4) Sottografo dei predecessori prodotto da BFS su $G = \langle V, E \rangle$:

$G_\pi = \langle V_\pi, E_\pi \rangle$ , dove

$V_\pi \Rightarrow \{ v \in V : v.\pi \neq nil \ / \ \{s\} \} \rightarrow$ ogni vertice raggiungibile da s, che ha quindi un $\pi$.

$E_\pi \Rightarrow \{ (v, \pi.v) : v \in \underline{V_\pi - \{s\}} \} \rightarrow$ ogni arco che porta ad s.

BFS $(G, v)$:

\# preparo i nodi:

```
for v ∈ V - {s} :
    v.color = "white" ;          O(V)
    v.parent = nil ;
    v.distance = ∞ ;
```

\# setting sorgente:                 $O(V) \rightarrow$ enqueue e dequeue di ogni vertice.

```
s.color = "gray" ;
s.parent = ∅ ;
s.distance = 0 ;
Enqueue (G, s) ;
```

\# esplorazione:

```
while (G ≠ ∅) :
    x = Dequeue (q) ;
    for all  y = Adj (x) :  ←——— O(E)      totale  O(V+E) ;
        if (y.color == "white") :
            y.color = "gray"
            y.π = x ;
            y.d = x.d + 1 ;
            Enqueue (y) ;
    x.color = "Black" ;
```

* Sottografo dei predecessori DFS:

$G = (V_\pi, E_\pi)$ :

$E_\pi = \{ (v.\pi, v) : v \in V \land v.\pi \neq nil \}$

Foresta Depth First i cui archi sono $E_\pi$


Archi three :

Archi back :

Archi Fw :

Archi Cross :


DFS:

DFS (G) :

\# preparo i nodi
for all $v \in V$ :          O (V)
| v.color = white ;
| v.$\pi$ = NIL ;
| v            .
|              .
|              .
   time = 0 ;
\# per ogni nodo chiamo DFS-VISITi
   for $v \in V$ :              O (V)
      if (v = "white") :
            DFS-VISIT (G,v) ;


\# chiamo DFS-VISIT :                    tempo: O(V + E)

DFS-VISIT ( G, v ) :

      v.color = "gray" ;
      v.time = time + 1 ;
      for [all x = adj (v) ] :   $\Theta (E)$

```
        if (x.color = "white");
              x.π = v;
              DFS-VISIT (G,x);
    v.color = "Black";
    v.find = time+s;
```

Acyclic :


for    all    v ∈ V
      v.col = white
      v.π = nil;

for    all    v ∈ V
          if    (v == white):
                DFS-VISIT (w)


DFS-VISIT (w):
      v.color = "gray";
      v.d = time.ts;

      For    all x ∈ Adj [v]
          if (x.color == "white")
              x.π = v;
              (TRUE ←
      return(DFS-VISIT (w));          ————→    else if (x.color = gray)
                                                return    false;
      v.color = "Black".
      v.f = time+s;
          return    true;
```