Examples

A Sorted Tree Dictionary

Searching a Maze

Searching Graphs

Sorted Tree Dictionary

Need Associations between
items of information

Purpose:

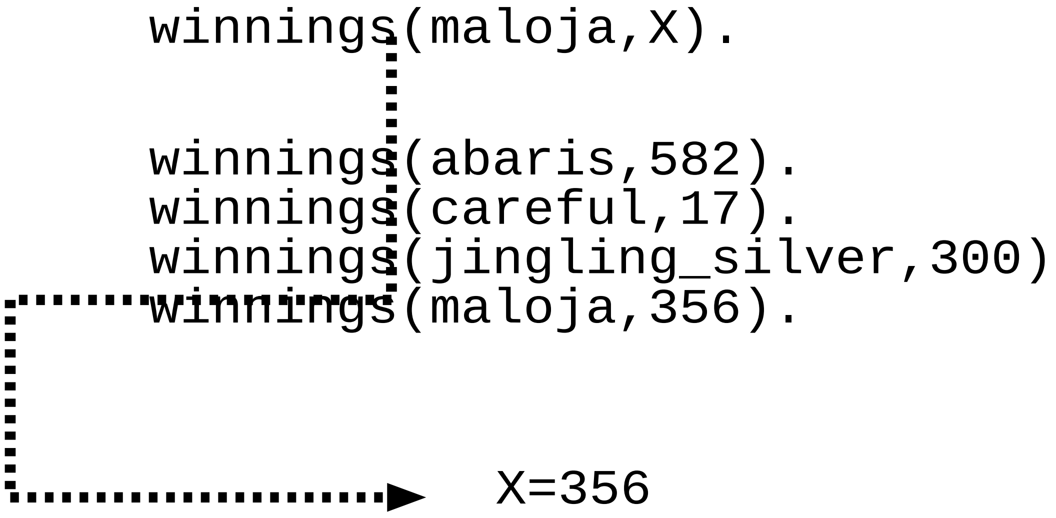
Retrieval

Dictionary

Associates word
with its definition
or
translation
or
with facts about it

Want to do this efficiently

Linear Search



```
winnings(maloja,X).


winnings(abaris,582).
winnings(careful,17).
winnings(jingling_silver,300)
winnings(maloja,356).
```

```
X=356
```

Data Search

If the database is large,
this linear search could be very inefficient

Another organization is needed
Give structure to the information

Sorted Tree

Sorted Tree

A set of connected nodes
forming a tree

Each node has information
about the nodes in the subtree

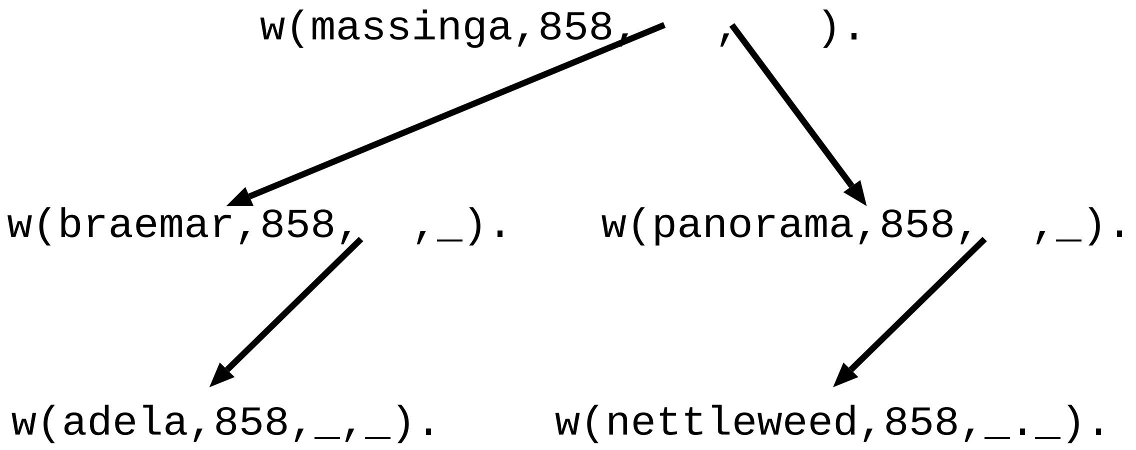
4 Components

Two associated items of information
as in `X` and `Y`
in `winnings`
Key: *The horse name*
Info: *The winnings*

3rd Element
A node with a key *less than* the current

4th Element
A node with a key *greater than* the current

Sorted Tree



```
w(massinga,858,  ,   ).

w(braemar,858,  ,_).    w(panorama,858,  ,_).

w(adela,858,_,_).    w(nettleweed,858,_._).
```

```
w(massinga,858,
    w(braemar,385,
       w(adela,588,_,_),
       _),
    w(panorama,158,
       w(nettleweed,579,_,_),
       _)
     ).
```

Program

```
lookup(H, w(H,G,_,_),G) :- !.

lookup(H, w(H1,_,Before,_), G) :-
     aless(H,H1),
     lookup(H,Before,G).

lookup(H, w(H1,_,_,After), G) :-
     not(aless(H,H1)),
     lookup(H,After,G).
```

With no data

```
| ?- lookup(ruby_vintage,X,582).
X = w(ruby_vintage,582,_B,_A) ? ;

| ?- lookup(ruby_vintage,X,582),
     lookup(maloja,X,356).
X = w(ruby_vintage,582,
             w(maloja,356,_C,_B),
             _A) ? ;



| ?- lookup(a,X,100),
     lookup(b,X,200),
     lookup(z,X,300),
     lookup(m,X,400).

X = w(a,100,_E,
        w(b,200,_D,
            w(z,300,w(m,400,_C,_B),
                        _A))) ? ;
```

Searching Mazes

Search for a telephone in a building

How do you search without getting lost?

How do you know that you have searched the whole building?

What is the shortest path to the telephone?

Steps

1

Go to the door of any room

2

If the room number in on the list (of already visited) ignore the room and go to step 1.

3

Add the room to the list

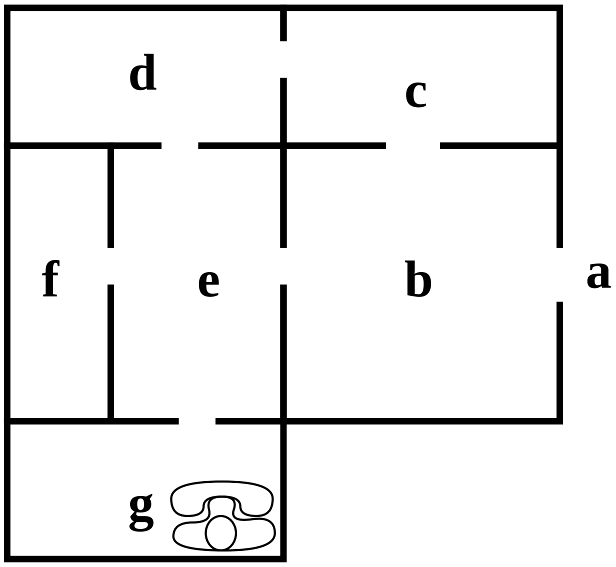
4

Look in the room for a telephone

5

If there is no telephone, go to step 1. Otherwise, we stop and out list has the path that we took to come to the correct room.

Maze



```
door(a,b).
door(b,e).
door(b,c).
door(c,d).
door(d,e).
door(e,f).
door(e,g).

hasphone(g).
```

Maze Program

When in a room:

We are in the room we want to be in

We have to pass through a door, and continue (recursively). We go into the other room if we have not been there yet (not on the list).

```
go(X,X,_).
go(X,Y,T) :- door(X,Z),
             write('Go into room'),
             write(Z),nl,
             not(member(Z,T)),
             go(Z,Y,[Z|T]).

go(X,Y,T) :- door(Z,X),
             write('Go into room'),
             write(Z),nl,
             not(member(Z,T)),
             go(Z,Y,[Z|T]).
```

Run

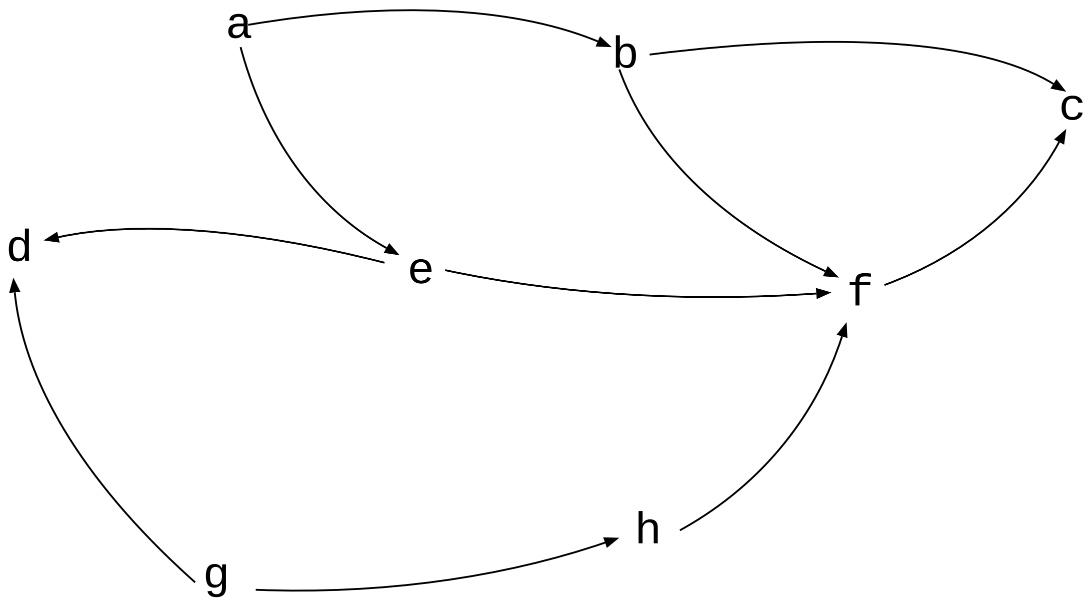
```
| ?-  hasphone(X),go(a,X,[]).
Go into room  b
Go into room  e
Go into room  f
Go into room  d
Go into room  c
Go into room  g

X = g ? ;
Go into room  c
Go into room  d
Go into room  e
Go into room  f
Go into room  g

X = g ? ;
Go into room  a

no
```

Graph Search



Moving Through Graph

```
go(X,X).
go(X,Y) :- a(X,Z),go(Z,Y).




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

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

Car Routes

```
a(newcastle,carlisle,58).
a(carlisle,penrith,23).
a(darlington,newcastle,40).
a(penrith,darlington,52).
a(workington,carlisle,33).
a(workington,penrith,39).

a(X,Y) :- a(X,Y,_).

go(Start,Dest,Route) :-
     go0(Start,Dest,[],.R).,
     rev(R,Route).

go0(X,X,T,[X|T]).
go0(Place,Dest,Route) :-
     legalnode(Place,T,Next),
     go0(Next,Y,[Place|T],R).

legalnode(X,Trail,,Y) :-
    (a(X,Y) ; a(Y,X)),
    legal(Y,Trail).

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

rev(L1,L2) :- revzap(L1,[],L2).

revzap([X|L],L2,L3) :-
     revzap(L,[X|L2],L3),
     revzap([],L,L).
```

Runs

```
| ?- go(darlington,workington,X).

X = [darlington,newcastle,
     carlisle,penrith,workington] ? ;

X = [darlington,newcastle,
     carlisle,workington] ? ;

X = [darlington,penrith,
     carlisle,workington] ? ;

X = [darlington,penrith,workington] ? ;

no
```

Findall

```
findall(X,G,_) :-
     asserta(found(mark)),
     call(G),
     asserta(found(X)),
     fail.
findall(_,_,L) :-
     collect_found([],M),
     !,
     L=M.

collect_found(S,L) :-
     getnext(X),
     !,
     collect_found([X|S],L).
collect_found(L,L).

getnext(X) :-
     retract(found(X)),
     !,
     X \== mark.
```

```
| ?-  findall(X,
          member(X,[a,b,c,d]),
          L).
L = [a,b,c,d] ? ;

| ?- findall(X,
        append(X,Y,[a,b,c,d]),
        L).
L = [[],
     [a],
     [a,b],
     [a,b,c],
     [a,b,c,d]] ? ;

| ?- findall([X,Y],
             append(X,Y,[a,b,c,d]),
             L).
L = [[[],[a,b,c,d]],
     [[a],[b,c,d]],
     [[a,b],[c,d]],
     [[a,b,c],[d]],
[[a,b,c,d],[]]] ? ;
```

Findall Paths

```
go(Start,Dest,Route) :-
     go0([[Start]],Dest,[],R),
     rev(R,Route).

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

Depth First

```
| ?- go(darlington,workington,X).

X = [darlington,newcastle,
     carlisle,penrith,workington] ? ;

X = [darlington,newcastle,
     carlisle,workington] ? ;

X = [darlington,penrith,
     carlisle,workington] ? ;

X = [darlington,penrith,
     workington] ? ;
```

Depth, Breadth First

```
go1([[Last|Trail]|Others],Dest,Route]
     :-
     findall([Z,Last|Trail],
             legalnode(Last,Trail,Z),
             List),
     append(List,Others,NewRoutes),
     go1(NewRoutes,Dest,Route).


go1([[Last|Trail]|Others],Dest,Route]
     :-
     findall([Z,Last|Trail],
             legalnode(Last,Trail,Z),
             List),
     append(Others,List,NewRoutes),
     go1(NewRoutes,Dest,Route).
```

Breath First

```
| ?- go(darlington,workington,X).

X = [darlington,penrith,
     workington] ? ;

X = [darlington,newcastle,
     carlisle,workington] ? ;

X = [darlington,penrith,
     carlisle,workington] ? ;

X = [darlington,newcastle,carlisle,
     penrith,workington] ? ;

n
```