Métodos de Pesquisa

Depth First Search
Breadth First Search
Best First Search

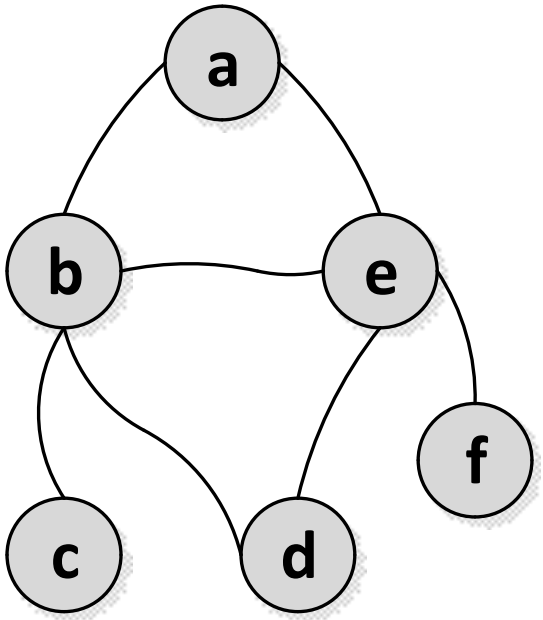
Depth First Search

Depth First Search (DFS)
Caracterização

Método de pesquisa em grafos/árvores não informado

- Ideia base: **Primeiro em profundidade**. Avançar no grafo/ árvore (em profundidade) enquanto for possível; se não for, **recuar (backtrack)** e tentar em alternativa outros caminhos
- **Estrutura de dados requerida muito simples e leve** (lista com nós visitados ou caminho actual)
- **Não garante caminho mais curto ou com menos passos em primeiro lugar**

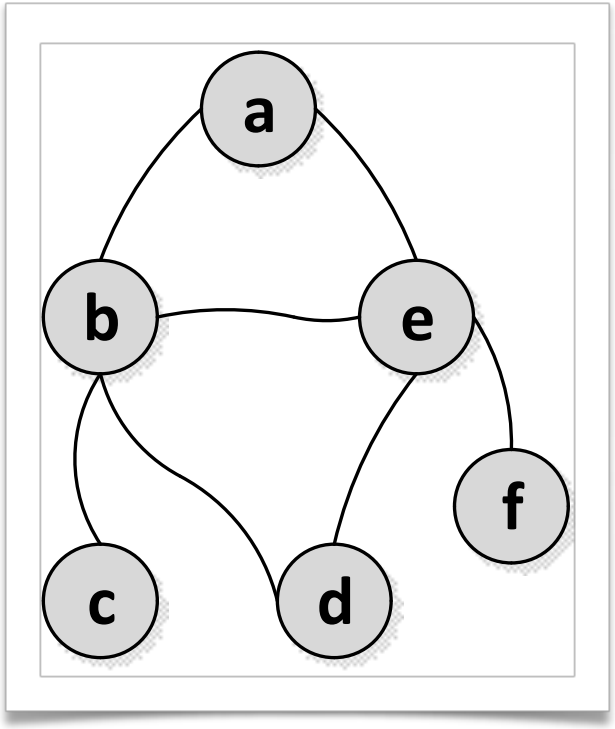
Depth First Search (DFS)
Grafo Exemplo

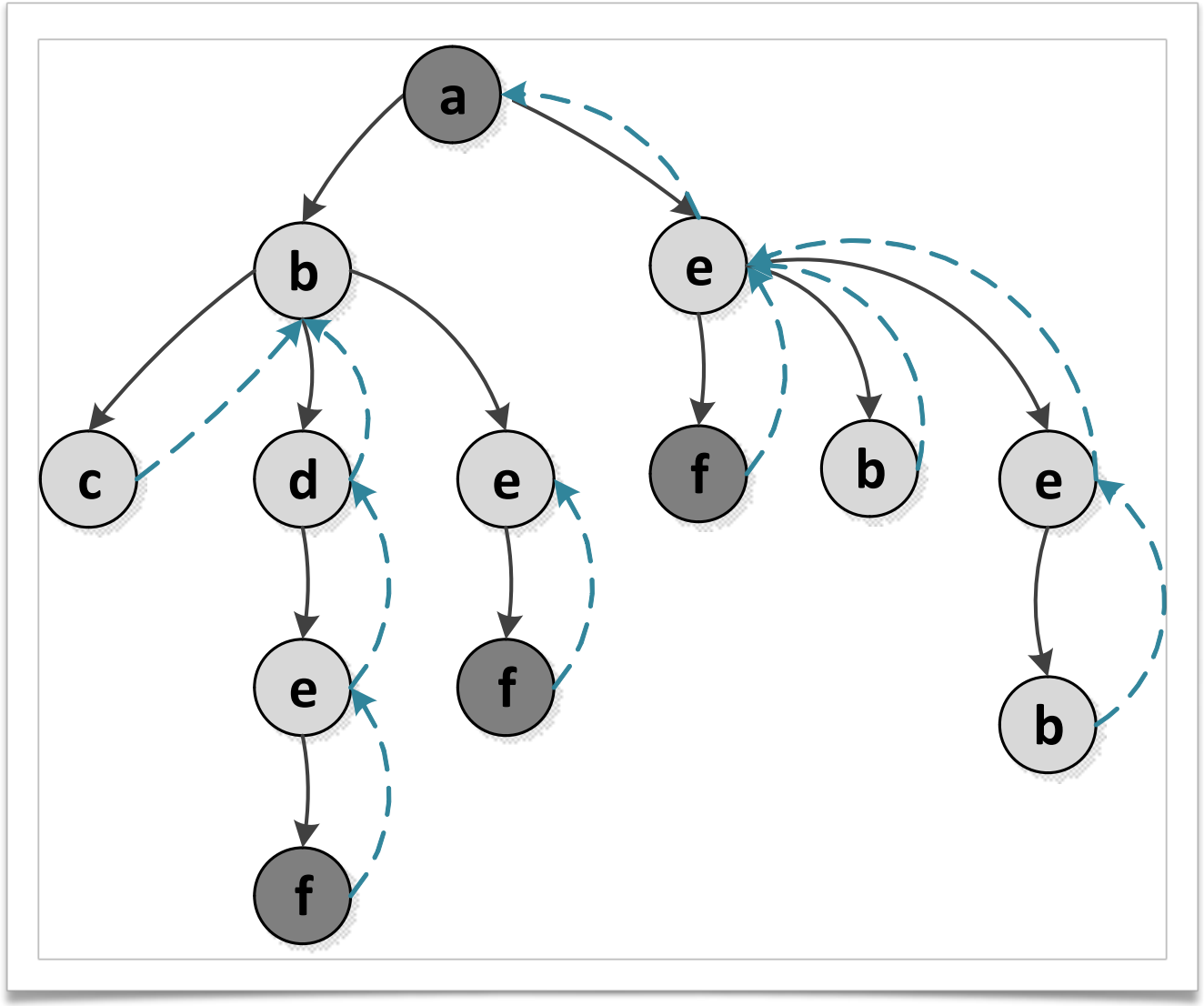



```
%edge(Node1,Node2)
edge(a,b).
edge(a,e).
edge(b,c).
edge(b,d).
edge(b,e).
edge(d,e).
edge(e,f).

% ligacoes bidireccionais
connect(X,Y):-
      edge(X,Y);edge(Y,X).
```

Depth First Search (DFS)

Implementação Prolog / Soluções





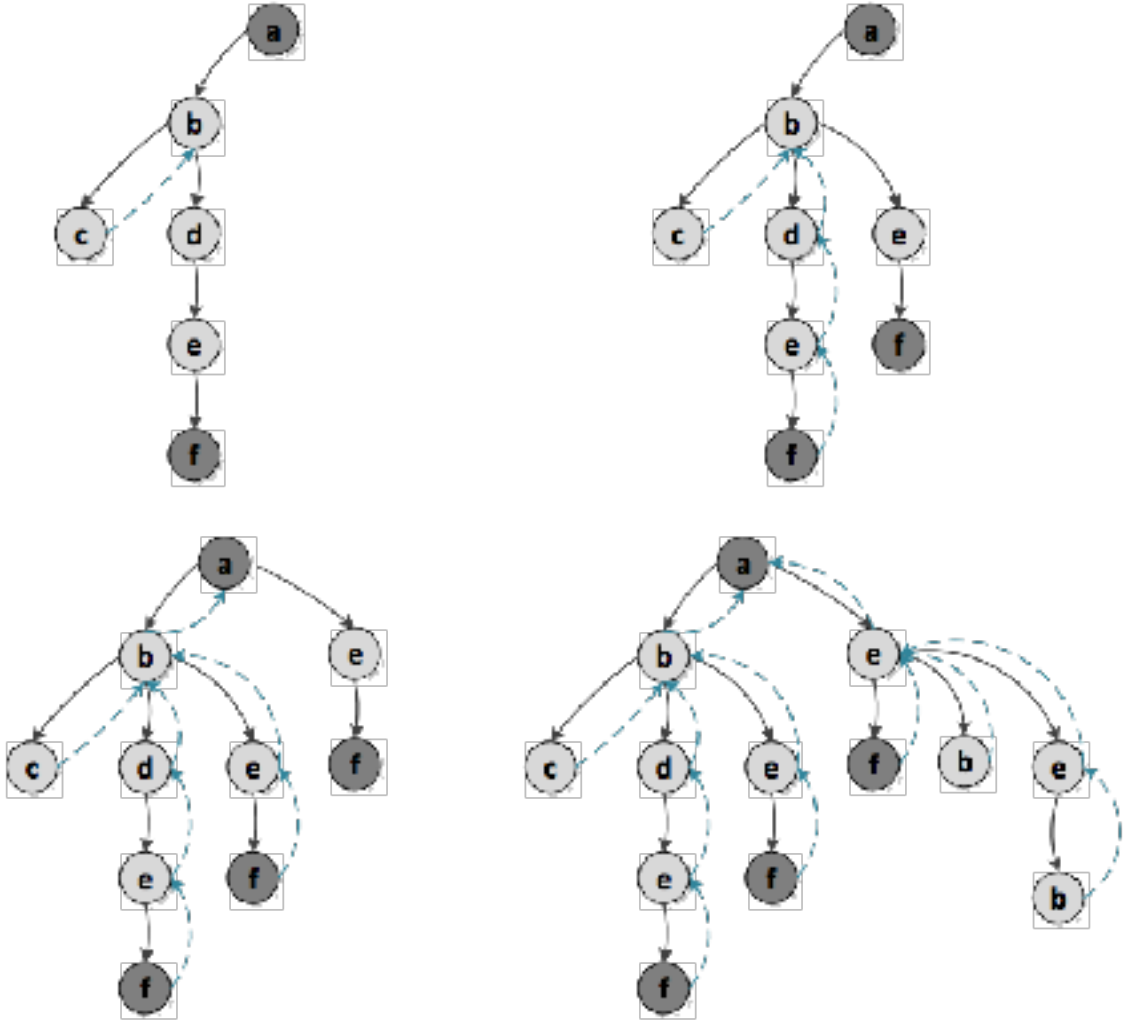
Depth First Search (DFS)

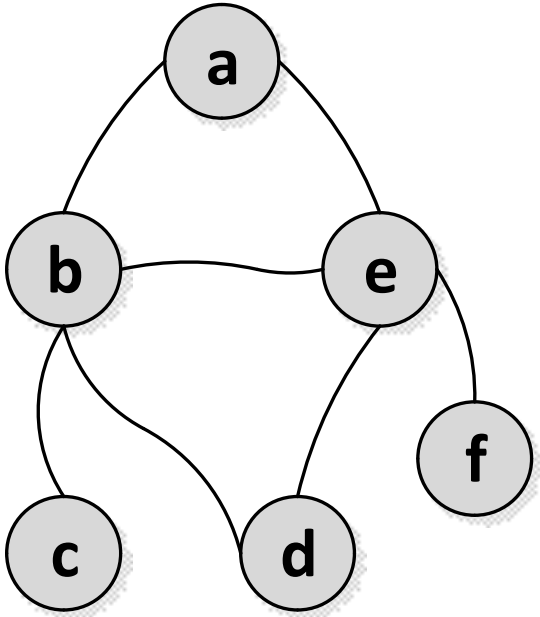
Implementação Prolog / Soluções

```
dfs(Orig,Dest,Cam):-
      dfs2(Orig,Dest,[Orig],Cam).

%condicao final: nó actual = destino
dfs2(Dest,Dest,LA,Cam):-
      %caminho actual esta invertido
      reverse(LA,Cam).

dfs2(Act,Dest,LA,Cam):-
      %testar ligacao entre ponto actual e um qualquer X
      connect(Act,X),

      %testar nao circularidade p/evitar nós ja visitados
      \+ member(X,LA),

      %chamada recursiva
      dfs2(X,Dest,[X|LA],Cam).
```

Depth First Search (DFS)

Espaço do problema / Soluções





Depth First Search (DFS)
Implementação Prolog / Soluções

```
dfs(Orig,Dest,Cam):-
      dfs(Orig,Dest,[Orig],Cam).

dfs(Dest,Dest,LA,Cam):-
      reverse(LA,Cam).

dfs(Act,Dest,LA,Cam):-
      connect(Act,X),
      \+ member(X,LA),
      dfs(X,Dest,[X|LA],Cam).
```

```
?- dfs(a,f,Caminho).
[a]
[b,a]
[b,a]
[d,b,a]
[e,d,b,a]
[f,e,d,b,a]
Caminho = [a, b, d, e, f] ;
[b,a]
[e,b,a]
[f,e,b,a]
Caminho = [a, b, e, f] ;
[e,b,a]
[a]
[e,a]
[f,e,a]
Caminho = [a, e, f] ;
[e,a]
[b,e,a]
[b,e,a]
[e,a]
[d,e,a]
[b,d,e,a]
false.
```

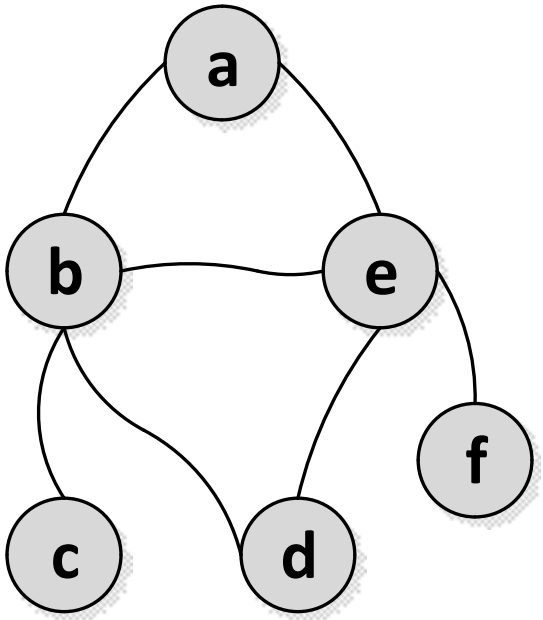
Breadth First Search

Breadth First Search (BFS)
Caracterização

Método de pesquisa em grafos/árvores não informado

- Ideia base: **Primeiro em largura.** A partir de um nó são explorados todos os nós adjacentes e só depois são explorados os nós acessíveis através dos adjacentes (nível seguinte) e assim sucessivamente
- **Estrutura de dados requerida pesada** pois é preciso armazenar todos os caminhos ainda por expandir (fila de caminhos)
- **Garante que o caminho com menos passos** é encontrado em primeiro lugar

Breadth First Search (BFS)
Grafo Exemplo




```
%edge(Node1,Node2)
edge(a,b).
edge(a,e).
edge(b,c).
edge(b,d).
edge(b,e).
edge(d,e).
edge(e,f).

% ligacoes bidireccionais
connect(X,Y):-
      edge(X,Y);edge(Y,X).
```

Breadth First Search (BFS)

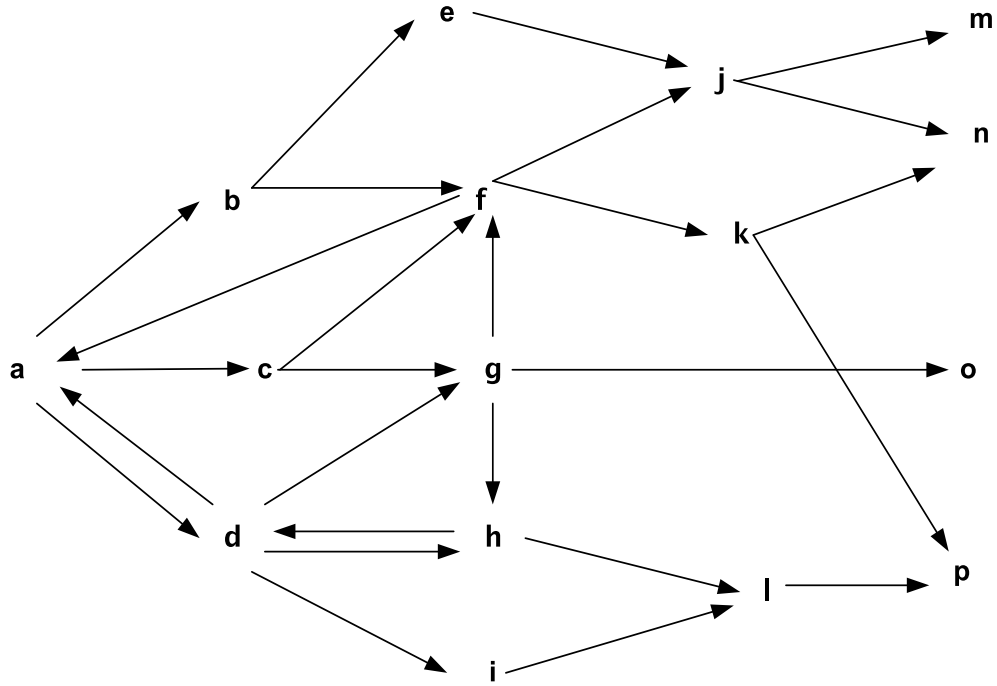
Implementação Prolog

```
bfs(Orig,Dest,Cam):-bfs2(Dest,[[Orig]],Cam).

%condicao final: destino = nó à cabeça do caminho actual
bfs2(Dest,[[Dest|T]|_],Cam):-
	%caminho actual está invertido
	reverse([Dest|T],Cam).

bfs2(Dest,[LA|Outros],Cam):-
	LA=[Act|_],
	% calcular todos os nós adjacentes não visitados e
	% gerar um caminho novo c/ cada nó e caminho actual
	findall([X|LA],
		(Dest\==Act,connect(Act,X),\+ member(X,LA)),
		Novos),
	%novos caminhos são colocados no final da lista p/ posterior
exploração
	append(Outros,Novos,Todos),
	%chamada recursiva
	bfs2(Dest,Todos,Cam).
```

Breadth First Search (BFS)

Lista de Percursos

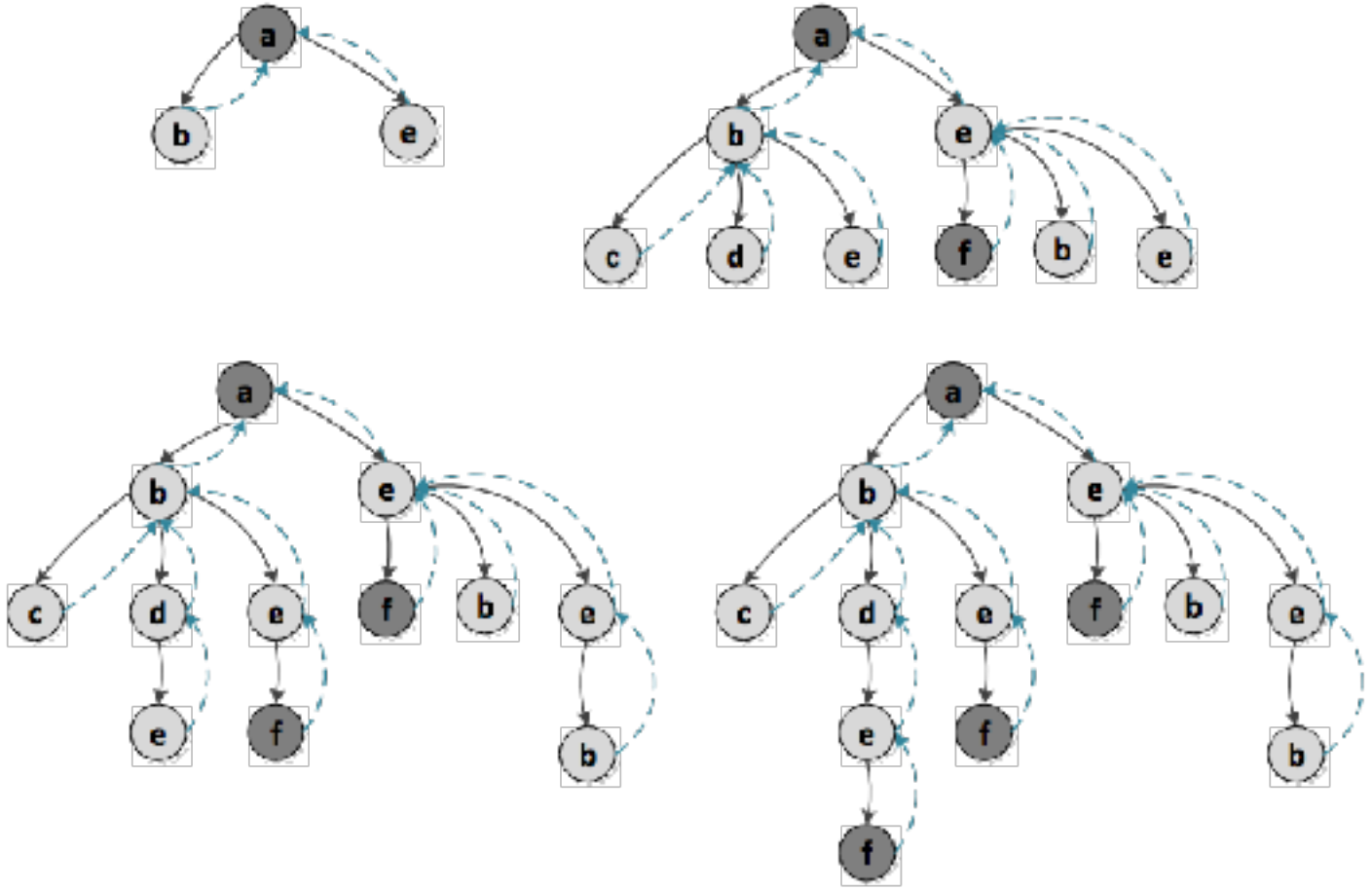
Lista de Percursos
após expansão do nó **a**

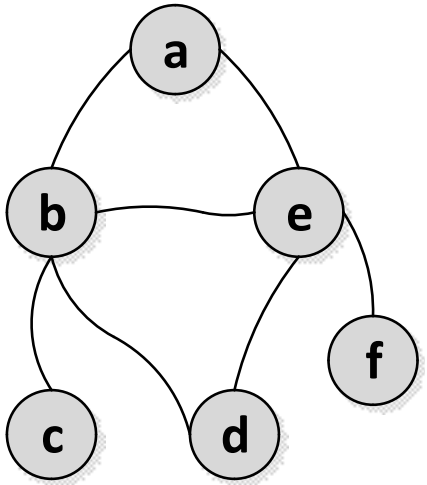


?- bfs(a,j,L).
[[b,a],[c,a],[d,a]]
[[c,a],[d,a],[e,b,a],[f,b,a]]
[[d,a],[e,b,a],[f,b,a],[f,c,a],[g,c,a]]
...
[[i,d,a],[j,e,b,a],[j,f,b,a],[k,f,b,a],[j,f,c,a],[k,f,c,a],[f,g,c,a],[o,g,c,a], ...]
[[j,e,b,a],[j,f,b,a],[k,f,b,a],[j,f,c,a],[k,f,c,a],[f,g,c,a],[o,g,c,a],[h,g,c,a],[f,g,d,a], ...]
L = [a,b,e,j]

Percursos resultantes da expansão do nó **b**

Breadth First Search (BFS)
Espaço do Problema / Soluções





Depth First Search (DFS)

Implementação Prolog / Soluções

```
bfs(Orig,Dest,Cam):-
       bfs2(Dest,[[Orig]],Cam).

bfs2(Dest,[[Dest|T]|_],Cam):-
       reverse([Dest|T],Cam).


bfs2(Dest,[LA|Outros],Cam):-
       LA=[Act|_],

       findall([X|LA],
       Dest\==Act,connect(Act,X),
       \+member(X,LA)),Novos),

       append(Outros,Novos,Todos),
       bfs2(Dest,Todos,Cam).
```

```
?- bfs(a,f,Caminho).
[a]
[b,a]
[e,a]
[c,b,a]
[d,b,a]
[e,b,a]
[f,e,a]
Caminho = [a, e, f] ;
[f,e,a]
[b,e,a]
[d,e,a]
[e,d,b,a]
[f,e,b,a]
Caminho = [a, b, e, f] ;
[f,e,b,a]
[d,e,b,a]
[c,b,e,a]
[d,b,e,a]
[b,d,e,a]
[f,e,d,b,a]
Caminho = [a, b, d, e, f] ;
[f,e,d,b,a]
[c,b,d,e,a]
false.
```

Depth First Search (DFS)

Usando Implementação do BFS com alteração

```
bfs(Orig,Dest,Cam):-bfs2(Dest,[[Orig]],Cam).

%condicao final: destino = nó à cabeça do caminho actual
bfs2(Dest,[[Dest|T]|_],Cam):-
        %caminho actual está invertido
        reverse([Dest|T],Cam).

bfs2(Dest,[LA|Outros],Cam):-
        LA=[Act|_],
        %calcular todos os nós adjacentes nao visitados e
        %gerar um caminho novo c/ cada nó e caminho actual
        findall([X|LA],
                (Dest\==Act,connect(Act,X),\+ member(X,LA)),
                Novos),
        %novos caminhos são colocados no inicio da lista
        %p/ exploracao   imediata
        append(Novos, Outros, Todos),
        %append(Outros, Novos, Todos),

        %chamada recursiva
        bfs2(Dest,Todos,Cam).
```

Best First Search

Best First Search (BestFS)

Caracterização

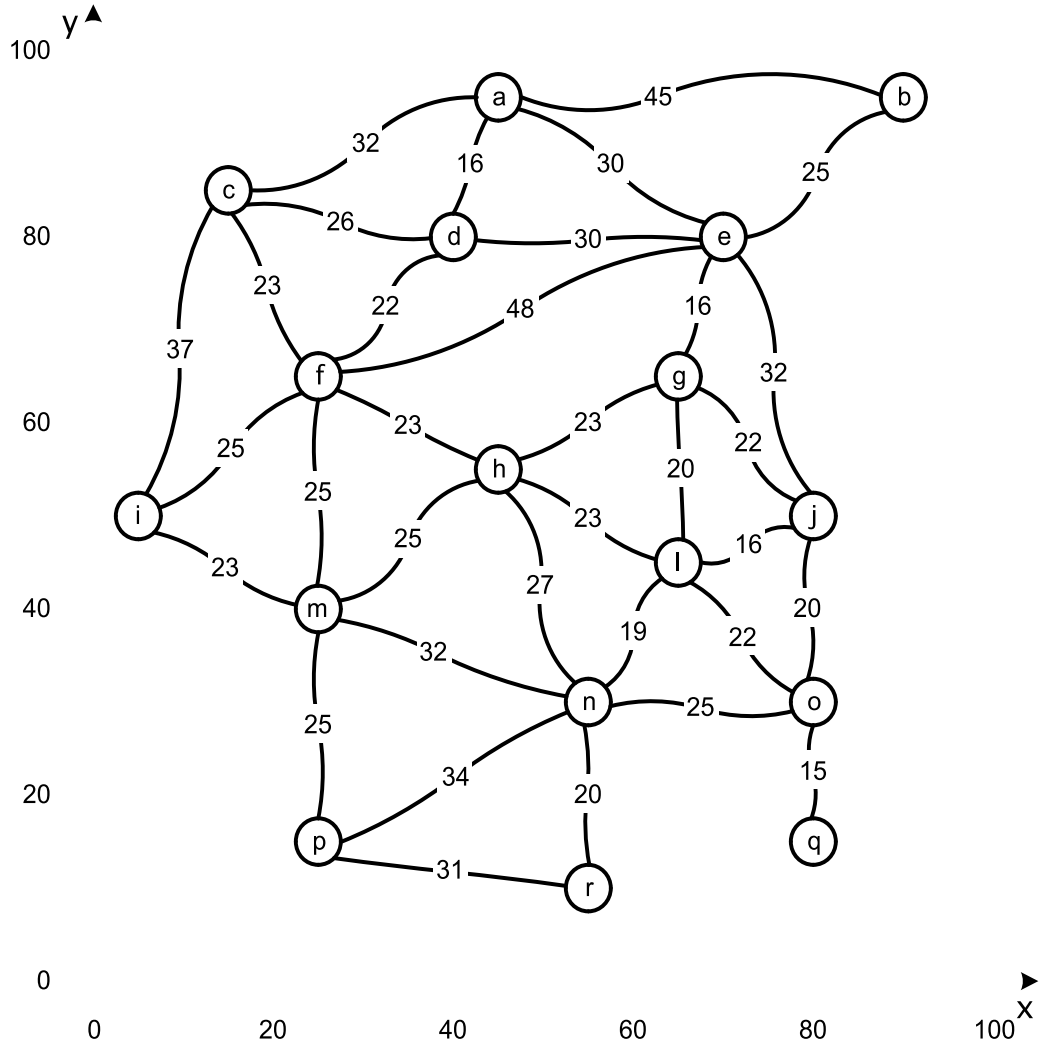
Método de pesquisa em grafos/árvores informado

- Ideia base: similar ao DFS mas a decisão sobre qual o nó a explorar de seguida é feita com base **num critério de decisão local**
- **Solução final é resultado da soma dos máximos locais** e como tal pode não ser o máximo global
- **No caso de produzir solução, esta é obtida muito rapidamente** pois não são exploradas múltiplas soluções; não existe backtracking
- **Não garante que caminho com menos passos ou mais barato** seja encontrado em primeiro lugar; pode mesmo não gerar qualquer solução

Best First Search (BestFS)

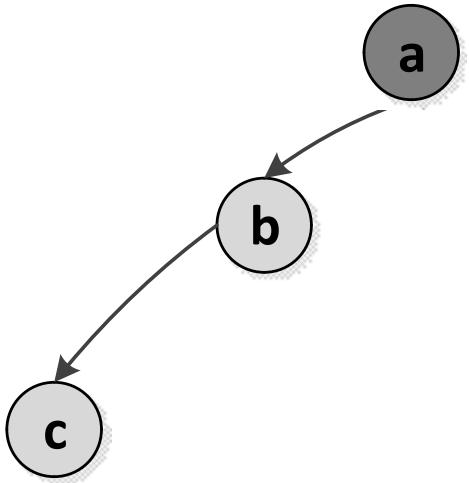
Implementação Prolog

```
bestfs(Orig,Dest,Cam):-
        bestfs2(Dest,[Orig],Cam).

%condicao final: destino = nó à cabeça do caminho actual
bestfs2(Dest,[Dest|T],Cam):- !,
        %caminho actual está invertido
        reverse([Dest|T],Cam).

bestfs2(Dest,LA,Cam):-
        LA=[Act|_],
        %calcular todos os nós adjacentes nao visitados e
        % guardar um tuplo com estimativa e novo caminho
        findall((EstX,[X|LA]),
                (connect(Act,X),\+ member(X,LA), estimativa(X,Dest,EstX)),Novos),
        %ordenar pela estimativa
        sort(Novos,NovosOrd),
        %extrair o melhor que está à cabeça
        NovosOrd = [(_,Melhor)|_],
        %chamada recursiva
        bestfs2(Dest,Melhor,Cam).
```


y
x
100
80
60
40
20
0
0
20
40
60
80
100
a
b
c
d
e
f
g
h
i
j
l
m
n
o
p
q
r
45
32
16
30
25
26
30
23
22
48
16
37
32
25
23
23
22
20
25
25
23
16
23
25
27
20
19
22
32
25
25
34
20
15
31

Breadth First Search (BFS)
Espaço do Problema / Soluções

Estimativa nula: método fica identico ao DFS mas sem backtracking pelo que não produz soluções



Situação em que estimativa e->f melhor do que b->f