MDE

Labwork2 – Class 3

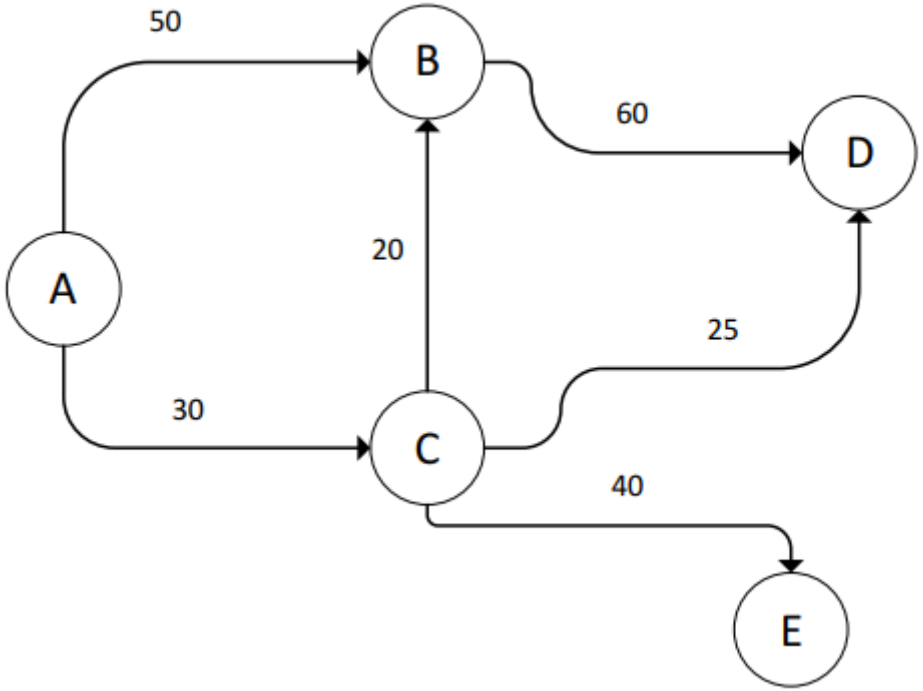
Graphs in Prolog

Examples

2024 - 2025

Graph Examples

- Directed Graph



- Facts

```
dist(a,b,50).
dist(a,c,30).
dist(b,d,60).
dist(c,b,20).
dist(c,d,25).
dist(c,e,40).
```

- Rules:

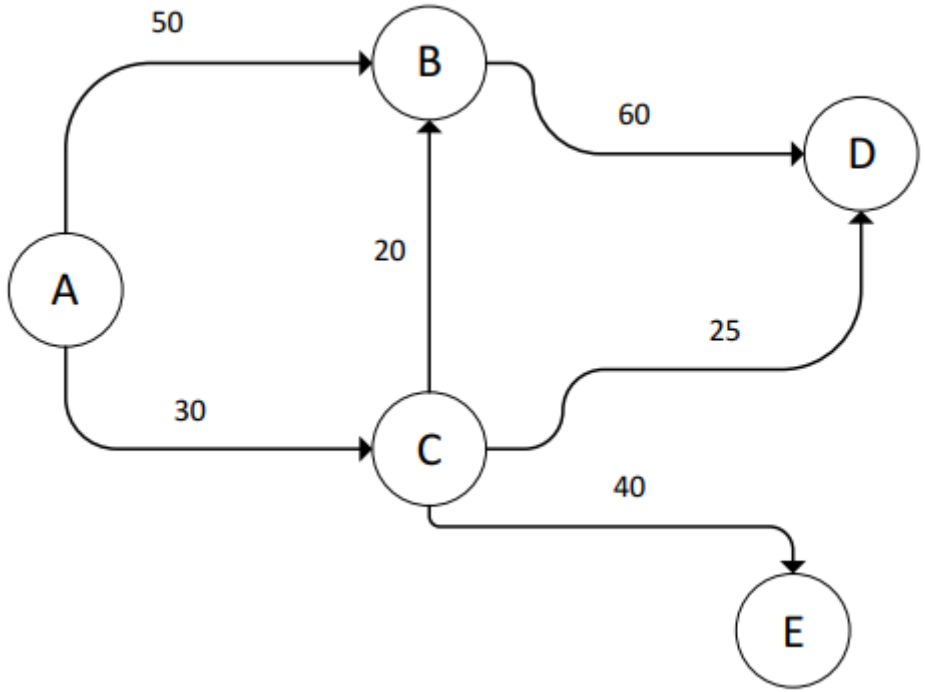
```
distance(X,Y,D) :- dist(X,Y,D).
distance(X,Y,D) :- dist(X,Z,D1),
                   distance(Z,Y, D2),
                   D is D1 + D2.
```

- Try:

```
?-distance(a,d,D).
```

Graph Examples

- Calculate the minimal distance



- Facts

```
dist(a,b,50).
dist(a,c,30).
dist(b,d,60).
dist(c,b,20).
dist(c,d,25).
dist(c,e,40).
```

- Rules:

```
distance(X,Y,D) :- dist(X,Y,D).
distance(X,Y,D) :- dist(X,Z,D1),
                   distance(Z,Y, D2),
                   D is D1 + D2.

mindist(X,Y,D) :- findall(Di, distance(X,Y,Di), LDi),
                  min(LDi, D).
min([X],X).
min([X|R], X) :- min(R,M), X =< M.
min([X|R],M) :- min(R,M), X > M.
```

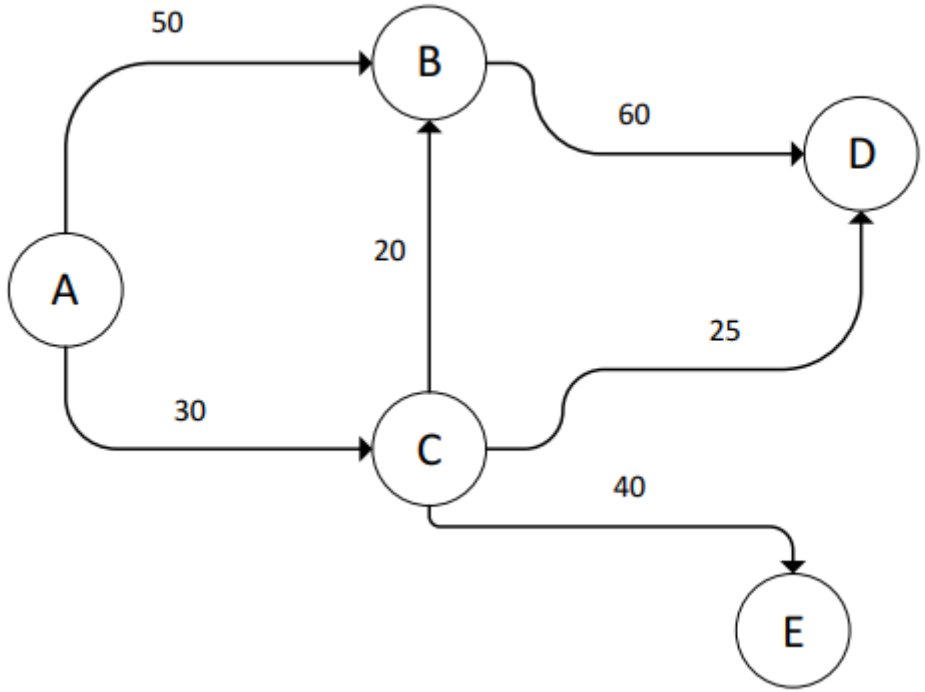
- Try:

```
? - mindist(a,d,Min).
```

```
?- mindist(a,d,M).
M = 55 ;
```

Graph Examples

- Calculate the maximum distance



- Facts

```
dist(a,b,50).
dist(a,c,30).
dist(b,d,60).
dist(c,b,20).
dist(c,d,25).
dist(c,e,40).
```

- Rules:

```
distance(X,Y,D) :- dist(X,Y,D).
distance(X,Y,D) :- dist(X,Z,D1),
                   distance(Z,Y, D2),
                   D is D1 + D2.

maxdist(X,Y,D) :- findall(Di, distance(X,Y,Di), LDi),
                  max(LDi, D).
max([X], X).
max([X | R], X) :- max(R, M),
                   X >= M.
max([X | R], M) :- max(R,M),
                   X < M.
```

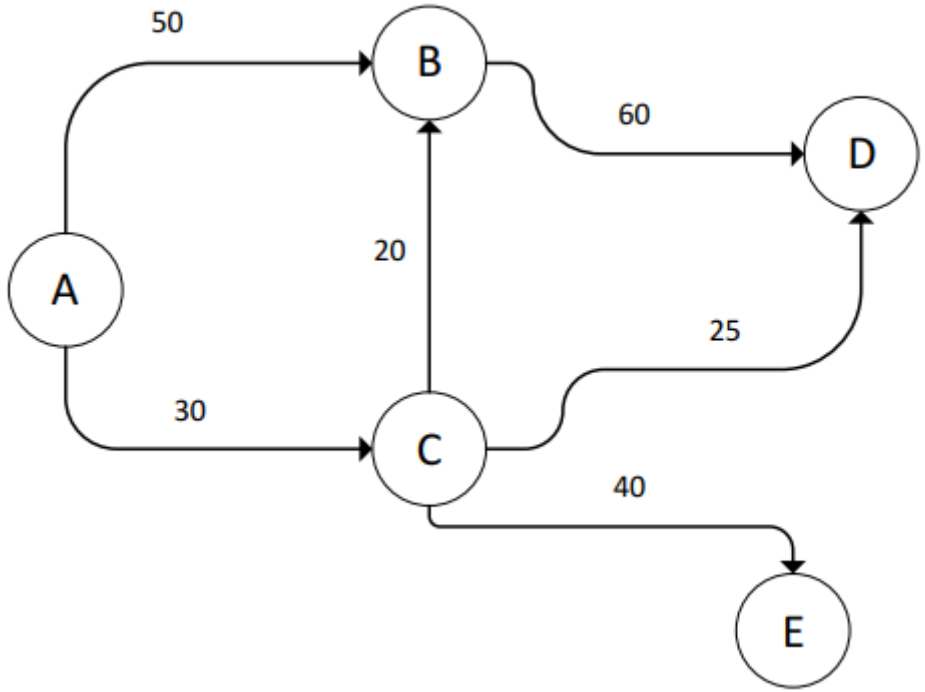
- Try:

```
? - maxdist(a,d,Max).
```

```
?- maxdist(a,d,Max).
Max = 110 ;
```

Graph Examples

- Path between two nodes using lists



- Facts

```
dist(a,b,50).
dist(a,c,30).
dist(b,d,60).
dist(c,b,20).
dist(c,d,25).
dist(c,e,40).
```

```
?- path(a,d,P).
P = [dist(a, b, 50), dist(b, d, 60)] ;
P = [dist(a, c, 30), dist(c, d, 25)] ;
P = [dist(a, c, 30), dist(c, b, 20), dist(b, d, 60)]
```

- Rules:

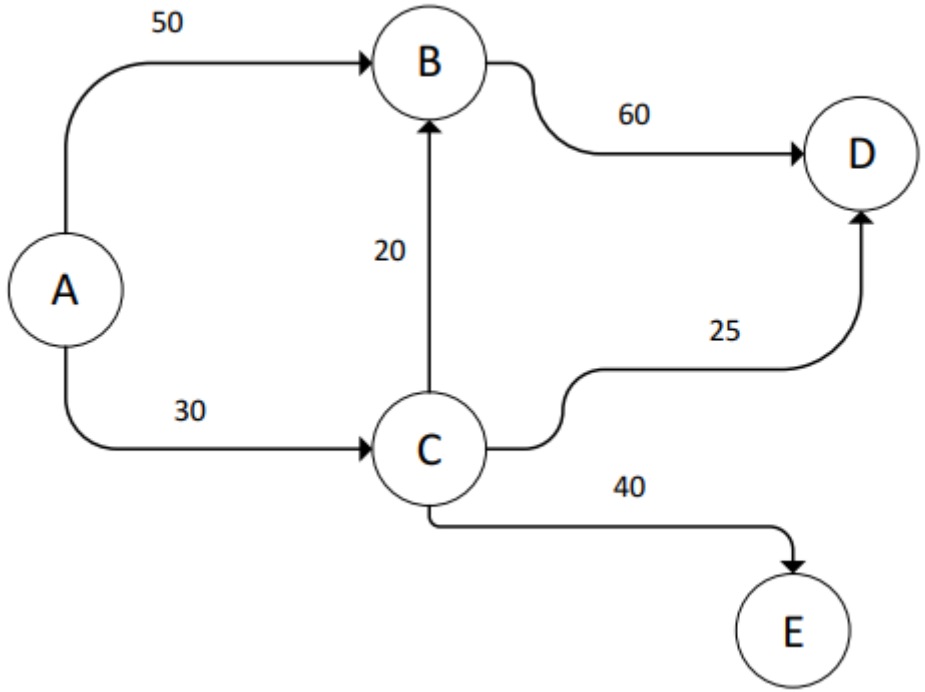
```
path(X,Y, [dist(X,Y,D)]) :- dist(X,Y,D).
path(X,Y, [dist(X,Z,D) | R]) :- dist(X,Z,D),
                                path(Z,Y,R).
```

- Try:

```
? - path(a,d,P).
```

Graph Examples

- Calculate the total distance between the two nodes



- Facts

  dist(a,b,50).
  dist(a,c,30).
  dist(b,d,60).
  dist(c,b,20).
  dist(c,d,25).
  dist(c,e,40).

- Rules:

  pathdist(X,Y, p([dist(X,Y,D)],D)) :- dist(X,Y,D).
  pathdist(X,Y, p([dist(X,Z,D1)|R],DT)) :-
  dist(X,Z,D1),
  pathdist(Z,Y,p(R,D2)),
  DT is D1 + D2.

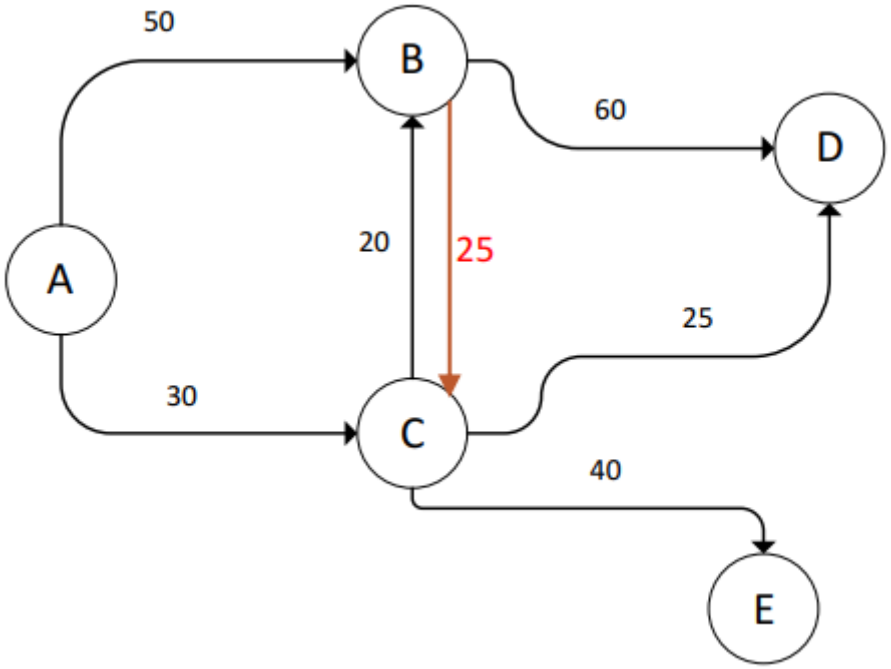
- Try:

  ? - pathdist(a,d,P).

```
?- pathdist(a, d, P).
P = p([dist(a, b, 50), dist(b, d, 60)], 110) ;
P = p([dist(a, c, 30), dist(c, d, 25)], 55) ;
P = p([dist(a, c, 30), dist(c, b, 20), dist(b, d, 60)], 110) ;
```

Graph Examples

- Now add a new arc with a different direction



- Facts

  dist(a,b,50).
  dist(a,c,30).
  dist(b,d,60).
  dist(c,b,20).
  dist(c,d,25).
  dist(c,e,40).
  dist(b,c,25).

- Rules:

  distance(X,Y,D) :- dist(X,Y,D).
  distance(X,Y,D) :- dist(X,Z,D1),
  distance(Z,Y, D2),
  D is D1 + D2.

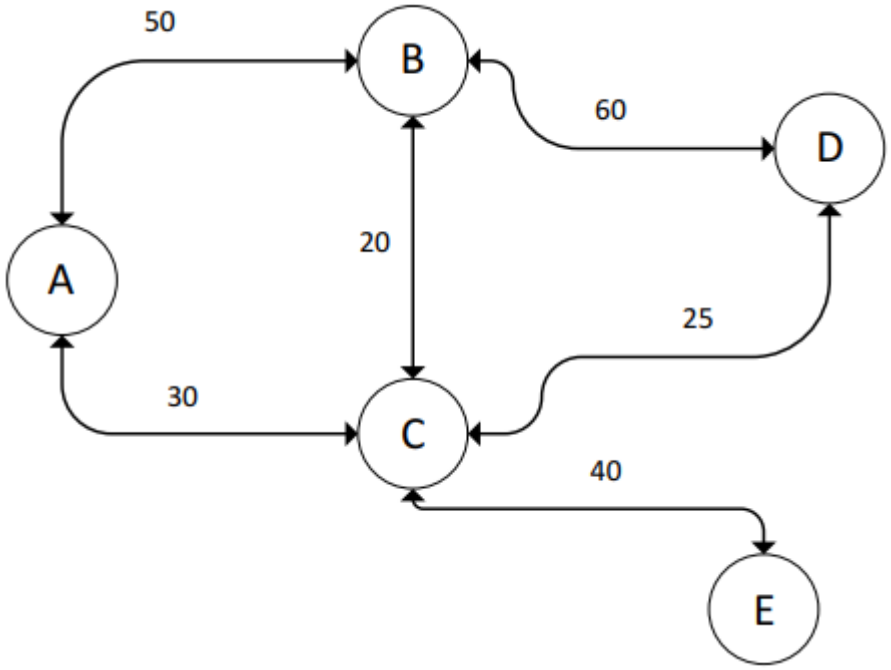
- Try:

  ? - distance(a,d,D).

```
?- distance(a,d,D).
D = 110 ;
D = 100 ;
D = 155 ;
D = 145 ;
D = 200 ;
D = 190 ;
D = 245 ;
D = 235 ;
D = 290 ;
D = 280 ;
D = 335 ;
D = 325 ;
D = 380 ;
D = 370 ;
D = 425 ;
D = 415 ;
D = 470 ;
D = 460 ;
D = 515 ;
D = 505 ;
D = 560 ;
D = 550 ;
D = 605
```

Bidirectional Graph Example

- Bi-directional graph



- To avoid the last slide problem, it is necessary to guarantee that we cannot pass more than once in each arc and node

- **Facts**

```
dist(a,b,50).
dist(a,c,30).
dist(b,d,60).
dist(c,b,20).
dist(c,d,25).
dist(c,e,40).
```

```
?- pass_once(a,d,P,D).
P = [dist(a, b, 50), dist(b, d, 60)],
D = 110 ;
P = [dist(a, b, 50), dist(b, c, 20), dist(c, d, 25)],
D = 95 ;
P = [dist(a, c, 30), dist(c, d, 25)],
D = 55 ;
P = [dist(a, c, 30), dist(c, b, 20), dist(b, d, 60)],
D = 110 ;
false.
```

- **Rules**:

```
biarc(X,Y,D) :- dist(X,Y,D).
biarc(X,Y,D) :- dist(Y,X,D).

pass_once(X,Y,TP, TD) :- stepnr(X,Y,[],TP, 0, TD).

stepnr(CP,FP,PP,TP, PD, TD) :- biarc(CP,FP,D),
                               addnorep(PP, dist(CP, FP, D),TP),
                               TD is PD + D.
stepnr(CP,FP,PP,TP, PD, TD) :- biarc(CP,NP,D),
                               addnorep(PP, dist(CP, NP, D),Pi),
                               Di is PD + D,
                               stepnr(NP, FP, Pi, TP, Di, TD).

addnorep(PP, dist(P1,P2, D), Pi) :- not(passed(PP, P2)),
                                    conc(PP,[dist(P1,P2, D)],Pi).

passed([dist(P,_,_)|_], P).
passed([dist(_,P,_)|_], P).
passed([_|R], P) :- passed(R, P).

conc([], L, L).
conc([C|R], L, [C|T]) :- conc(R, L, T).
```

- **Try**:
  ? – pass_once(a,d,P,D).

Summary

✓ Represent a graph (directed or bidirectional)

✓ Calculate maximum/minimum distance between node using lists

✓ Find paths using lists and recursion

✓ Add arcs dynamically

NEXT

Use inference rules to reason over the graph and fulfill the corresponding functional requirements!