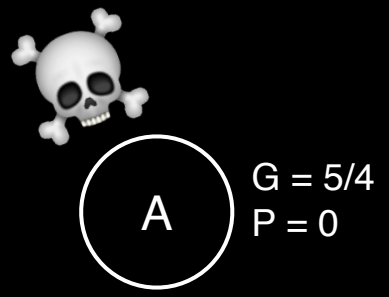
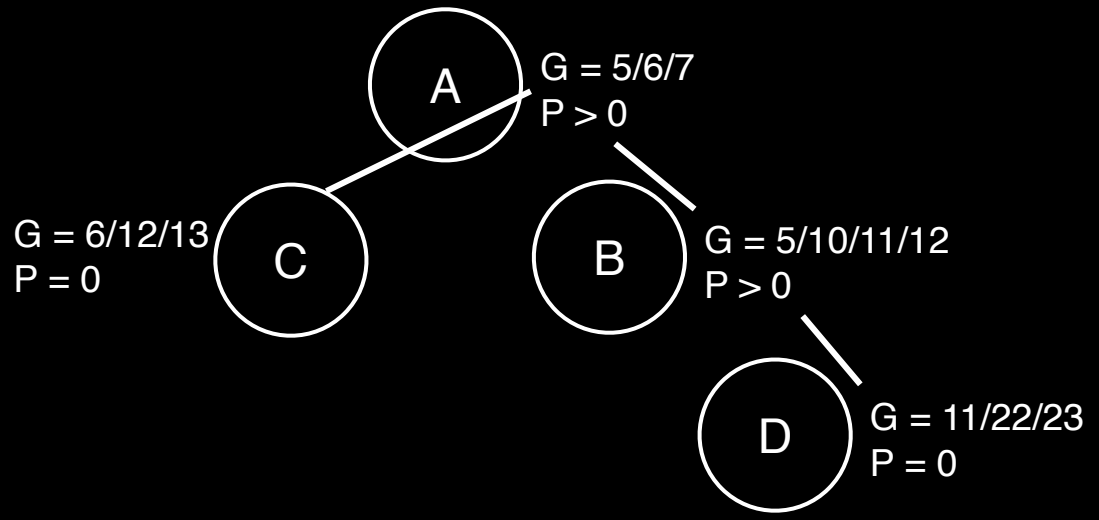
```c
int glob = 5;
int pid = 0;
    glob --;

    if (pid != 0)
    {
        pid = fork();
        glob --;
    }
```

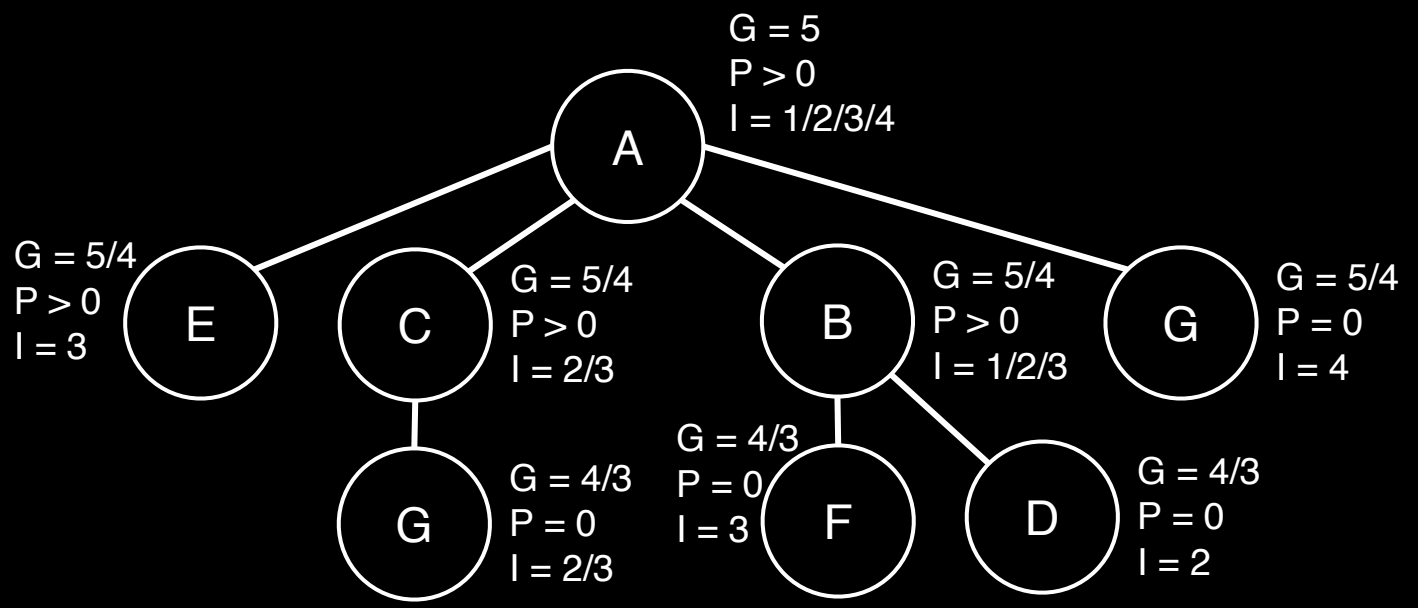


```c
int glob = 5;
int pid = 0;
    int i = 0;
    for (i = 1; i < 3; i ++)
    {
        pid = fork();
        if (pid == 0)
        {
            glob = glob * 2;
            sleep (i + 1);
        }

        glob = glob + 1;
    }
```



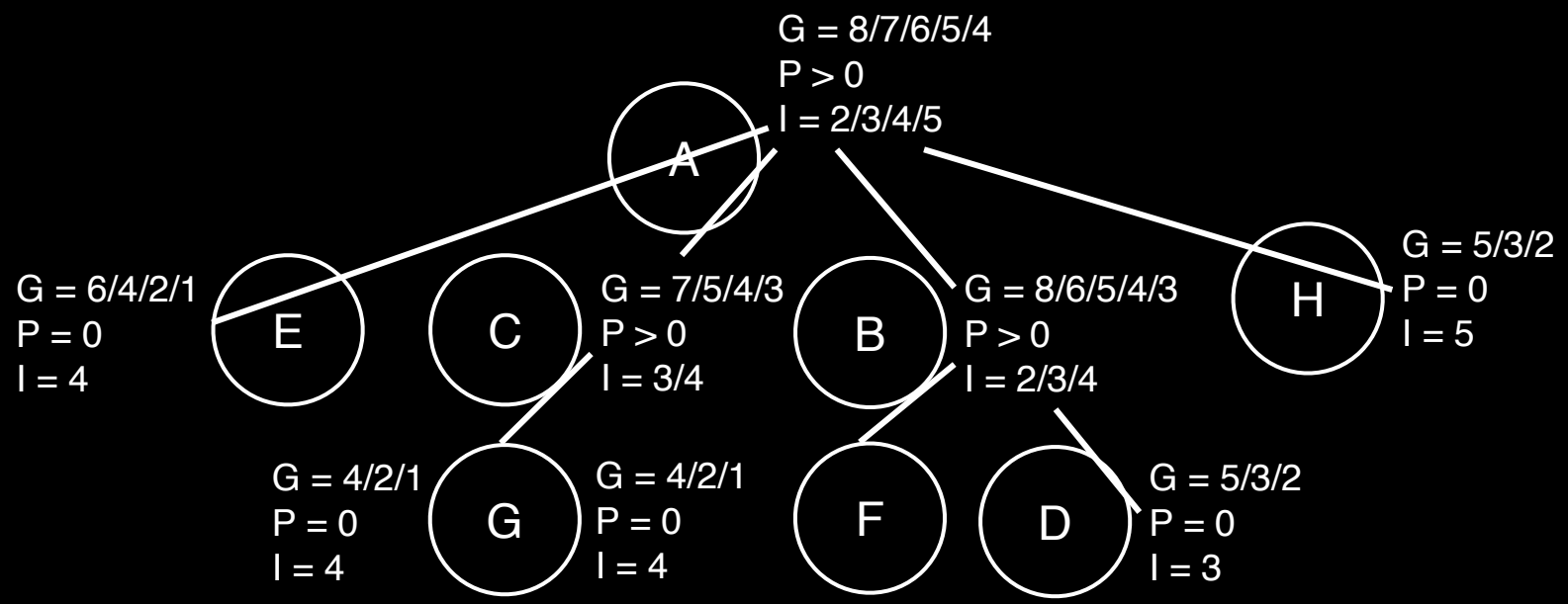
```c
int glob = 5;
    pid_t pid;
    int i;
    for (i = 1; i < glob; i
++)
    {
        pid = fork();
        if (pid == 0)
        {
            glob = glob - 1;
        }
    }
```

I = 1/2/3/4/5

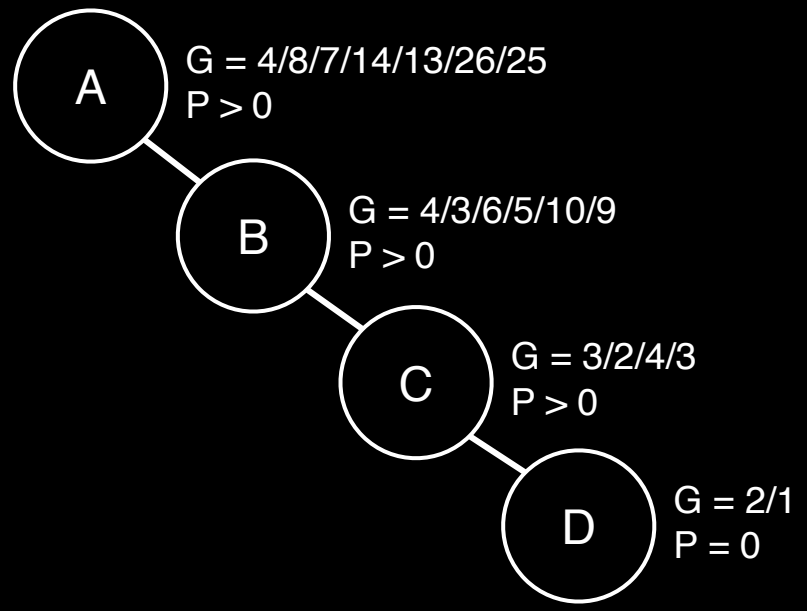


```c
int glob = 8;
int pid = 0;
    int i;
    for (i = 2; i <= glob; i ++)
    {
        pid = fork();
        if (pid == 0)
        {
            glob = glob - 2;
        }

        glob --;
    }
```

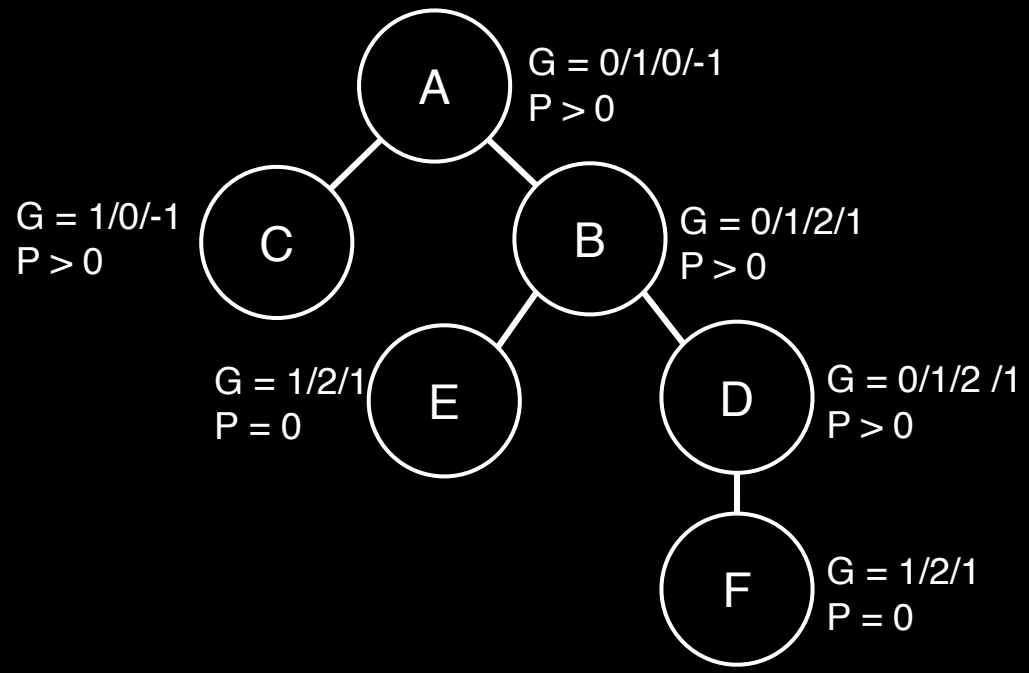
I = 2/3/4/5



```c
int glob = 4;
int pid = 0;
    int i;
    for (i = 1; i < 4; i ++)
    {
        if (pid == 0)
        {
            pid = fork();
        }

        if (pid != 0)
        {
            glob = glob * 2;
        }

        glob = glob - 1;
    }
```

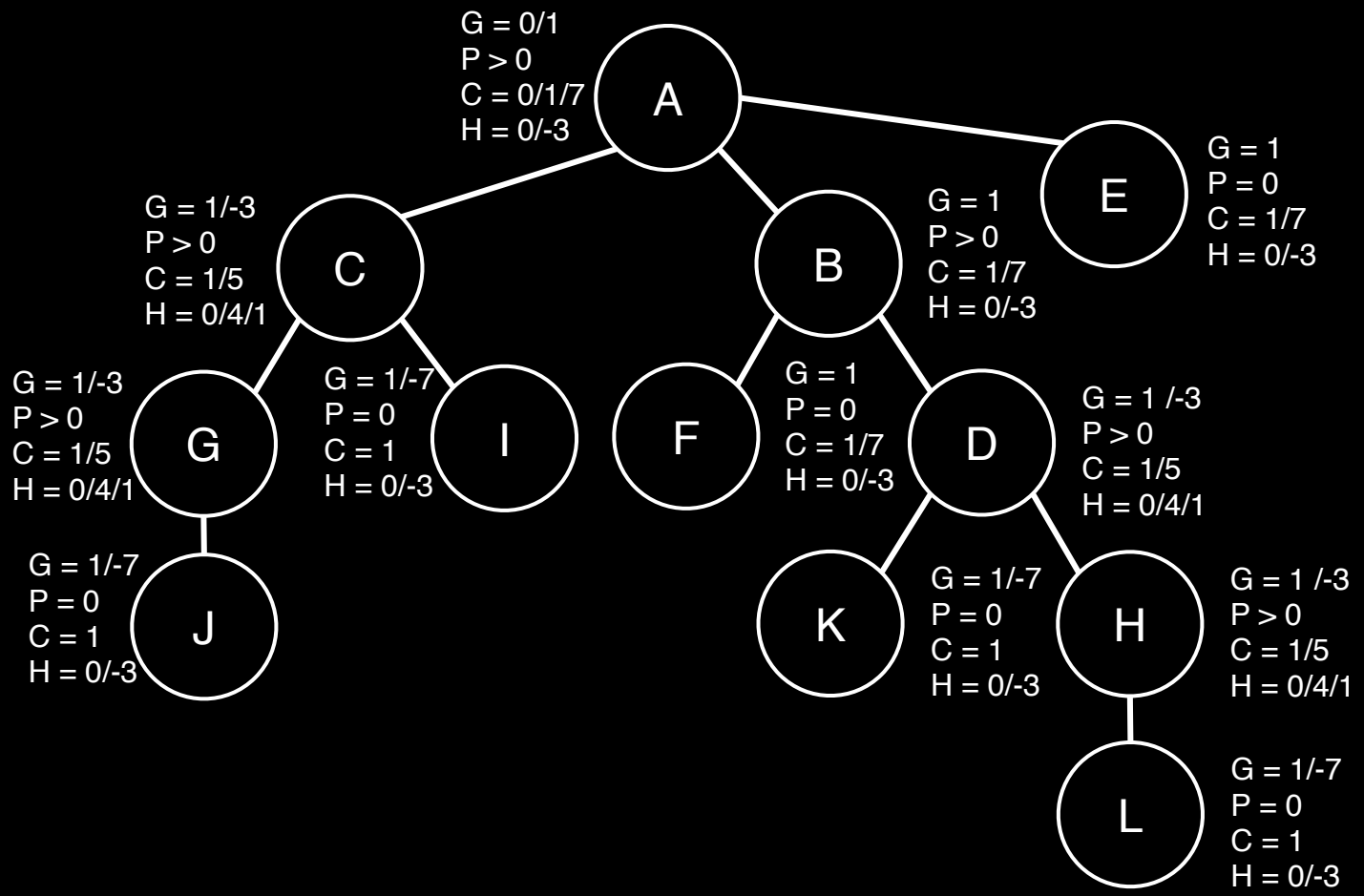


```c
int glob = 0;
    int i;
    int pid = 0;
    pid = fork();
    if (pid != 0)
    {
        glob++;
        pid = fork();
        glob--;
    }
    else if (pid == 0)
    {
        for (i = 1; i <= 2; i +
+)
        {
            pid = fork();
            glob++;
        }
    }
    glob--;
```

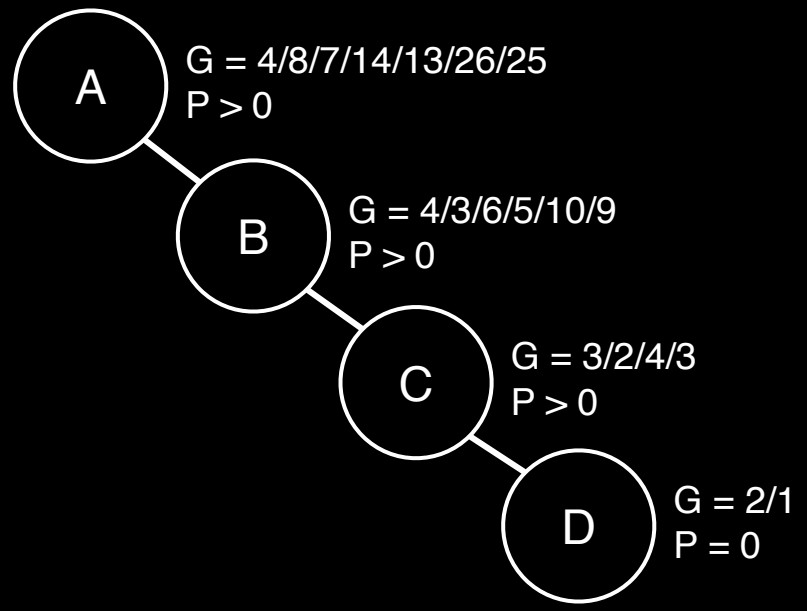


•Nel primo fork(), vengono creati due processi: il **padre** e il **figlio**. Il padre esegue il blocco if (pid != 0), mentre il figlio esegue il blocco else if (pid == 0).
•Nel padre, dopo il primo incremento di glob++, viene eseguito un secondo fork(), che crea un **nuovo figlio**. Questo nuovo figlio eredita il codice che segue il fork() nel padre, cioè eseguirà il glob-- che si trova subito dopo il fork().
•**Il nuovo figlio** generato dal secondo fork() non entrerà mai nel blocco else if (pid == 0) perché in quel momento la variabile pid (il valore restituito dalla seconda chiamata a fork()) non sarà uguale a 0 per lui. Invece, sarà uguale a 0 solo per il figlio generato dalla prima chiamata a fork().

```c
int glob = 0;
int cocc = 0;
int hug = 0;
    int i = 0;
    int pid = 0;
    glob++;
    cocc++;
    pid  = fork();
    glob = cocc;
    cocc = glob;
    pid  = fork();
    if (pid != 0)
    {
        for (i = 1 ; i < 2;
i++)
            pid = fork();

        cocc = glob + 2 * 3;
    }
    else if (pid == 0)
    {
        for (i = 2; i <= 3;
i++)
            pid = fork();

        if(pid != 0)
        {
            hug = 4;
            cocc = cocc * 5;
        }
        glob = cocc - 2 * 4;
```



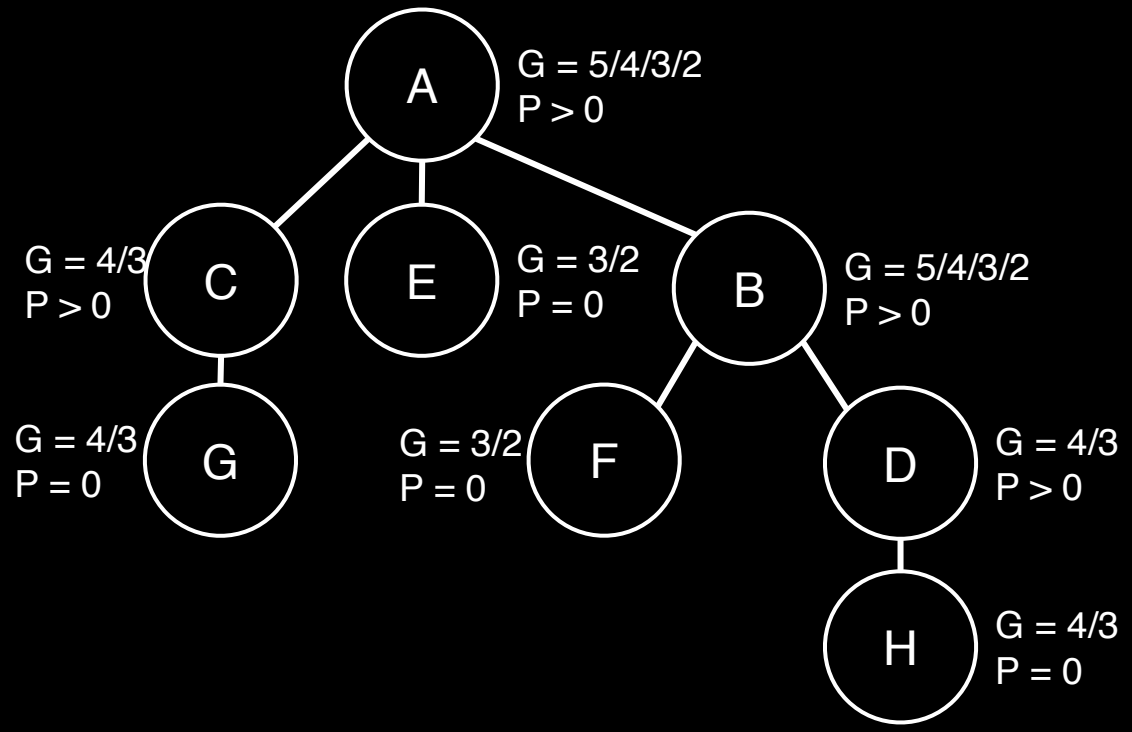
```c
int glob = 4;
int pid = 0;
    int i;
    for (i = 1; i < 4; i ++)
    {
        if (pid == 0)
        {
            pid = fork();
        }

        if (pid != 0)
        {
            glob = glob * 2;
        }

        glob = glob - 1;
    }
```



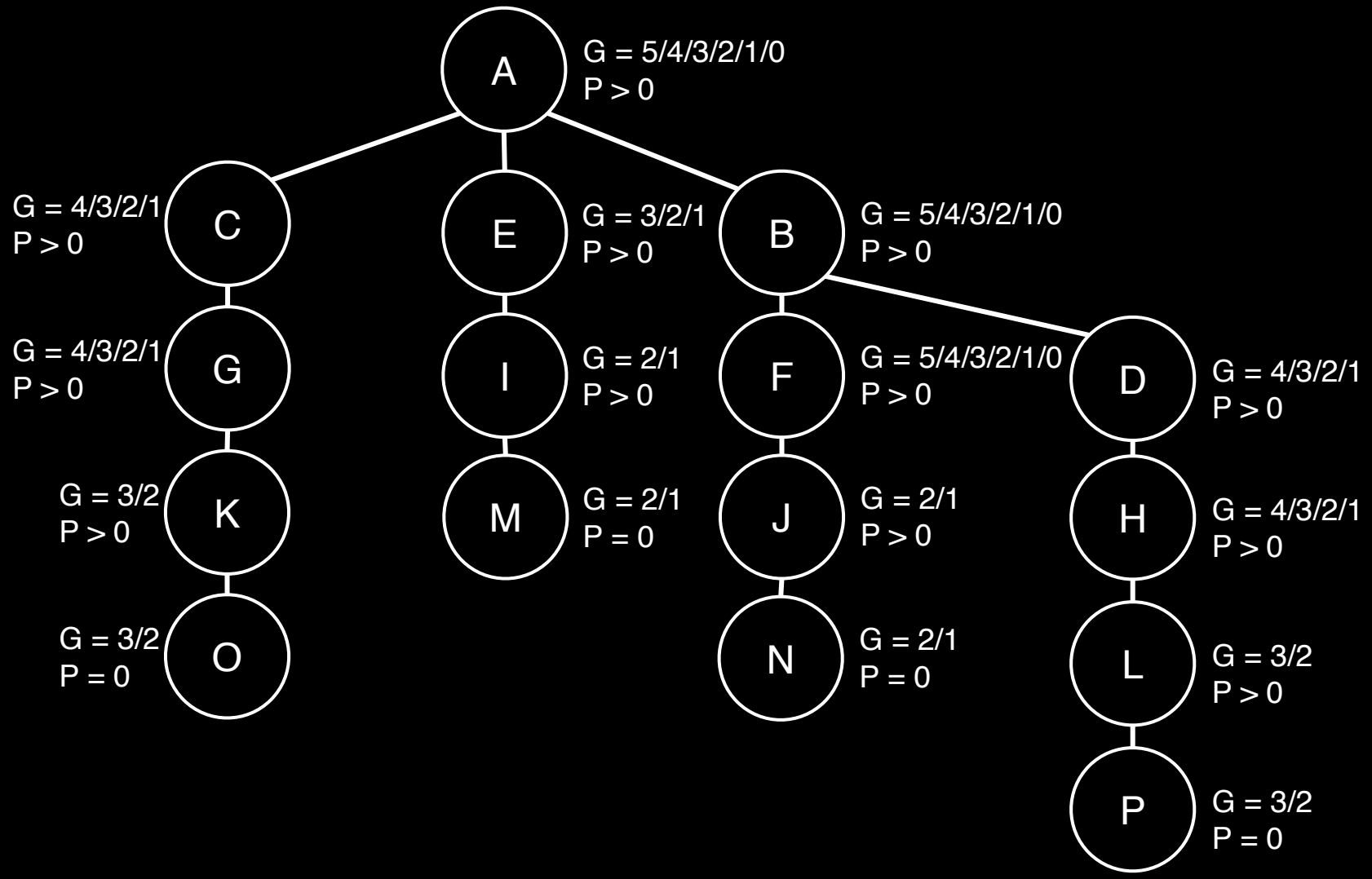
Mi rifiuto di fare 32 processi

Il 10 non c’è

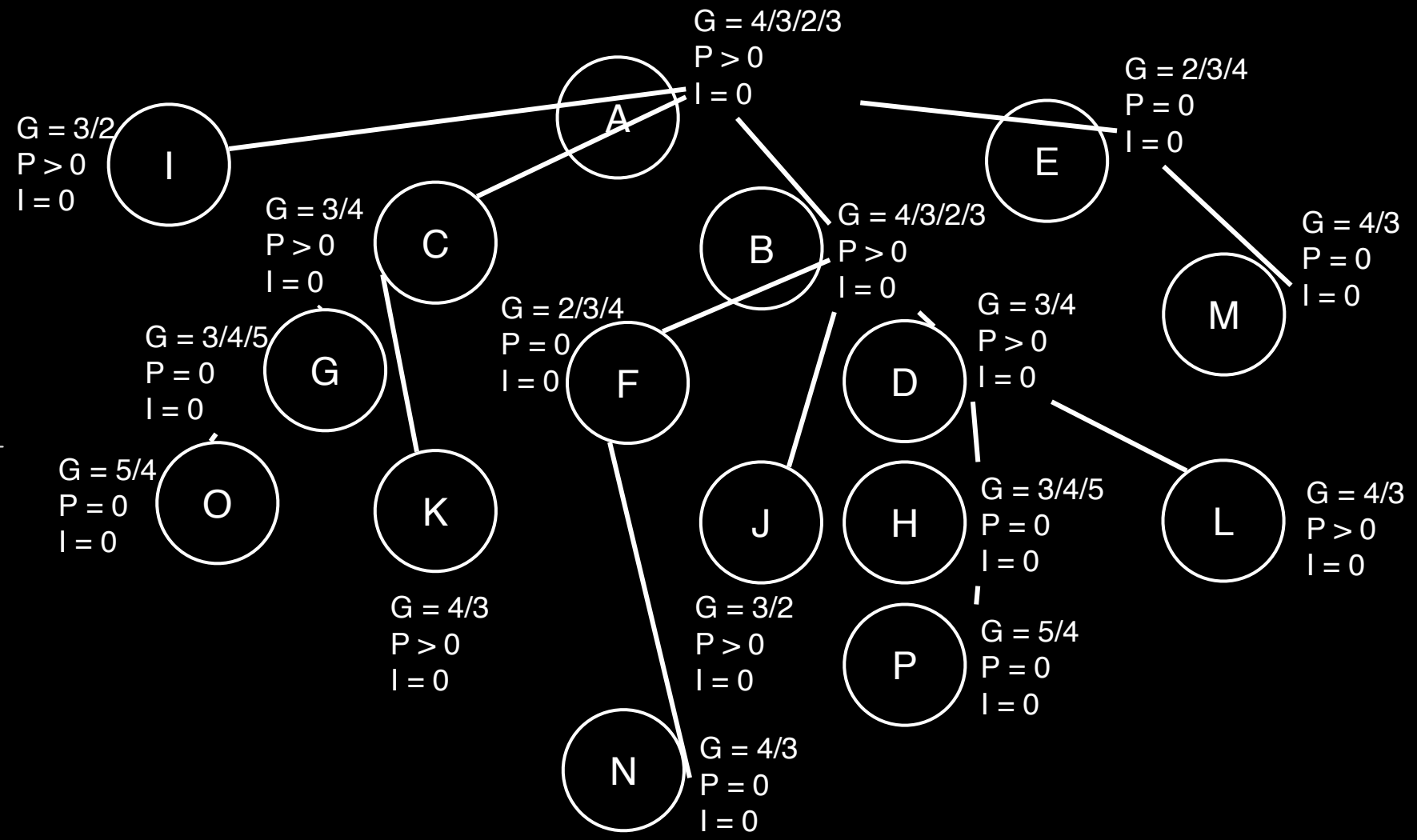
```c
int glob = 5;
int pid = 0;
    fork();
    glob--;

    if (fork())
        glob--;

    if (pid == 0)
    {
        pid = fork();
        glob--;
    }
```



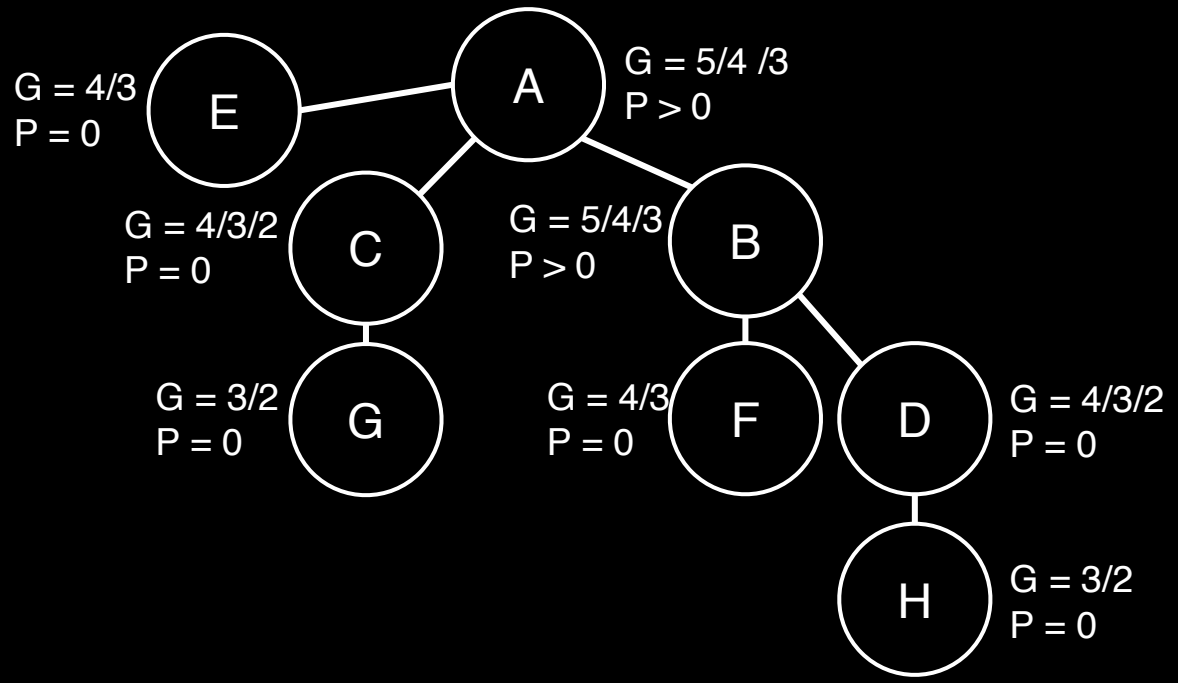
```c
int glob = 5;
int pid = 0;
int i = 0;
    fork();
    glob--;

    if (fork())
        glob--;

    if (pid == 0)
    {
        pid = fork();
        glob--;
    }
    for (i = 0; i < 2; i+
+)
    {
        if (pid == 0)
            pid = fork();
        if (pid != 0)
            glob--;
    }
```



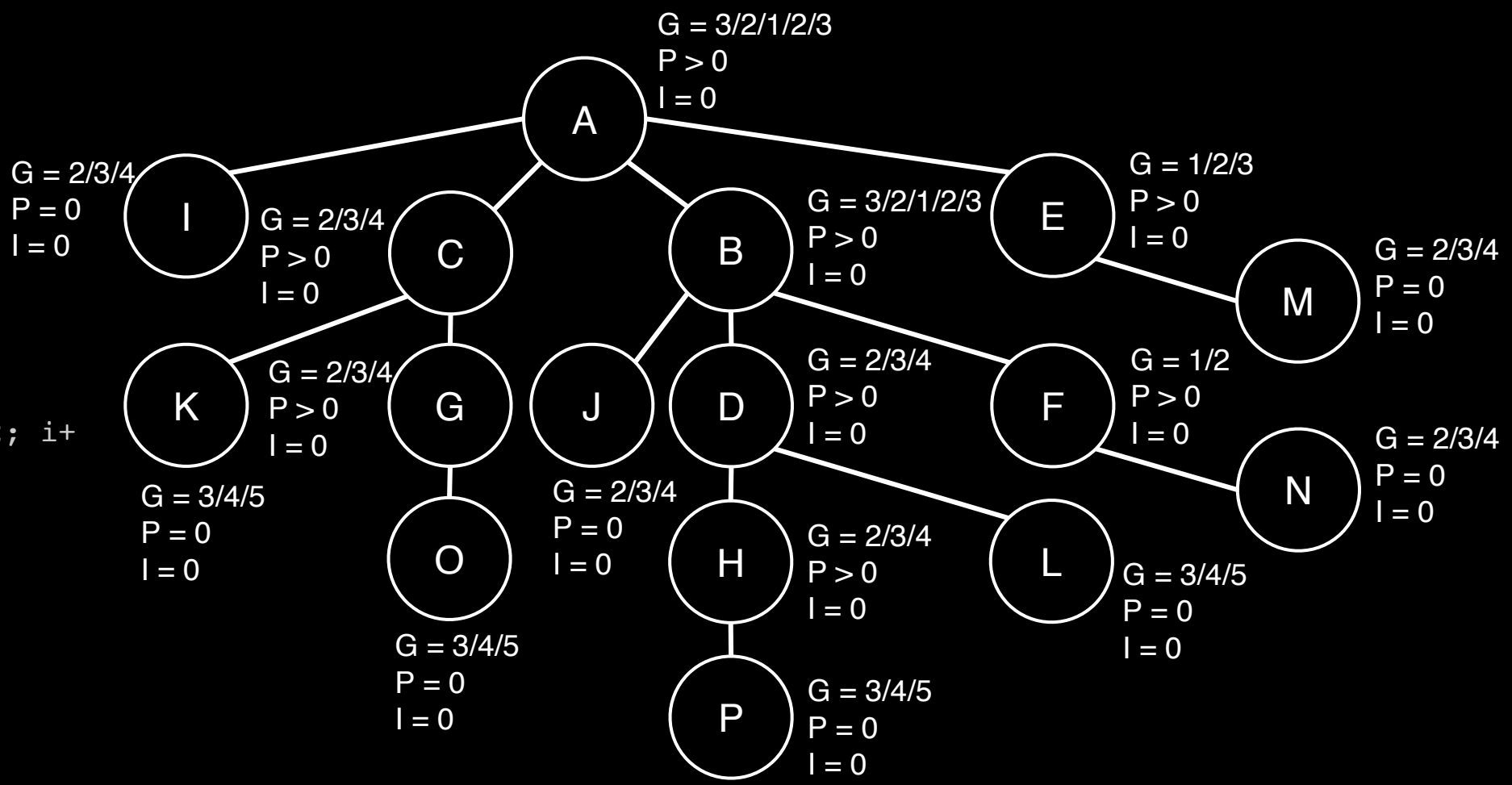
```c
int glob = 4;
int pid = 0;
int i = 0;
    fork();
    glob--;
    if (fork())
        glob--;
    if (pid == 0)
    {
        for (i = 1; i < 2; i+
+)
        {
            pid = fork();
            glob++;
        }
    }
    if (pid == 0)
        glob++;
    if (!fork())
        glob--;
```



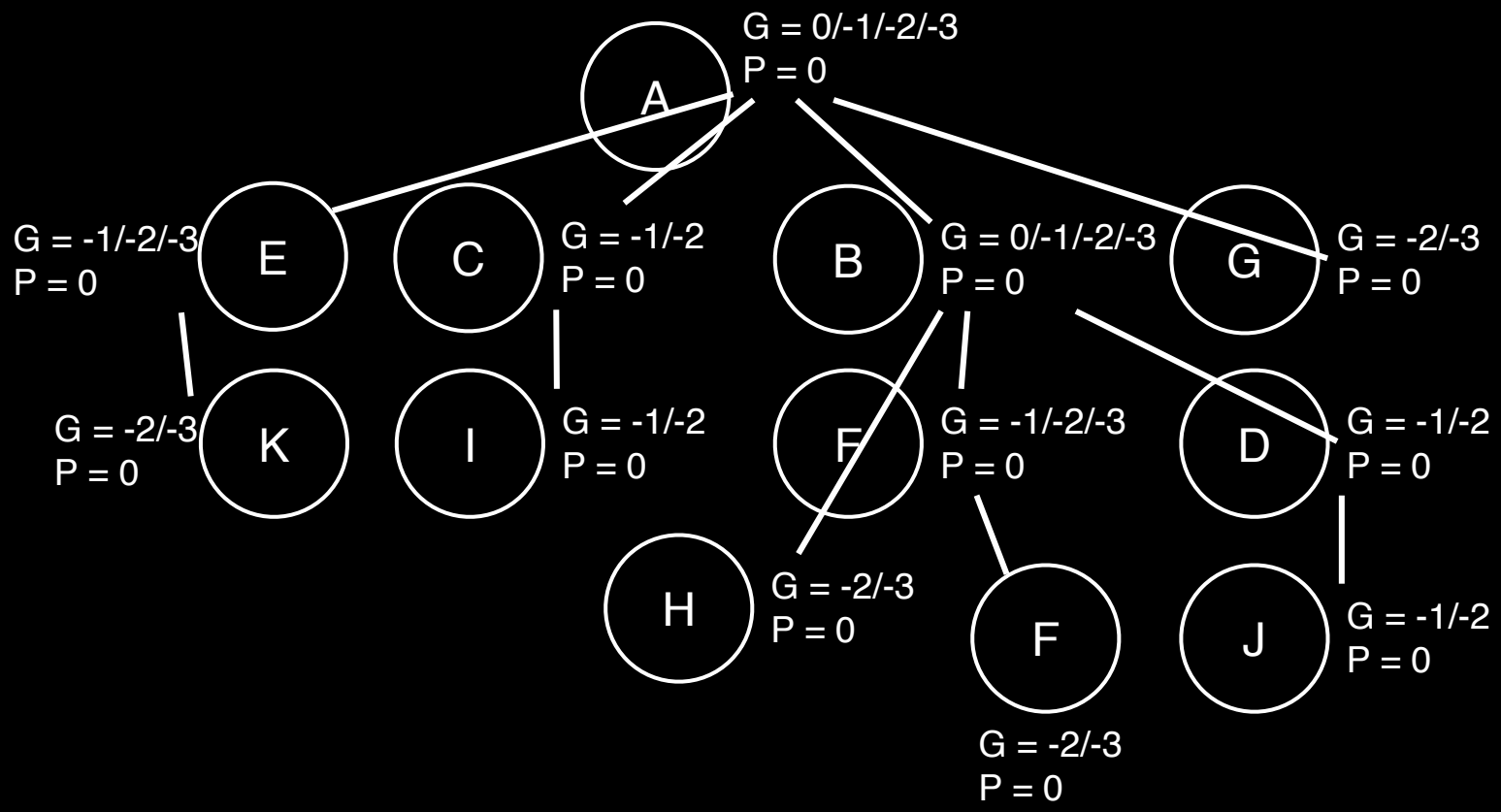
```c
int glob = 5;
int pid = 0;
    fork();
    glob--;
    if (!fork())
    {
        glob--;
    }
    if (pid == 0)
    {
        pid = fork();
        glob--;
    }
```



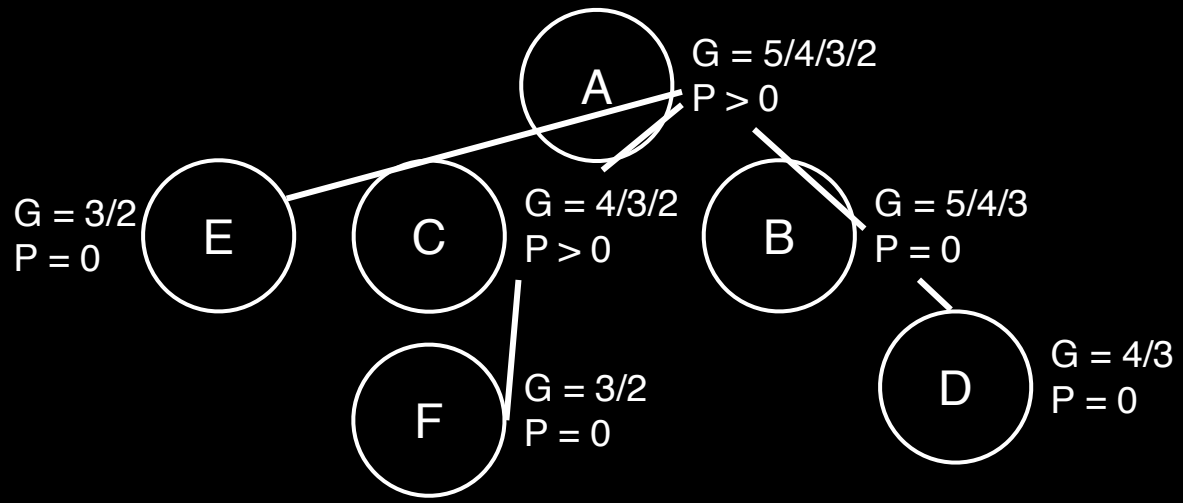
```c
int glob = 3;
int pid = 0;
int i = 0;
    fork();
    glob--;
    if (fork())
        glob--;
    if (pid == 0)
    {
        for (i = 0; i < 2; i+
+)
        {
            pid = fork();
            glob++;
        }
    }
    if (!pid)
        glob++;
```



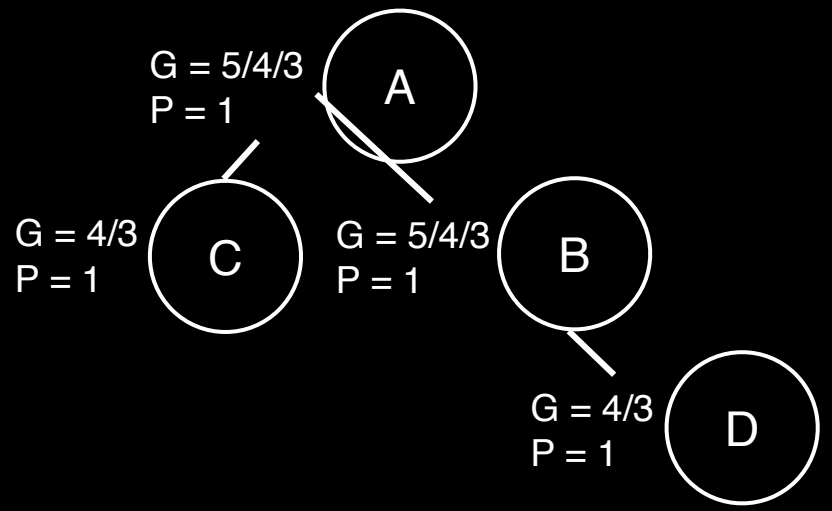
```c
int glob = 0;
int pid = 0;
    fork();
    glob--;
    if (fork())
    {
        pid = fork();
        glob--;
        pid = 0;
    }
    if (pid)
    {
        glob--;
    }
    else
    {
        fork();
        glob--;
    }
```



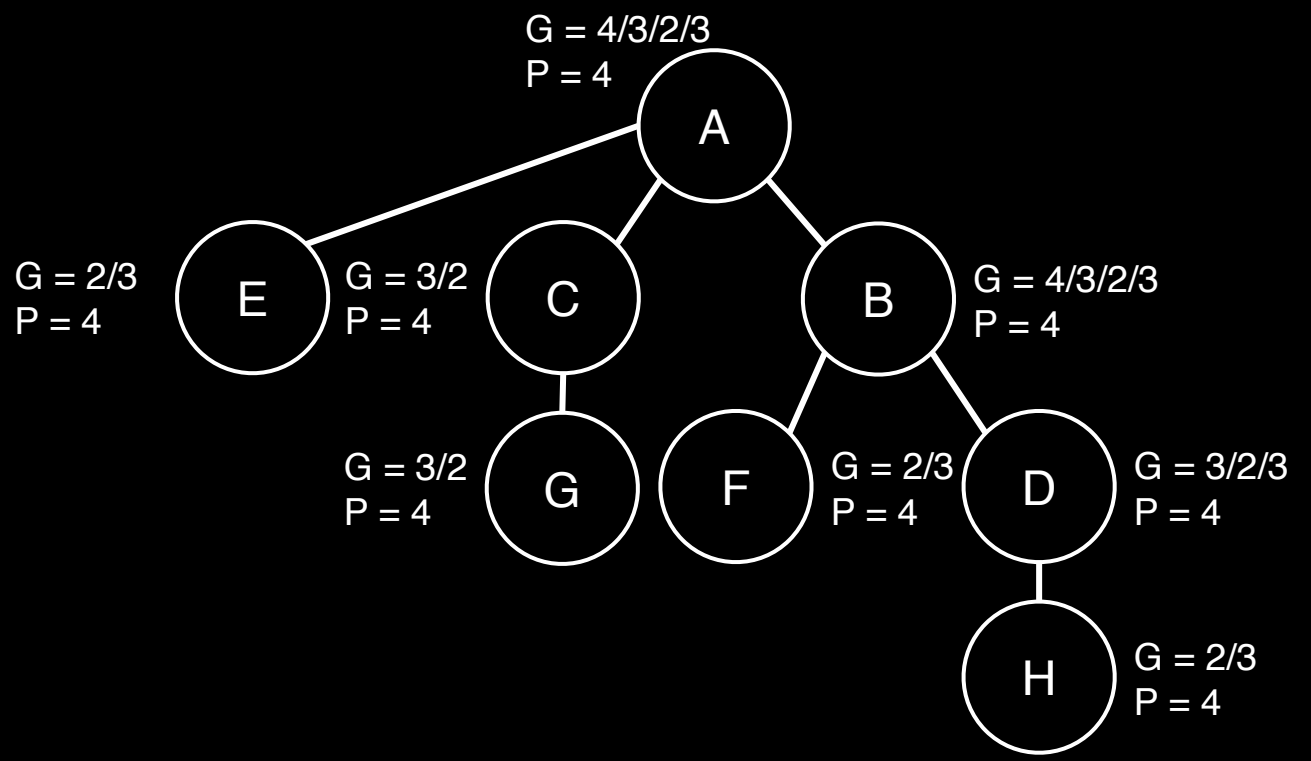
```c
int glob = 5;
int pid = 0;
    pid = fork();
    glob--;
    fork();
    glob--;
    if (!pid)
    {
        pid = fork();
        glob--;
    }
```



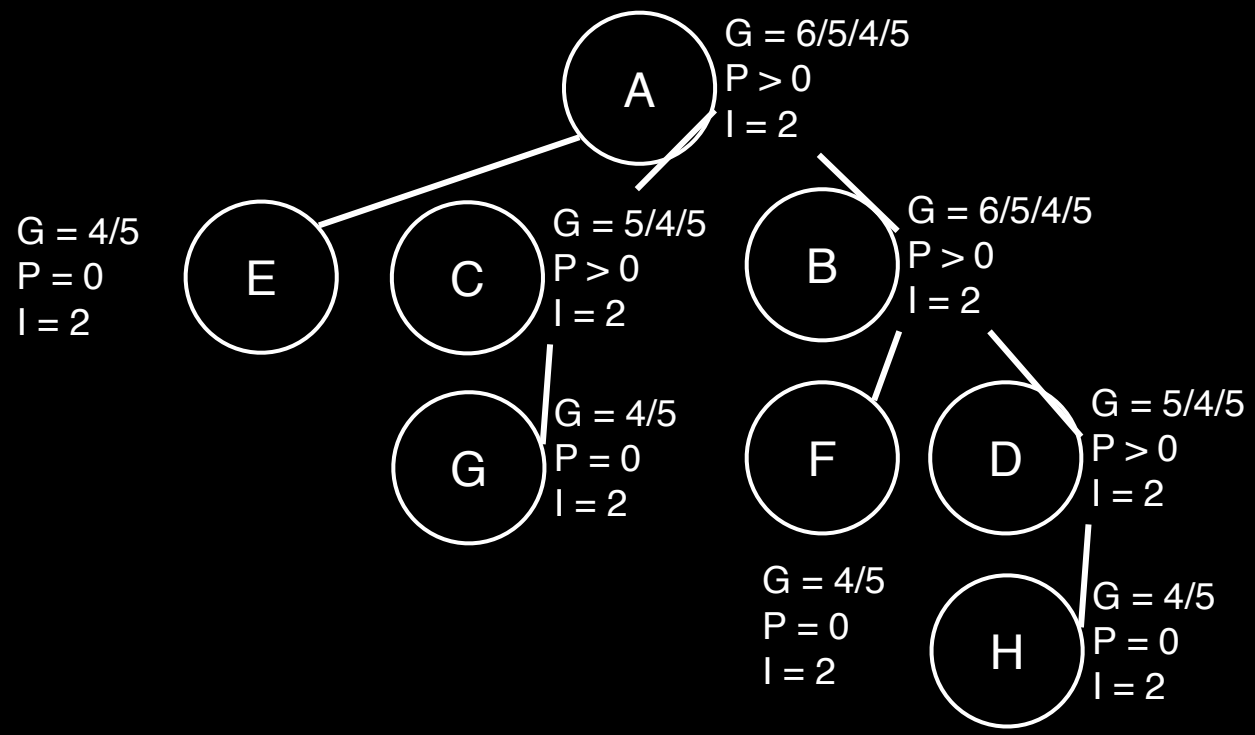
```c
int glob = 5;
int pid = 0;
    pid = fork();
    glob--;
    fork();
    pid = 1;
    glob--;
    if (!pid)
    {
        pid = fork();
        glob--;
    }
```



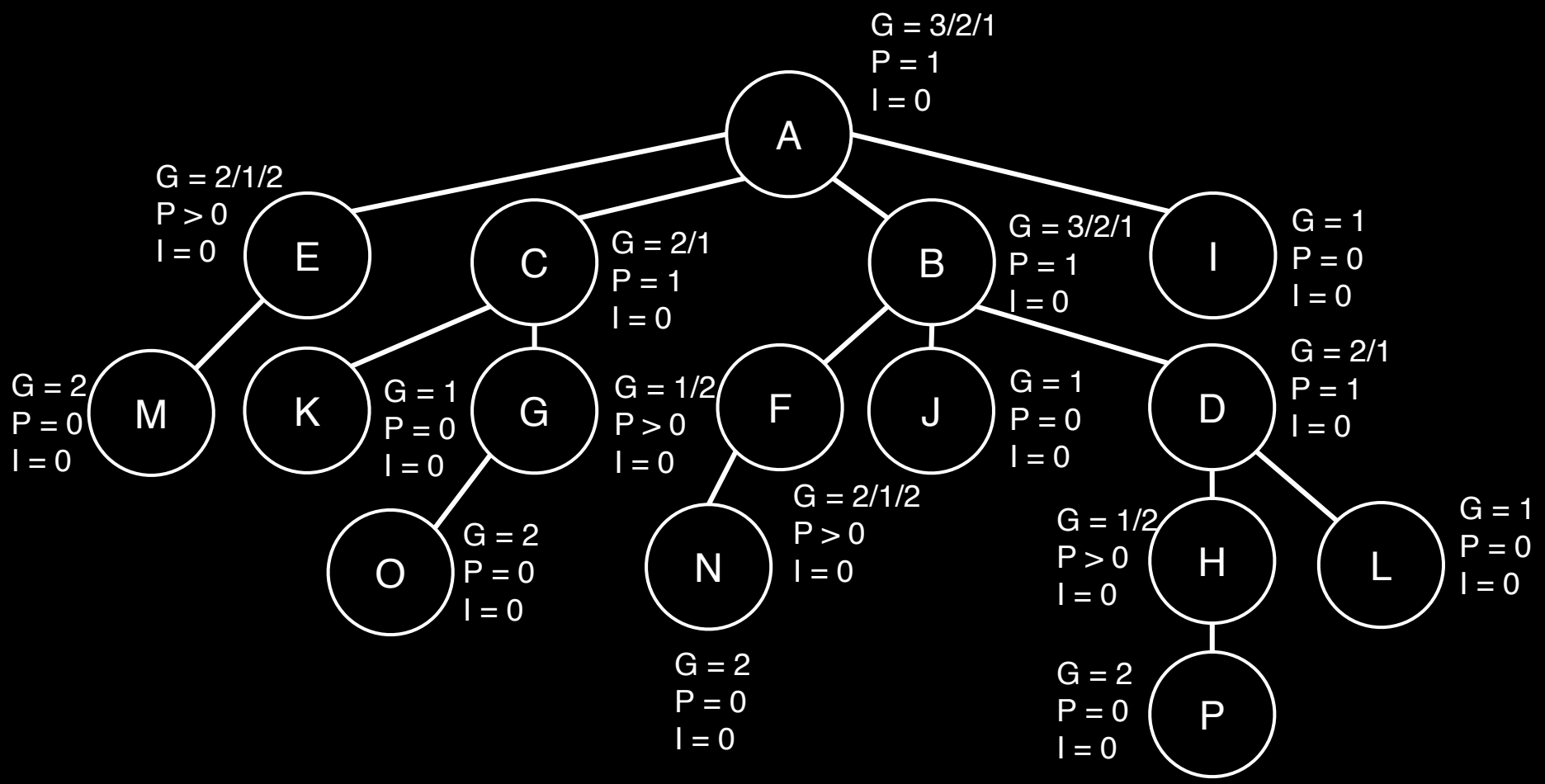
```c
int glob = 4;
int pid = 0;
int i;
    pid = fork();
    glob--;
    fork();
    pid = 4;
    glob--;
    if (!pid)
    {
        pid = fork();
        glob--;
    }
    for (i = 1; i < 2;i++)
    {
        pid = fork();
        glob++;
    }
```



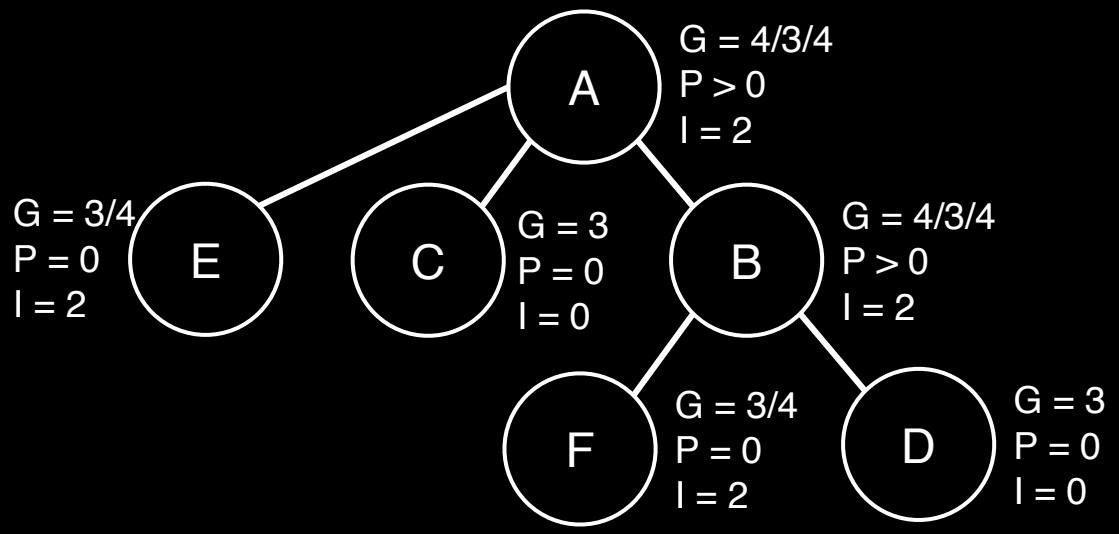
```c
int glob = 6;
int pid = 1;
int i;
    pid = fork();
    glob--;
    fork();
    pid = 0;
    glob--;
    if (pid)
    {
        pid = fork();
        glob--;
    }
    if (!pid)
    {
        for (i = 1; i < 2; i++)
        {
            pid = fork();
            glob++;
        }
    }
```



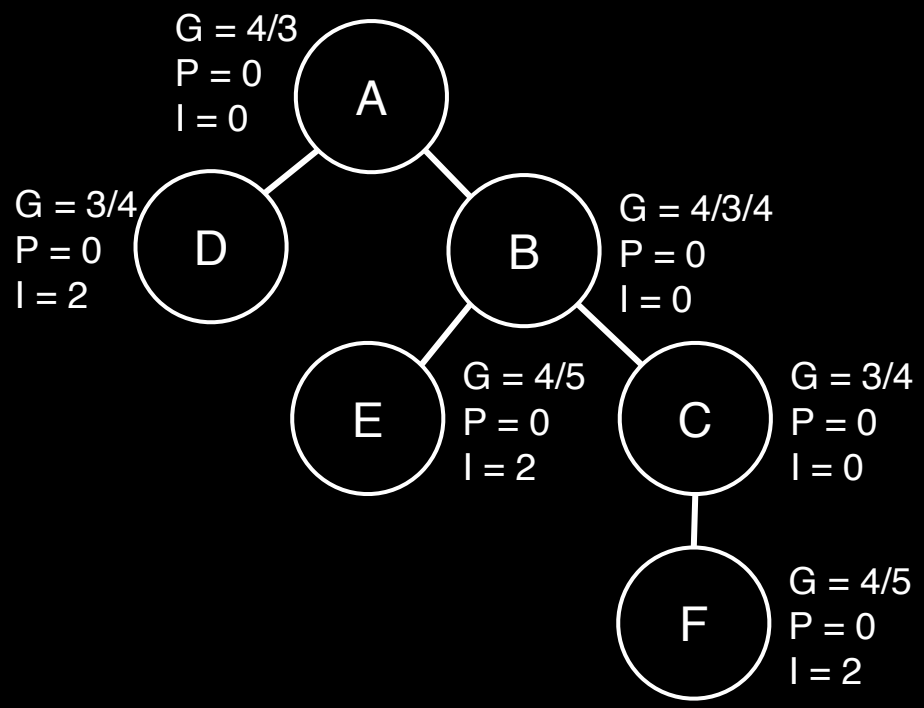
```c
int glob = 3;
int pid = 0;
int i;
    pid = fork();
    glob--;
    fork();
    pid = 1;
    if (pid)
    {
        pid = fork();
        glob--;
    }
    if (pid == 0)
        glob++;
    for (i = 1; i < 2; i+
+)
        pid = fork();
```



```c
int glob = 4;
int pid = 0;
int i;
    pid = fork();
    pid = 1;
    glob--;
    if (!pid)
    {
        pid = fork();
        glob++;
    }
    pid = 0;
    if (fork())
    {
        for (i = 1; i < 2; i++)
{
            pid = fork();
            glob++;
        }
    }
```



```c
int glob = 4;
int pid = 0;
int i;
    pid = fork();
    glob--;
    if (!pid)
    {
        pid = fork();
        glob++;
    }
    pid = 0;
    if (!fork())
    {
        for (i = 1; i < 2; i++)
        {
            glob++;
        }
    }
```

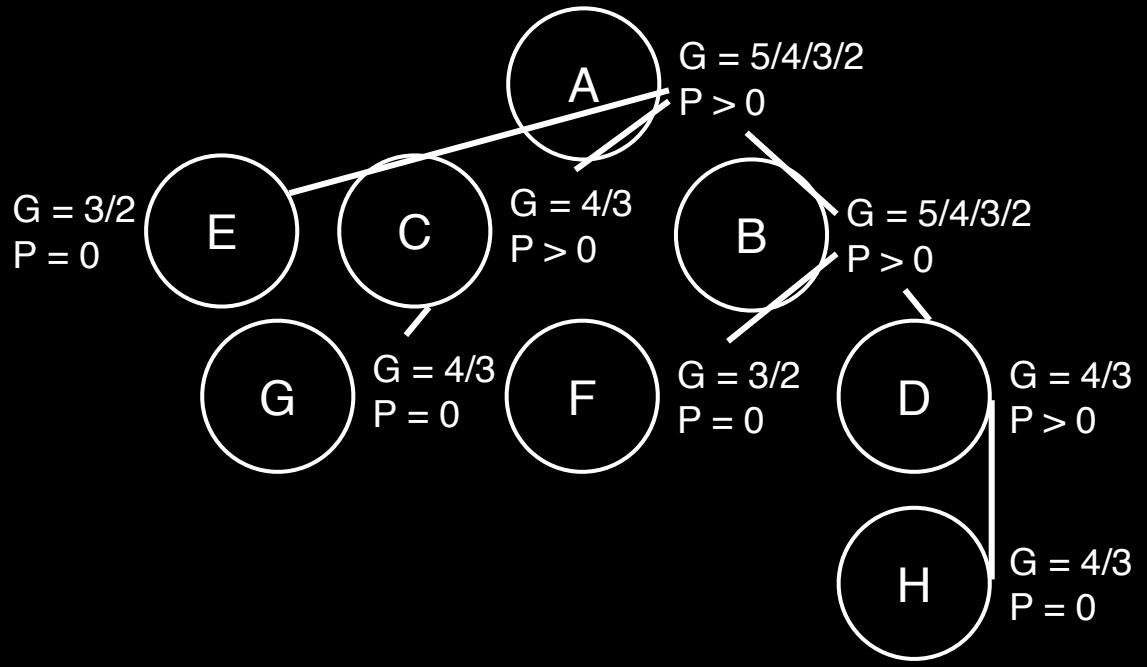


```c
int glob = 20;
int pid = 0;
for(int i = 1; i < 3; i++)
{
  if(pid)
    fork();
  if(!pid)
    glob = glob * 2;
  glob = glob - 1;
}
```

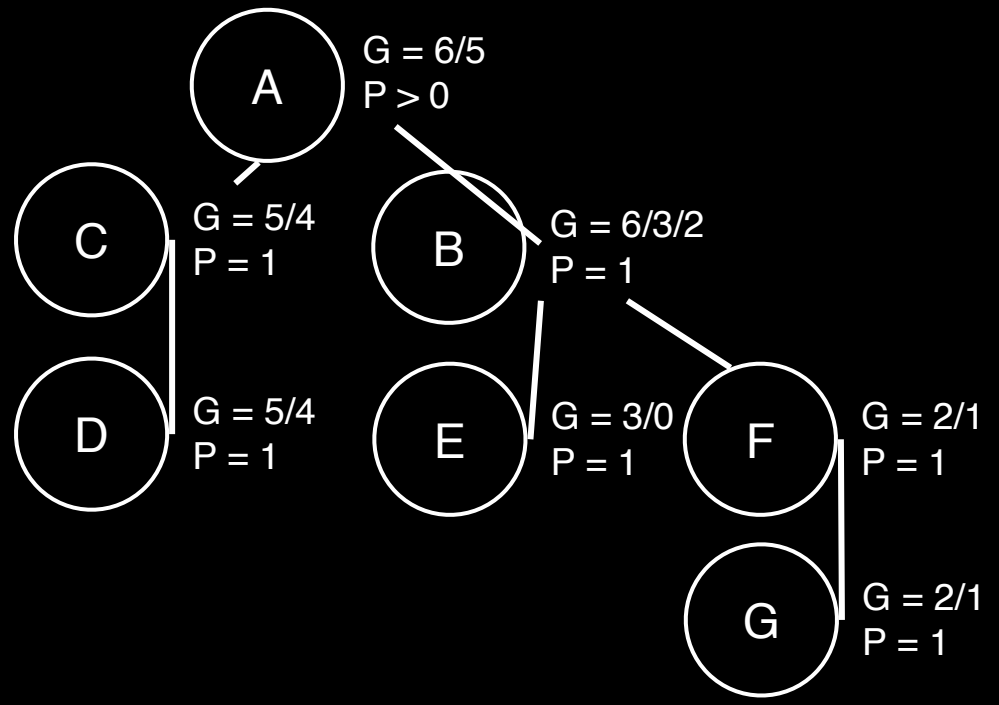
A

G = 20/40/39/78/77
P = 0

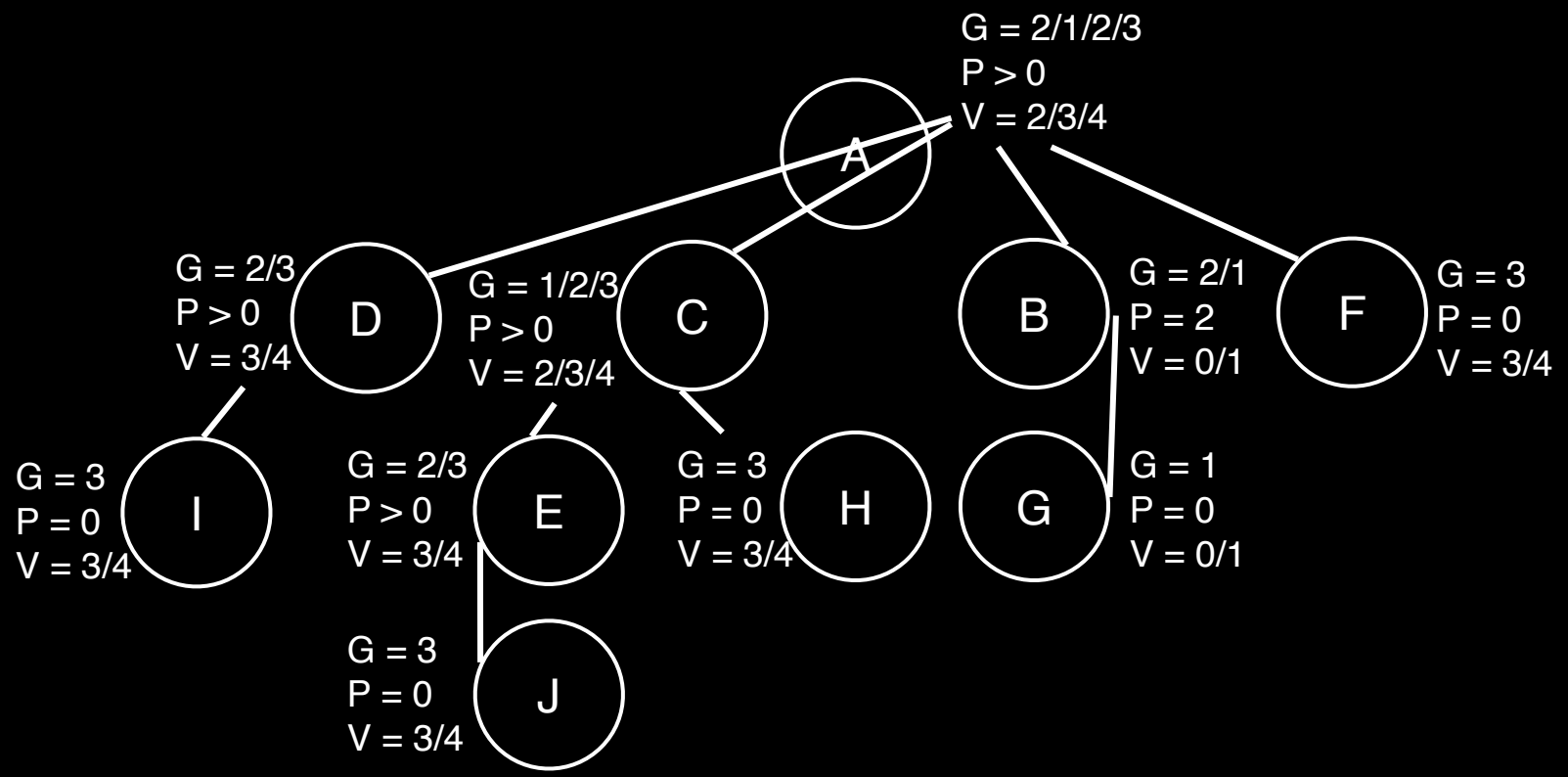
```c
int glob = 5;
int pid = 0;
fork();
glob--;
if(fork())
  glob--;
if(pid == 0)
{
  pid = fork();
  glob--;
}
```



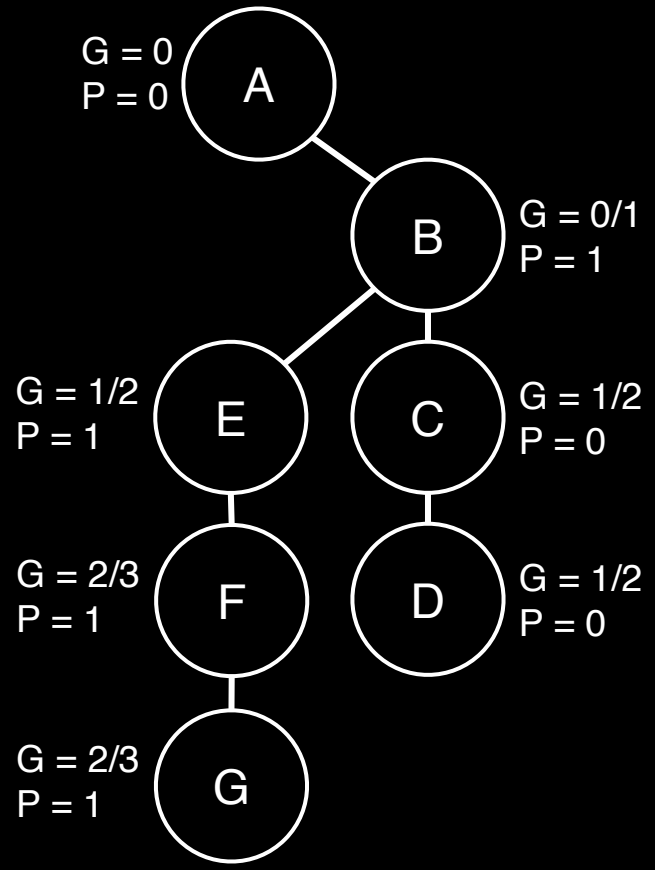
```c
int glob = 6;
int pid = 0;
    while (!pid)
    {
        if (!fork())
            glob -=3;
        else
        {
            glob --;
            if ((pid = fork()) ==
0)
            {
                pid = 1;
                fork();
                glob --;
            }
        }
        if (!glob)
        pid = 1;
    }
```



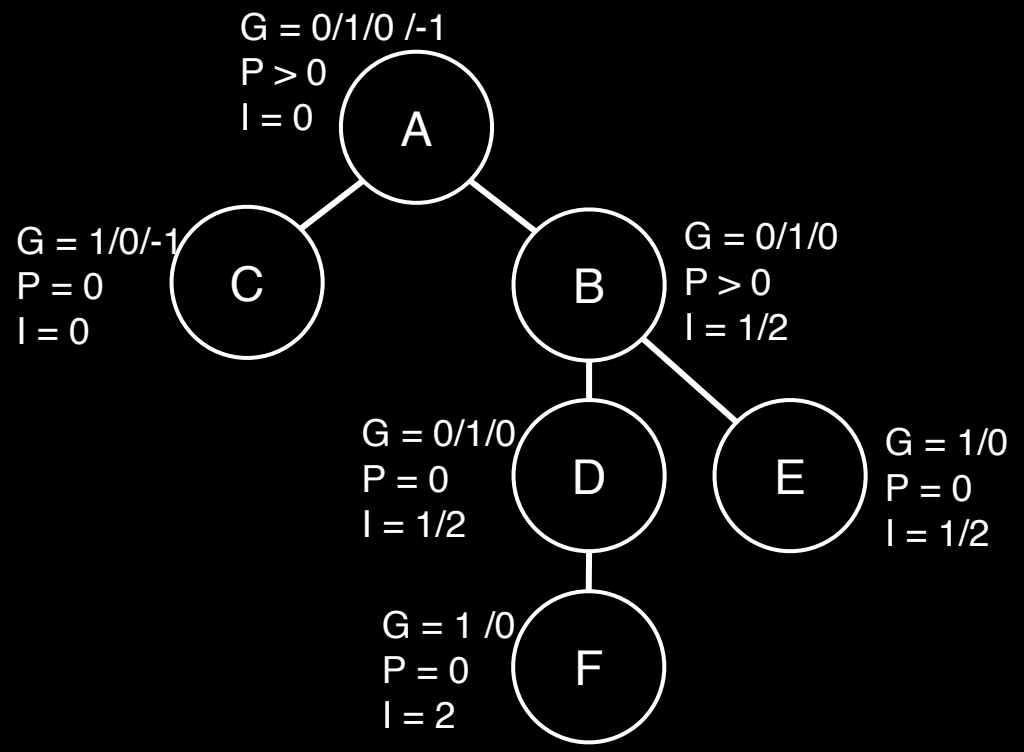
```c
int glob = 2;
int pid = 2;
int var = 2;
var = fork();
glob--;
if (var)
{
    pid =
fork();
    glob++;
    pid = 0;
}
if(!pid)
{
    var++;
    fork();
    glob++;
}

pid = fork();
var++;
```



```c
int glob = 0;
int pid = 0;
while (!fork())
{
  glob += 1;
  if (fork())
    pid = 1;
  else
  {
    fork();
    glob++;
  }
  if (glob > 1)
    exit(-3);
}
```

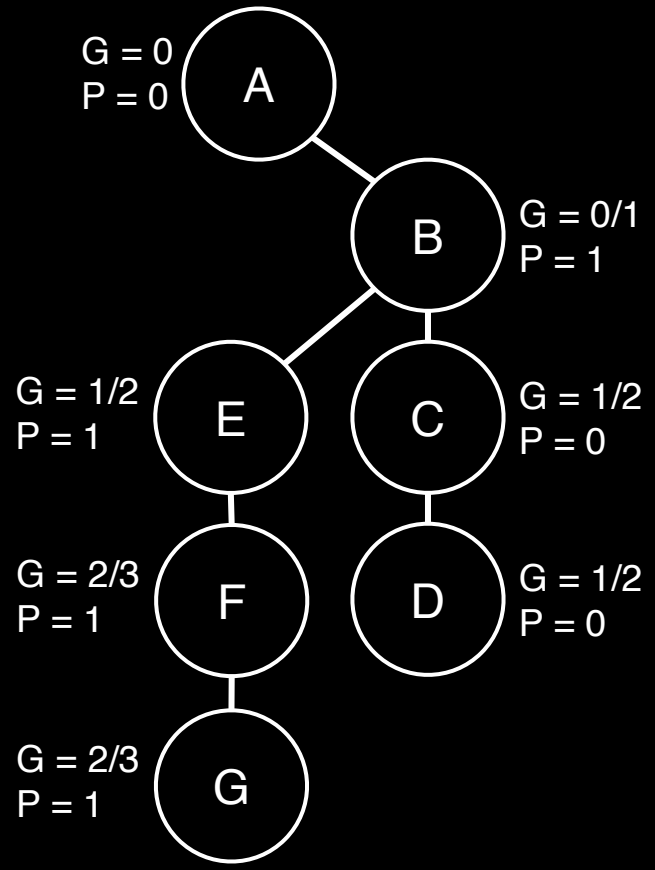


```c
int glob = 0;
    int i;
    int pid = 0;
    pid = fork();
    if (pid != 0)
    {
        glob++;
        pid = fork();
        glob--;
    }
    else if (pid == 0)
    {
        for (i = 1; i <= 2; i+
+)
        {
            pid = fork();
            glob++;
        }
    }
    glob --;
```

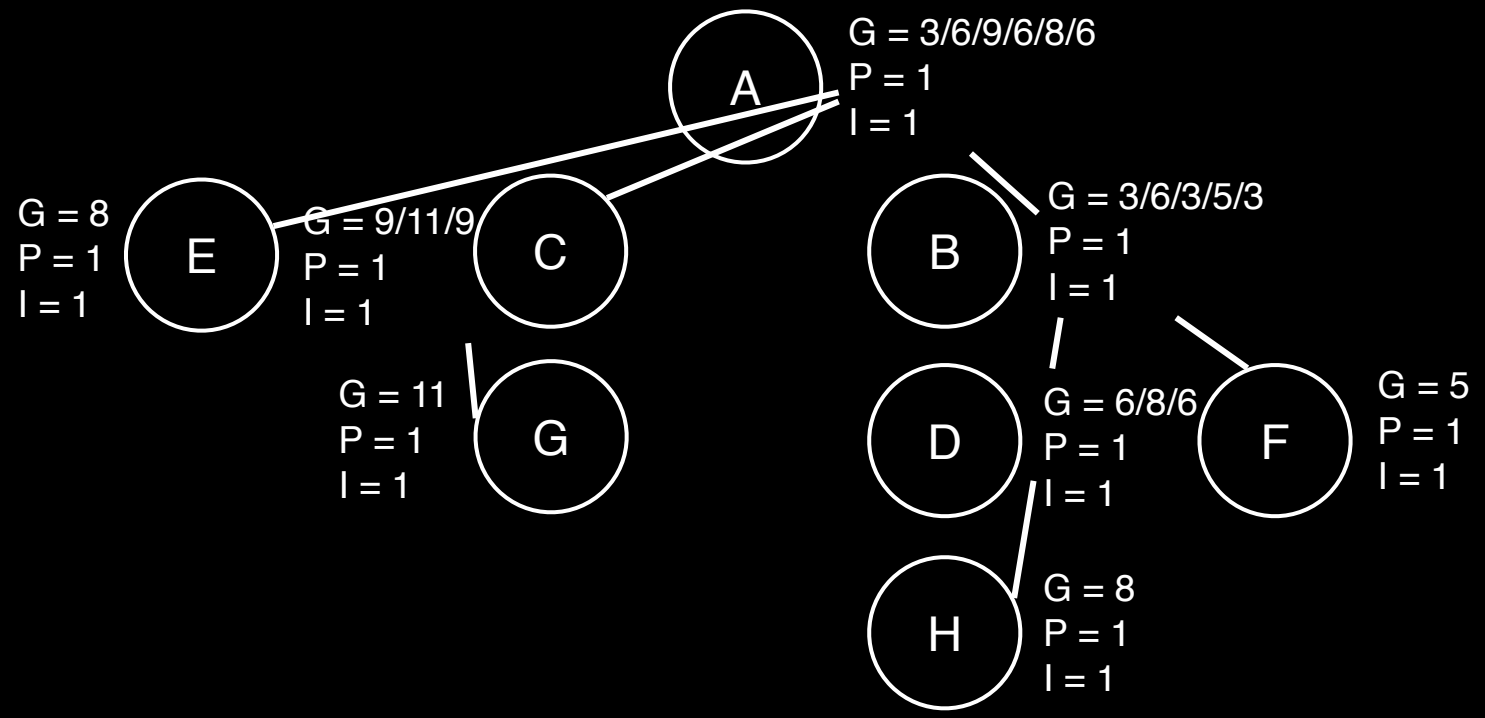


•Nel secondo blocco, ovvero else if (pid == 0), entra solo B in quanto C (figlio di A) è appena stato creato e non eseguendo il controllo della condizione if(pid != 0), non entrerà in else if (pid == 0)

```c
int glob = 0;
int pid = 0;
    while (!fork())
    {
        glob += 1;
        if (fork())
            pid = 1;
        else
        {
            fork();
            glob++;
        }
        if (glob > 1)
            exit(-3);
    }
```

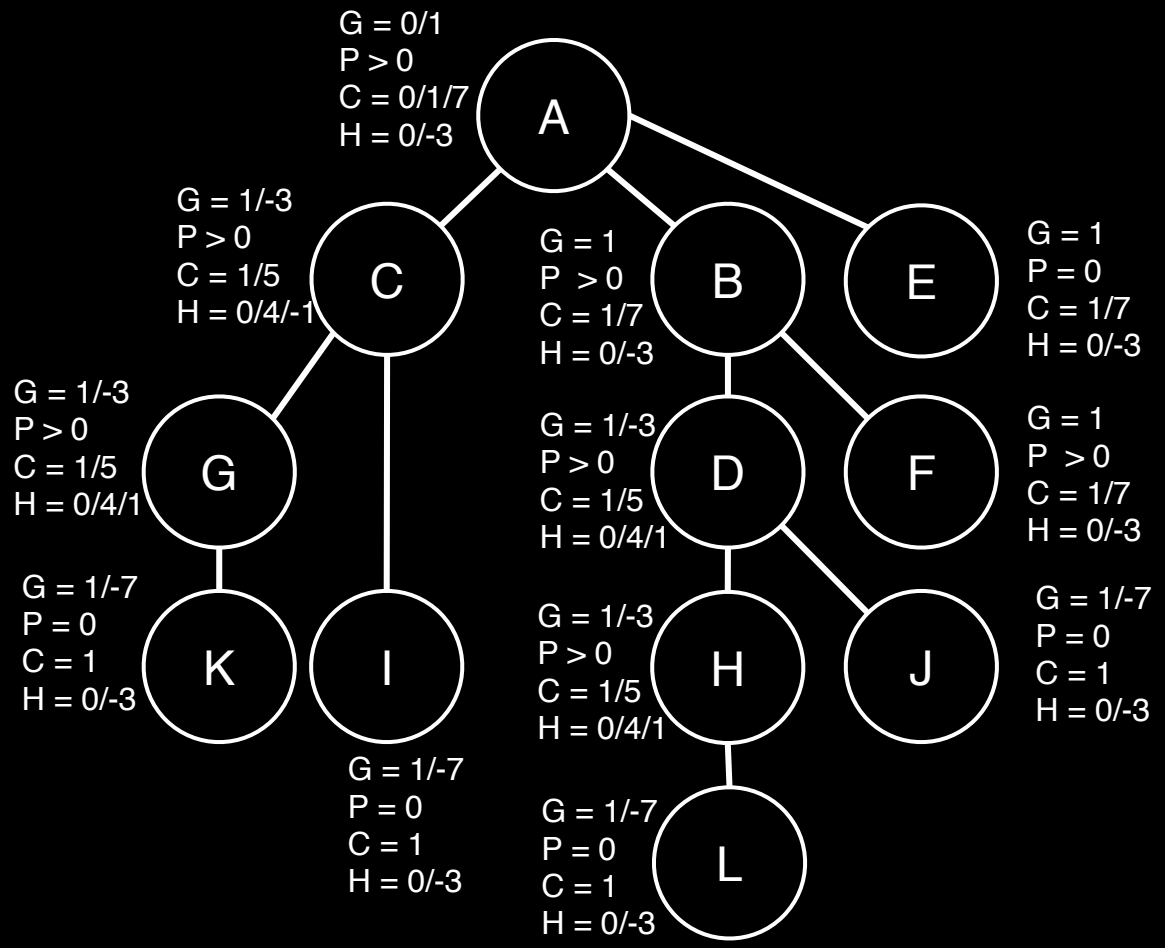


```c
int glob = 3;
int pid = 0;
int i;
    if (fork())
        if (!pid)
            glob = glob + 3;

    for(i = 3; i > 1; i--)
    {
        glob = glob + i;
        pid = fork();
        if (pid)
            glob = glob - i;
        pid = 1;
    }
    if (!pid)
        glob--;
```

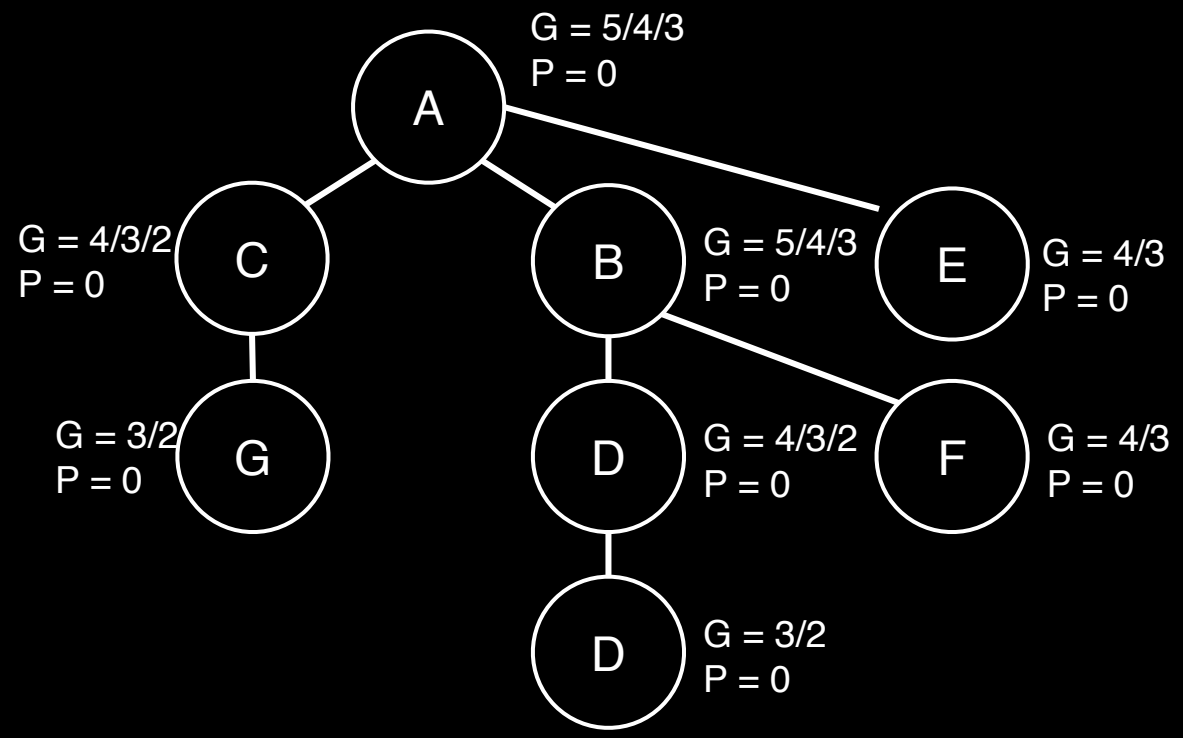


Senza senso

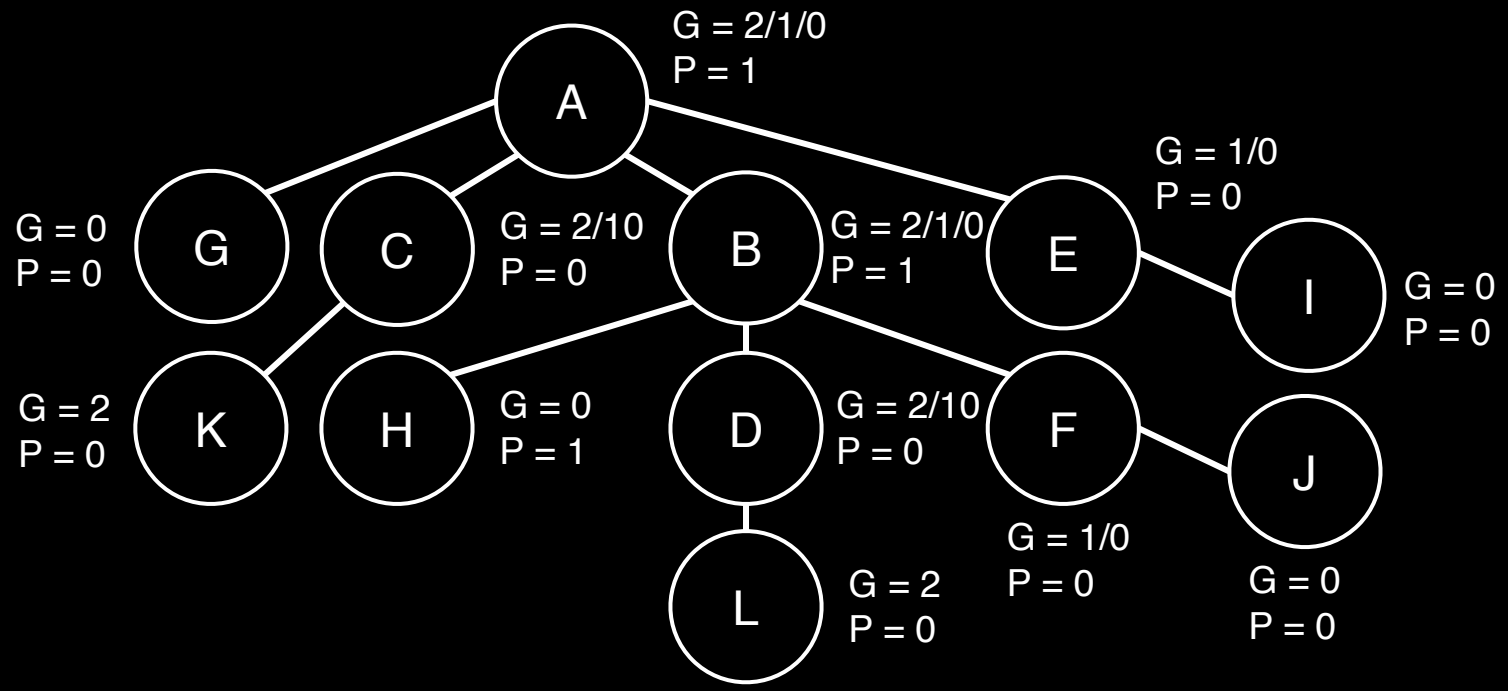
```c
int glob = 0;
int c = 0;
int h = 0;
    int i = 0;
    int pid = 0;
    glob++;
    c++;
    pid = fork();
    glob = c;
    c = glob;
    pid = fork();
    if (pid != 0)
    {
        for (i = 1; i < 2; i++)
            pid = fork();
        c = glob + 2 * 3;
    }
    else if (pid == 0)
    {
        for (i = 2; i <= 3; i++)
            pid = fork();
        if (pid != 0)
        {
            h = 4;
            c = c * 5;
        }
        glob = c - 2 * 4;
    }
    h = h - 3;
```



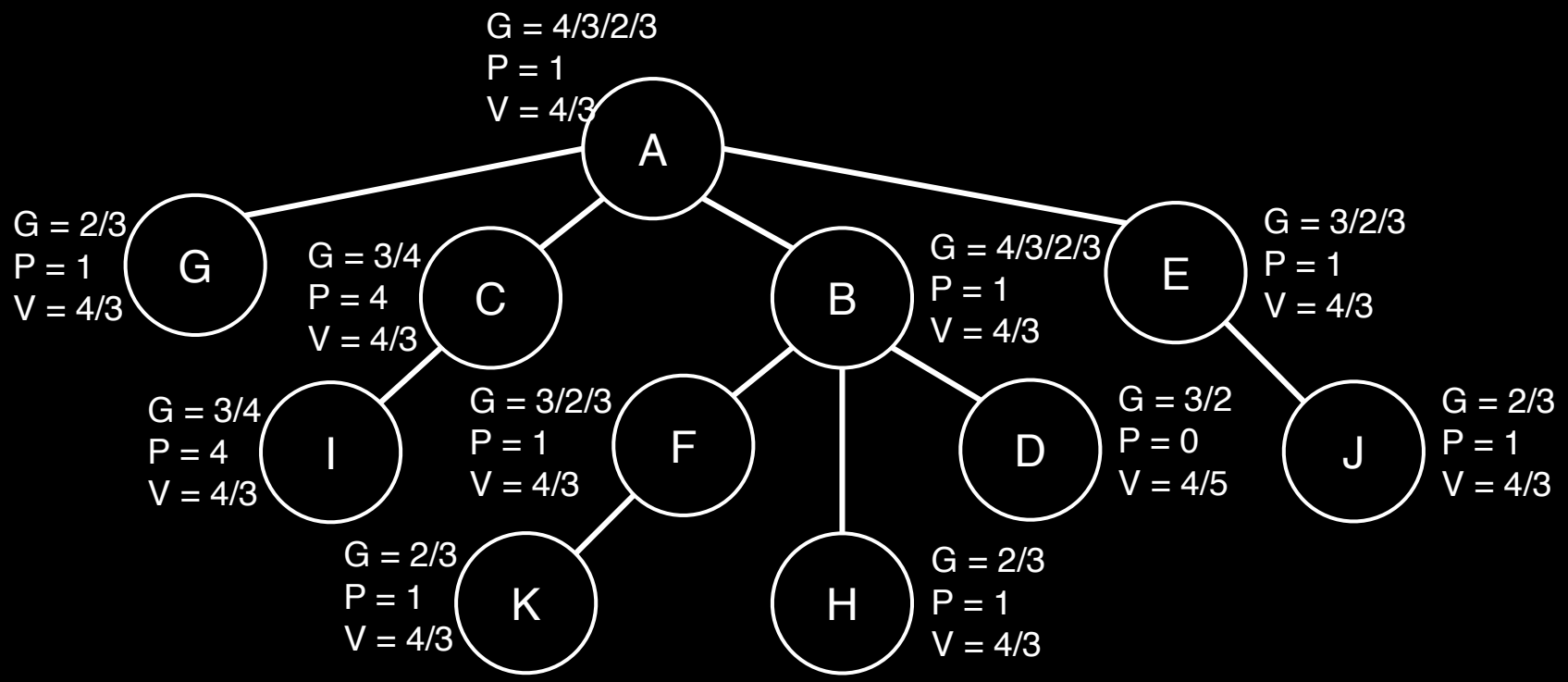
```c
int glob = 5;
int pid = 0;
    fork();
    glob--;
    if (!fork())
    {
        glob--;
    }
    if (pid == 0){
        pid =
fork();
        glob--;
    }
```



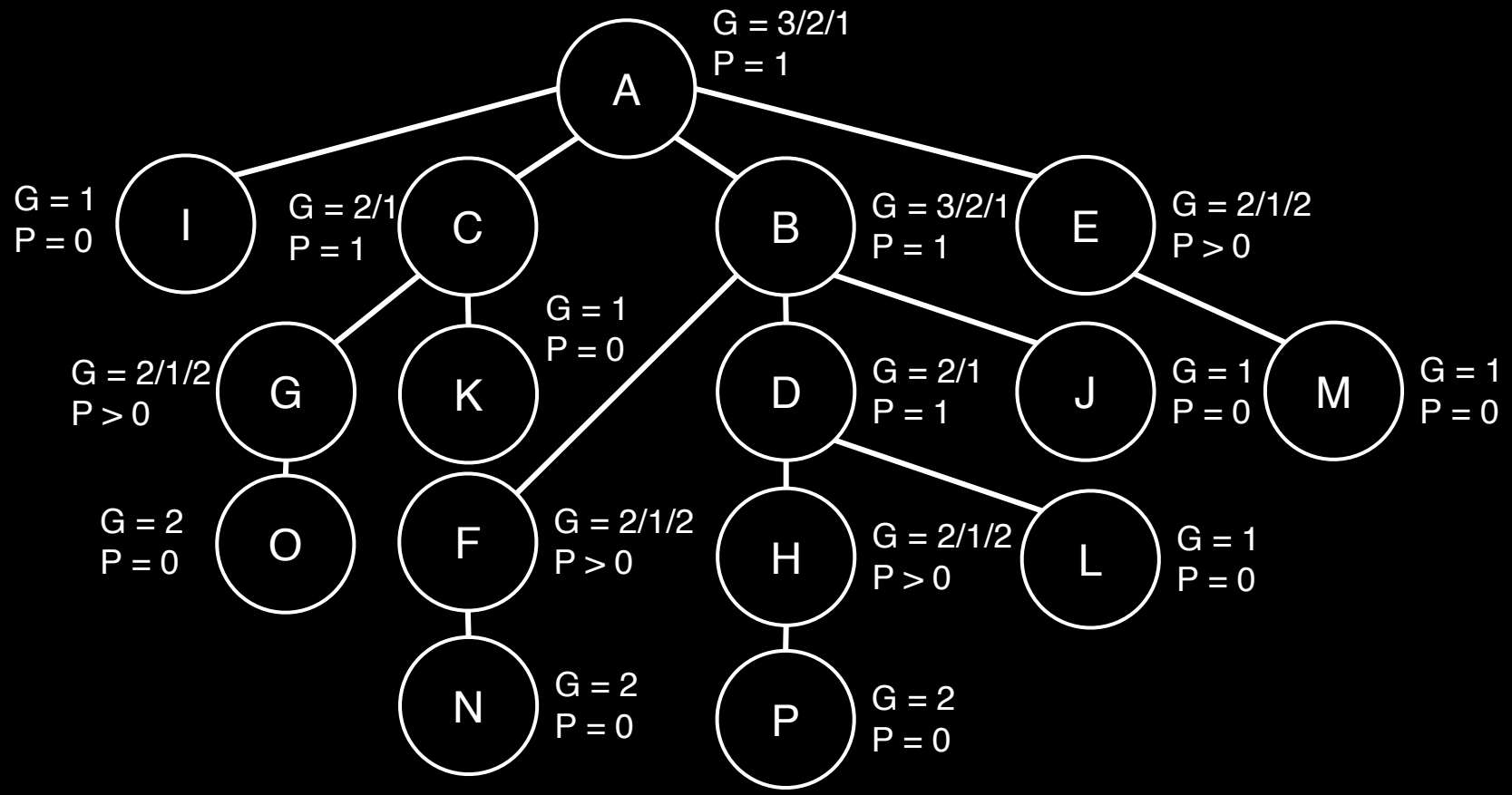
```c
int glob = 2;
int pid = 1;
fork();
pid = fork();
if (pid)
{
    while (glob)
    {
        glob--;
        pid =
fork();
    }
}
else
{
    if (fork())
        glob = 10;
}
```



```c
int glob = 4;
int var = 4;
int pid = 4;
pid = fork();
glob --;
if (fork())
{
    pid = fork();
    glob--;
    pid = 1;
}
if (!pid)
{
    glob-- && var++;
}
else
{
    fork();
    glob++;
    var--;
}
```



```c
int glob = 3;
int pid = 0;
pid = fork();
glob--;
fork();
pid = 1;
if (pid)
{
    pid = fork();
    glob--;
}
if (pid == 0)
    glob++;
for (int i = 1; i < 2; i+
+)
    pid = fork();
```



```c
int glob = 6;
int pid = 0;
if(pid = fork())
{
    fork();
    glob--;
}
fork();
glob--;
if (pid > 0)
{
    pid = fork();
    glob--;
}
```