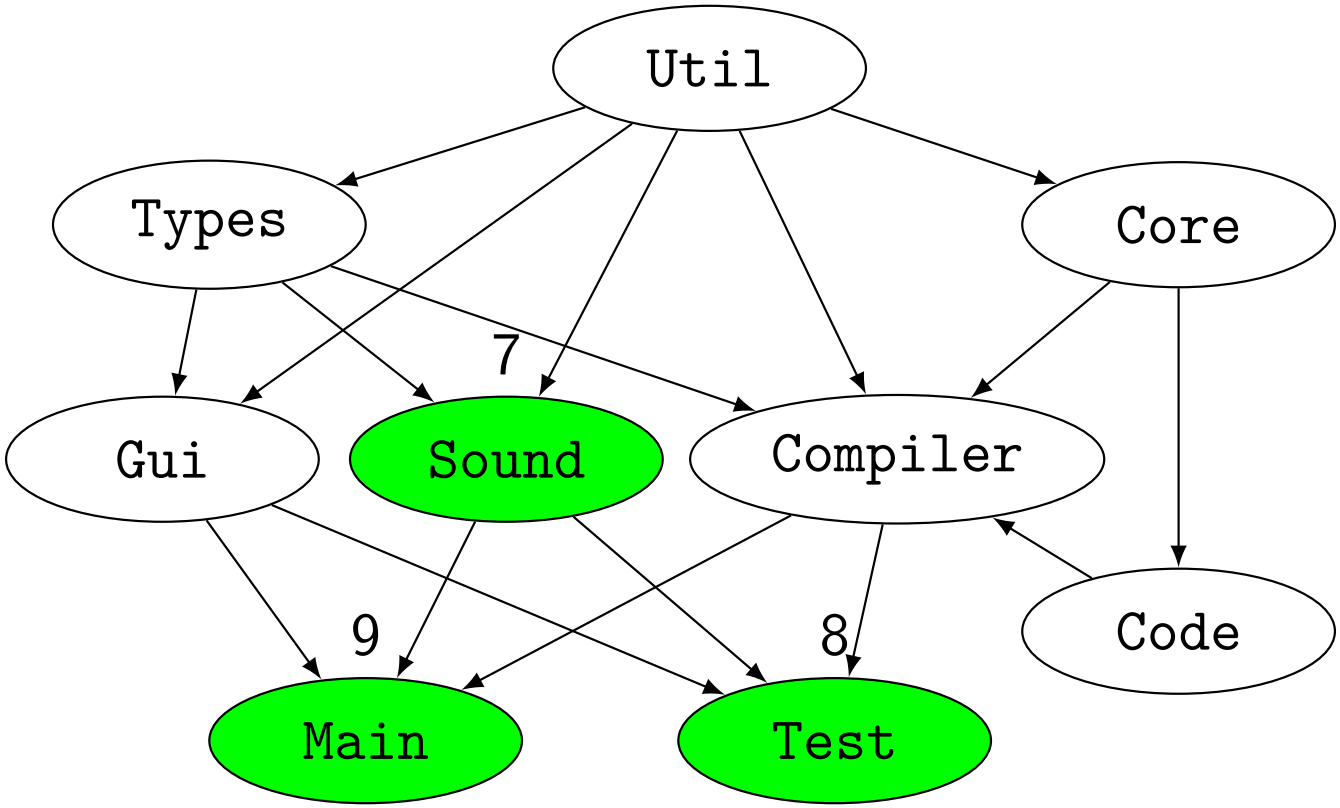
Topologisk sortering – DFS



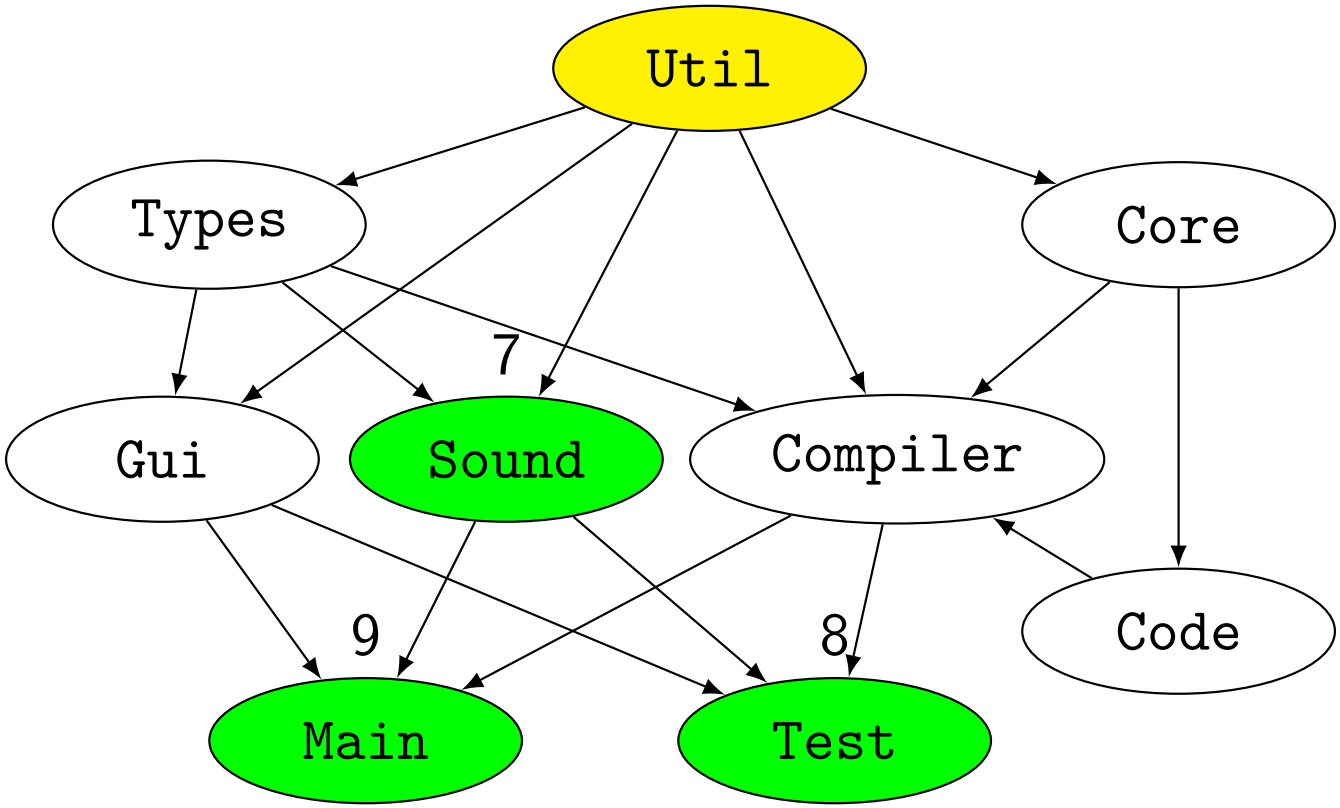
Topologisk sortering – DFS



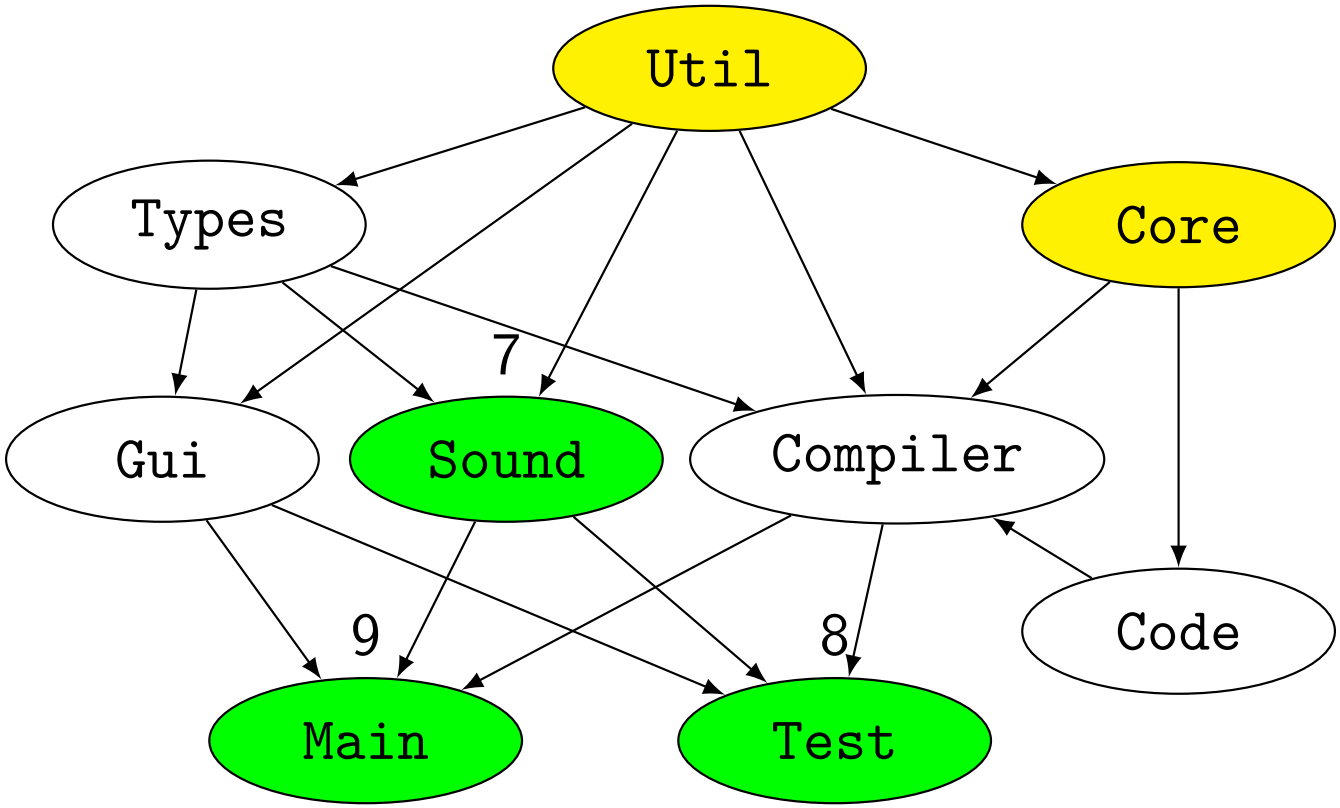
Topologisk sortering – DFS



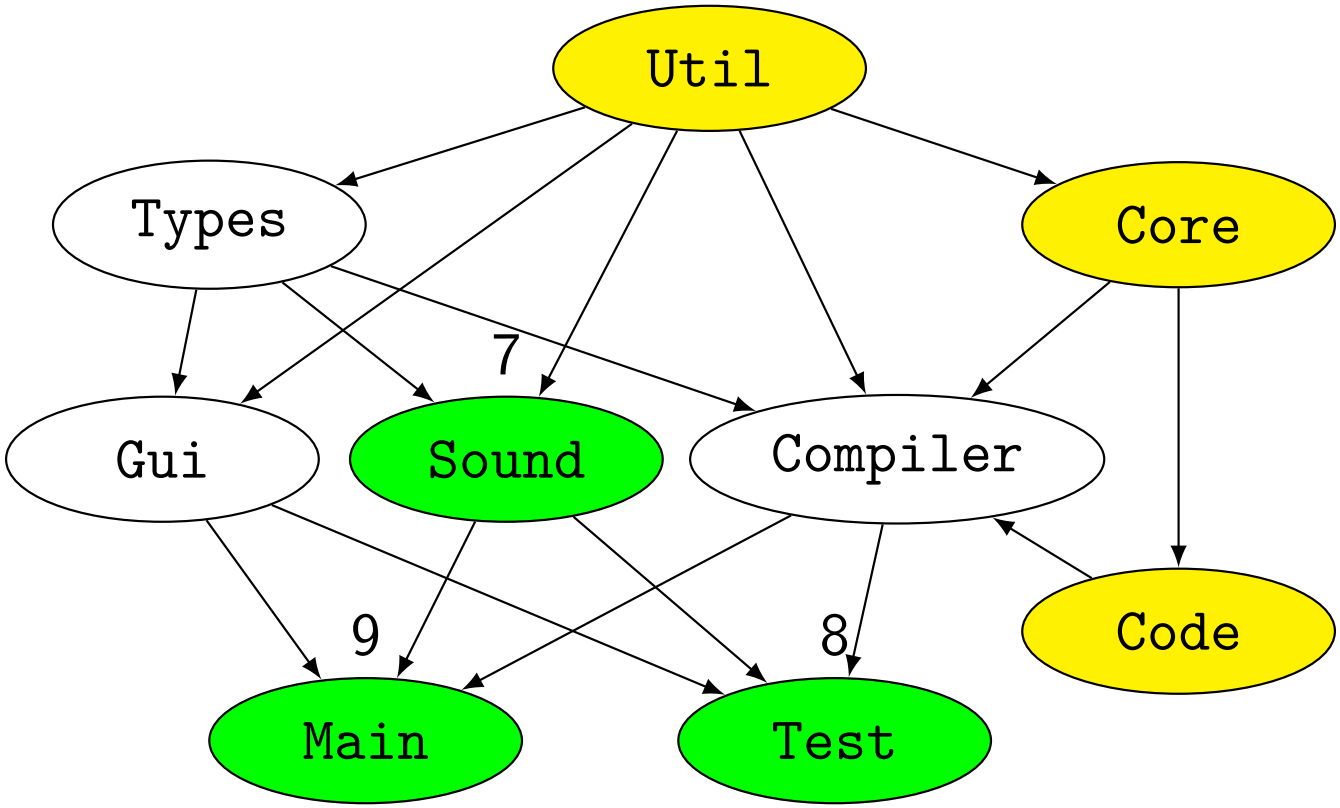
Topologisk sortering – DFS



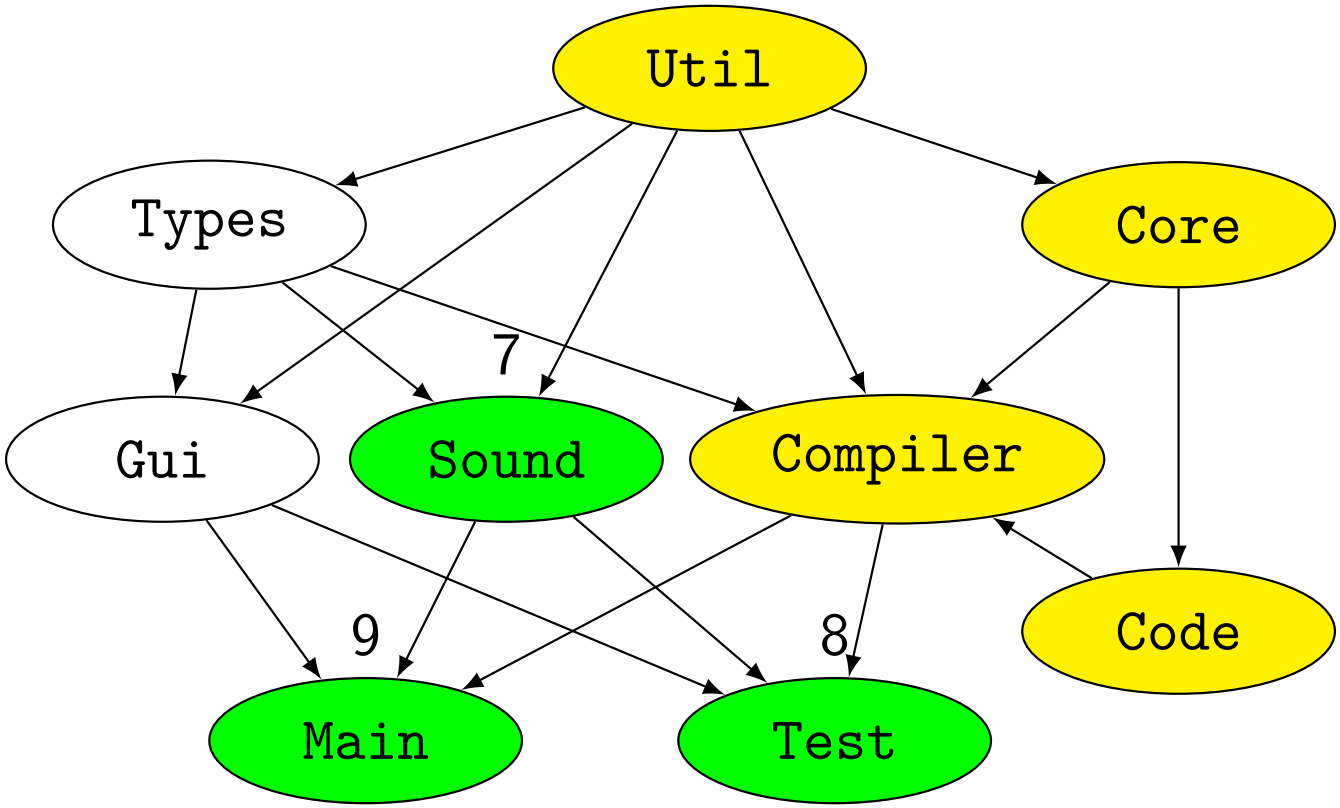
Topologisk sortering – DFS



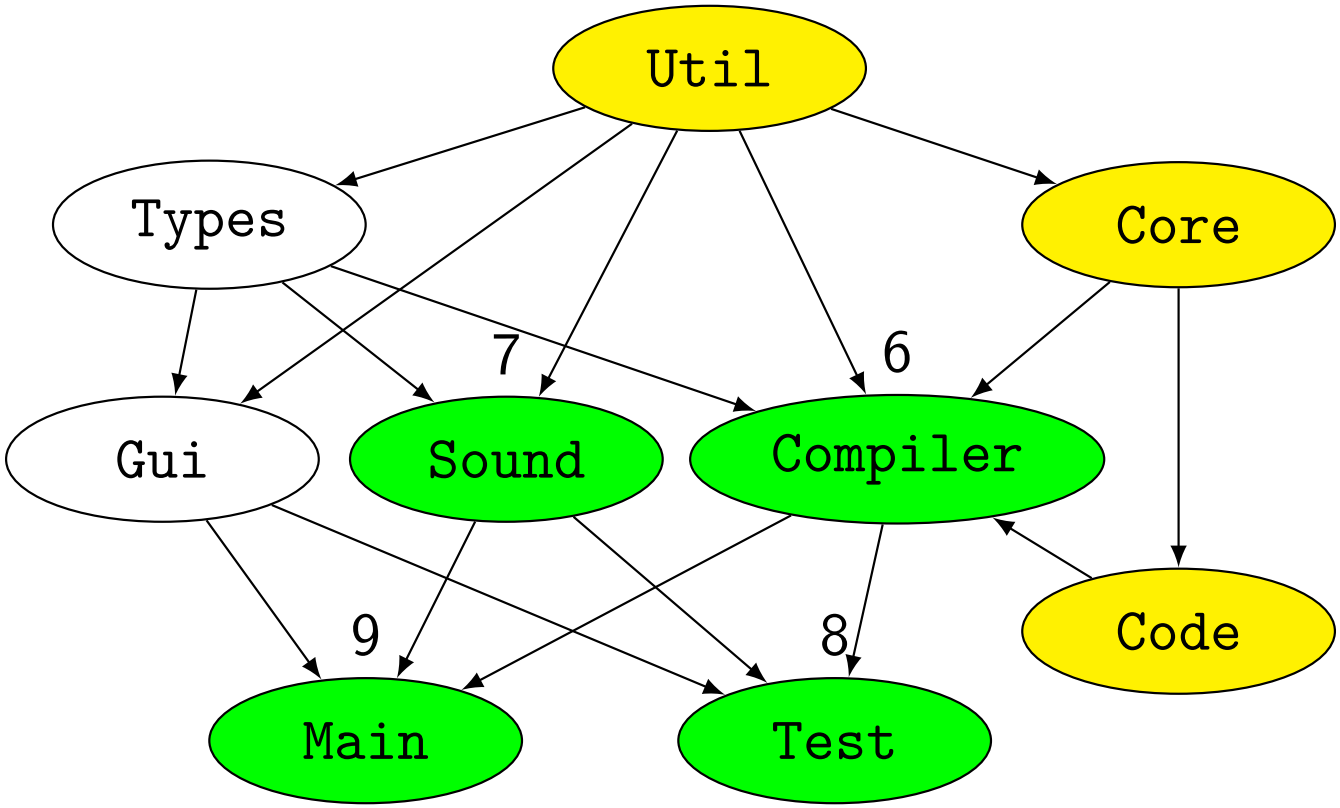
Topologisk sortering – DFS



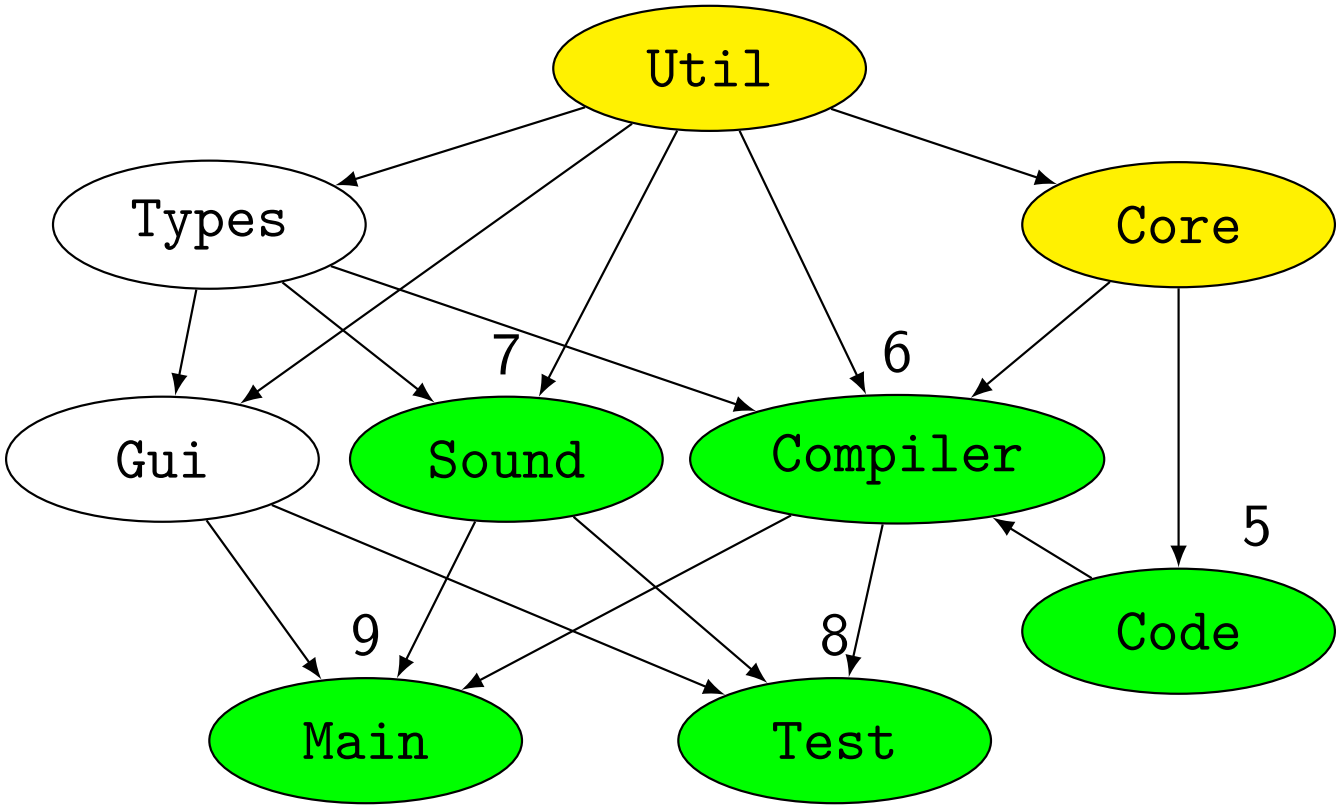
Topologisk sortering – DFS



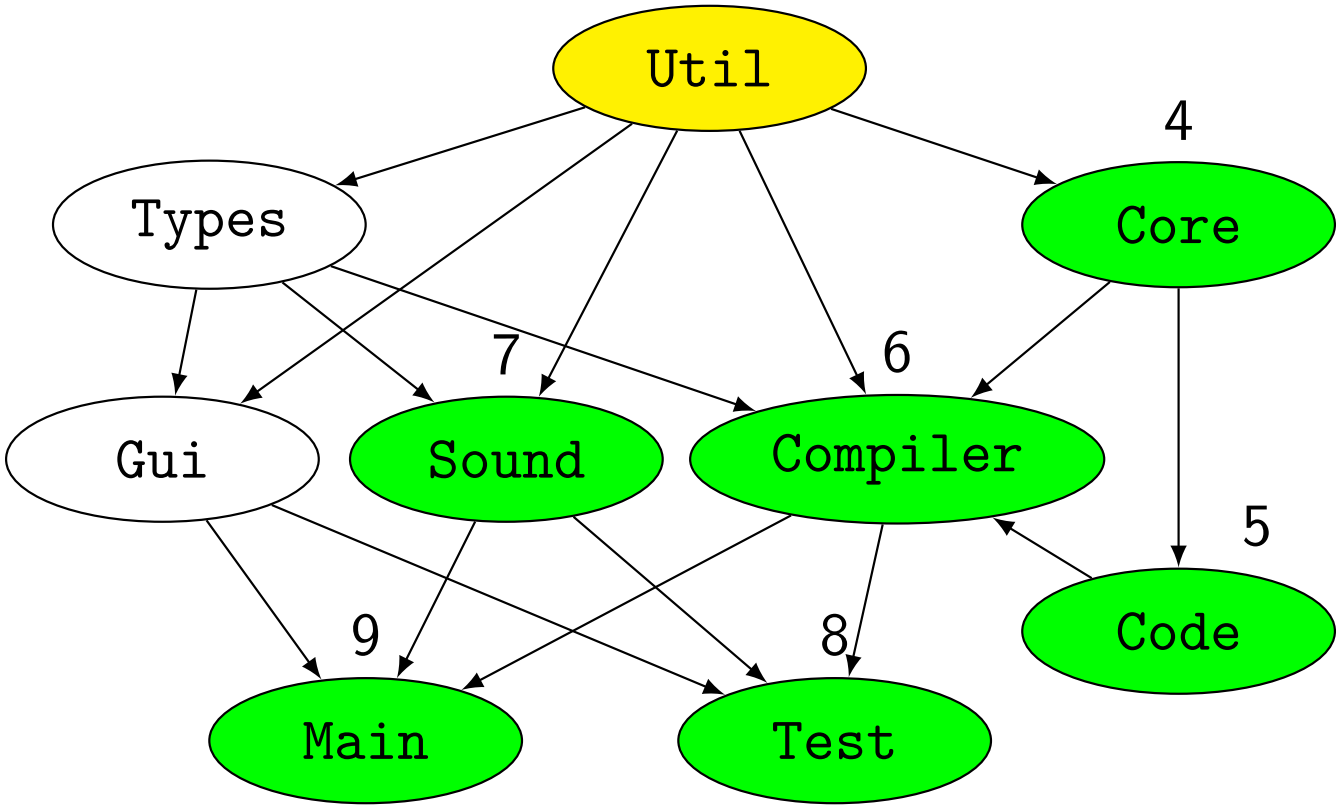
Topologisk sortering – DFS



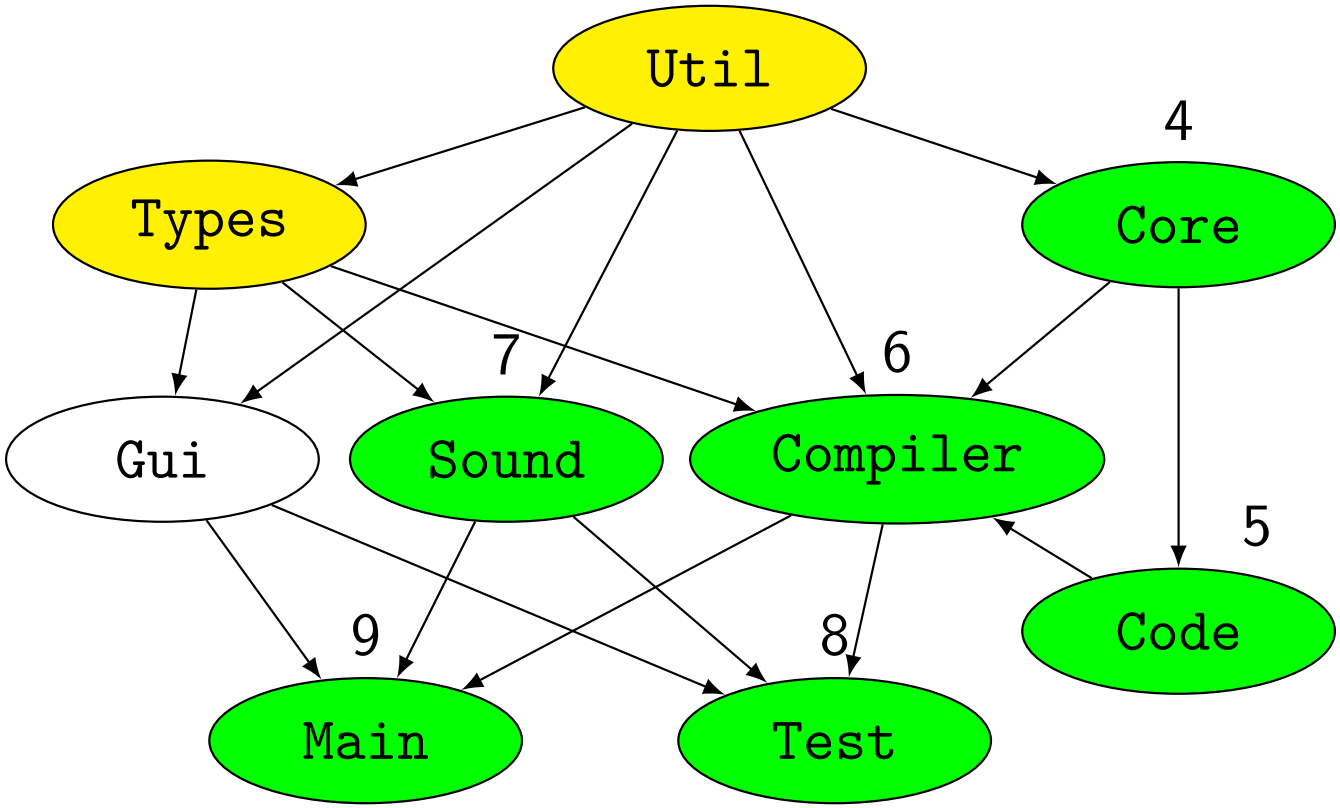
Topologisk sortering – DFS



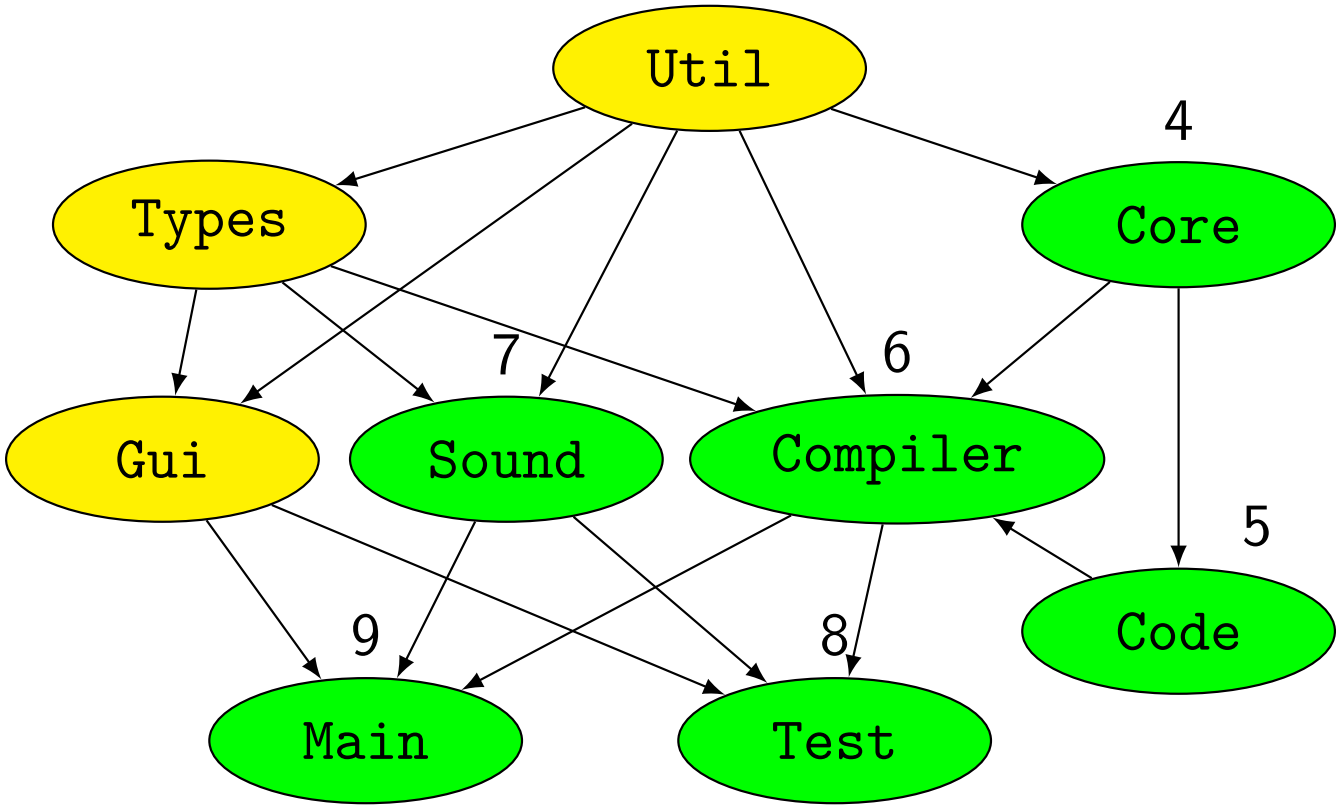
Topologisk sortering – DFS



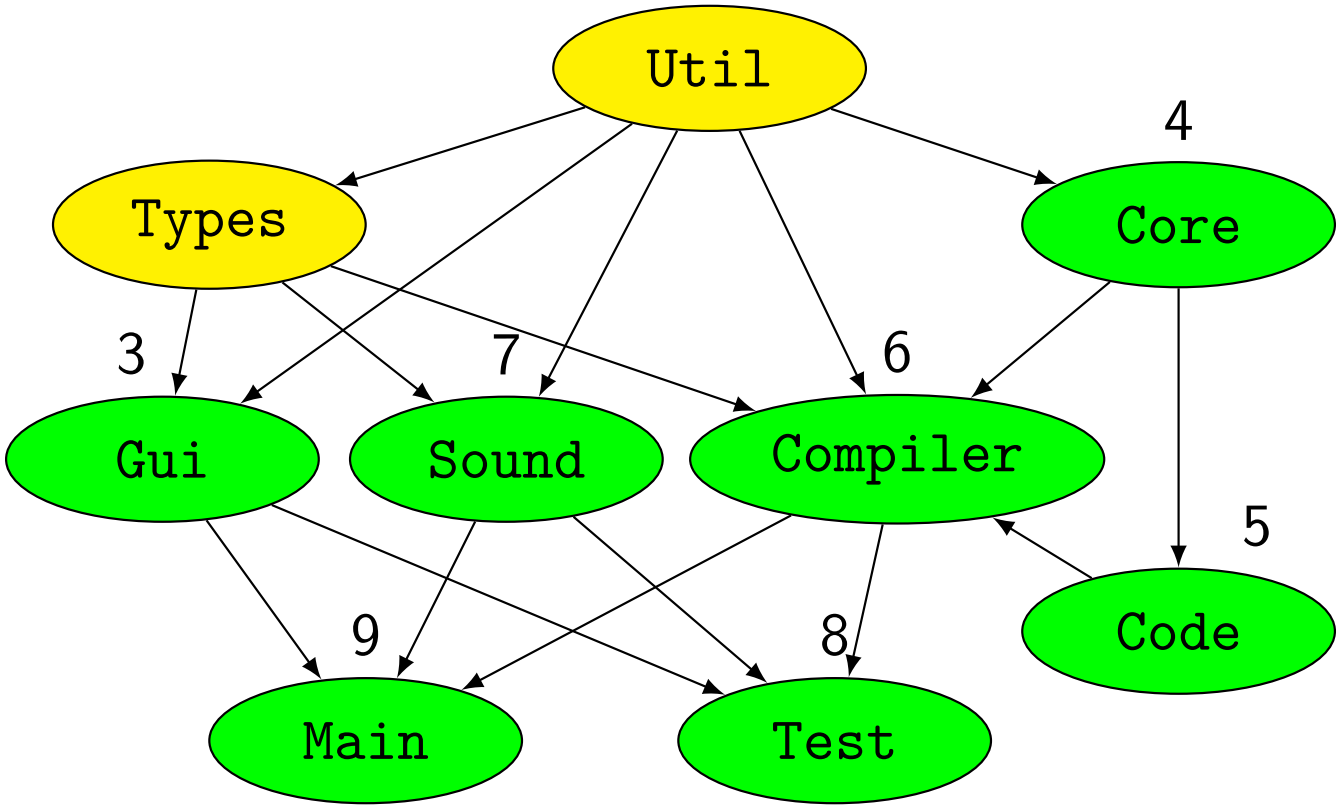
Topologisk sortering – DFS



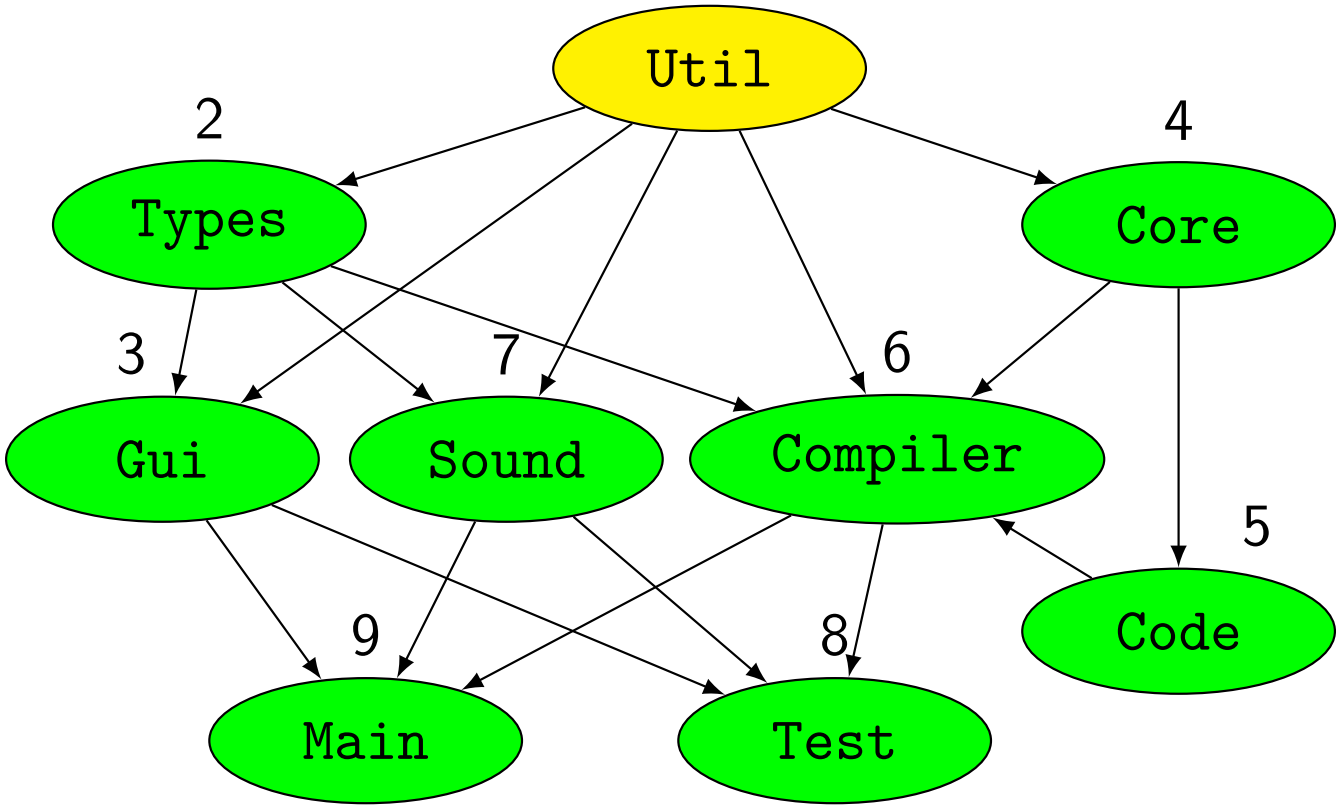
Topologisk sortering – DFS



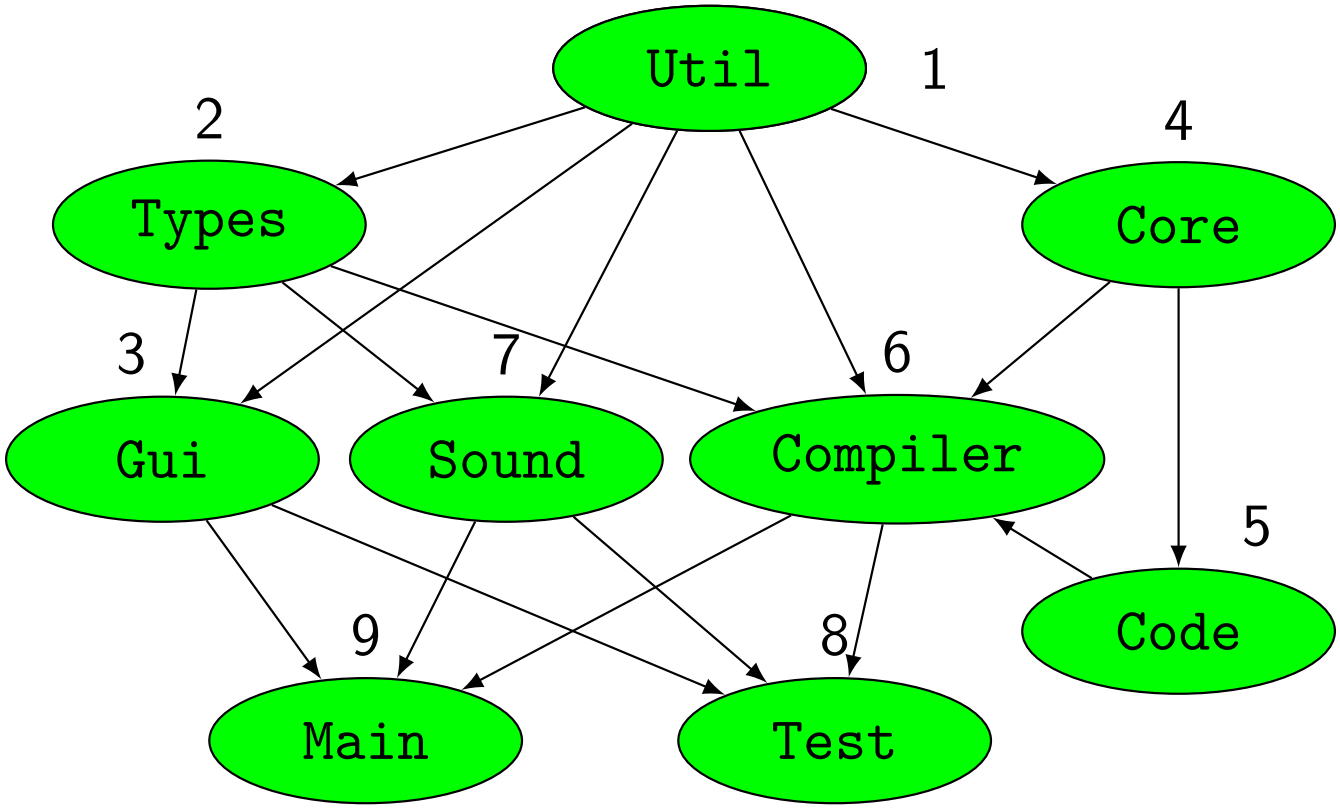
Topologisk sortering – DFS



Topologisk sortering – DFS



Topologisk sortering – DFS

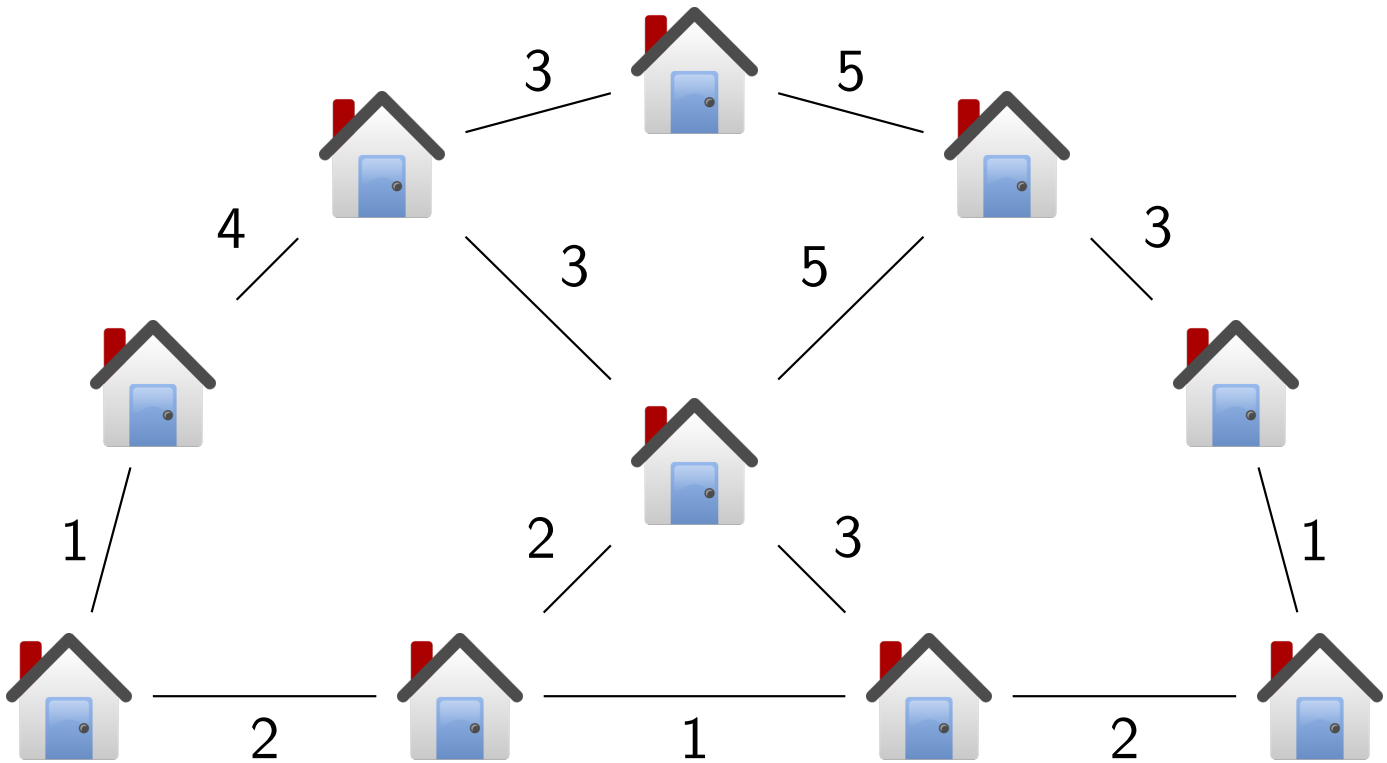


LINKÖPING UNIVERSITY

Problem

Invånarna i en liten by har bestämt sig för att dra in fiber i alla hus i staden. Du som leder projektet har fått ansvaret för att planera hur fibern ska dras i staden. Du har kommit fram till att det är enklast att dra fibern längs vägar. För att hålla kostnaderna nere, och invånarna glada, vill du gräva upp så lite väg som möjligt. Till din hjälp har du en karta över byn och din trogna dator.

Vilka vägar ska vi använda?



lakttagelser:

Det här är ju en graf!

- Vill välja bågar så att summan blir så låg som möjligt
- Vill välja tillräckligt många för att koppla ihop alla hus

$\Rightarrow$ Vi vill inte ha några **cykler** i grafen

$\Rightarrow$ Vi vill hitta ett träd, det **minsta uppspännande trädet**

Minsta uppspännande träd

Det finns två välkända algoritmer:

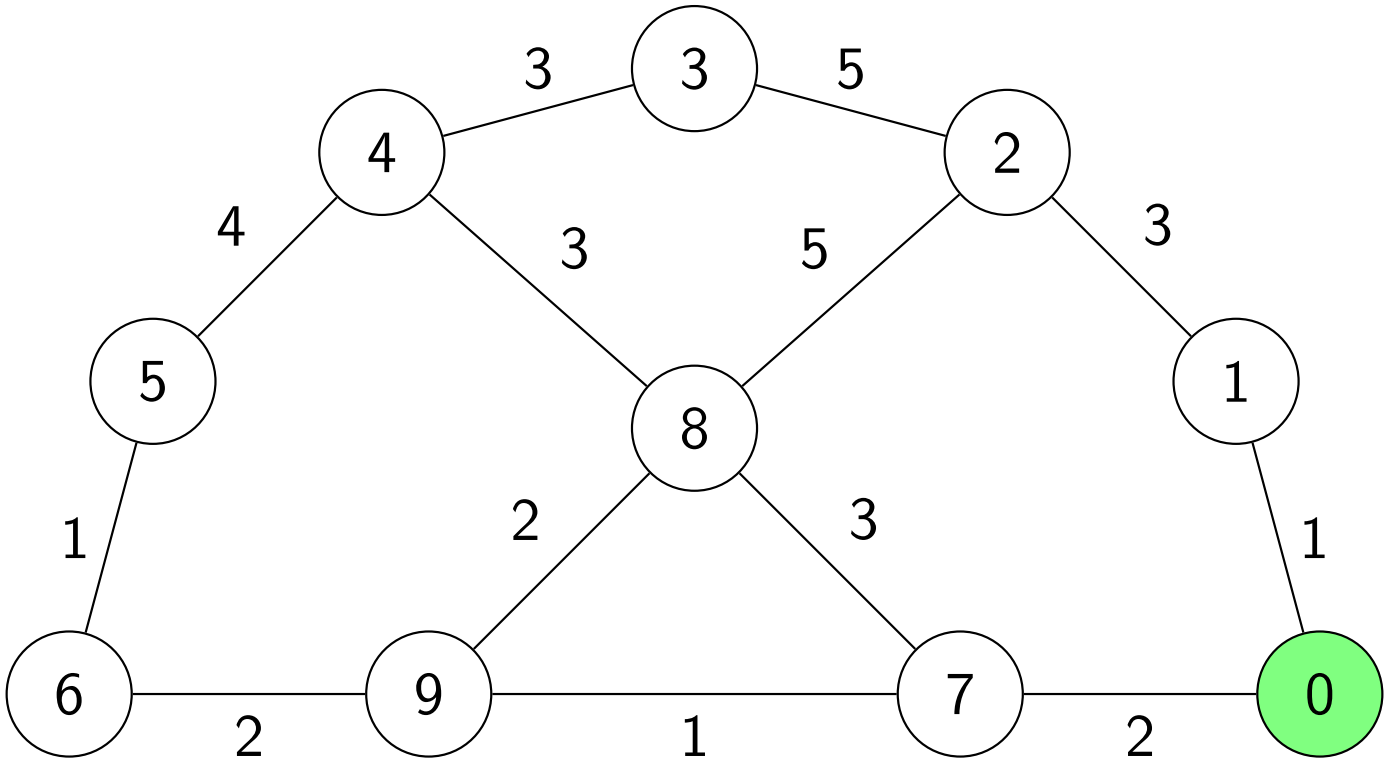
Prims algoritm: $\mathcal{O}(|E|\log(|V|))$

Börja med en nod, utöka sedan trädet successivt genom att lägga till den billigaste bågen som går till en ny nod.

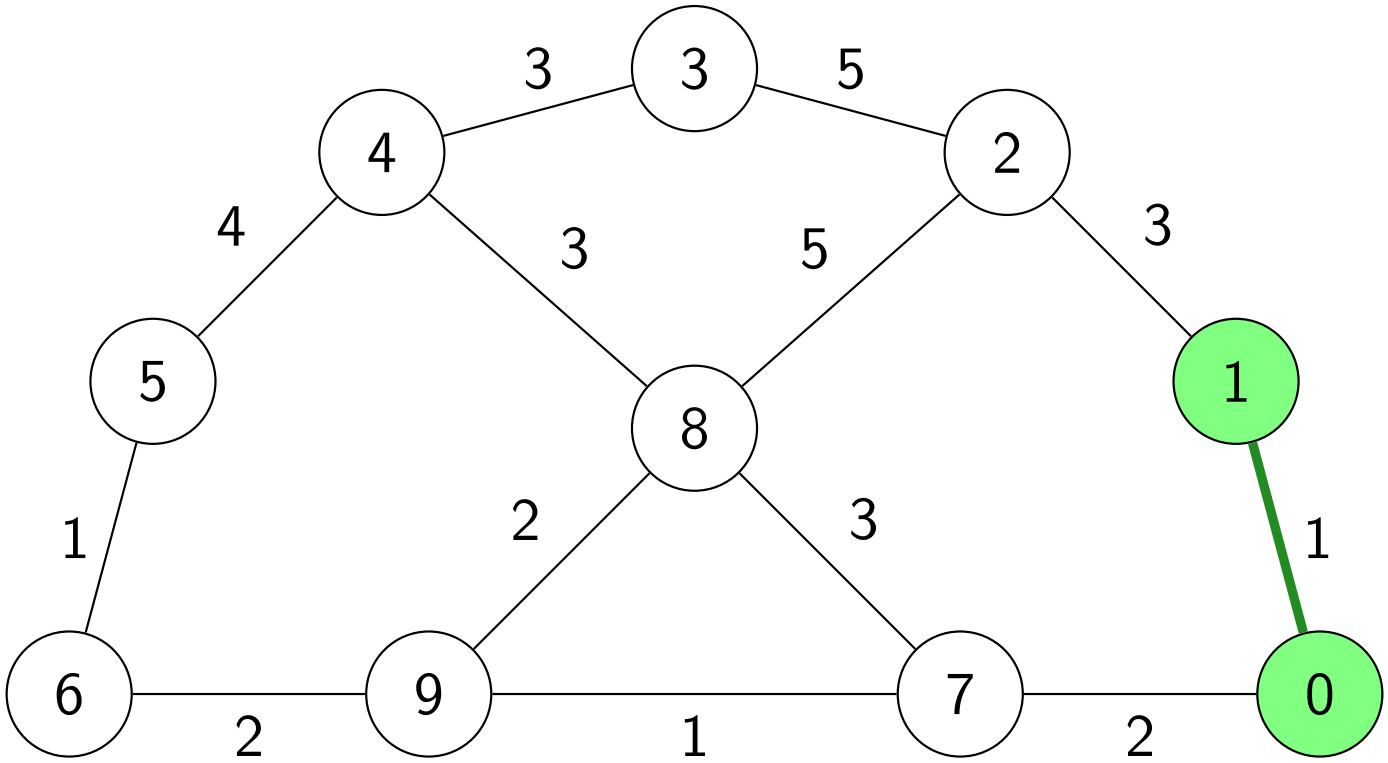
Kruskals algoritm: $\mathcal{O}(|E|\log(|E|)) = \mathcal{O}(|E|\log(|V|))$

Börja med ett tomt träd, utöka sedan trädet med den minsta bågen som inte skapar en cykel tills vi fått ett träd.

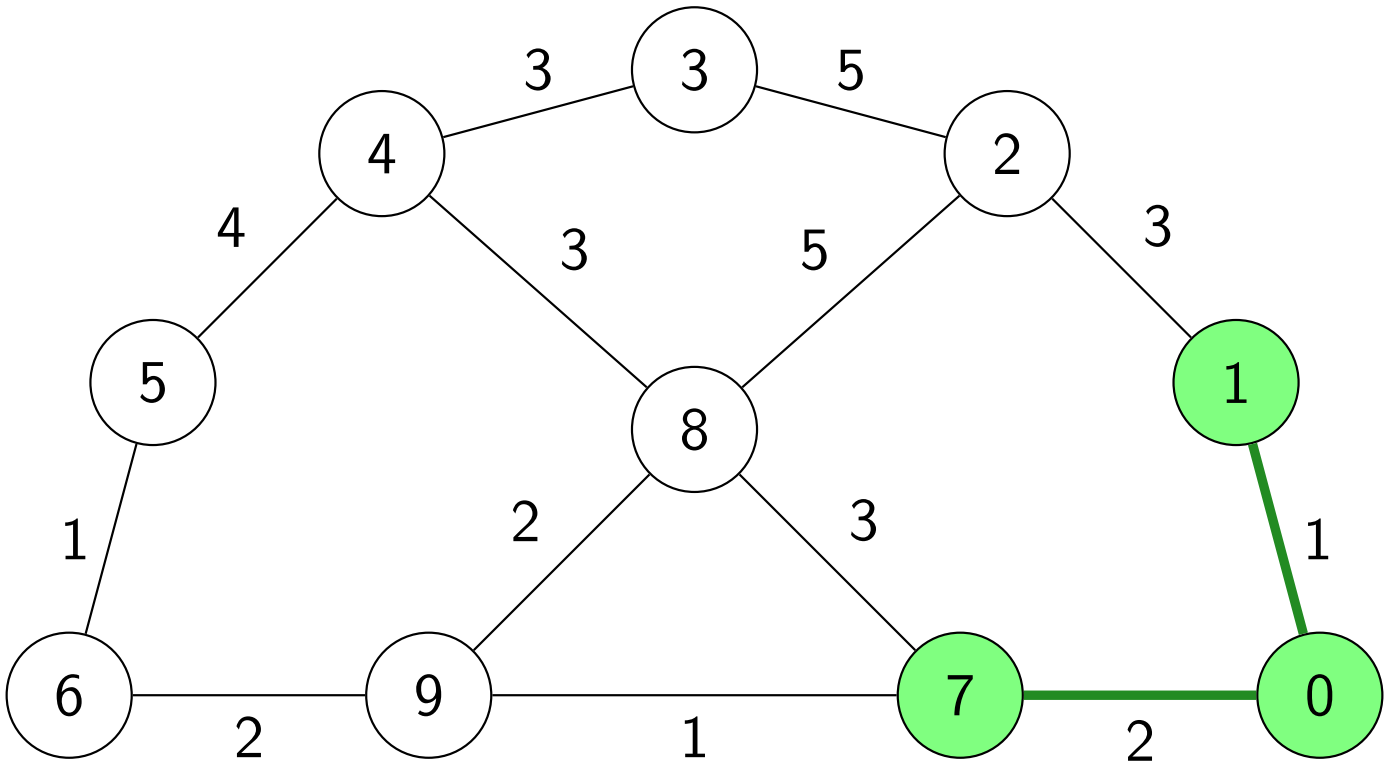
Prims algoritm



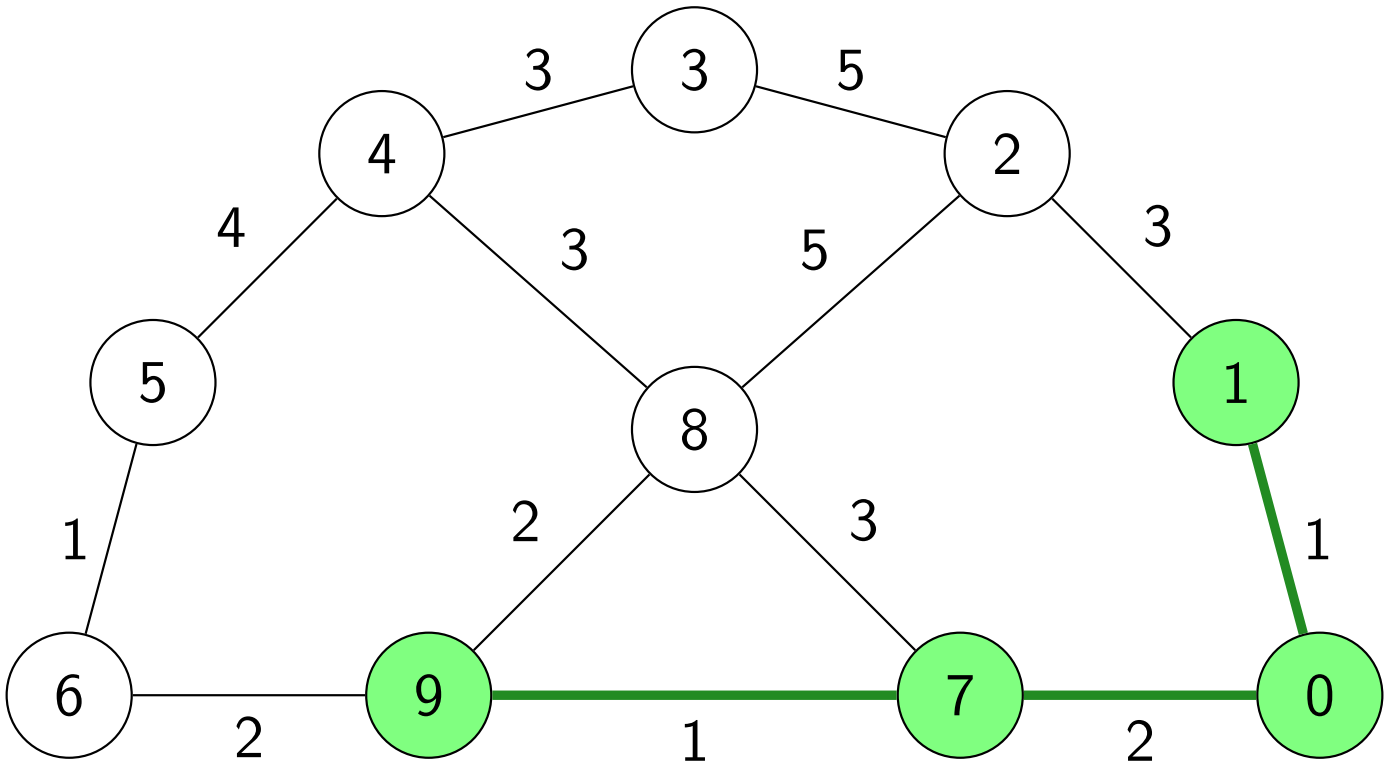
Prims algoritm



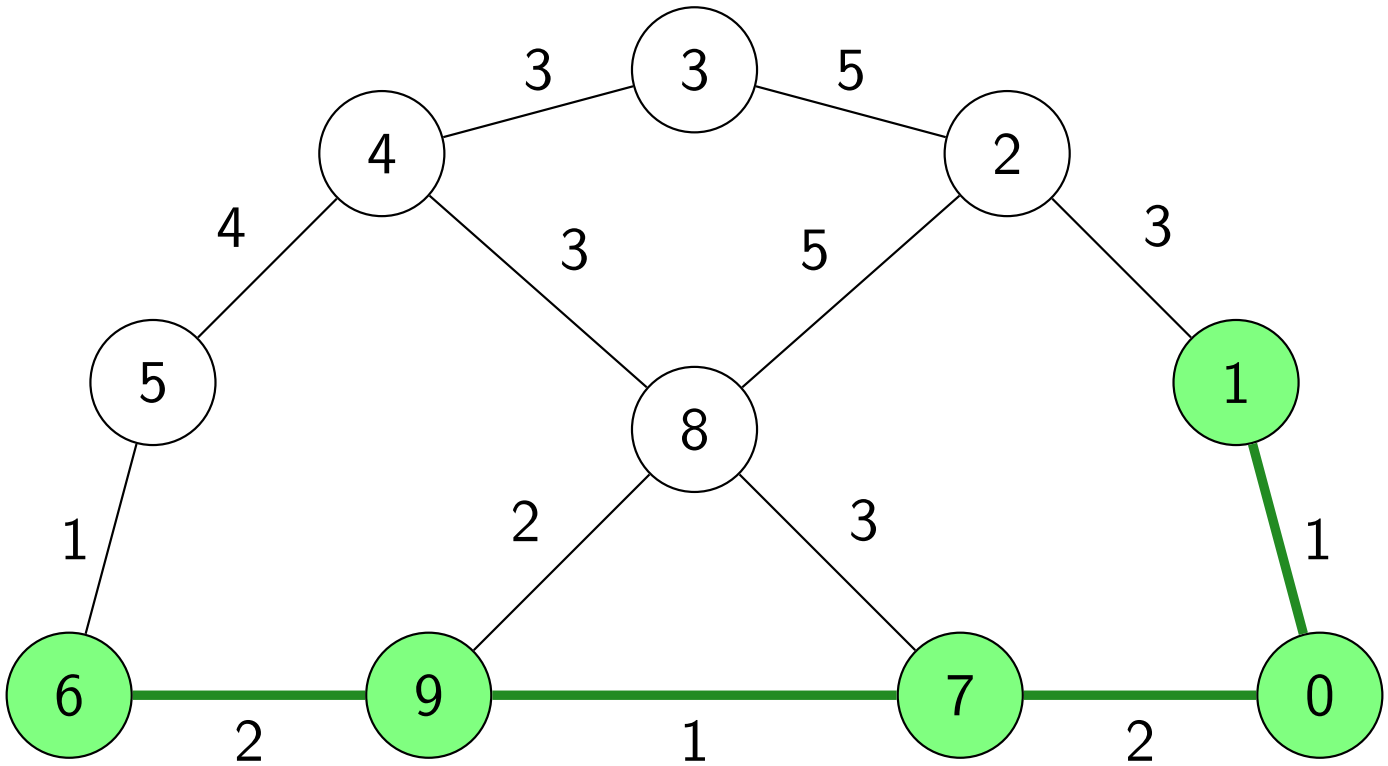
Prims algoritm



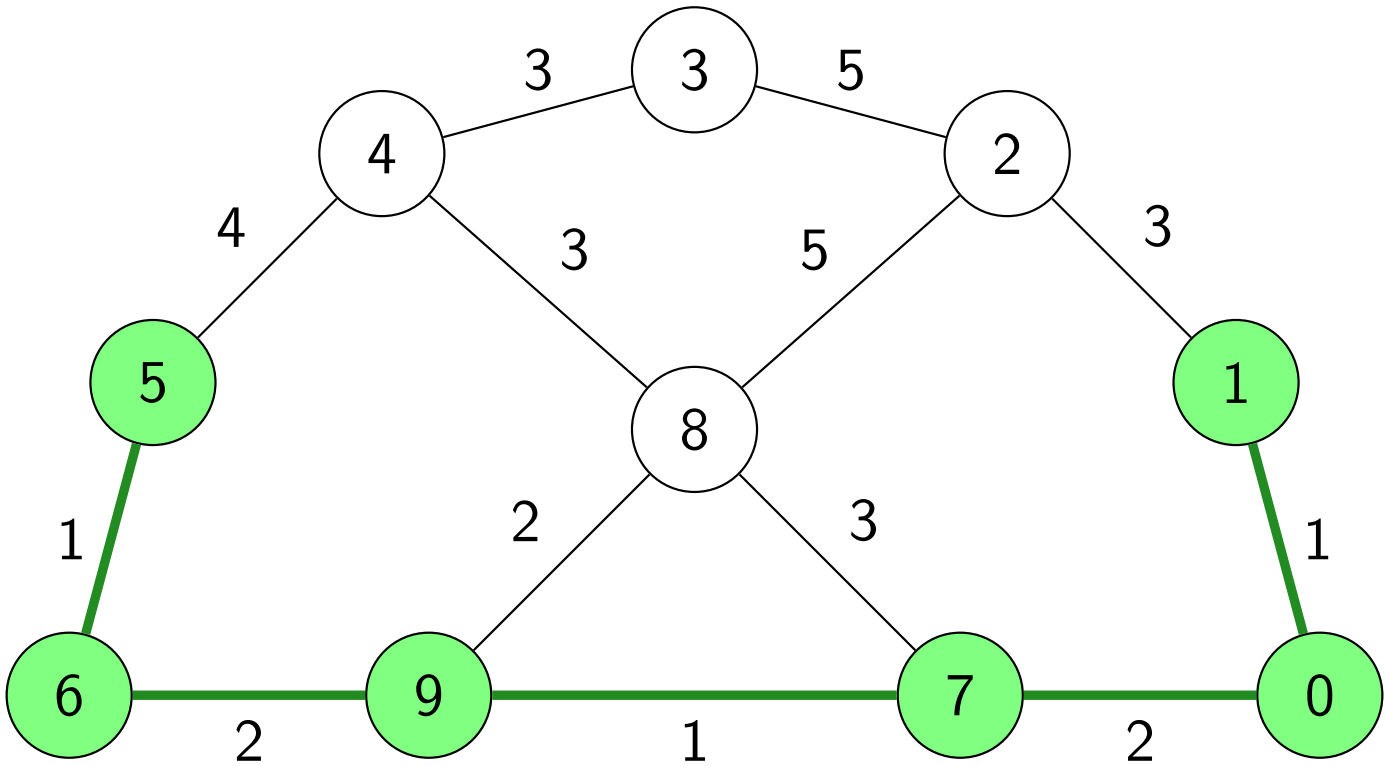
Prims algoritm



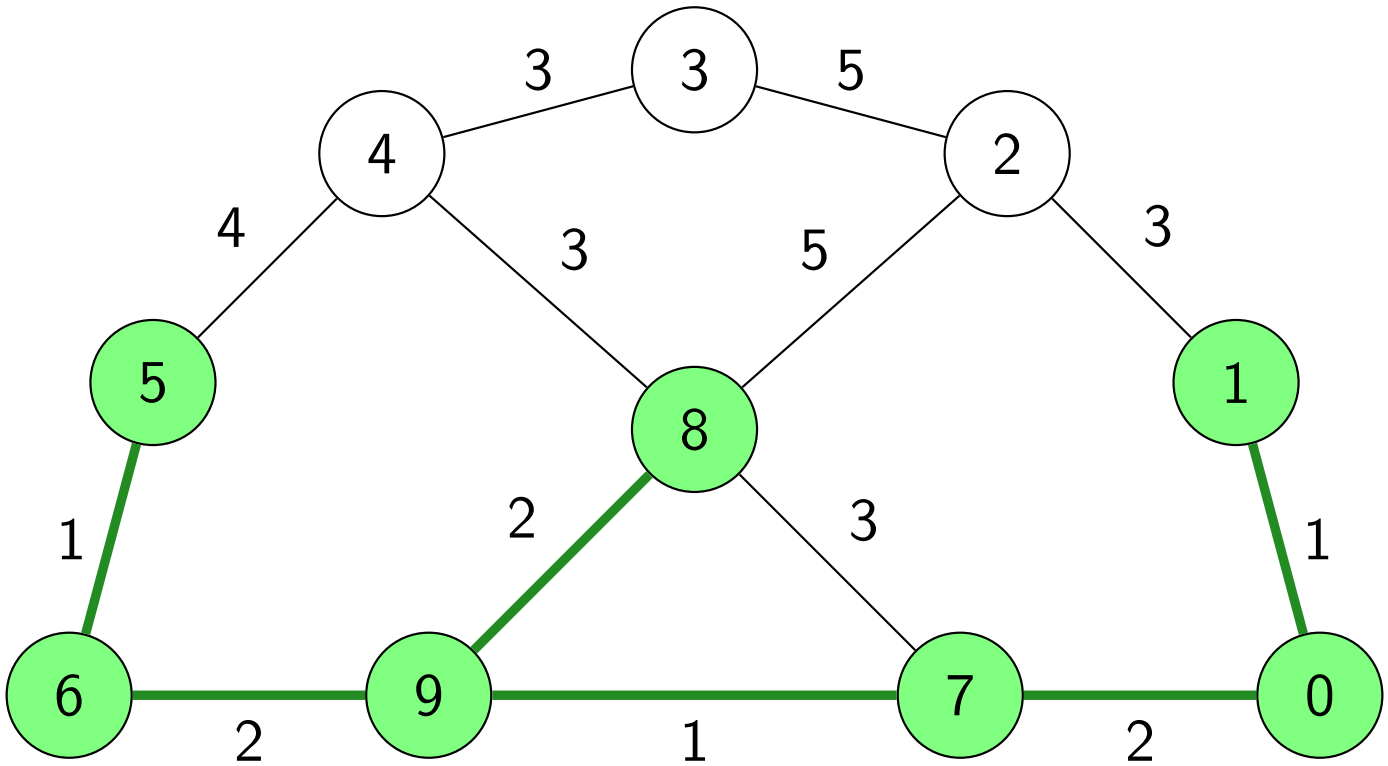
Prims algoritm



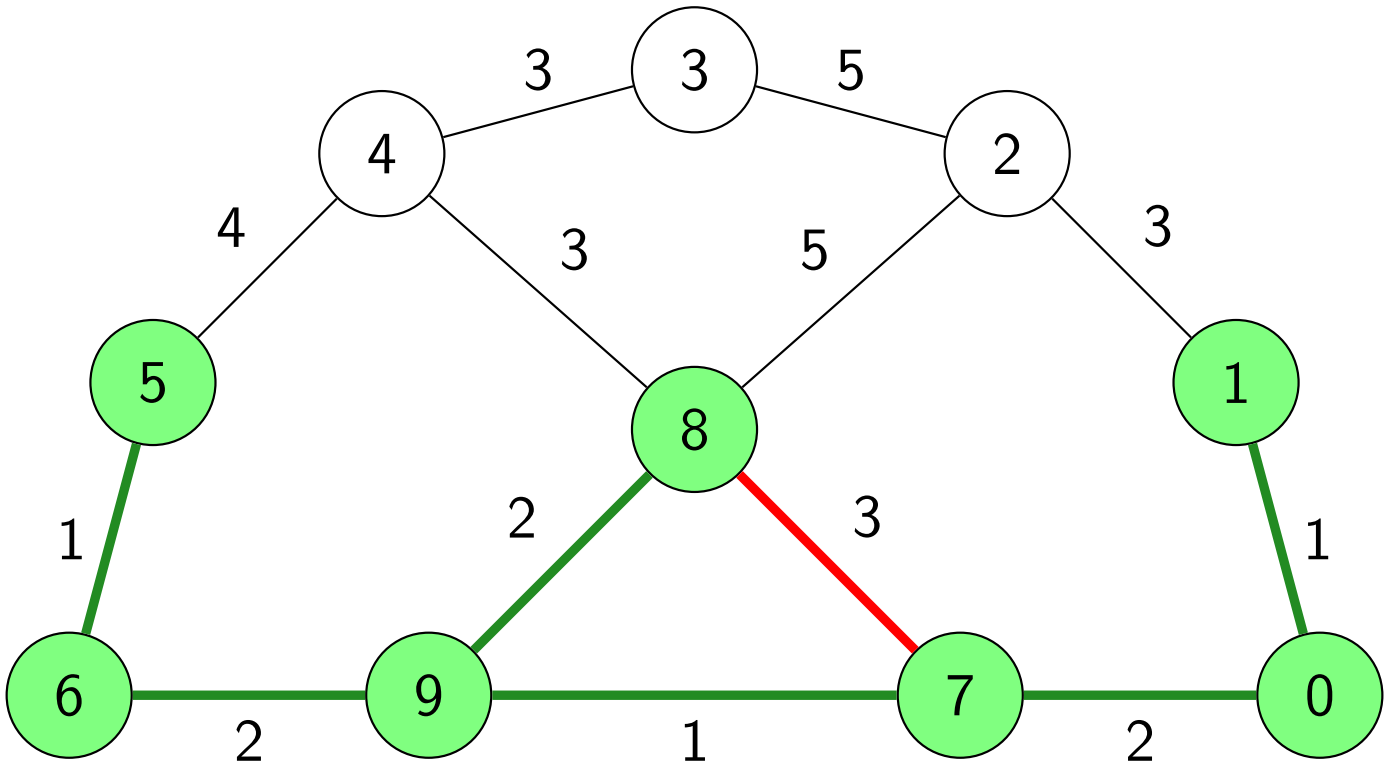
Prims algoritm



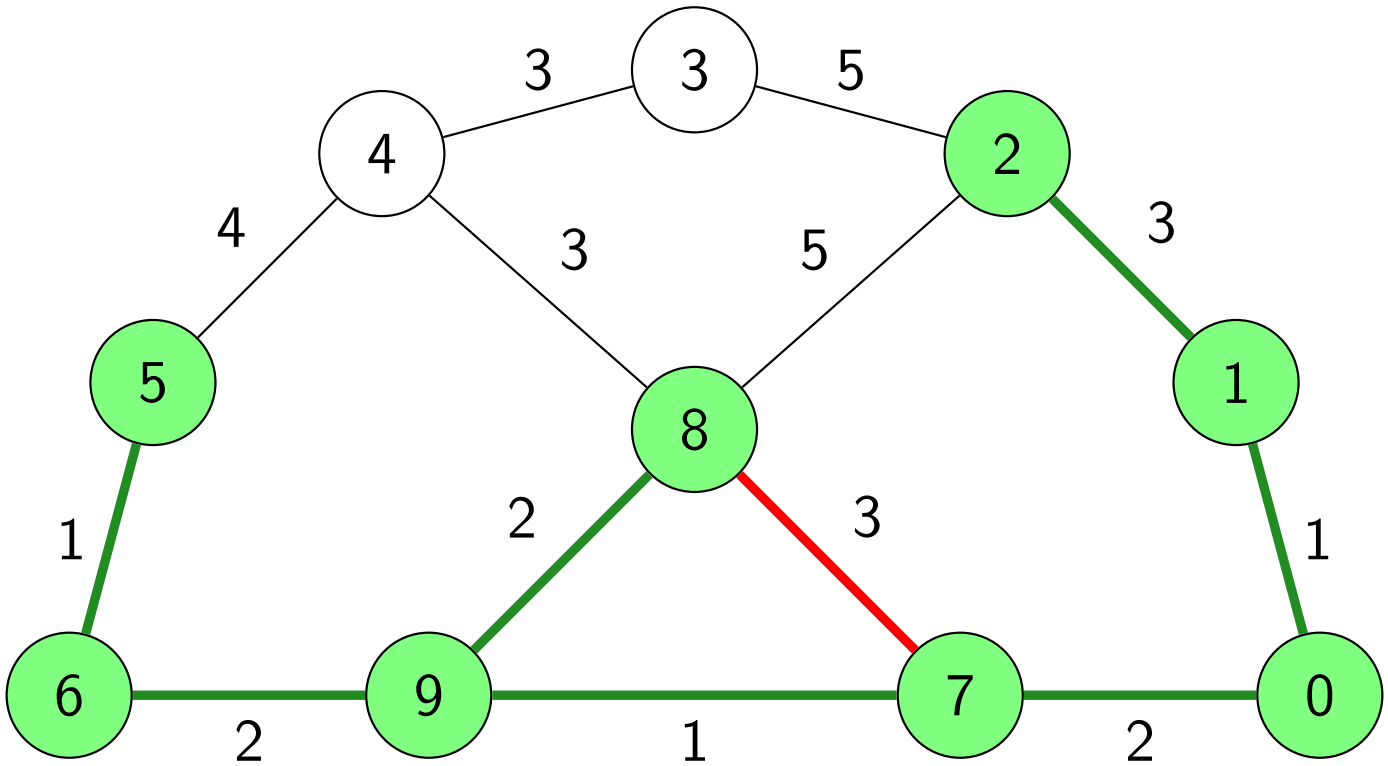
Prims algoritm



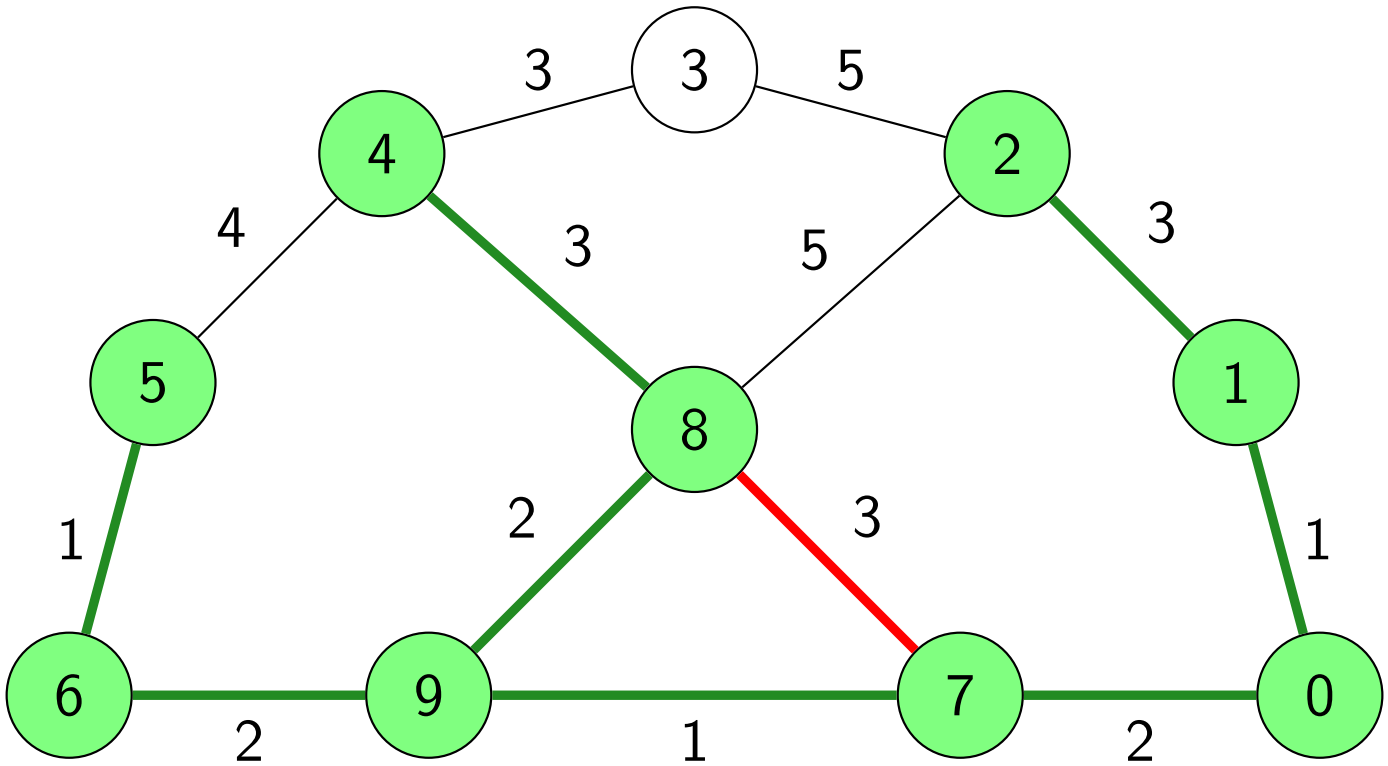
Prims algoritm



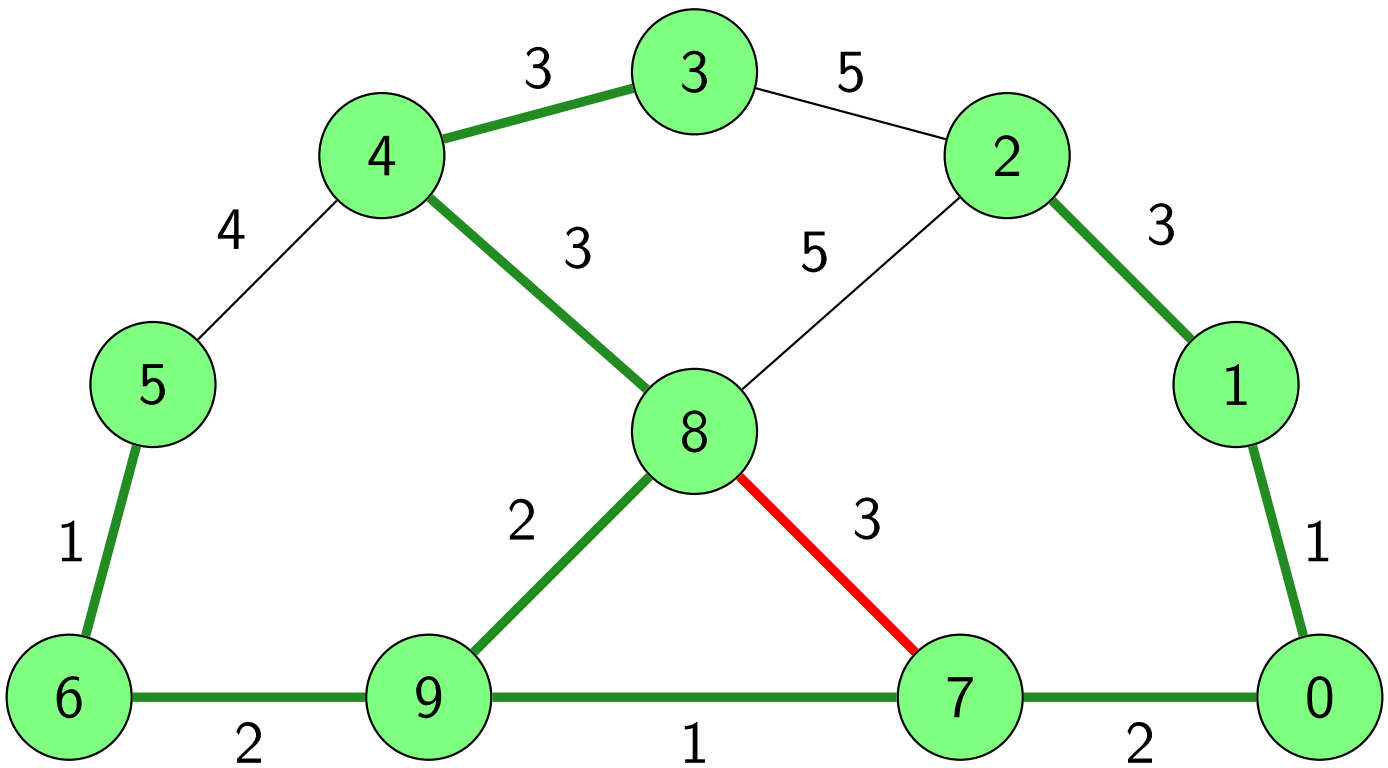
Prims algoritm



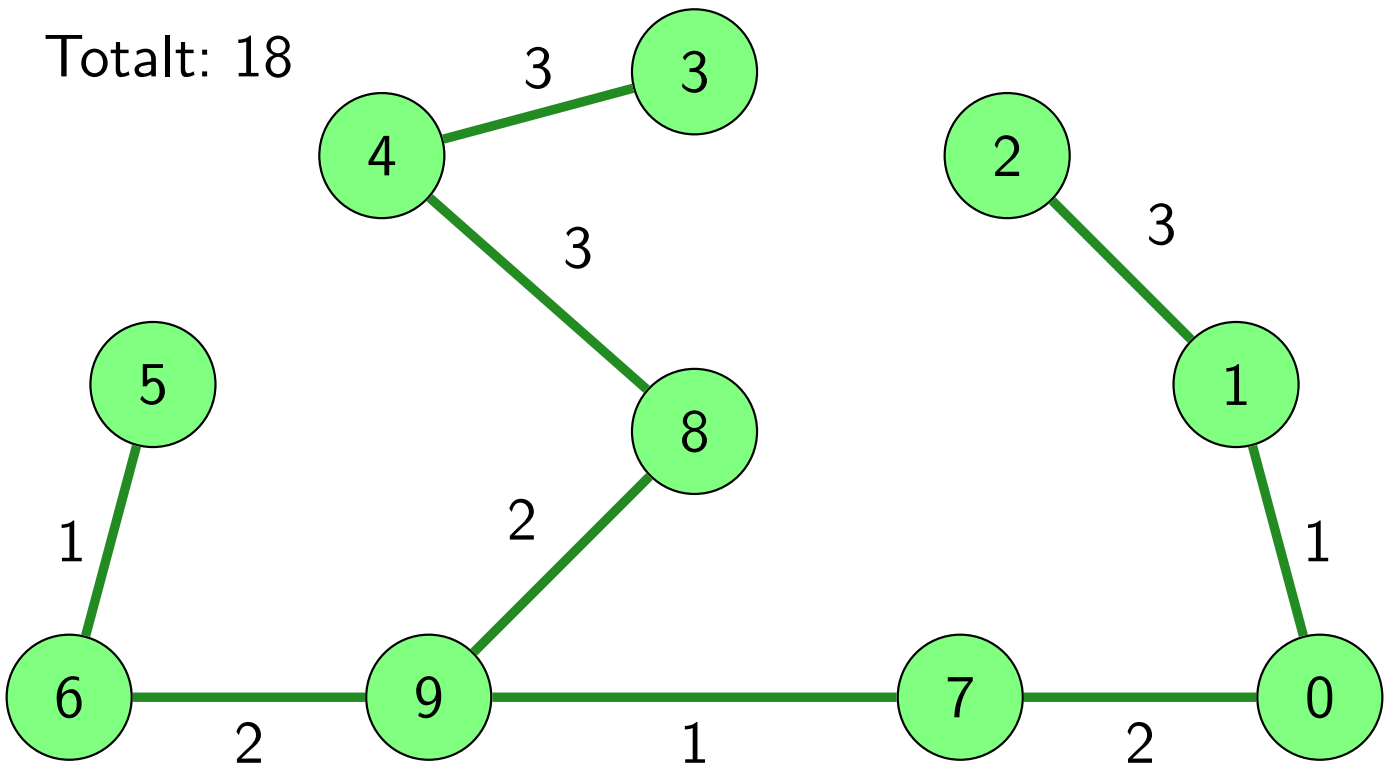
Prims algoritm



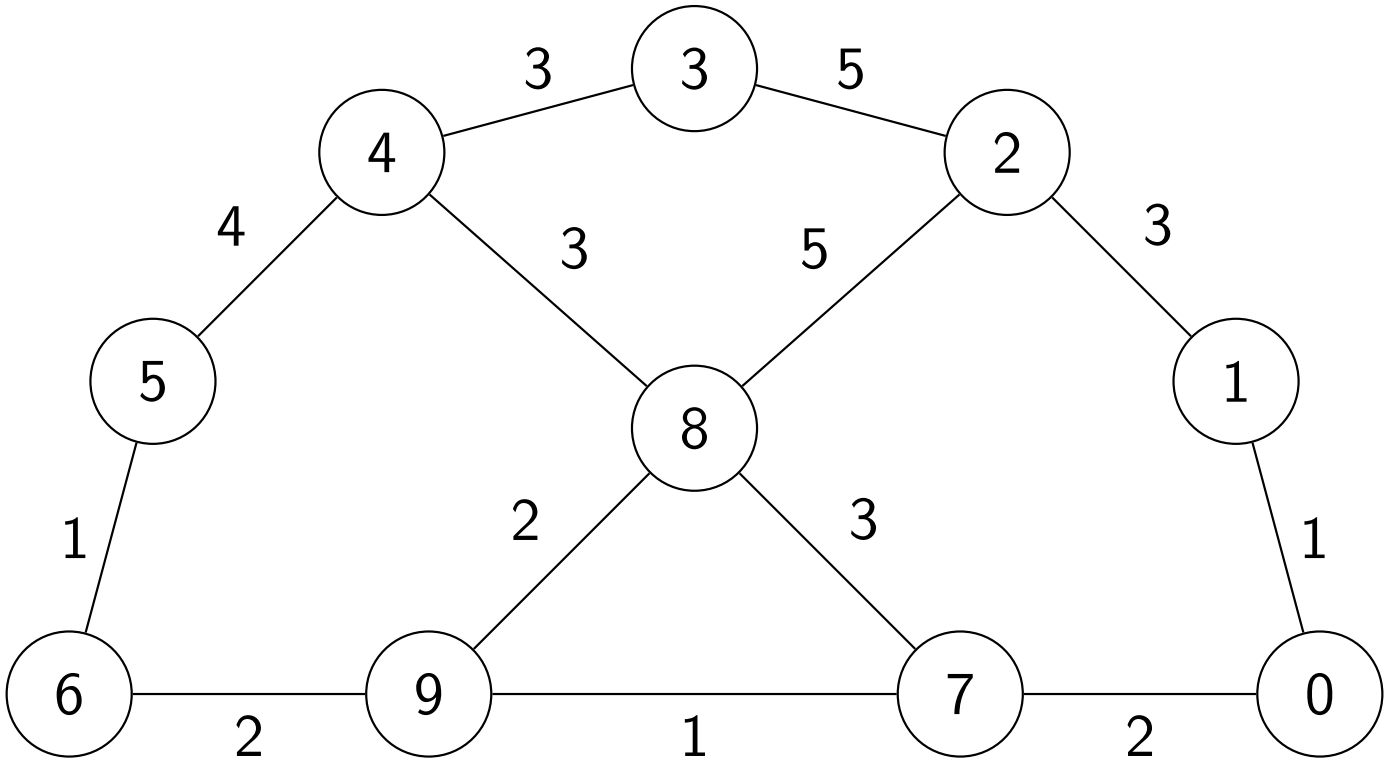
Prims algoritm



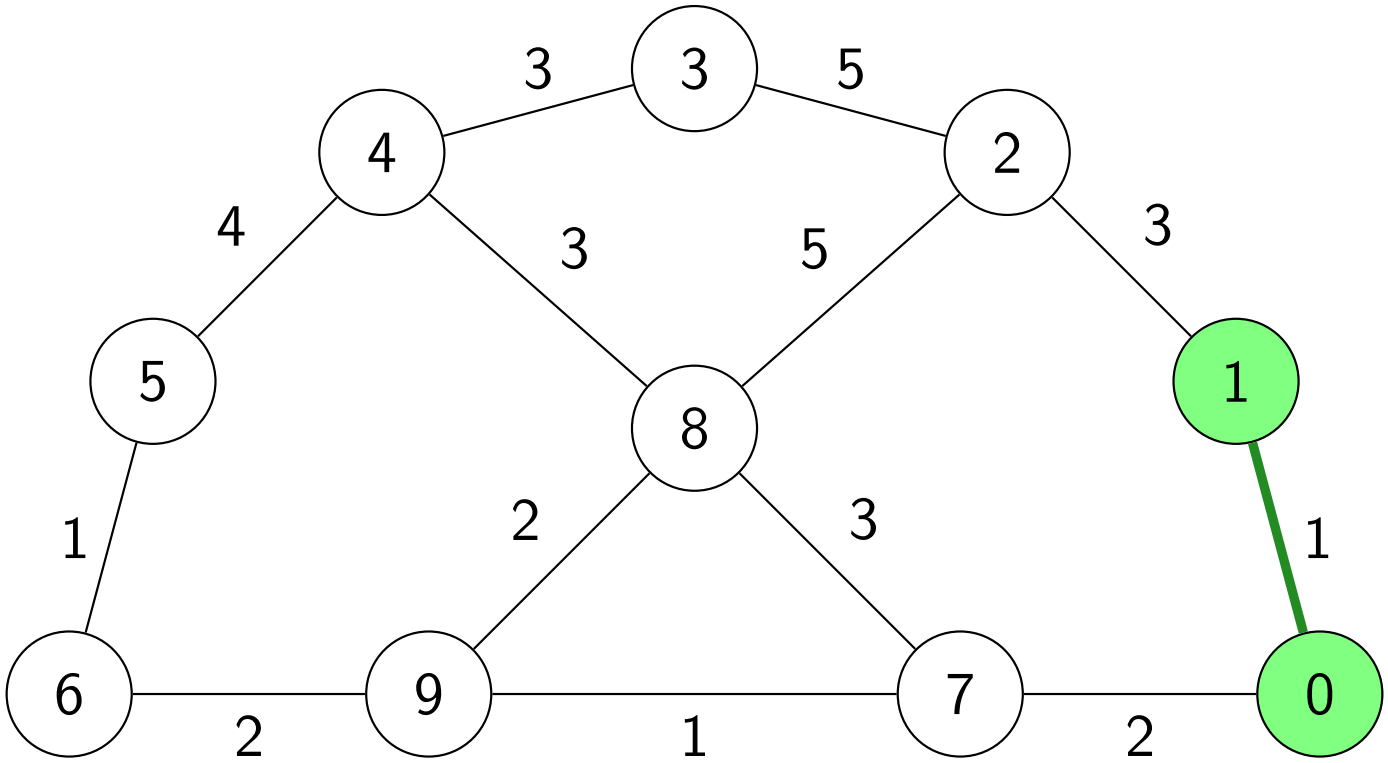
Prims algoritm



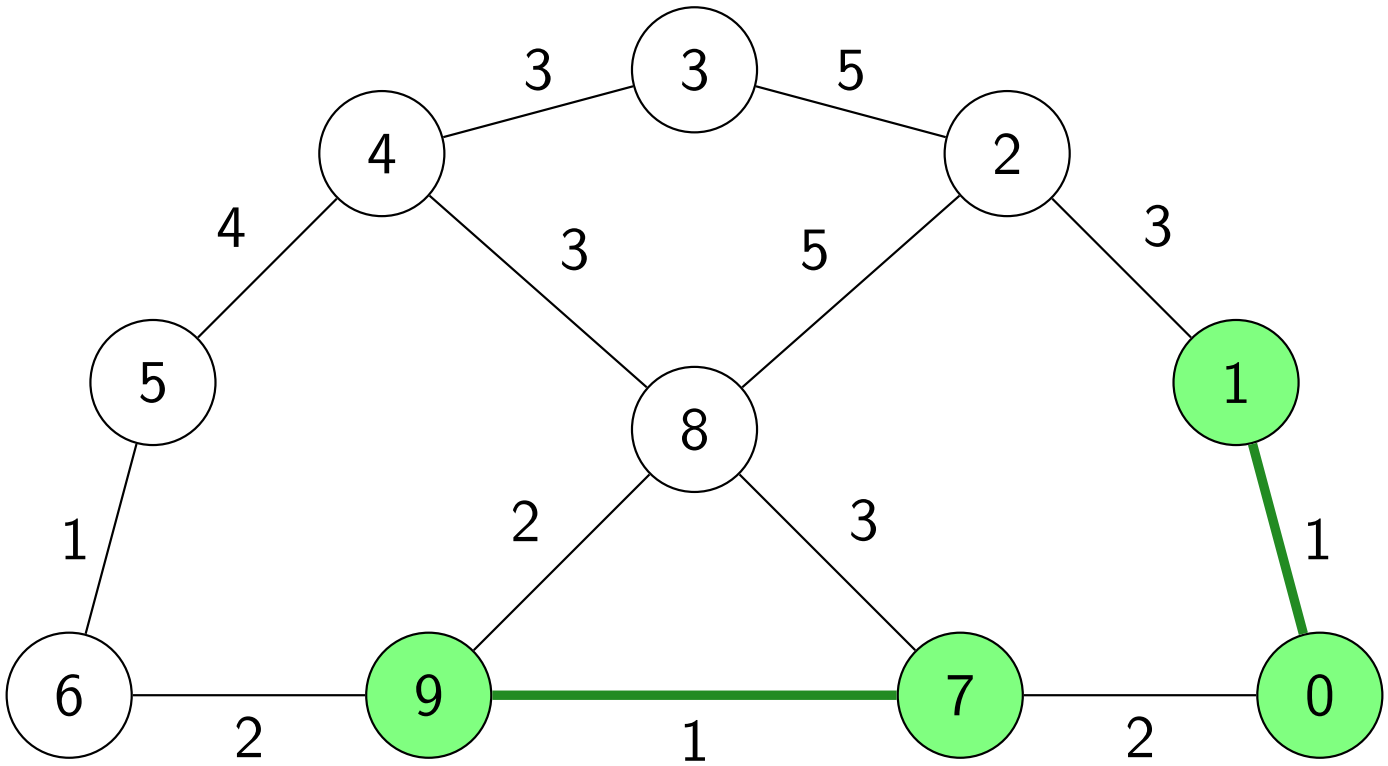
Kruskals algoritm



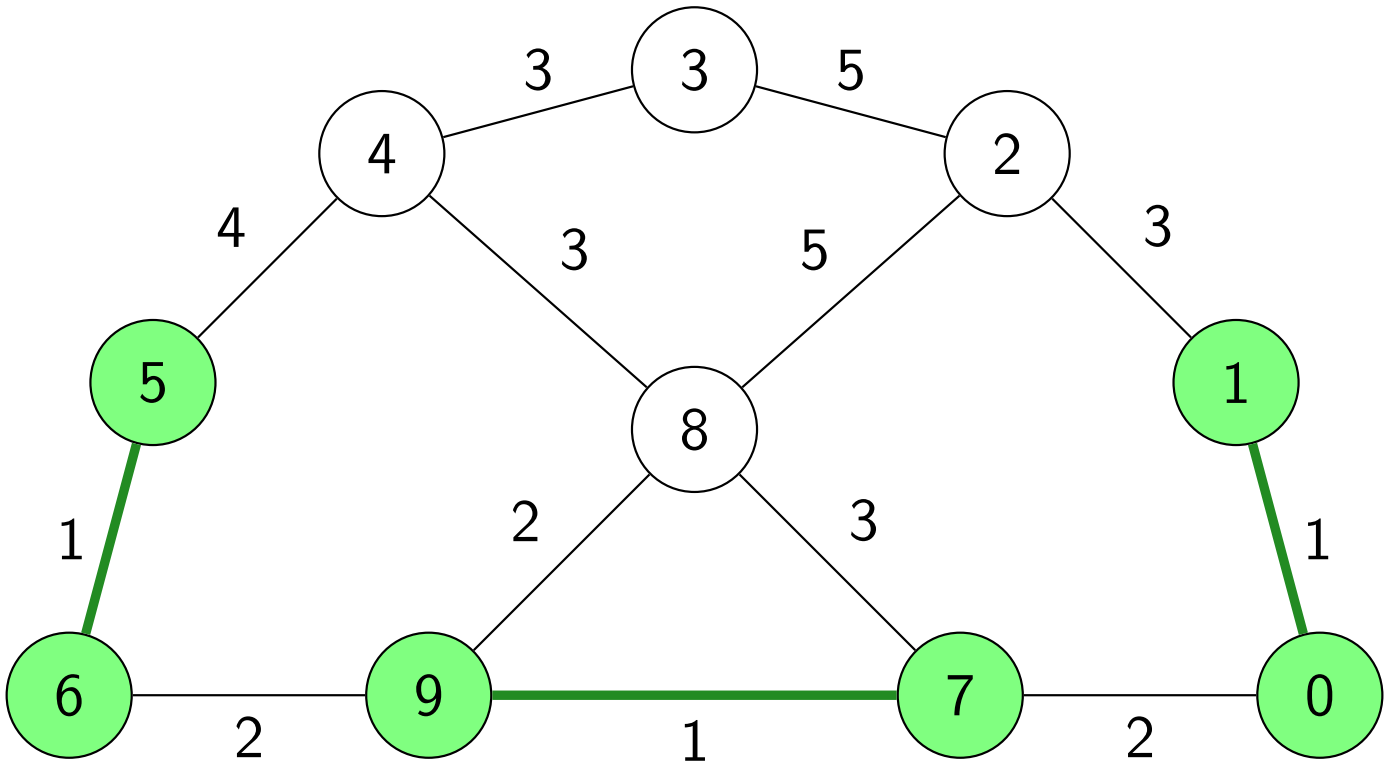
Kruskals algoritm



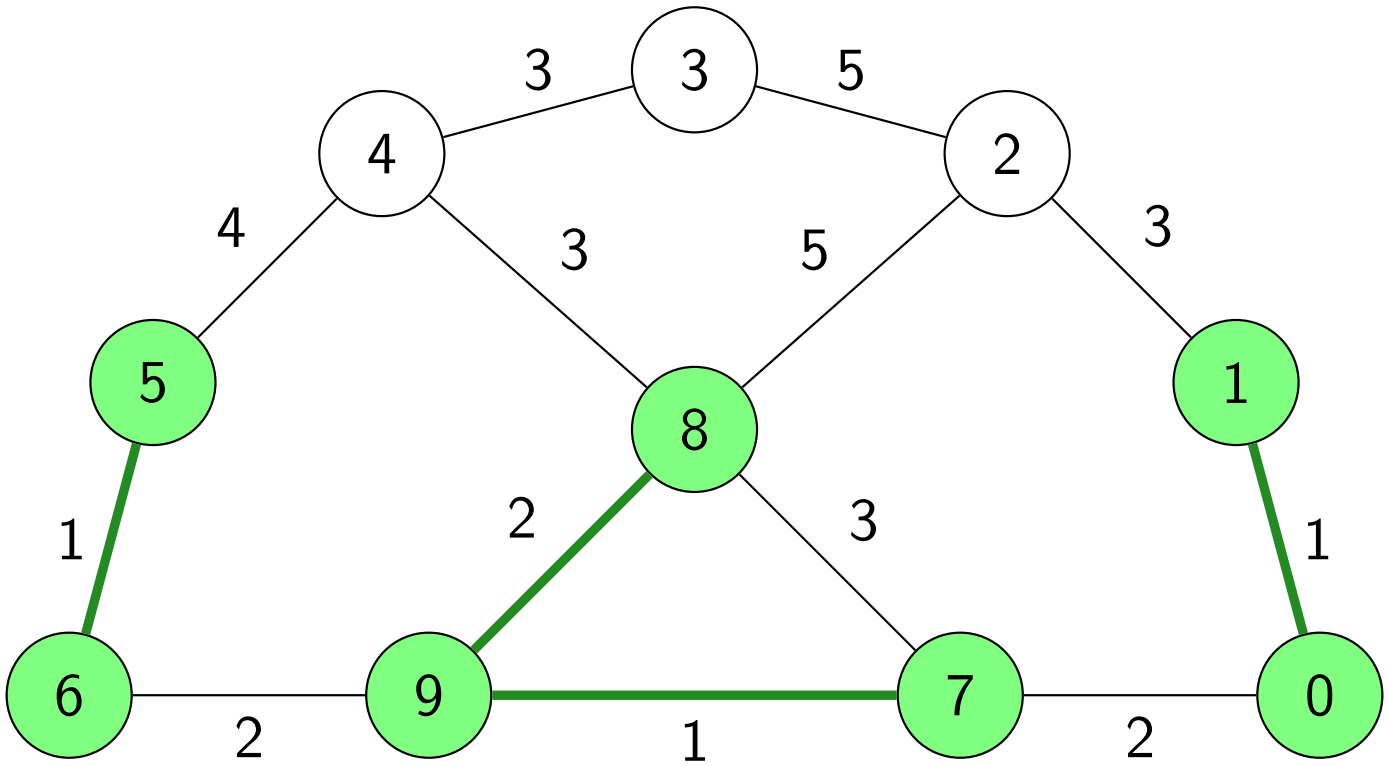
Kruskals algoritm



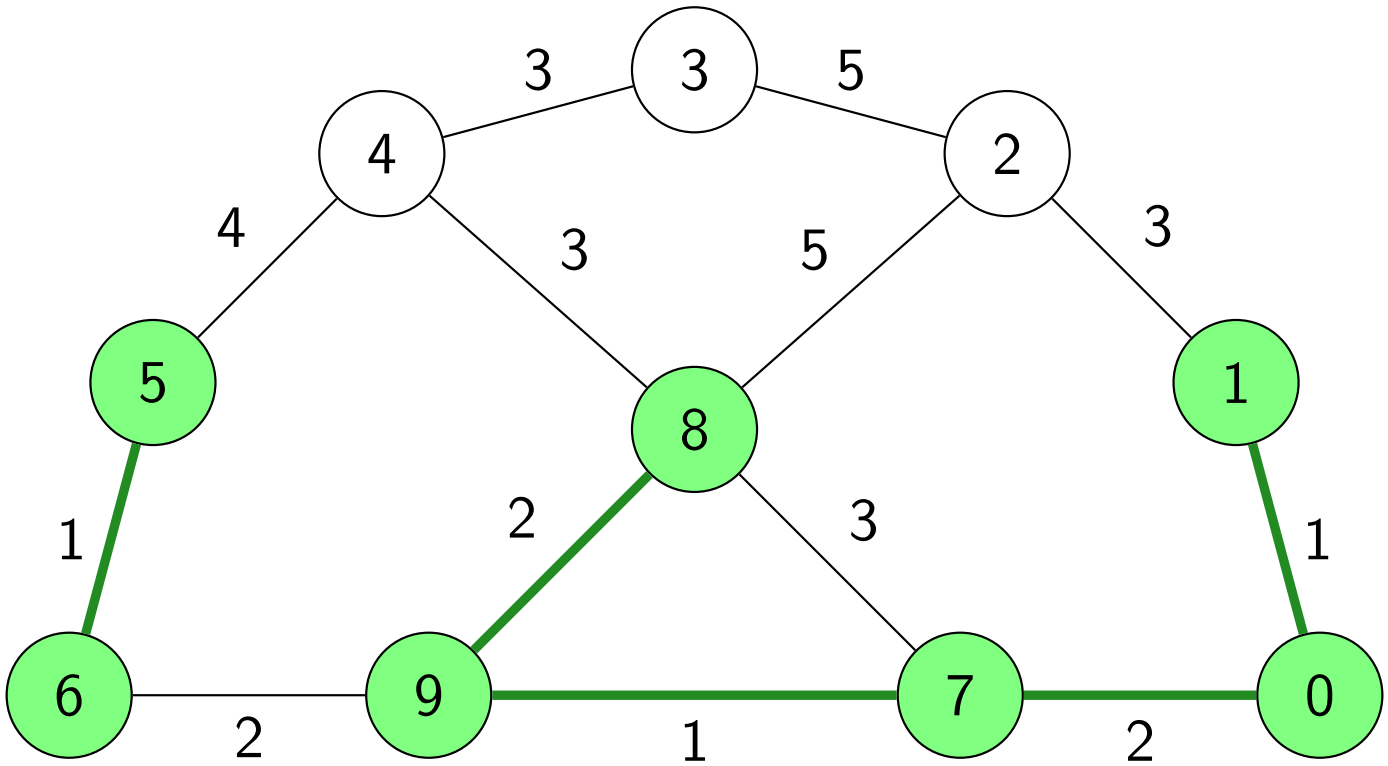
Kruskals algoritm



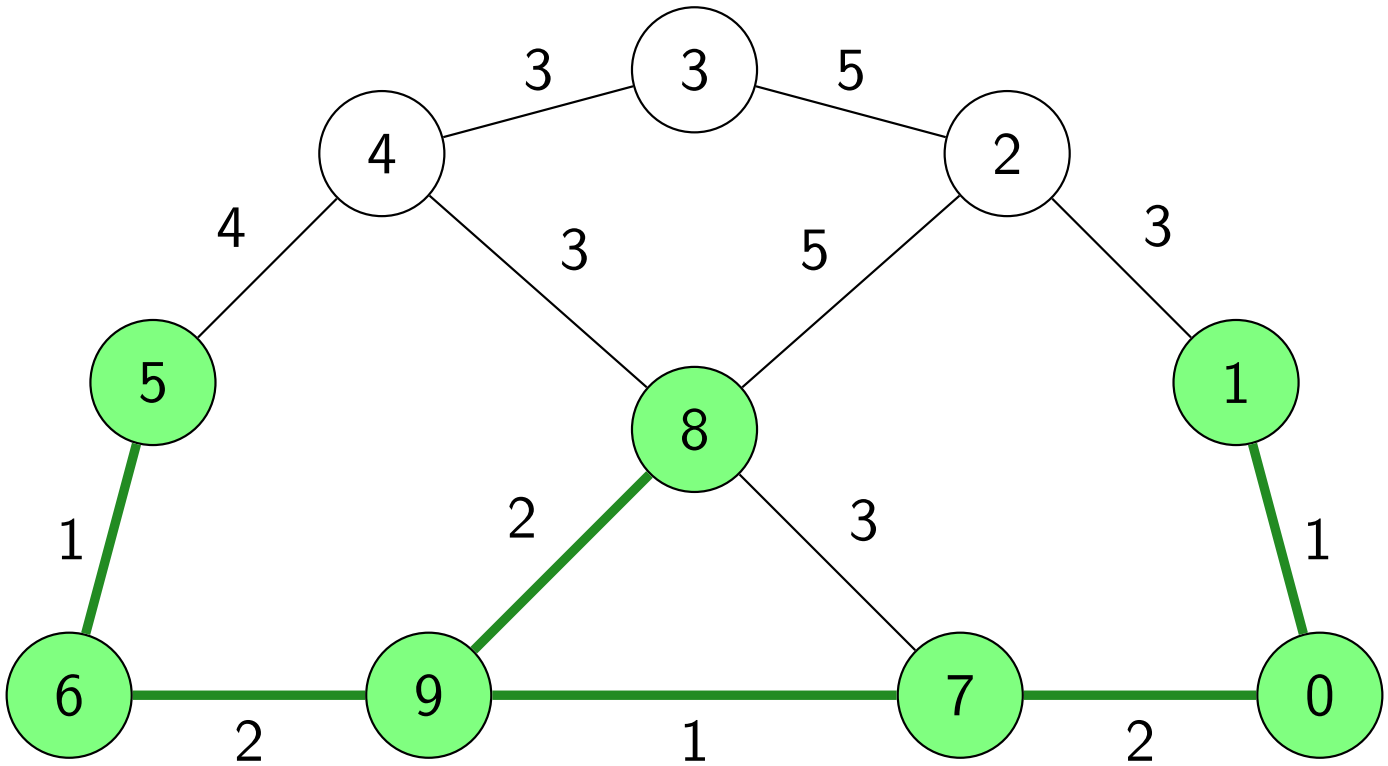
Kruskals algoritm



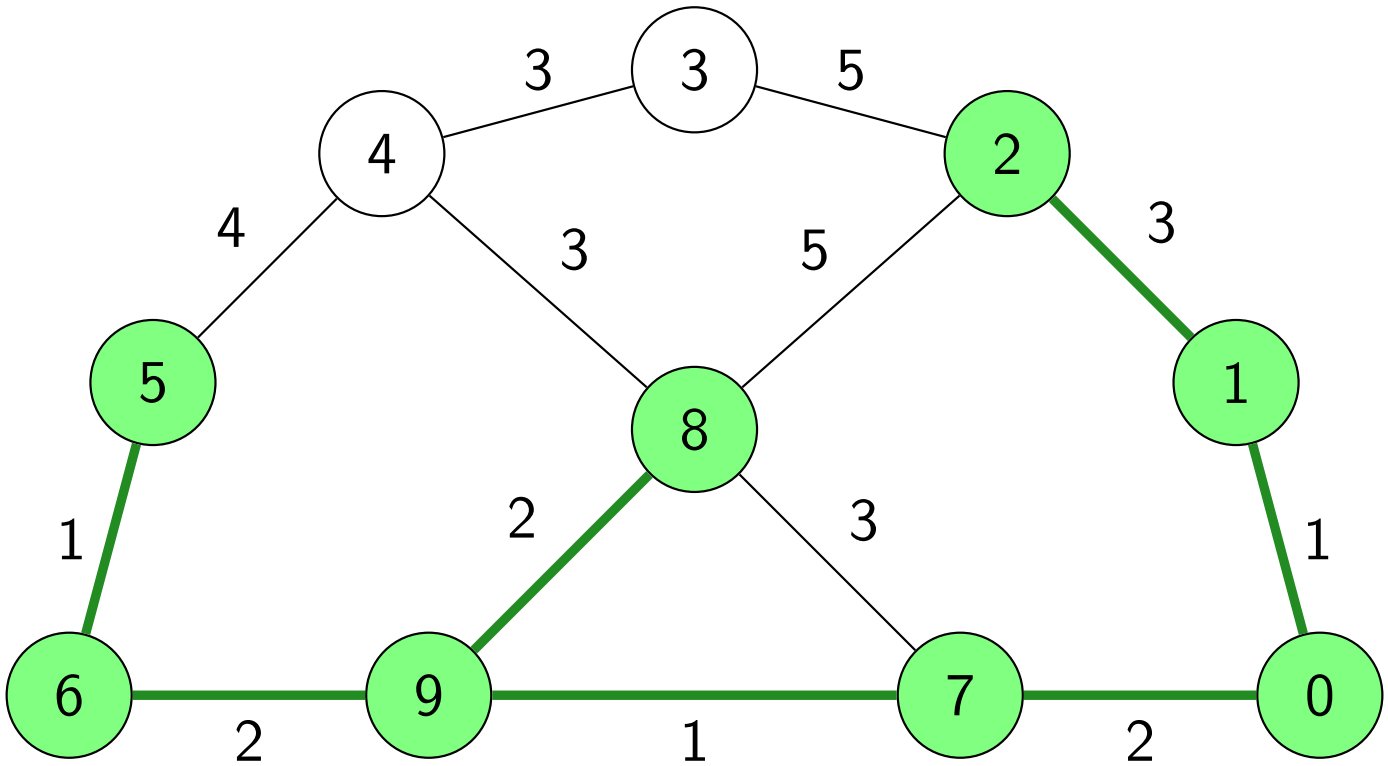
Kruskals algoritm



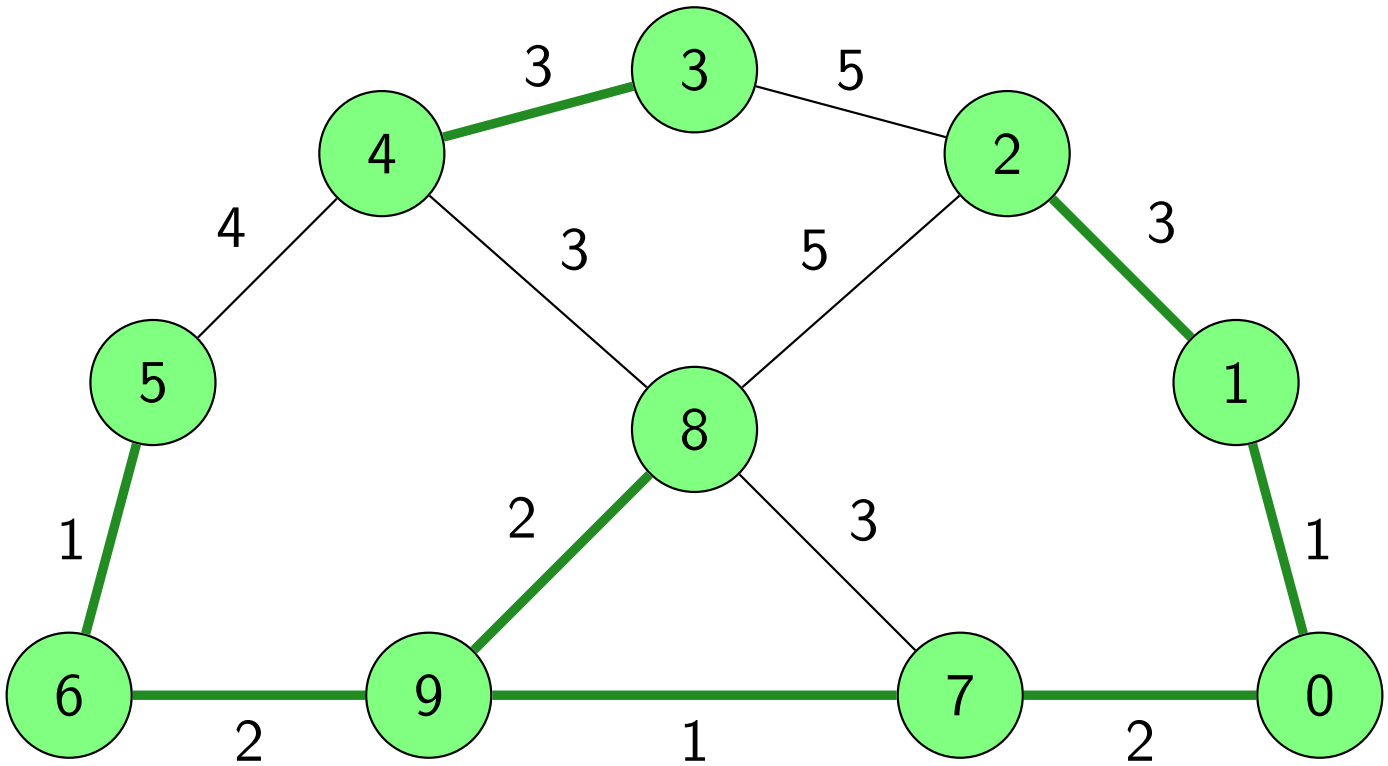
Kruskals algoritm



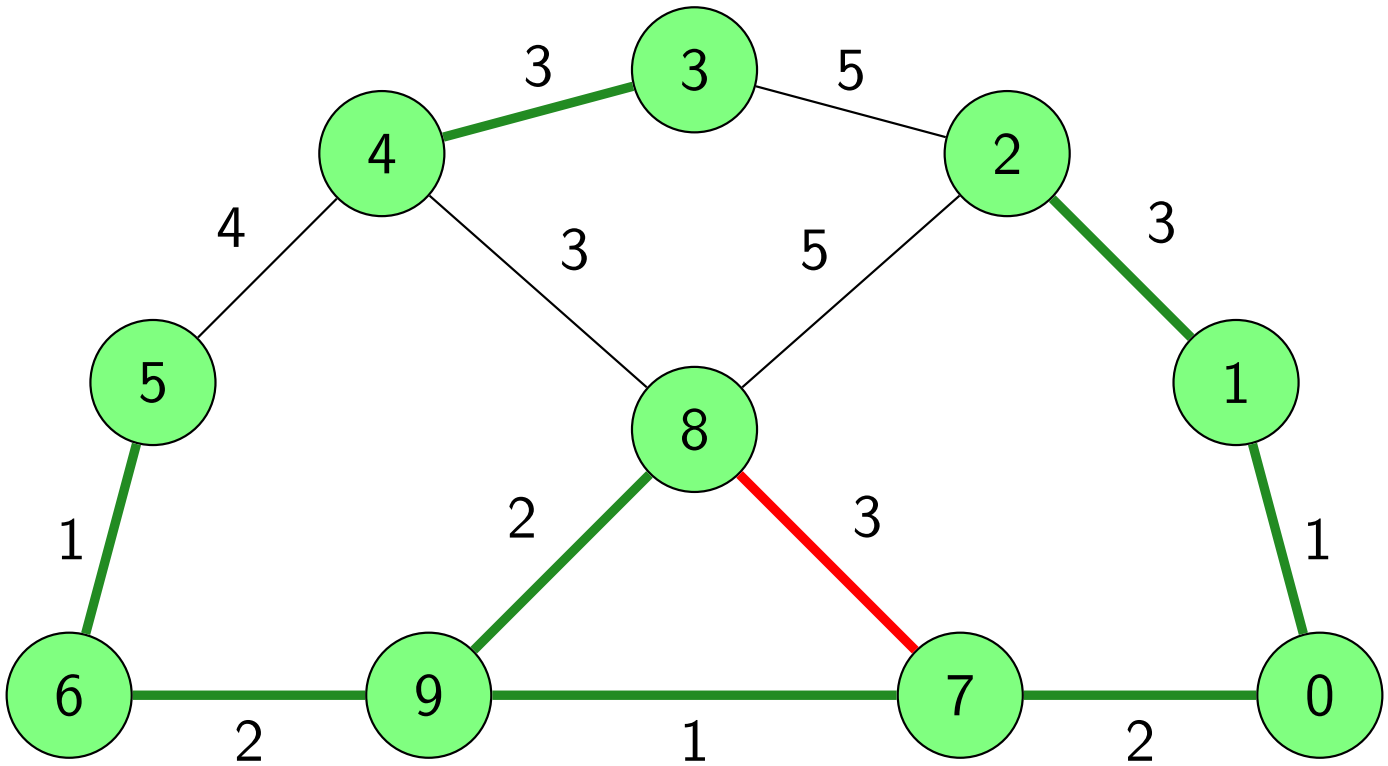
Kruskals algoritm



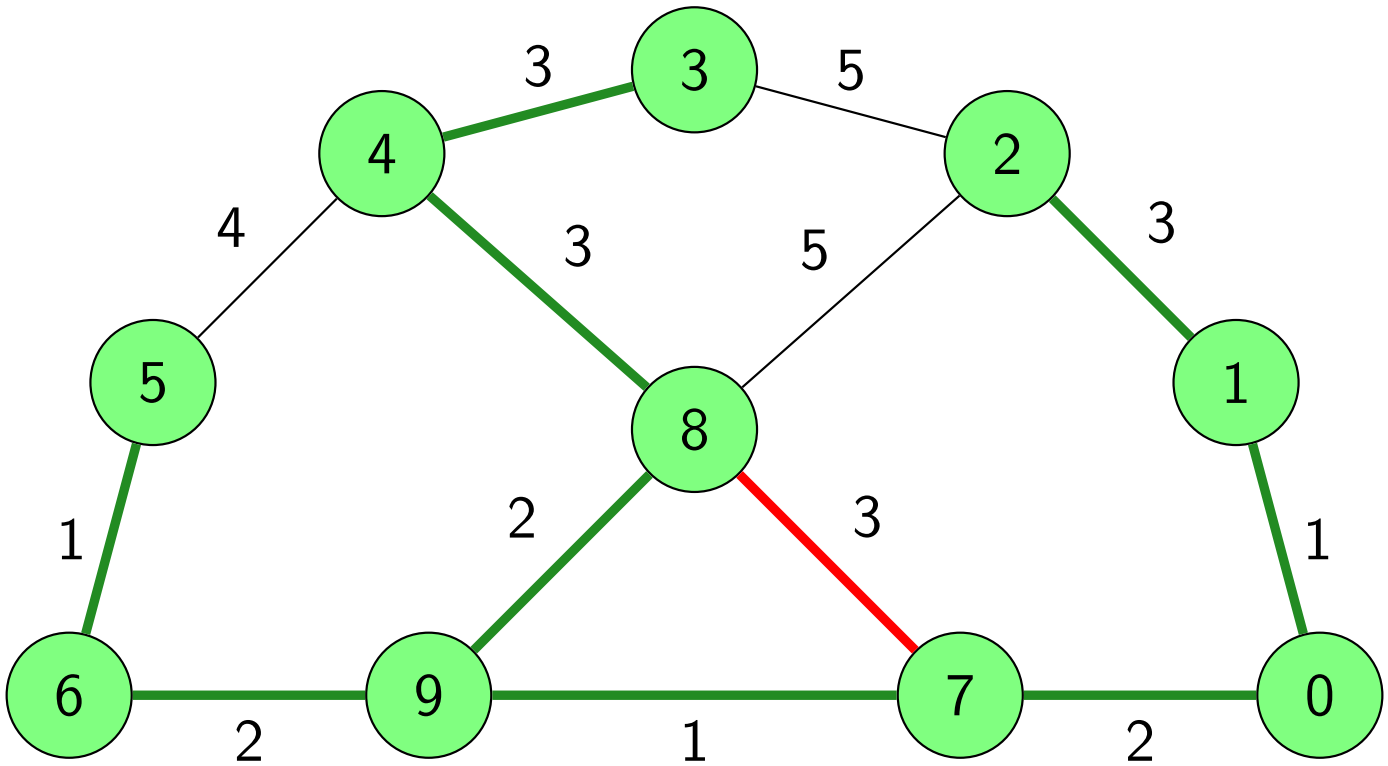
Kruskals algoritm



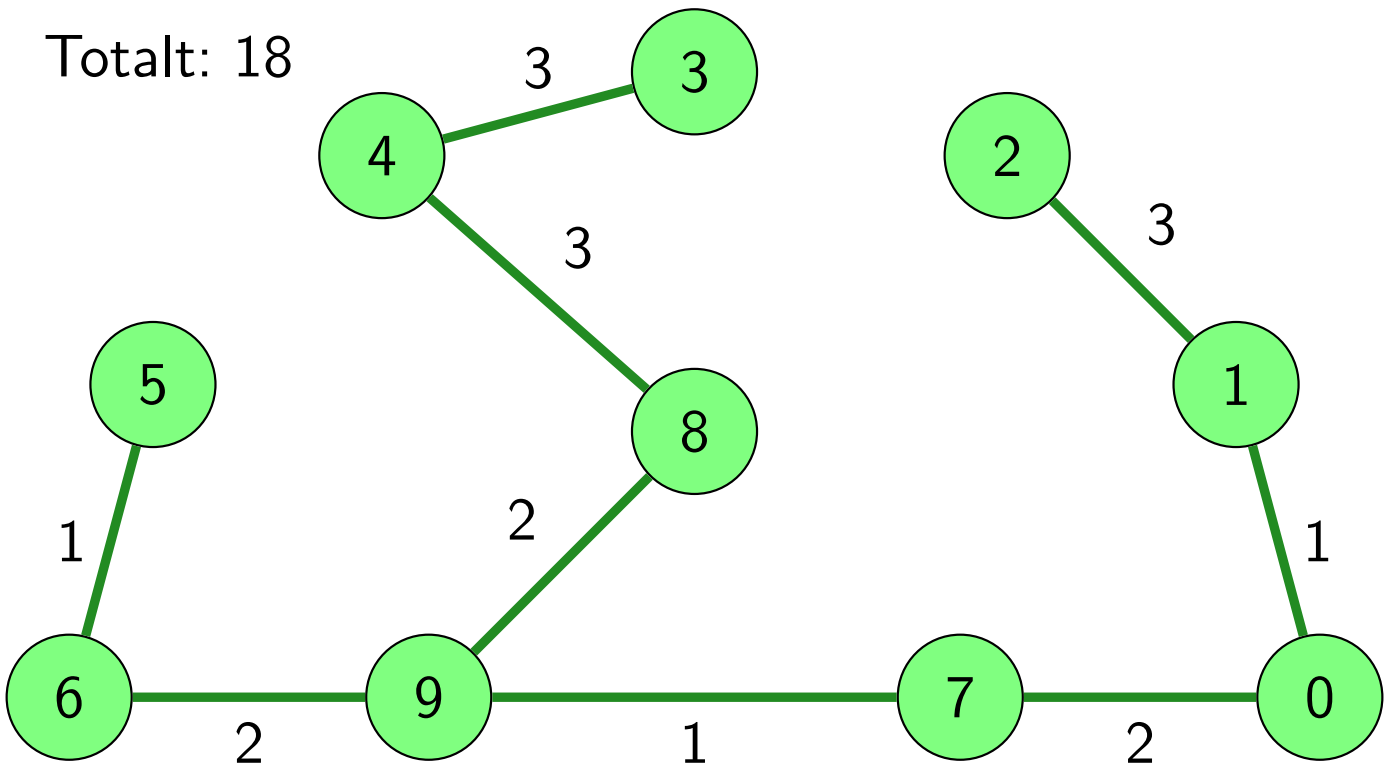
Kruskals algoritm



Kruskals algoritm



Kruskals algoritm



Datastruktur för bågar

Hur vet vi om vi ska lägga till en båge mellan två noder?

Behöver två operationer:

- Sätt ihop två element
- Kontrollera om två element sitter ihop

Datastruktur för bågar

Hur vet vi om vi ska lägga till en båge mellan två noder?

Behöver två operationer:

- Sätt ihop två element (`union`)
- Kontrollera om två element sitter ihop (`find`)

Datastruktur för bågar

Hur vet vi om vi ska lägga till en båge mellan två noder?

Behöver två operationer:

- Sätt ihop två element (`union`)
- Kontrollera om två element sitter ihop (`find`)

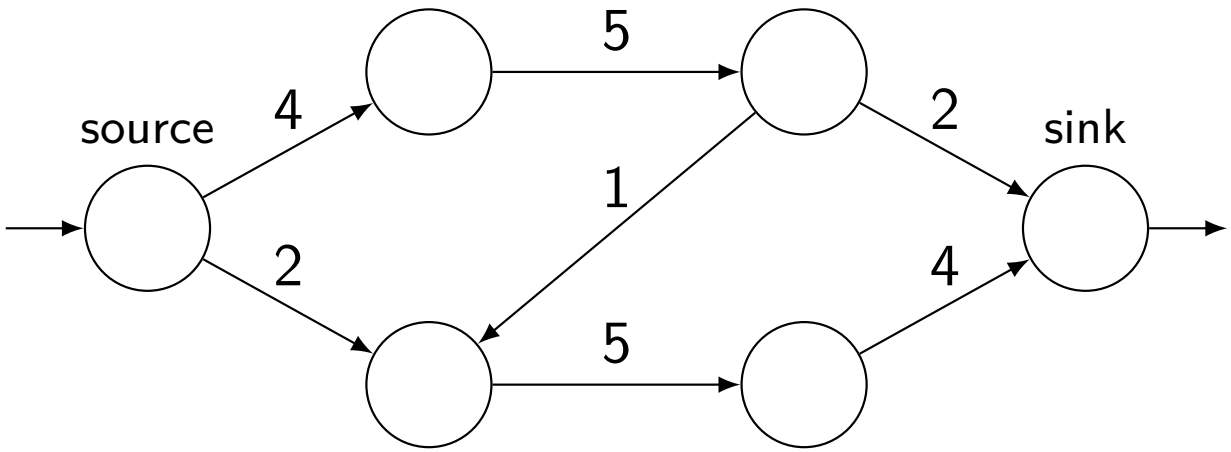
$\Rightarrow$ Union-find

- `union`: $\mathcal{O}(\log^* n) \approx \mathcal{O}(1)$
- `find`: $\mathcal{O}(\log^* n) \approx \mathcal{O}(1)$

LINKÖPING UNIVERSITY

Maxflow – Maximalt flöde i en graf

1. Initiera flödet i alla bågar till 0
2. Skapa en *residualgraf*
3. Hitta en väg (bredden först). Om det finns en väg, uppdatera flödet och repetera steg 2



Maxflow – Egenskaper

Två algoritmer (fler finns):

- **Ford-Fulkerson**
  Använd DFS för att hitta nytt flöde.
  Tar $\mathcal{O}(f \cdot |V||E|^2)$, f beror på det maximala flödet.
- **Edmonds-Karp**
  Använd BFS för att hitta nytt flöde.
  Tar $\mathcal{O}(|V||E|^2)$.

Problem

Du ska anordna ett kalas. Du vill ge var och en av gästerna en godispåse, och därför har du köpt lösgodis av diverse sorter som du vet gästerna gillar. Gästerna är olika förtjusta i de olika sorterna, så du vill inte ge för många av en viss typ till en gäst som inte gillar den typen. Du vill dessutom att gästerna ska få lika många bitar totalt.

Hur ska du fördela godiset för att det ska bli så bra som möjligt?

Problem

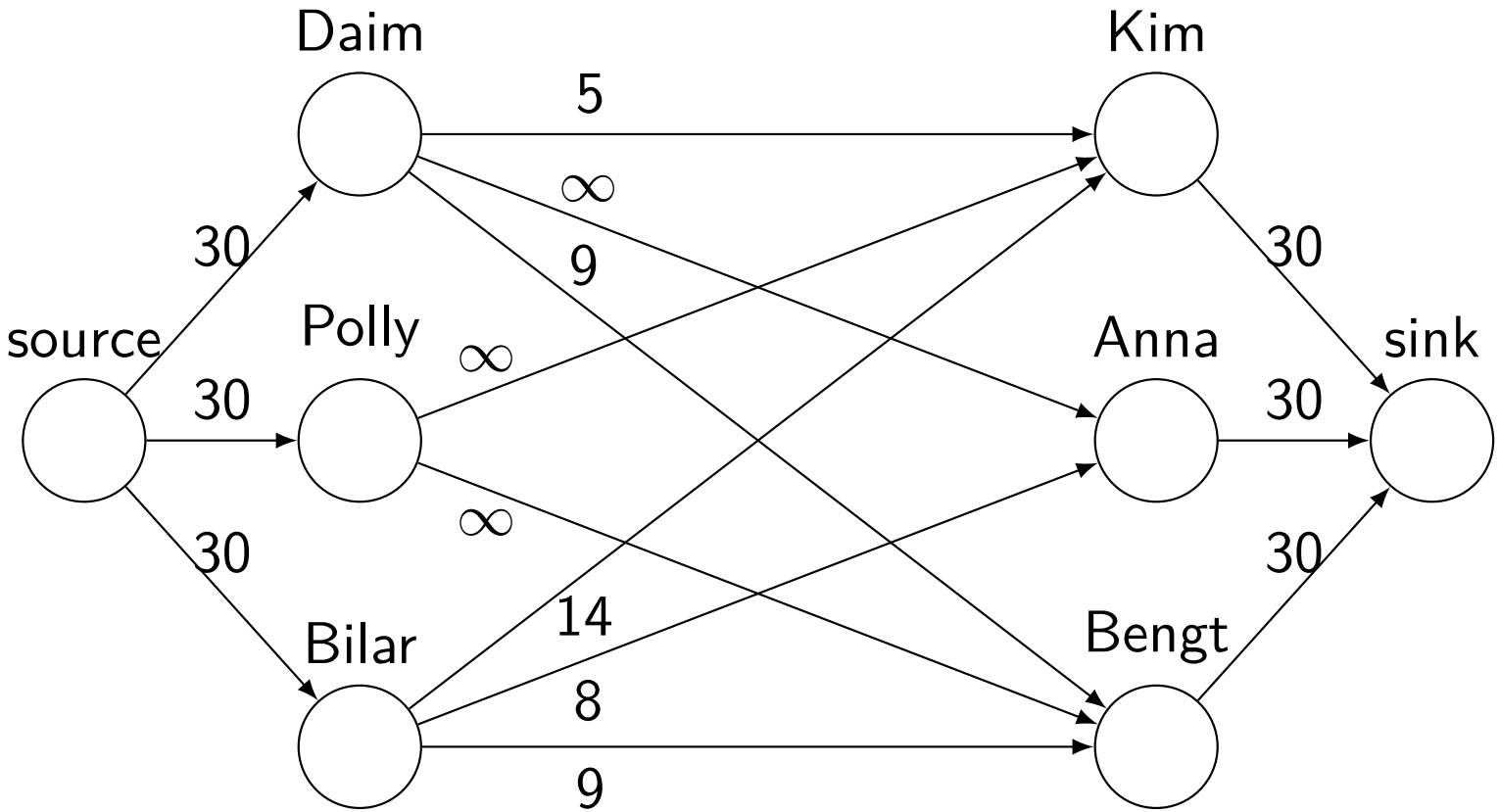
	Person	Kim	Anna	Bengt
Sort	Mängd	30	30	30
Daim	30	≤ 5		≤ 9
Polly	30		≤ 0	
Bilar	30	≤ 14	≤ 8	≤ 9

Problem

	Person	Kim	Anna	Bengt
Sort	Mängd	30	30	30
Daim	30	≤ 5		≤ 9
Polly	30		≤ 0	
Bilar	30	≤ 14	≤ 8	≤ 9

Det här är ju en graf!

Grafrepresentation



En lösning

	Person	Kim	Anna	Bengt
Sort	Mängd	30	30	30
Daim	30	5	22	3
Polly	30	12	0	18
Bilar	30	13	8	9

LINKÖPING UNIVERSITY

I kursen framöver

- Denna veckan
  - Se till att börja på lab 3
- Nästa vecka
  - Sortering, introduktion till lab 4
- Extrauppgifter
  - 10305 (enklare)
    Är en topologisk sortering.
  - 12274 (svårare)
    Går att lösa med grafsökning, frågan är bara i vilken graf!

Filip Strömbäck

www.liu.se