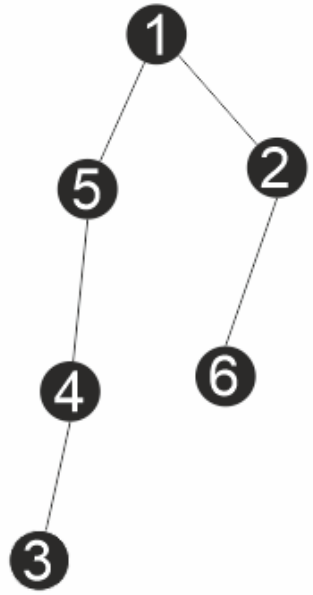
Recorridos en profundidad

Usaremos en esta ocasión, para este recorrido, una gestión tipo pila para el conjunto de nodos pendientes de ser explorados, ToDo. Usando una pila, el recorrido se denomina, en profundidad (*dfs, deep first search,* en inglés).
Por tanto, el esquema del recorrido pasa a ser:

Recorrido en profundidad: usando marcas de visitas y pila de nodos por explorar

```
{inicialización}
Para todo nodo v,
        visitado[v] = falso;
{preparamos el inicio del recorrido desde el nodo i}
Visitado[i] = verdadero;
MeterenlaPila(ToDo, i);
{bucle principal}
Mientras PilaNoVacía(ToDo) hacer
                k = SacaDePila(ToDo);
                Para todo adyacente j de k hacer
                        Si visitado[j] = falso entonces
                                Visitado[j]=verdadero;
                                MeteEnPila(ToDo, j);
```



$\Gamma_1 = \{5, 2\}$

$\Gamma_2 = \{1, 6\}$

$\Gamma_3 = \{4\}$

$\Gamma_4 = \{5, 3\}$

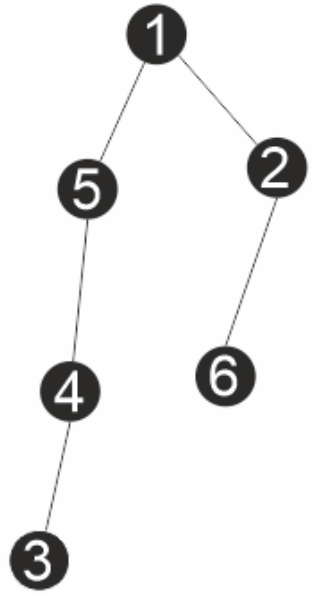
$\Gamma_5 = \{1, 4\}$

$\Gamma_6 = \{2\}$

```
Mientras PilaNoVacía(ToDo) hacer
        k = SacaDePila(ToDo);
        Para todo adyacente j de k hacer
            Si visitado[j] = falso entonces
                Visitado[j]=verdadero;
                MeteEnPila(ToDo, j);
```

ToDo

- Comenzamos desde el nodo 1, visitado y apilado en ToDo
- Sacamos k=1 de la pila, y revisamos su adyacencia {5, 2}
- El nodo 5 no está visitado, visitamos y apilamos
- El nodo 2 no está visitado, visitamos y apilamos
- Sacamos k=2 de la pila y revisamos su adyacencia {1, 6}
- El nodo 1 ya ha sido visitado
- EL nodo 6 no está visitado, visitamos y apilamos
- El nodo 5 espera pues está debajo de la pila
- Sacamos k=6 de la pila y revisamos la adyacencia {2}
- El nodo 2 ya ha sido visitado
- Sacamos k=5 de la pila y revisamos su adyacencia {1,4}
- El nodo 1 ya ha sido visitado
- El nodo 4 no está visitado, visitamos y apilamos
- Sacamos k=4 de la pila y revisamos la adyacencia {5, 3}
- El nodo 5 ya ha sido visitado
- El nodo 3 no está visitado, visitamos y apilamos
- Sacamos k=3 de la pila y revisamos la adyacencia {4}
- El nodo 4 ya ha sido visitado
- Pila vacía: FIN




```
Mientras PilaNoVacía(ToDo) hacer
        k = SacaDePila(ToDo);
        Para todo adyacente j de k hacer
            Si visitado[j] = falso entonces
                Visitado[j]=verdadero;
                MeteEnPila(ToDo, j);
```

$$\Gamma_1 = \{5, 2\}$$
$$\Gamma_2 = \{1, 6\}$$
$$\Gamma_3 = \{4\}$$
$$\Gamma_4 = \{5, 3\}$$
$$\Gamma_5 = \{1, 4\}$$
$$\Gamma_6 = \{2\}$$

Conclusiones

- El orden que la gestión de ToDo como una pila hace que visitemos los nodos en el siguiente orden:

$$1 < 2, 6 < 5, 4, 3$$

Es decir: prioriza desplazarse por las ramas hasta finalizarlas, y luego regresa para recorrer otra rama. En este caso, finalizó primero la rama **1, 2, 6** y luego la **1, 5, 4, 3**.

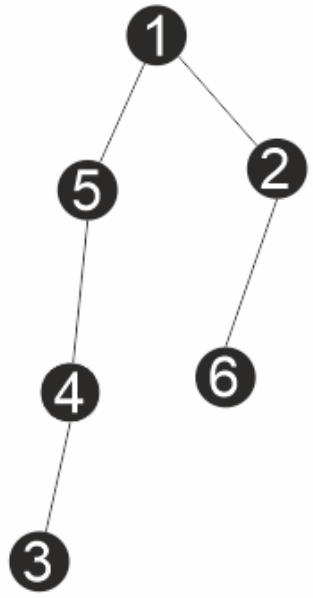
Recorridos en profundidad

Sin embargo, la expresión más común de un recorrido es la versión recursiva, en donde la pila de llamadas recursivas sustituye a la pila ToDo y se simplifica. El esquema es el siguiente:

```
dfs (k, &visitado, &enum, &onum)

visitado[k] = TRUE;
prenum[enum++] = k;

{bucle principal}
Para todo j adyacente a k hacer
        Si visitado[j] = FALSE entonces
                dfs(j, visitado, enum, onum);
postnum[onum++]=k
```

Y para un recorrido en profundidad desde el nodo i, tenemos:

```
{inicialización}
para k = 1 hasta n hacer
        visitado[k] = FALSE
onum = 0; enum = 0;

dfs (i, visitado, enum, onum)
```



$\Gamma_1 = \{5, 2\}$

$\Gamma_2 = \{1, 6\}$

$\Gamma_3 = \{4\}$

$\Gamma_4 = \{5, 3\}$

$\Gamma_5 = \{1, 4\}$

$\Gamma_6 = \{2\}$

```
dfs (k, var visitado, var enum, var onum)
visitado[k] = TRUE;
prenum[enum++] = k;
{bucle principal}
Para todo j adyacente a k hacer
        Si visitado[j] = FALSE entonces
                dfs(j, visitado, enum, onum);
postnum(onum++)=k
```

enum

1	5	4	3	2	6

prenum

onum

3	4	5	6	2	1

postnum

dfs(1)

- Visitamos el nodo k=1, actualizamos prenum, y analizamos su adyacencia {5, 2}
- El nodo 5 no ha sido visitado
    - dfs(5)
        - dfs(4)
            - dfs(3)
- El nodo 2 no ha sido visitado
    - dfs(2)
        - Visitamos el nodo k=2, actualizamos prenum y analizamos su adyacencia {1, 6}
        - El nodo 1 ya ha sido visitado
        - El nodo 6 no ha sido visitado
            - dfs(6)
                - Visitamos el nodo k=6, actualizms prenum y analizms la adyacencia {2}
                - El nodo 2 ya ha sido visitado
                - Actualizamos postnum
        - Actualizamos postnum
- Actualizamos postnum
- FIN