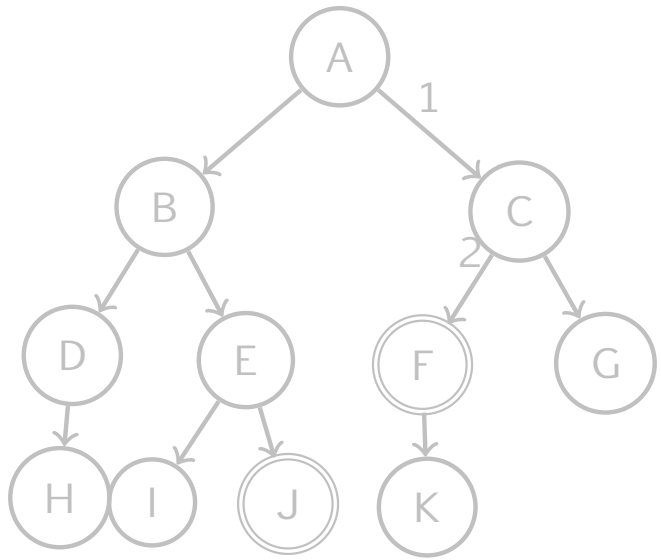
Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

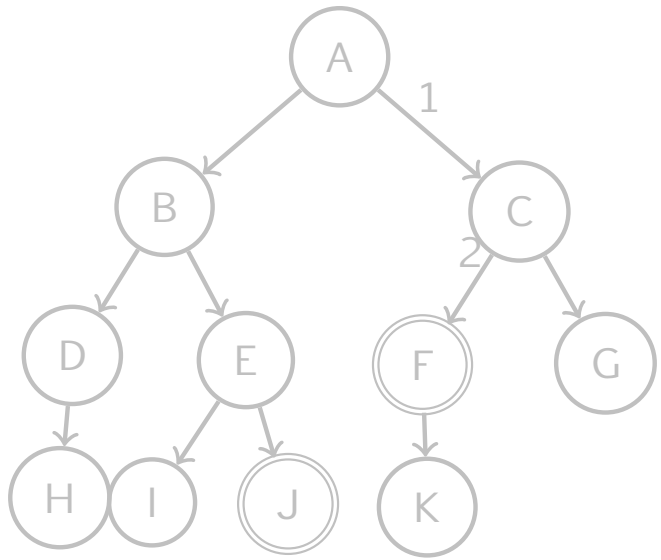
Ch	n	n1	fin	Liste	Sol
-	-	-	-	{{A}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

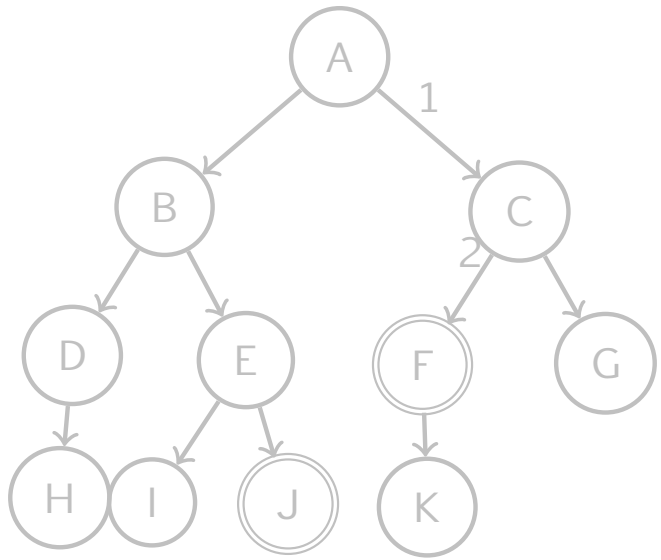
Ch	n	n1	fin	Liste	Sol
{A}	-	-	Faux	{{A}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

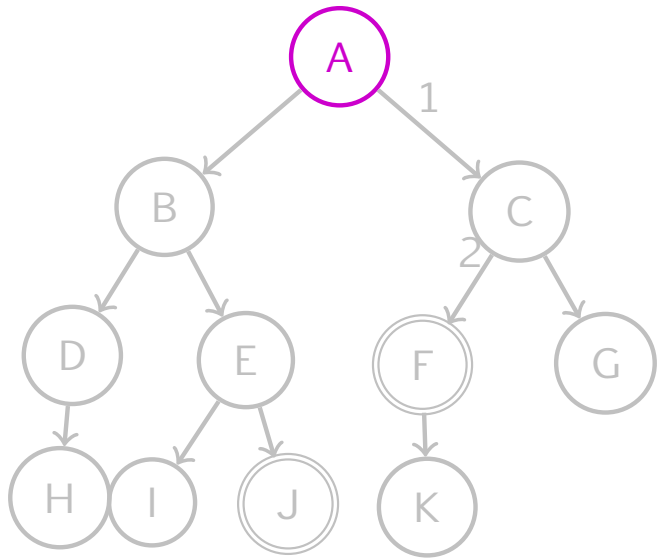
Ch	n	n1	fin	Liste	Sol
{A}	-	-	Faux	{{A}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

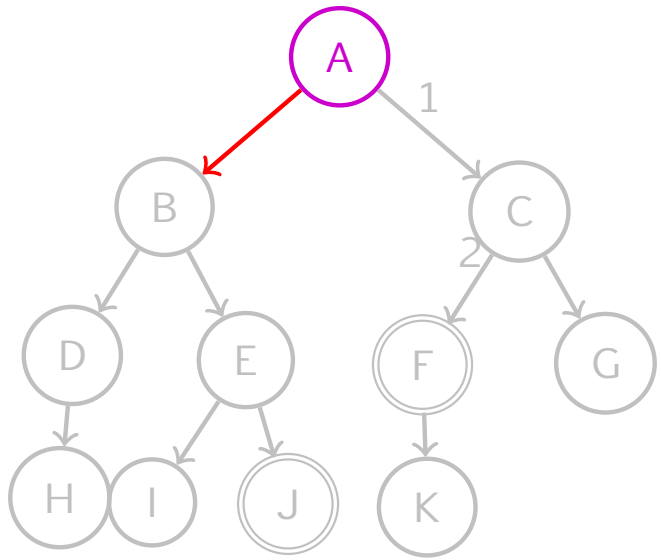
Ch	n	n1	fin	Liste	Sol
{A}	A	-	Faux	{}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

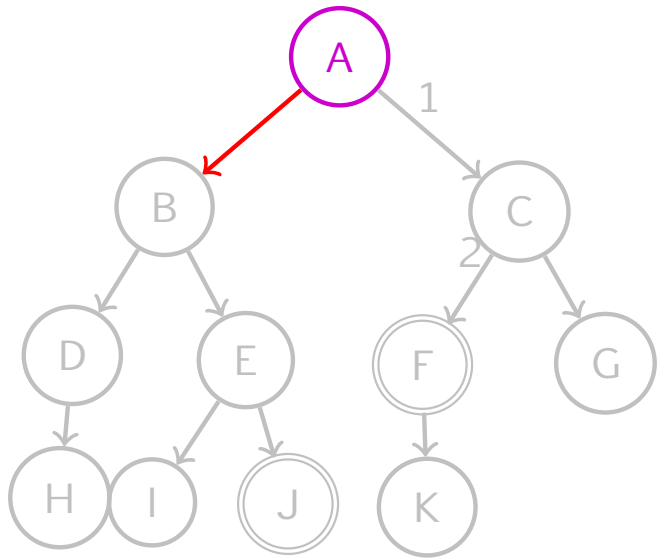
Ch	n	n1	fin	Liste	Sol
{A}	A	B	Faux	{}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

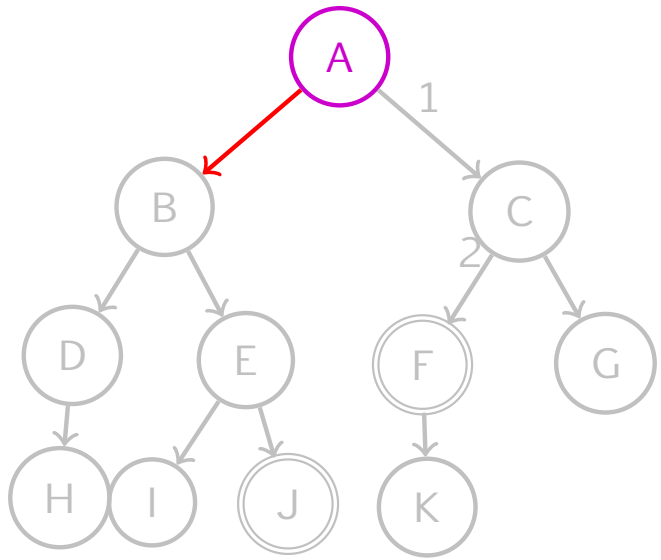
Ch	n	n1	fin	Liste	Sol
{A}	A	B	Faux	{}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

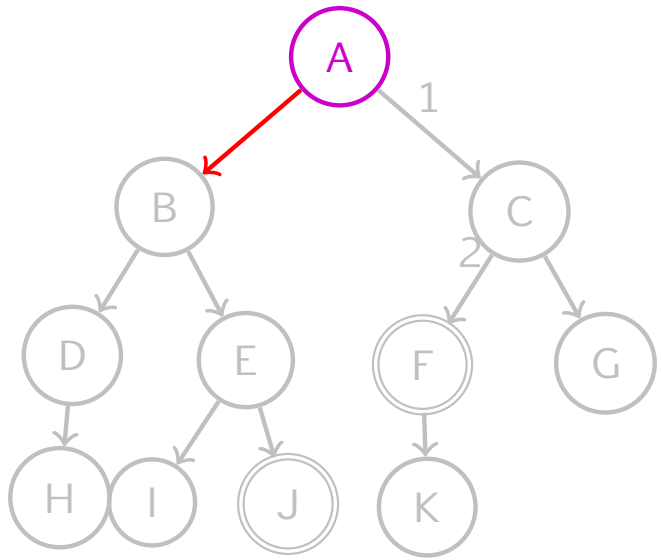
Ch	n	n1	fin	Liste	Sol
{A}	A	B	Faux	{}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

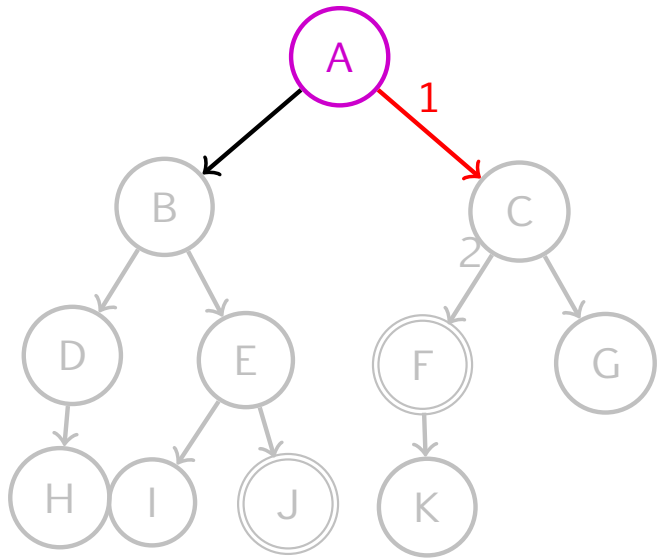
Ch	n	n1	fin	Liste	Sol
{A}	A	B	Faux	{{AB}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

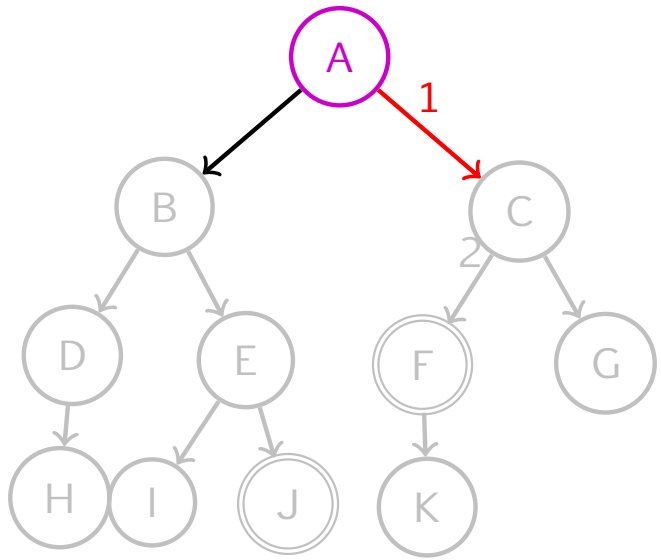
Ch	n	n1	fin	Liste	Sol
{A}	A	C	Faux	{{AB}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

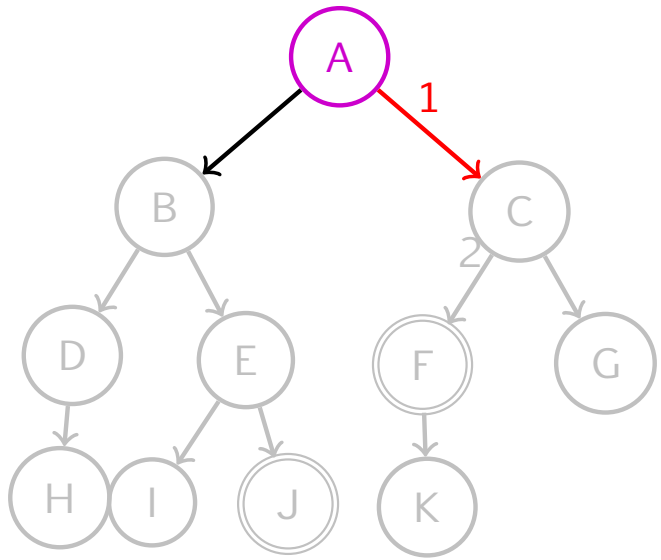
Ch	n	n1	fin	Liste	Sol
{A}	A	C	Faux	{{AB}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

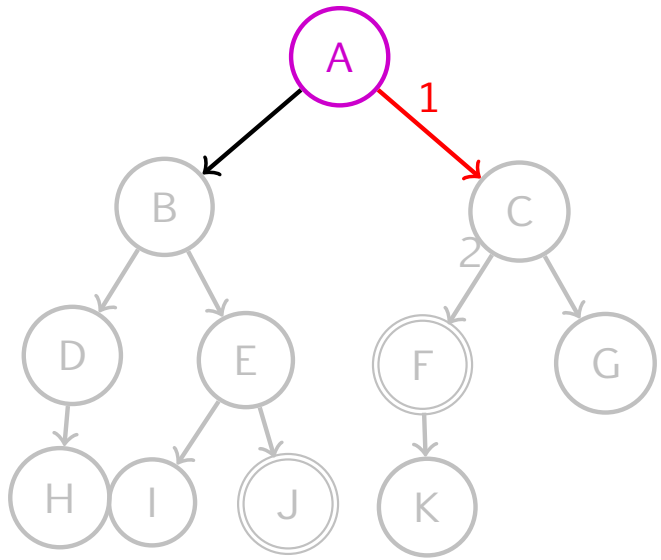
Ch	n	n1	fin	Liste	Sol
{A}	A	C	Faux	{{AB}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

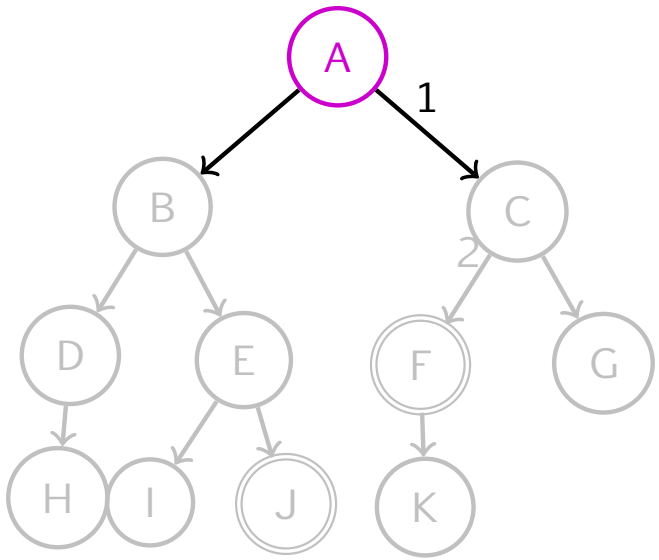
Ch	n	n1	fin	Liste	Sol
{A}	A	C	Faux	{{AB},{AC}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

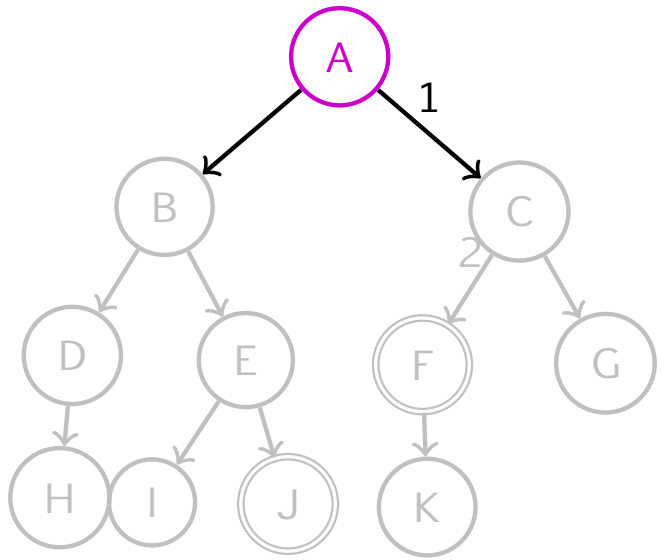
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

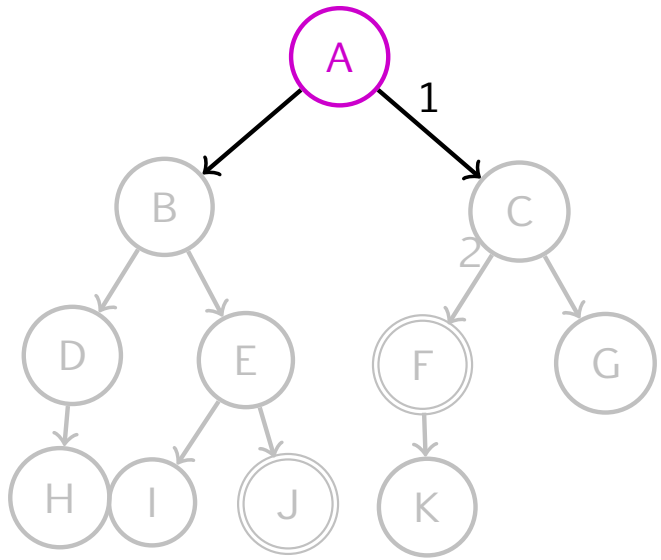
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

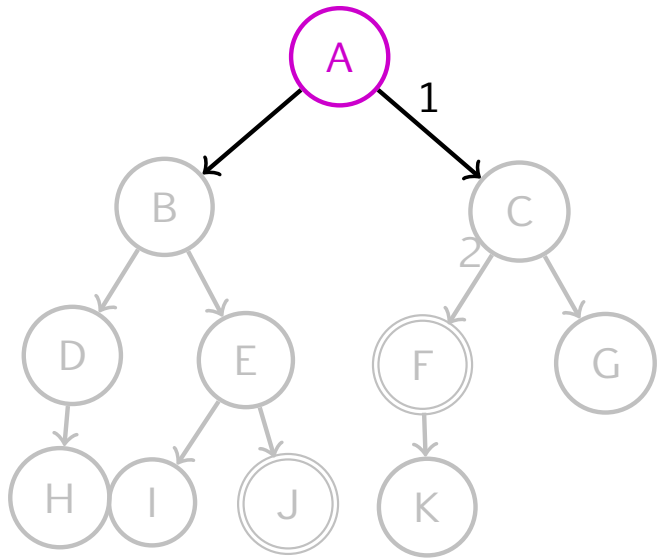
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A	⊥	Faux	{{AB},{AC}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

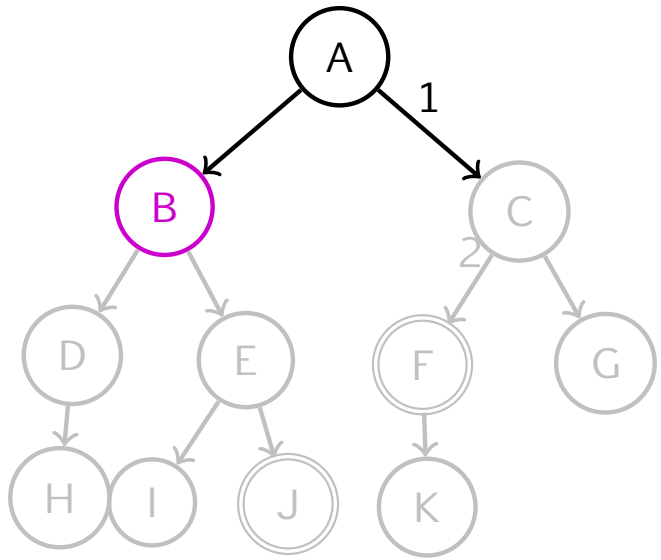
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A	$\perp$	Faux	{{AB},{AC}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

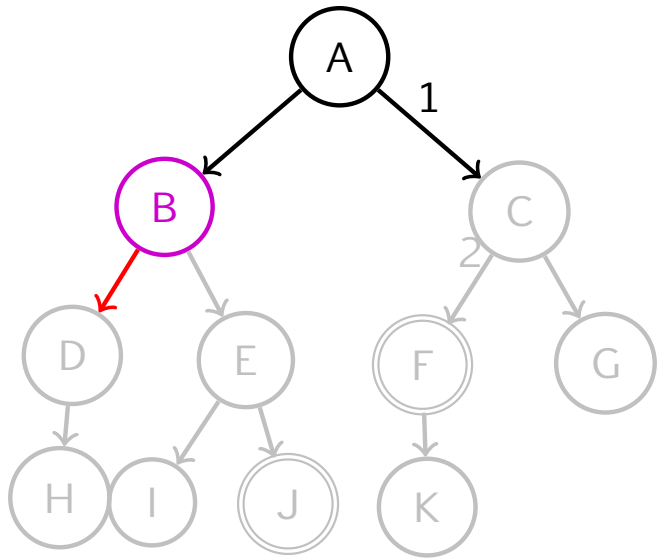
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥	Faux	{{AB},{AC}} {AC}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

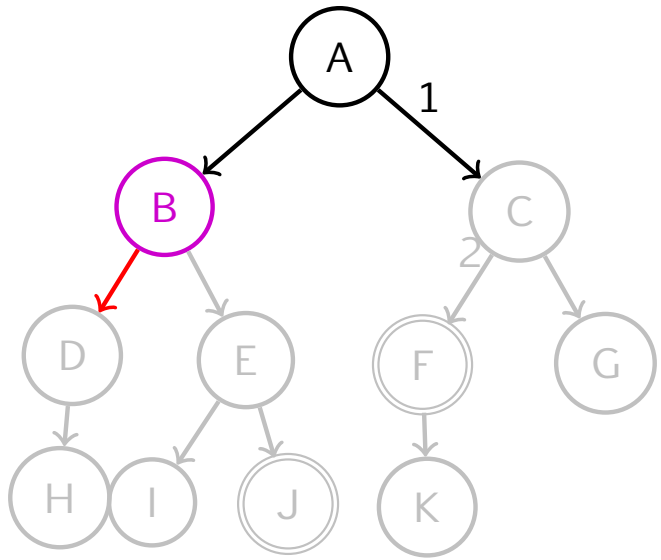
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ D	Faux	{{AB},{AC}} {AC}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

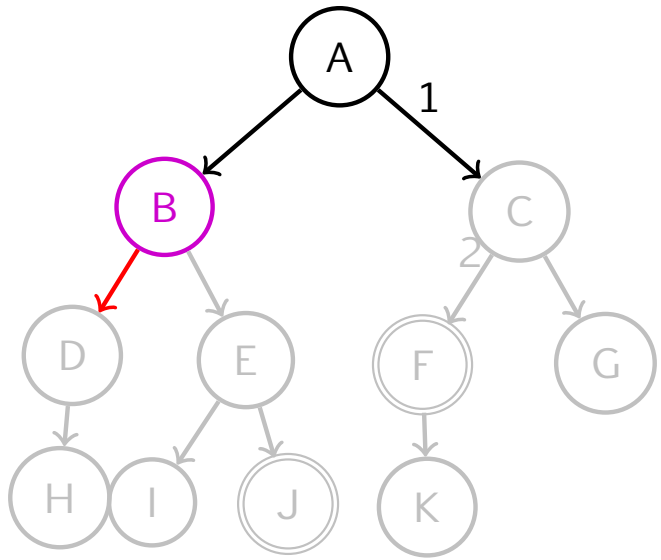
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ D	Faux	{{AB},{AC}} {AC}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

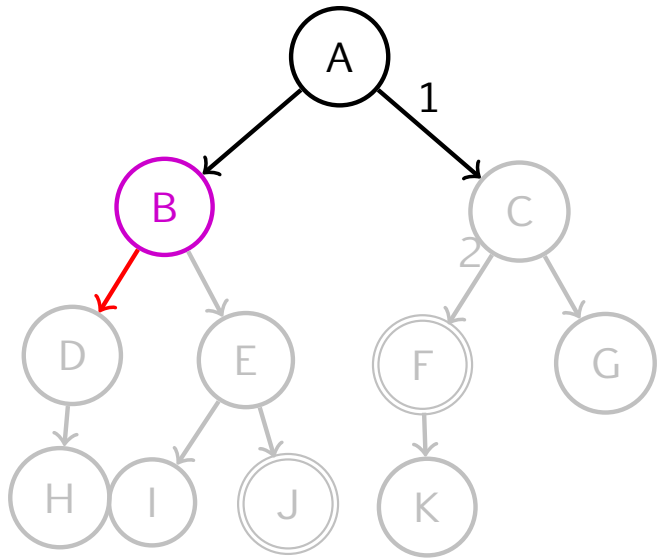
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ D	Faux	{{AB},{AC}} {AC}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

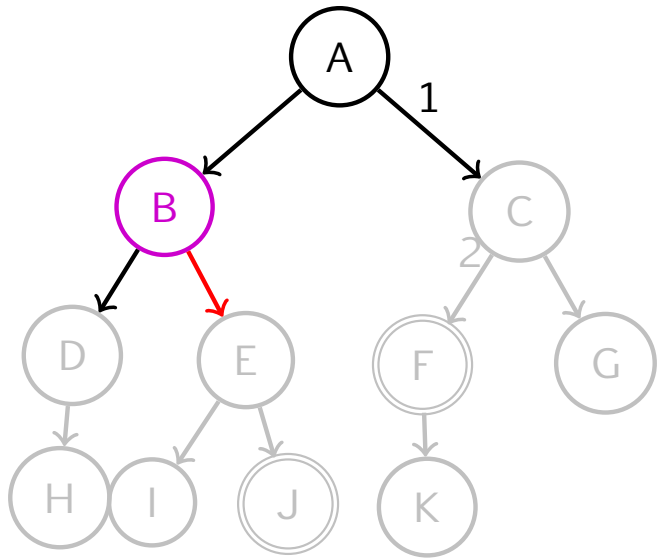
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ D	Faux	{{AB},{AC}} {{AC},{ABD}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

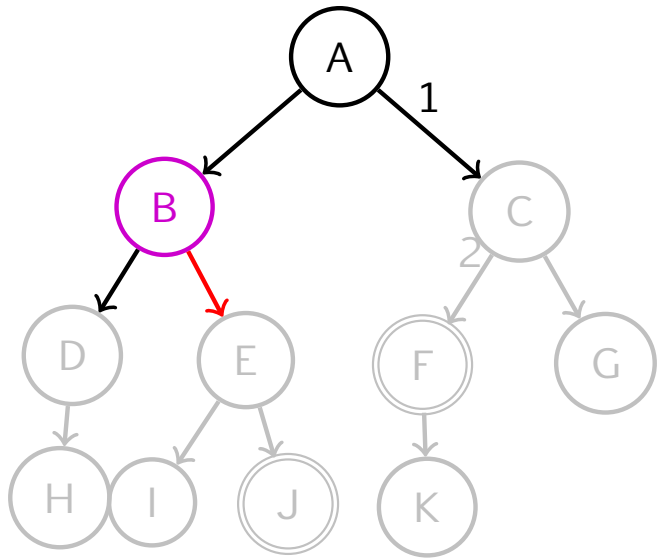
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ E	Faux	{{AB},{AC}} {{AC},{ABD}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

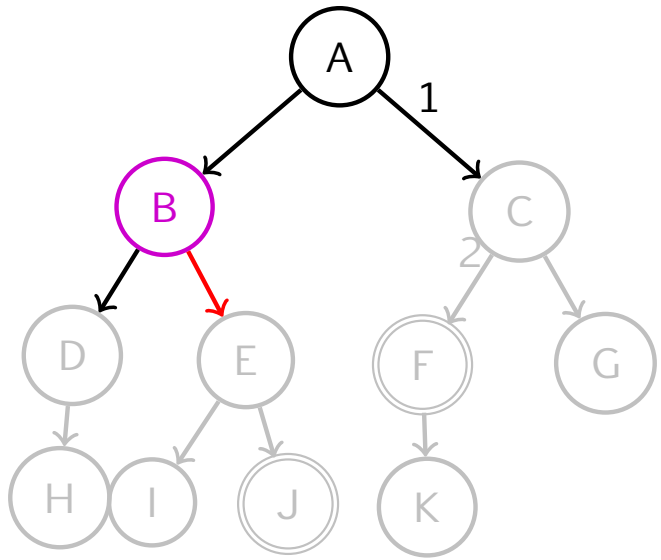
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ E	Faux	{{AB},{AC}} {{AC},{ABD}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

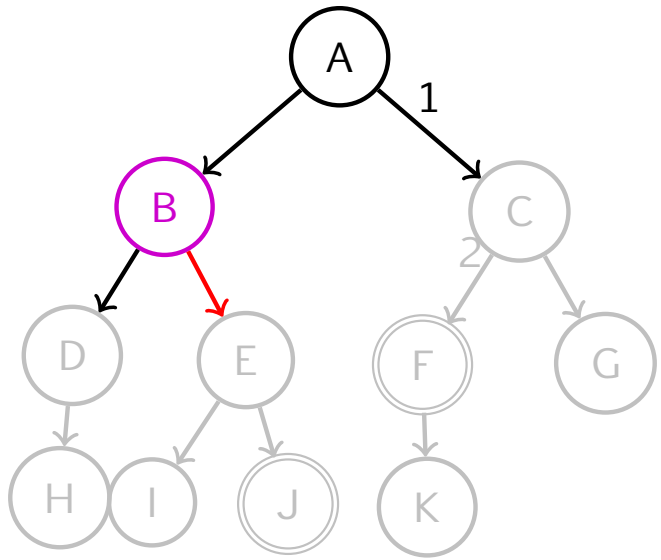
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ E	Faux	{{AB},{AC}} {{AC},{ABD}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

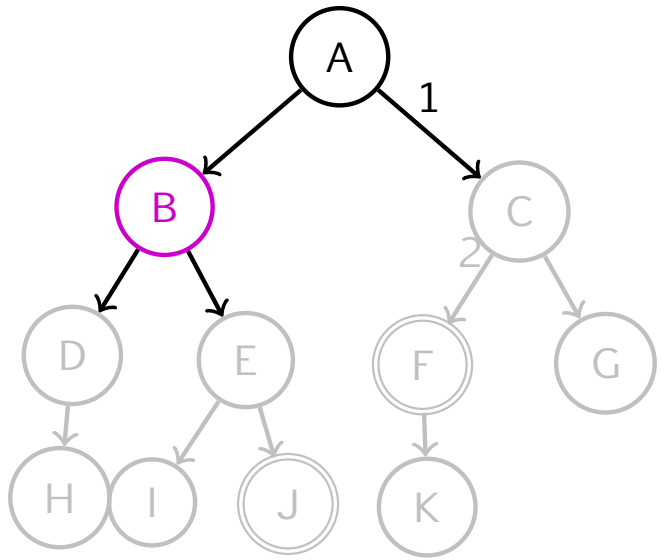
Ch	n	n1	fin	Liste	Sol
{A} {AB}	A B	⊥ E	Faux	{{AB},{AC}} {{AC},{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

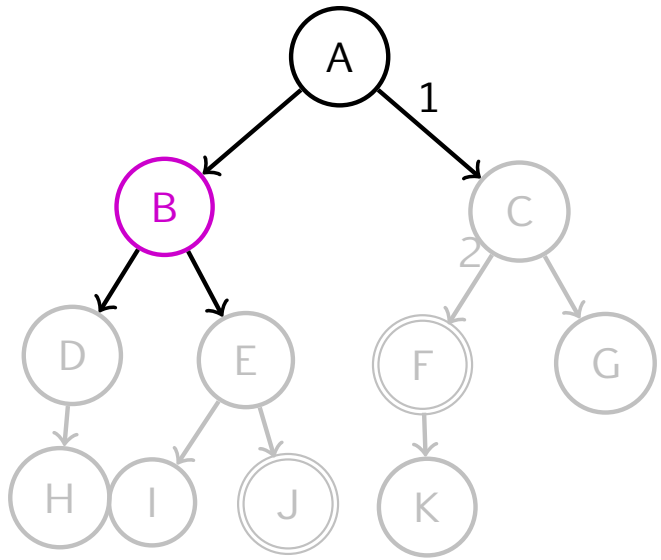
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

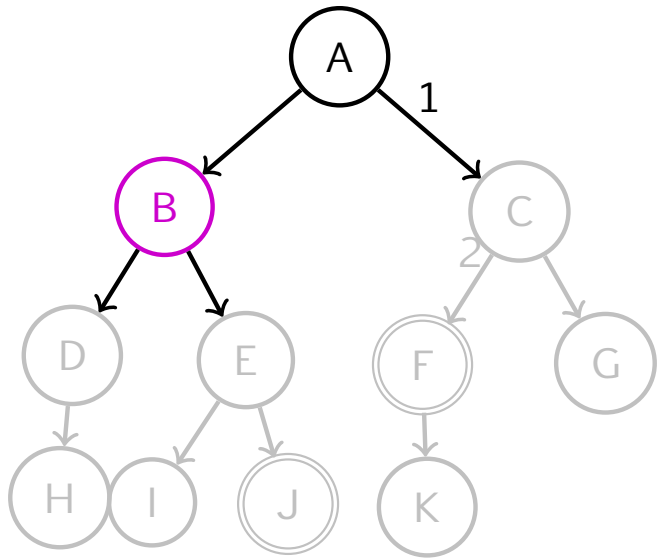
Ch	n	n1	fin	Liste	Sol
{A}	A	$\perp$	Faux	{{AB},{AC}}	
{AB}	B	$\perp$		{{AC},{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

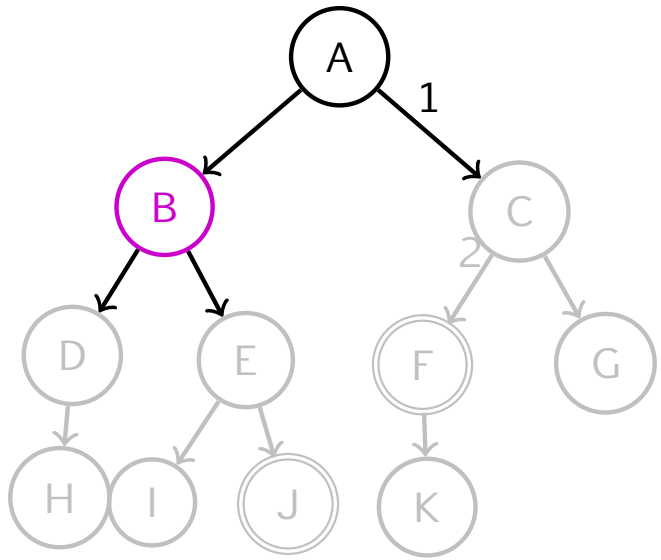
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}					



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

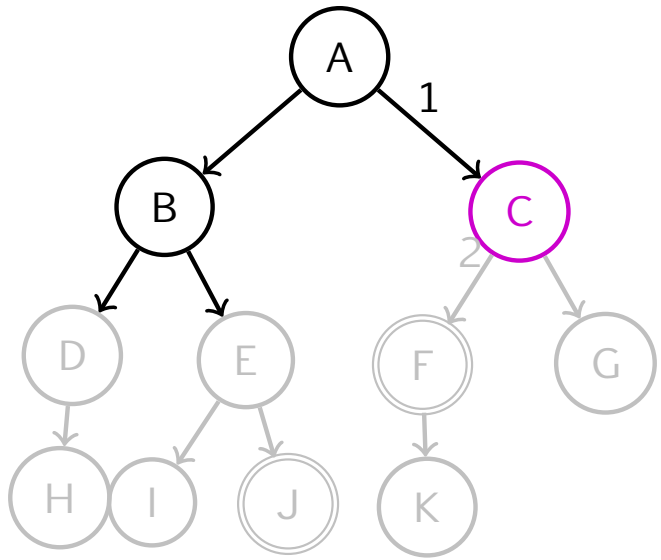
Ch	n	n1	fin	Liste	Sol
{A} {AB} {AC}	A B	⊥ ⊥	Faux	{{AB},{AC}} {{AC},{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

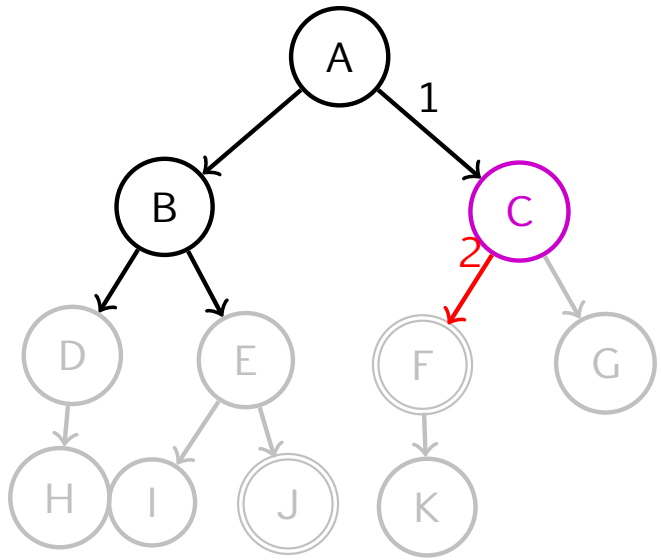
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C			{{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

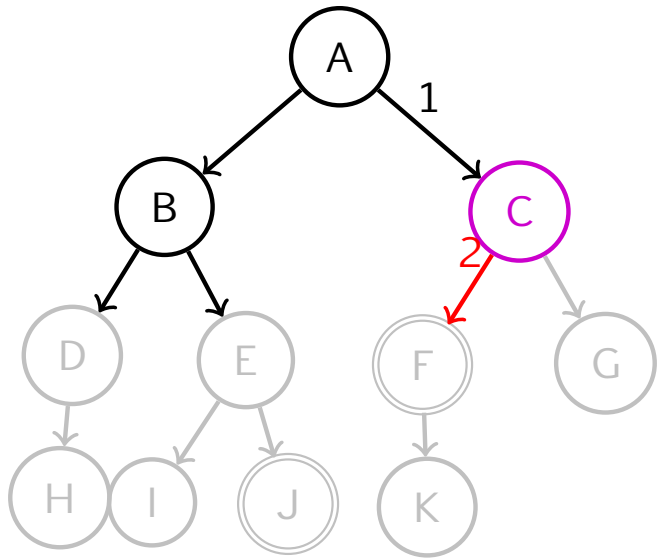
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	F		{{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

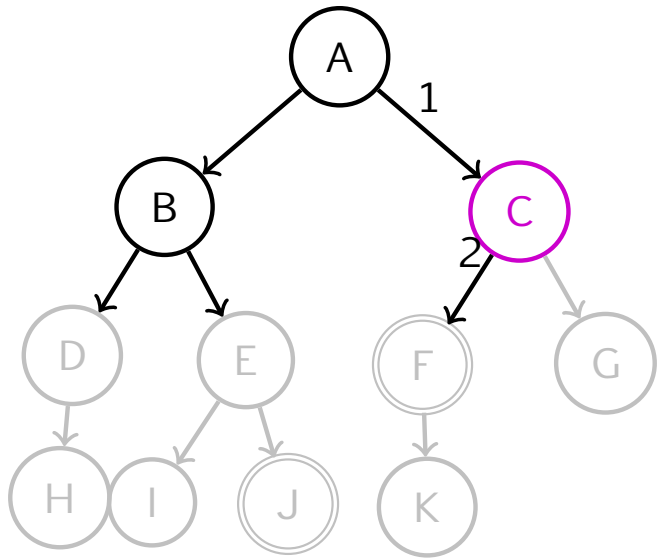
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	F		{{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

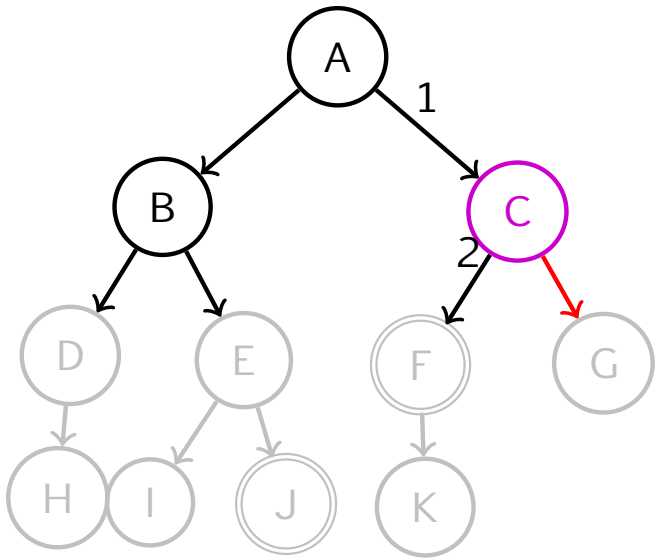
Ch	n	n1	fin	Liste	Sol
{A}	A	$\perp$	Faux	{{AB},{AC}}	
{AB}	B	$\perp$		{{AC},{ABD},{ABE}}	
{AC}	C	F		{{ABD},{ABE}}	



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

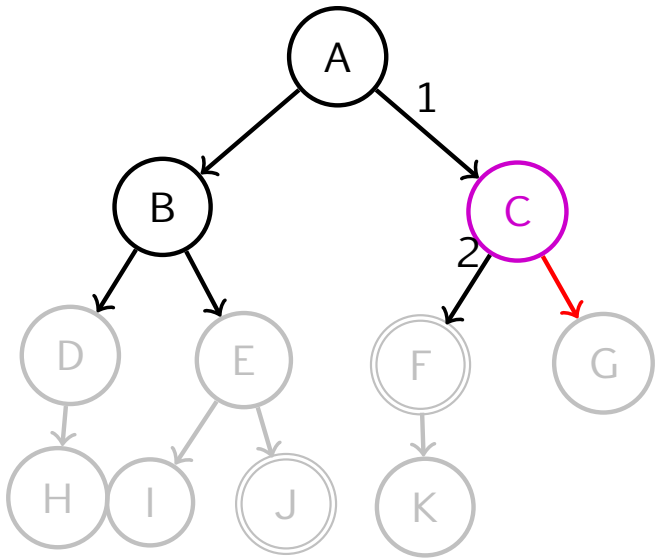
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	F	Vrai	{{ABD},{ABE}}	{A,C,F}



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

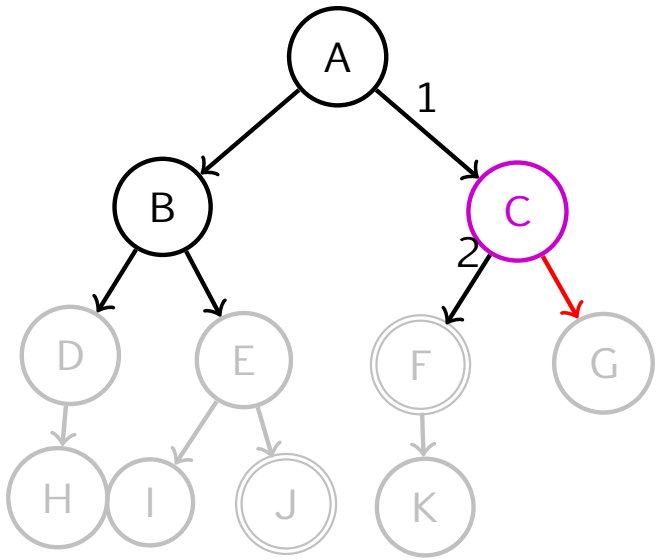
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	G	Vrai	{{ABD},{ABE}}	{A,C,F}



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

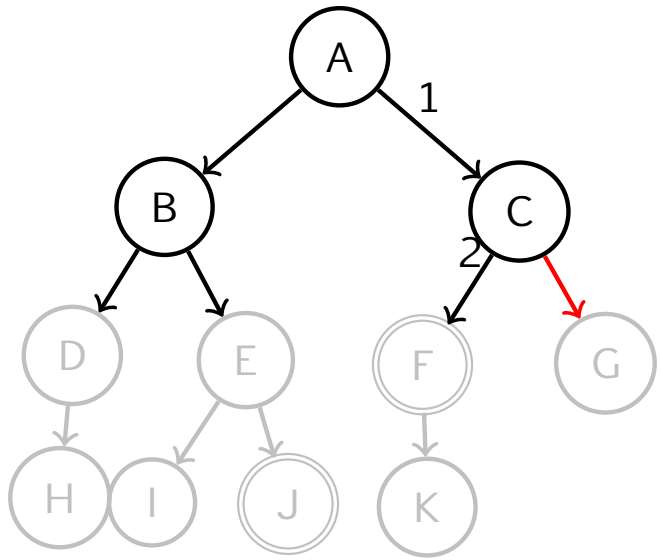
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	G	Vrai	{{ABD},{ABE}}	{A,C,F}



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

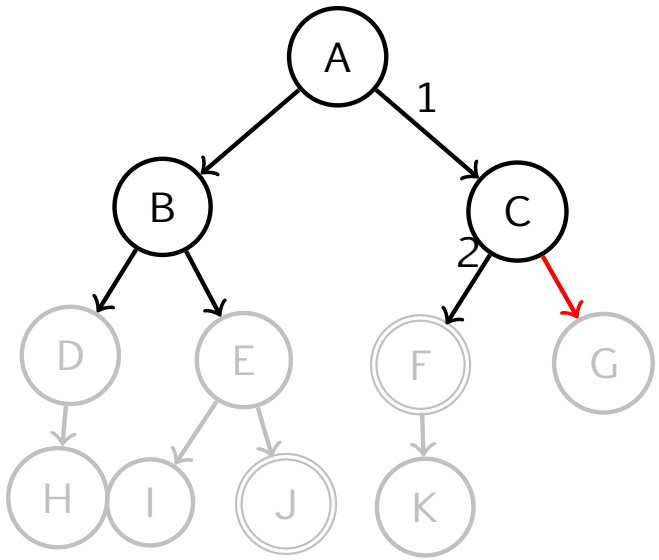
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	G	Vrai	{{ABD},{ABE}}	{A,C,F}



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
  while Ch != {} et non(fin) do
  Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
  n1 <- successeur(n);
  while non(fin) et n1 est valide do
  if n1 est solution then
  Sol Ch Uf {n1}; fin <- Vrai;
  else Liste <- Liste Uf {(Ch Uf {n1})};
  n1 <- successeur(n);
  Ch Premier(Liste);
  return fin;
```

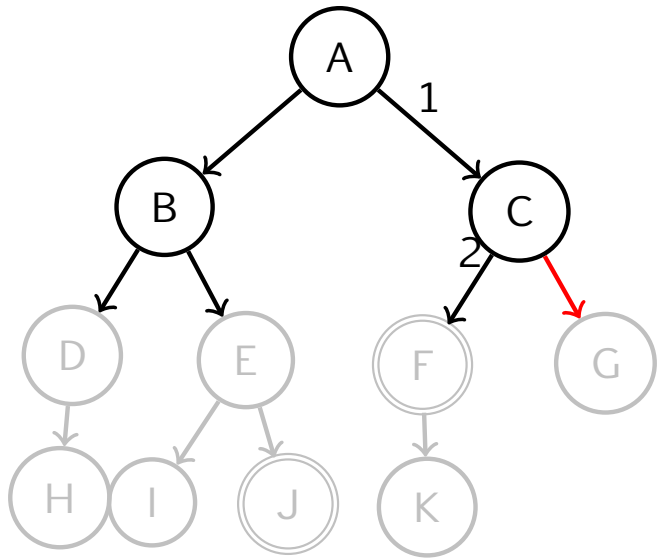
Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	G	Vrai	{{ABD},{ABE}}	{A,C,F}



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

Ch	n	n1	fin	Liste	Sol
{A}	A	⊥	Faux	{{AB},{AC}}	
{AB}	B	⊥		{{AC},{ABD},{ABE}}	
{AC}	C	G	Vrai	{{ABD},{ABE}}	{A,C,F}



Algorithme Largeur(List,Sol)

```
Ch <- Premier(Liste); fin <- Faux;
   while Ch != {} et non(fin) do
   Liste <- Liste / {Ch}; n <- dernierNoeud(Ch);
   n1 <- successeur(n);
   while non(fin) et n1 est valide do
   if n1 est solution then
   Sol Ch Uf {n1}; fin <- Vrai;
   else Liste <- Liste Uf {(Ch Uf {n1})};
   n1 <- successeur(n);
   Ch Premier(Liste);
   return fin;
```

Ch	n	n1	fin	Liste	Sol
{A}	A	$\perp$	Faux	{{AB},{AC}}	
{AB}	B	$\perp$		{{AC},{ABD},{ABE}}	
{AC}	C	G	Vrai	{{ABD},{ABE}}	{A,C,F}