CS 520 Advanced Programming Languages
Fall Semester, 2009
Doc 5 Prolog Examples
Sept 17, 2009

Copyright ©, All rights reserved. 2009 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

Remove K'th Element

remove_at(RemovedElement,L,K,ResultantList)

```
remove_at(X,[X|Xs],1,Xs).
remove_at(X,[Y|Xs],K,[Y|Ys]) :- K > 1,
   K1 is K - 1, remove_at(X,Xs,K1,Ys).
```

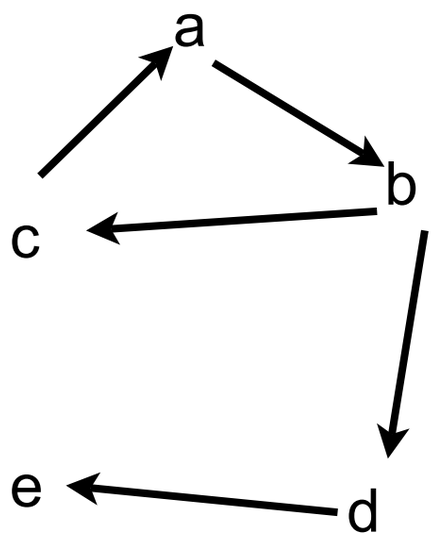
insert

insert_at(Element,L,K,ResultantList)

insert_at(X,L,K,R) :- remove_at(X,R,K,L).

Graphs

```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).
```

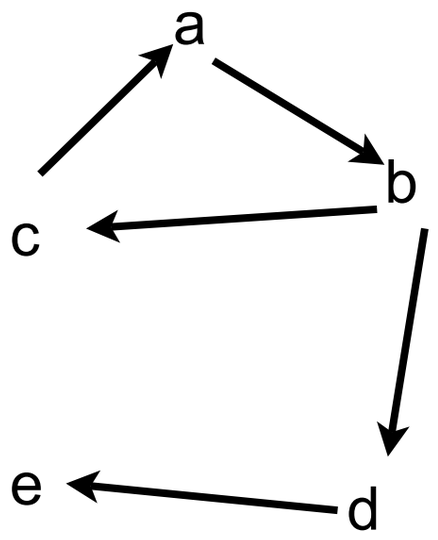



```
graph([a,b,c,d,e],[e(a,b),e(b,c),e(c,a), e(b,d),e(d,e)]).
```

```
[n(a,[b], n(b,[c,d]), n(c,[a]), n(d,[e]), n(e, [])]
```

```
[a-b, b-c, c-a, b-d, d-e]
```

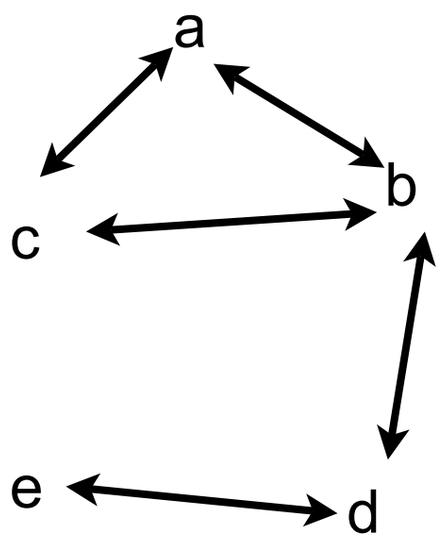
Search



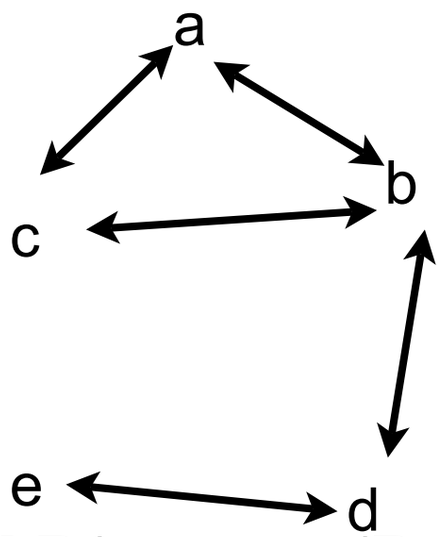

```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(X,Y) :- go(X,Y,[]), !.
go(X,X,T).
go(X,Y,T) :- edge(X,Z), legal(Z,T),go(Z,Y, [Z|T]).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```

Search

```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(X,Y) :- go(X,Y,[]), !.
go(X,X,T).
go(X,Y,T) :-
     (edge(X,Z);edge(Z,X)),
     legal(Z,T),
     go(Z,Y, [Z|T]).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```

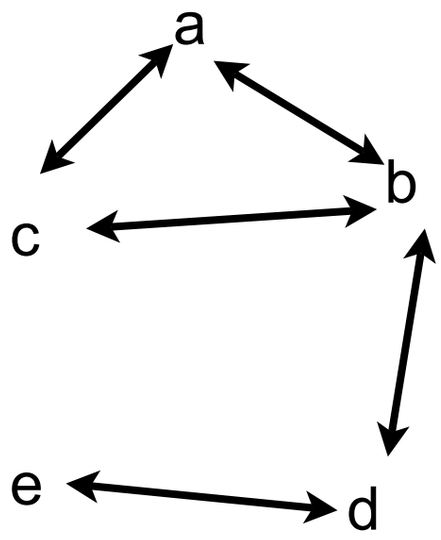


With Route




```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(Start,Destination,Route) :- go(Start,Destination,[],R ), reverse(R, Route).
go(X,X,T, [X|T]).
go(Place,Y,T,R) :-
      legal_node(Place,T, Next),
      go(Next,Y,[Place|T], R).

legal_node(X, Trail, Y):- (edge(X,Y);edge(Y,X)), legal(Y, Trail).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```

With Explicit Routes




```
edge(a,b).
edge(b,c).
edge(c,a).
edge(b,d).
edge(d,e).

go(Start,Destination,Route) :- go1([[Start]],Destination,R ), reverse(R, Route).

go1([First|Rest],Destination, First) :- First = [Destination|_].
go1([[Last|Trail]|Others],Destination,Route) :-
      findall([Z,Last|Trail],legal_node(Last,Trail,Z),List),
      append(List,Others, NewRoutes),
      go1(NewRoutes,Destination, Route).

legal_node(X, Trail, Y):- (edge(X,Y);edge(Y,X)), legal(Y, Trail).

legal(X,[]).
legal(X, [H|T]) :- \+X = H, legal(X,T).
```