GRÁFELMÉLET

Gráfok bejárásai

3. előadás

Bejárni egy gráfot annyit jelent, hogy egy bizonyos stratégia szerint, a gráf élein haladva, meglátogatni a csúcspontokat. Alapvetően két bejárási algoritmus létezik: **szélességi bejárás** (BFS) és **mélységi bejárás** (DFS).

Bejárni egy gráfot annyit jelent, hogy egy bizonyos stratégia szerint, a gráf élein haladva, meglátogatni a csúcspontokat. Alapvetően két bejárási algoritmus létezik: **szélességi bejárás** (BFS) és **mélységi bejárás** (DFS).

A továbbiakban az alábbi jelöléseket fogjuk használni:

G	– gráf
V(G)	– csúcspontok halmaza: $\{1, 2, \ldots, n\}$
n	– csúcspontok száma
E(G)	– élek halmaza
m	– élek száma
`szomszéd(u)`	– az u csúcspont szomszédainak halmaza
`szín[u]`	– FEHÉR (érintetlen csúcs) SZÜRKE (elért csúcs) FEKETE (elhagyott csúcs)
`apa[u]`	– az u csúcspont apa-csúcsa

1) Szélességi bejárás (BFS – Breadth First Search)

Stratégia: Meglátogatjuk a kiindulási csúcspontot (s), majd ennek szomszédait (azonosítóik növekvő sorrendjében), azután a szomszédok szomszédait és így tovább. A gráf azon éleit, amelyeken keresztül elérjük az egyes csúcspontokat, faéleknek nevezzük. Minden faél egy apa-fiú kapcsolatot képvisel a gráf csúcspontjai között. Egy csúcspont azoknak a csúcspontoknak az apja, amelyek az ő szomszédaiként érhetők el.

1) Szélességi bejárás (BFS – Breadth First Search)

Stratégia: Meglátogatjuk a kiindulási csúcspontot (s), majd ennek szomszédait (azonosítóik növekvő sorrendjében), azután a szomszédok szomszédait és így tovább. A gráf azon éleit, amelyeken keresztül elérjük az egyes csúcspontokat, faéleknek nevezzük. Minden faél egy apa-fiú kapcsolatot képvisel a gráf csúcspontjai között. Egy csúcspont azoknak a csúcspontoknak az apja, amelyek az ő szomszédaiként érhetők el.

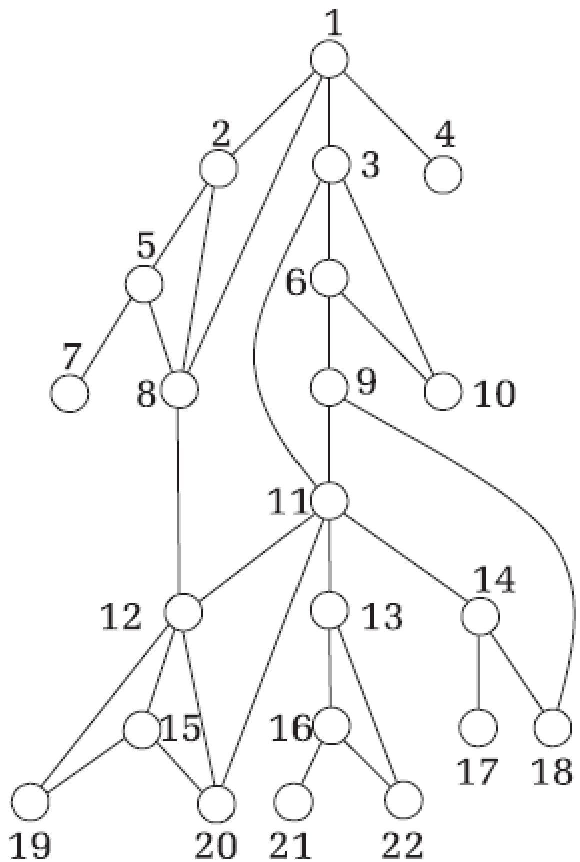
Ebből a szempontból tekintve a dolgokat, úgy is mondhatjuk, hogy a szélességi bejárás generációról generációra halad: meglátogatja s-t, majd s fiait, azután a fiúk fiait és így tovább. A faélek alkotta s gyökerű fát **szélességi fá**nak nevezzük.

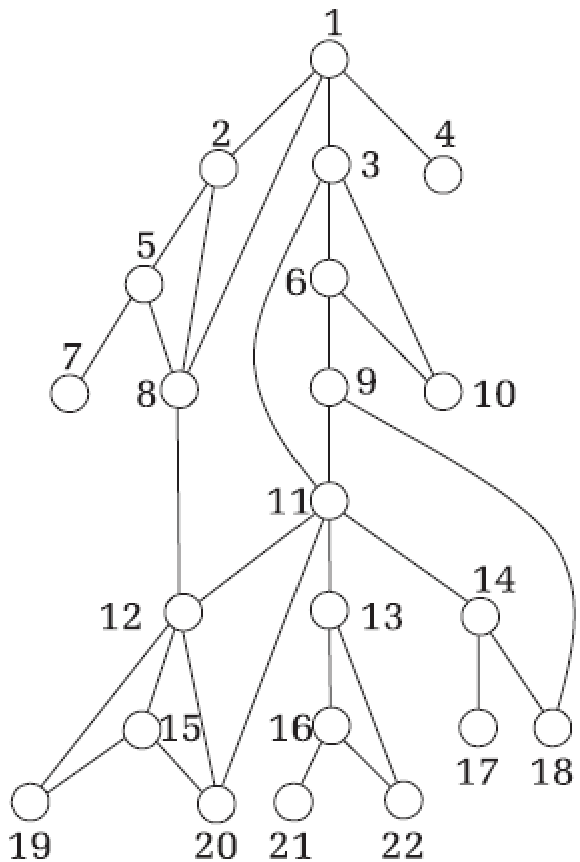
1) Szélességi bejárás (BFS – Breadth First Search)

Stratégia: Meglátogatjuk a kiindulási csúcspontot (s), majd ennek szomszédait (azonosítóik növekvő sorrendjében), azután a szomszédok szomszédait és így tovább. A gráf azon éleit, amelyeken keresztül elérjük az egyes csúcspontokat, faéleknek nevezzük. Minden faél egy apa-fiú kapcsolatot képvisel a gráf csúcspontjai között. Egy csúcspont azoknak a csúcspontoknak az apja, amelyek az ő szomszédaiként érhetők el.

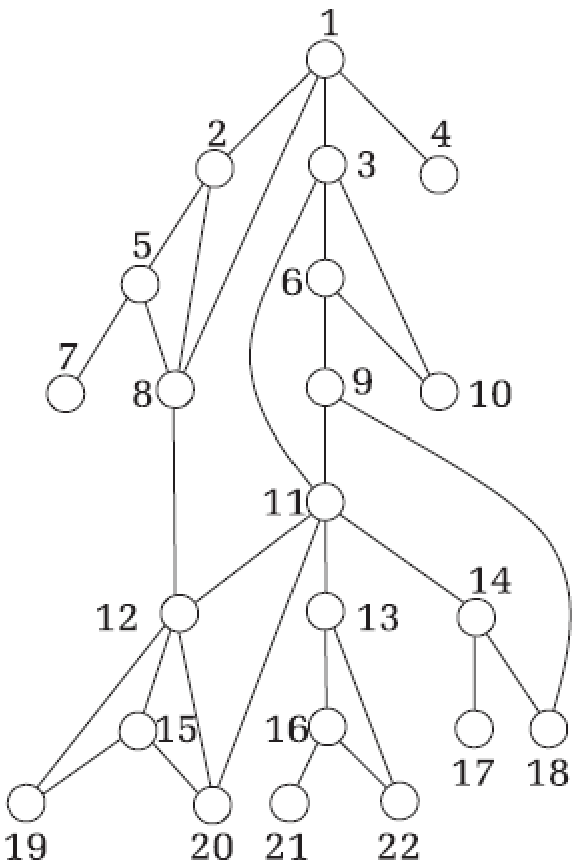
Ebből a szempontból tekintve a dolgokat, úgy is mondhatjuk, hogy a szélességi bejárás generációról generációra halad: meglátogatja s-t, majd s fiait, azután a fiúk fiait és így tovább. A faélek alkotta s gyökerű fát **szélességi fá**nak nevezzük.

A szélességi bejárás csak azokat a csúcspontokat éri el, amelyek ahhoz a komponenshez tartoznak, melynek s is része.

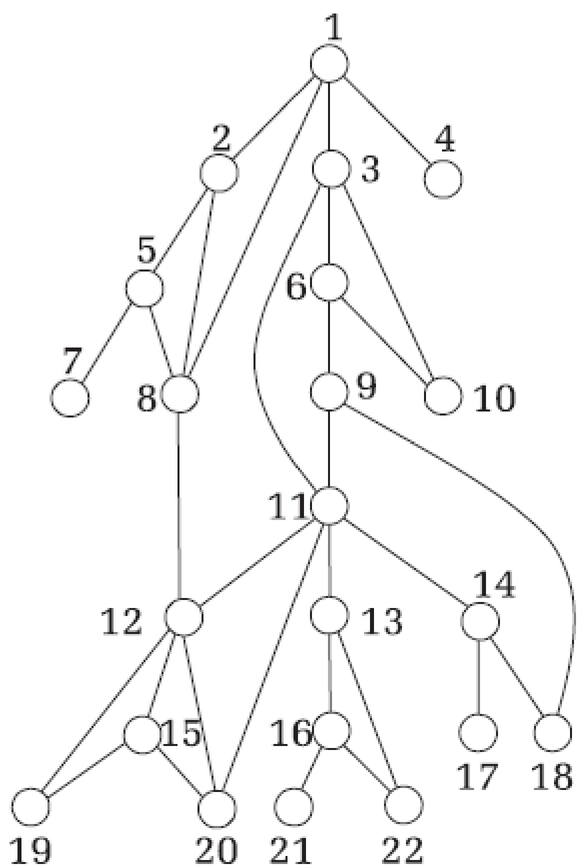

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22



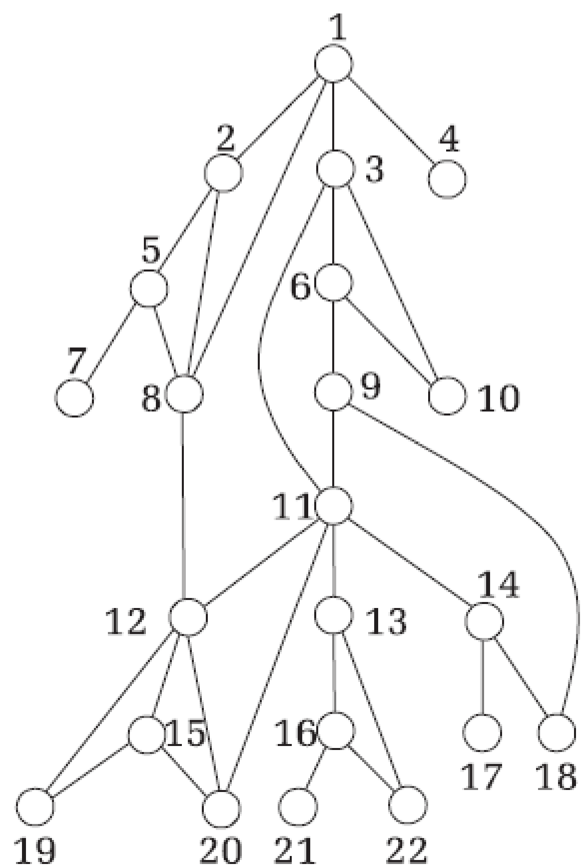
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21



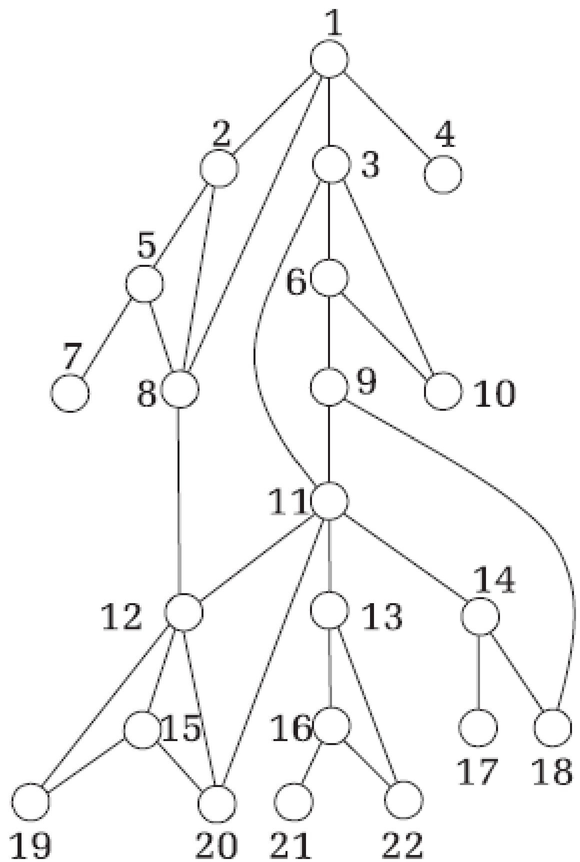
1,



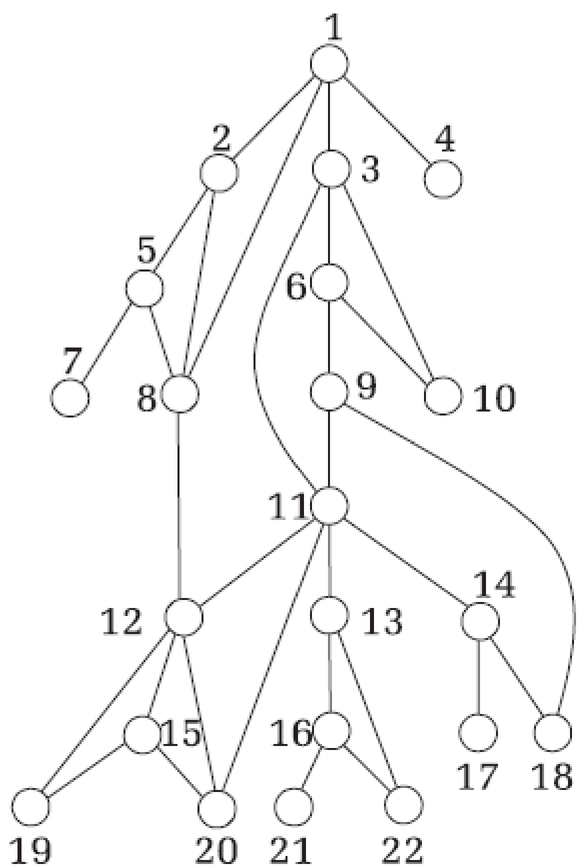
1, 2, 3, 4, 8,



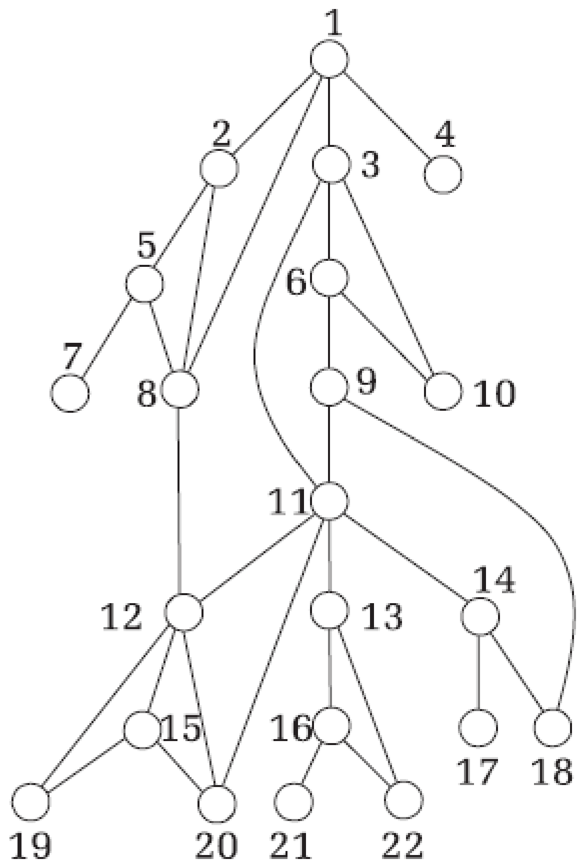
1, 2, 3, 4, 8, 5,



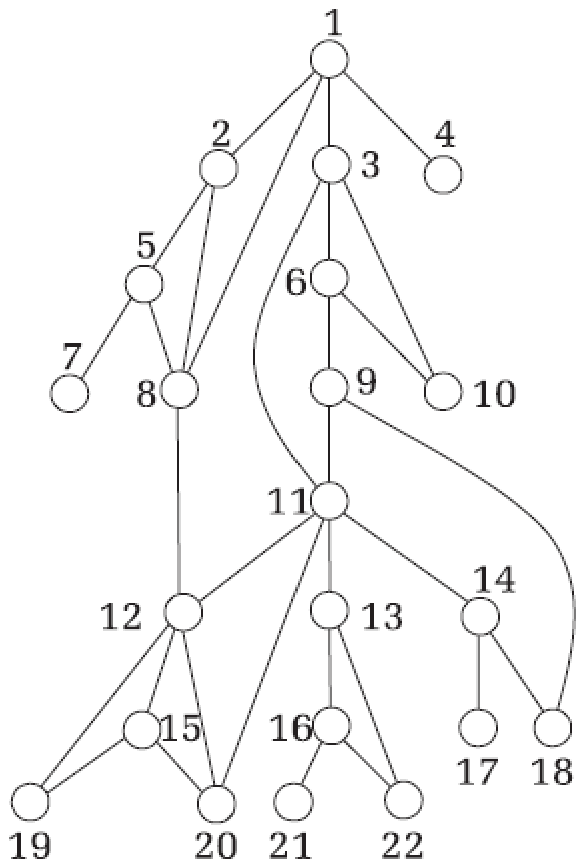
1, 2, 3, 4, 8, 5, 6, 10, 11,



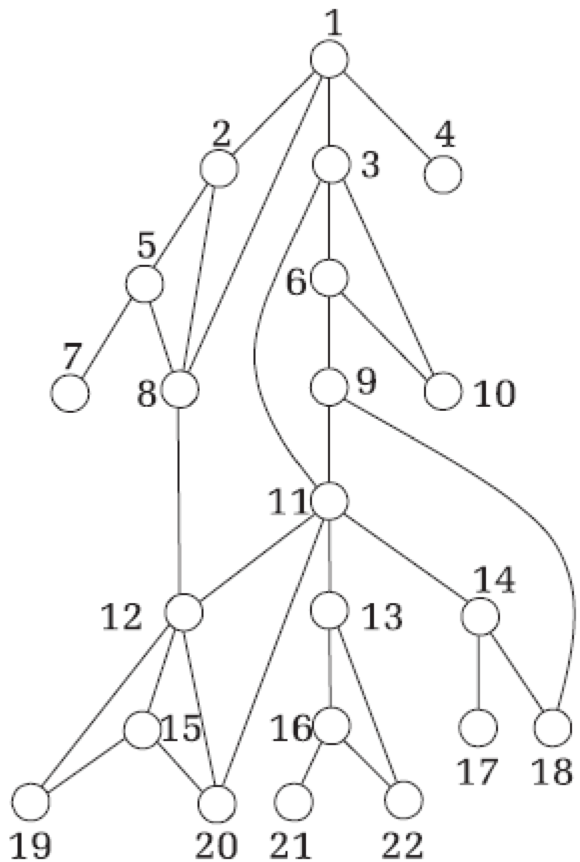
1, 2, 3, 4, 8, 5, 6, 10, 11,



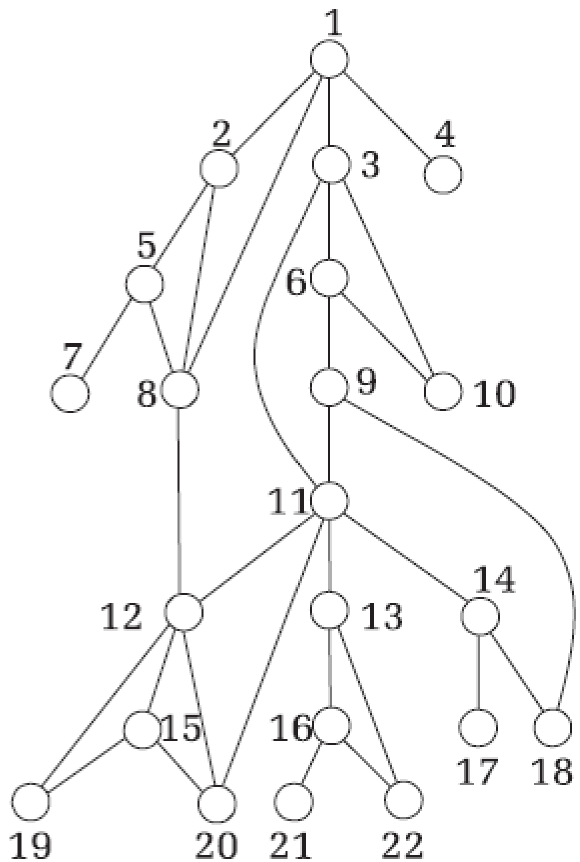
1, 2, 3, 4, 8, 5, 6, 10, 11, 12,



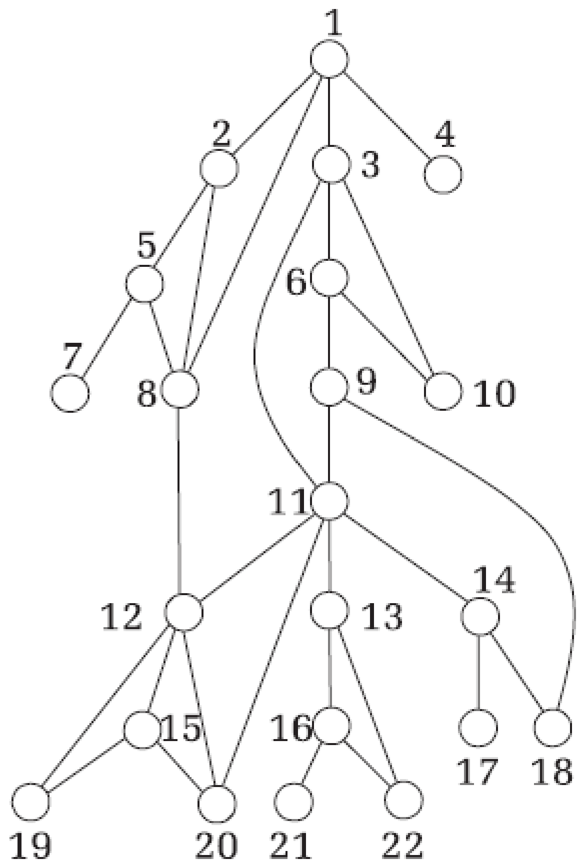
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7,



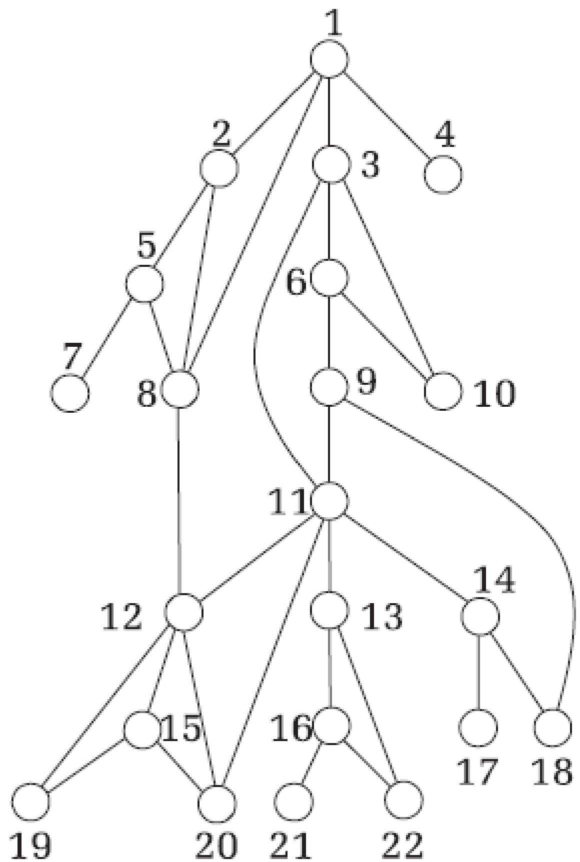
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9,



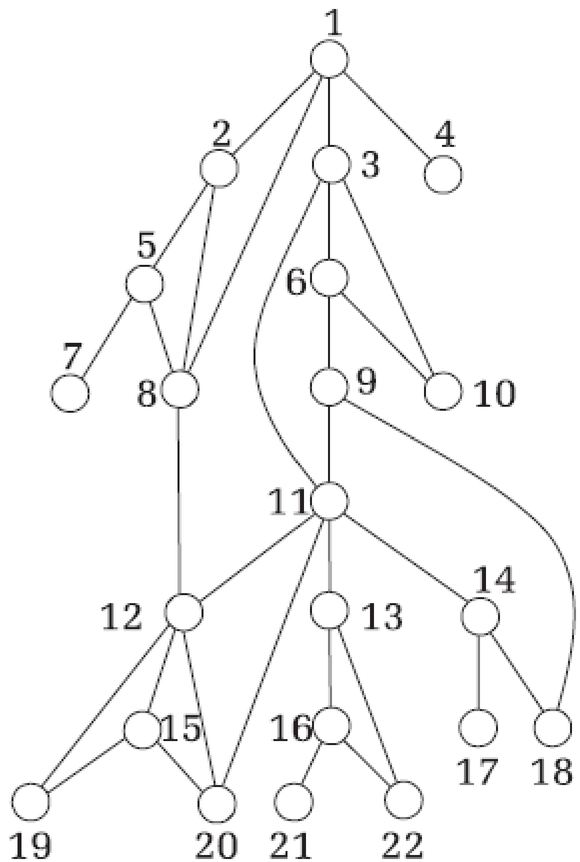
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9,



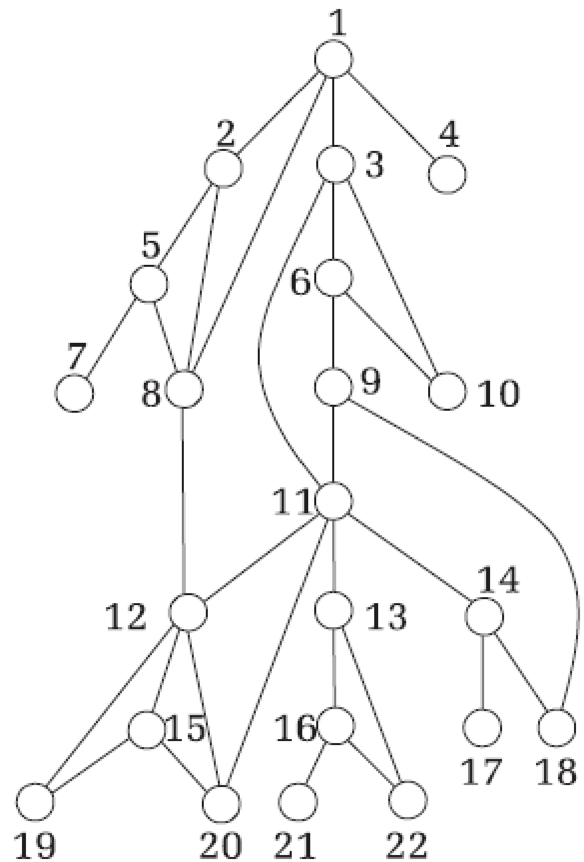
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20,



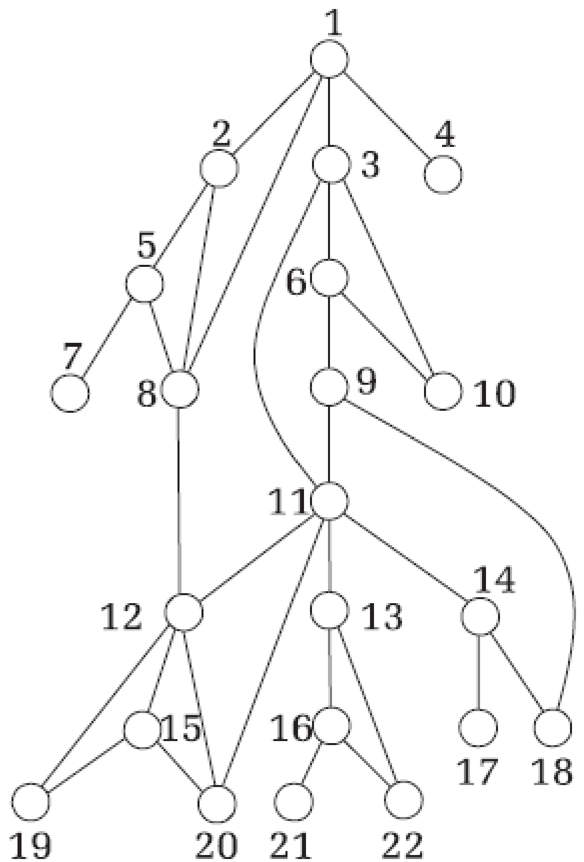
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19,



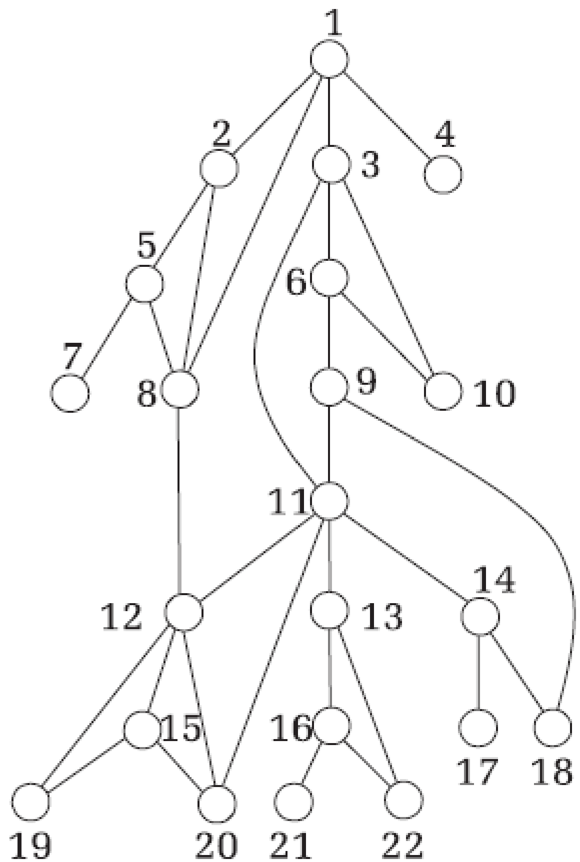
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19,



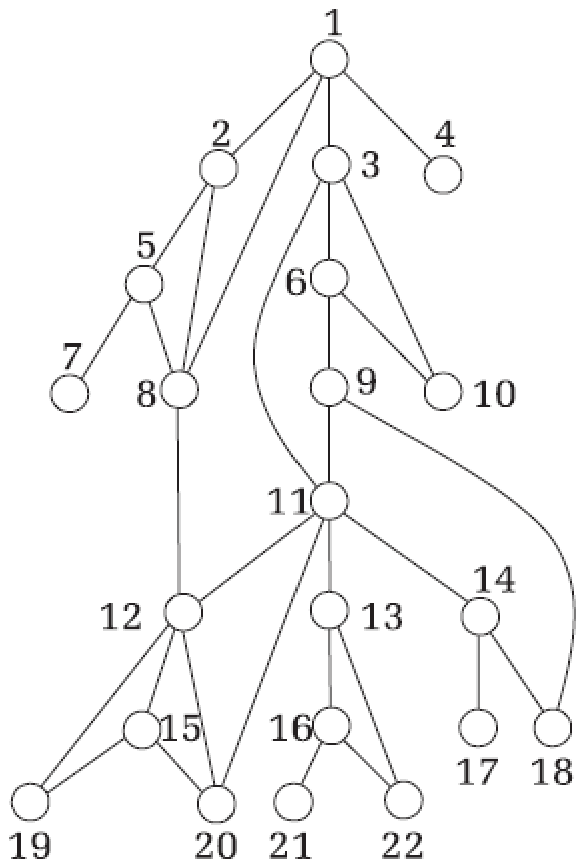
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18,



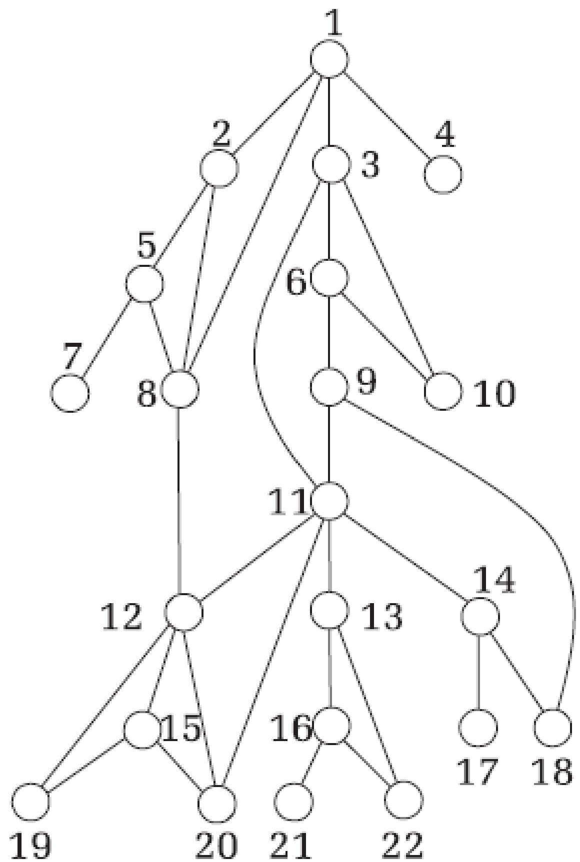
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22,



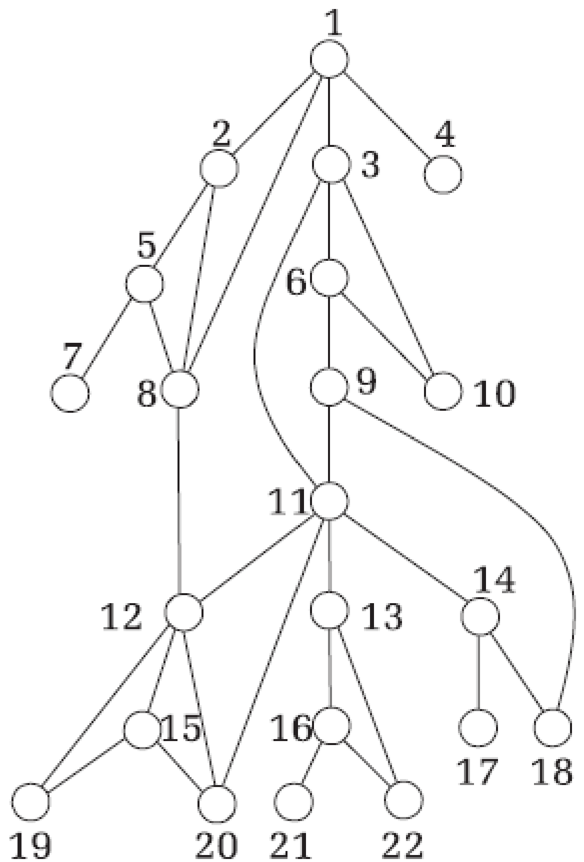
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17,



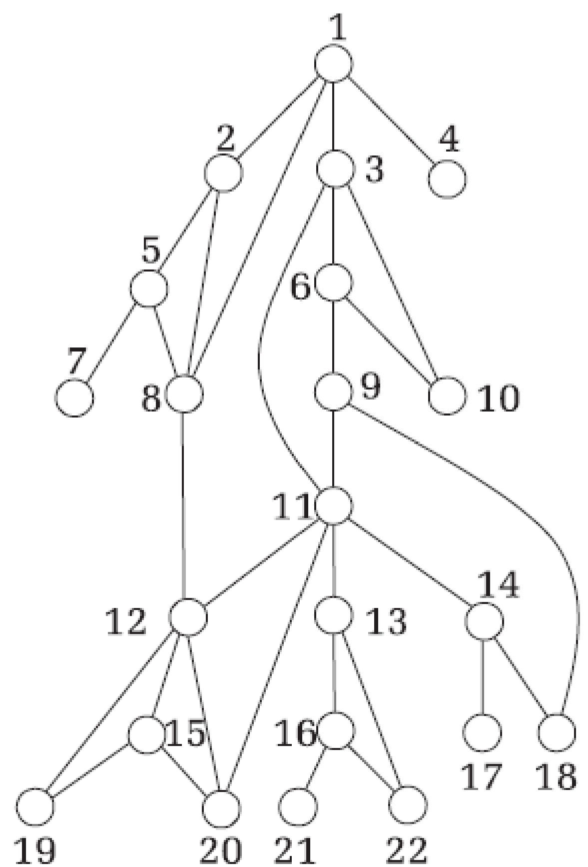
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17,



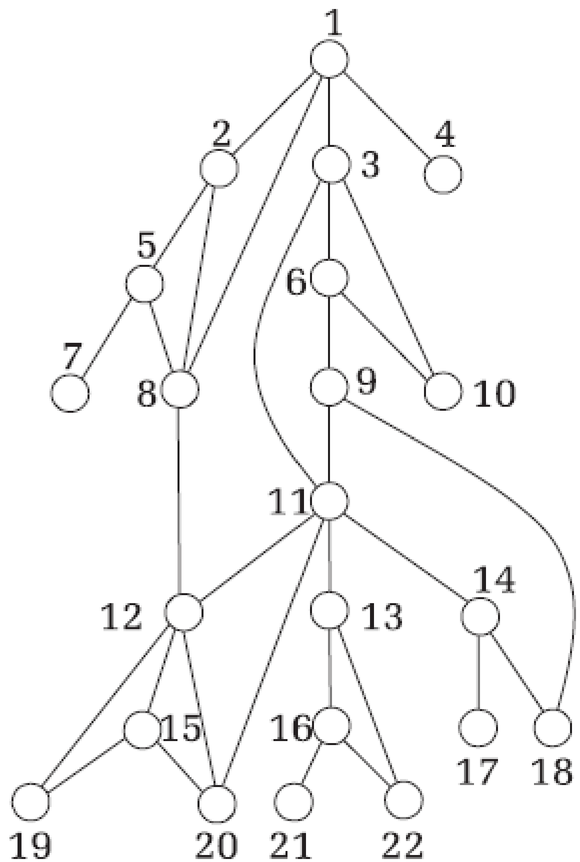
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17,



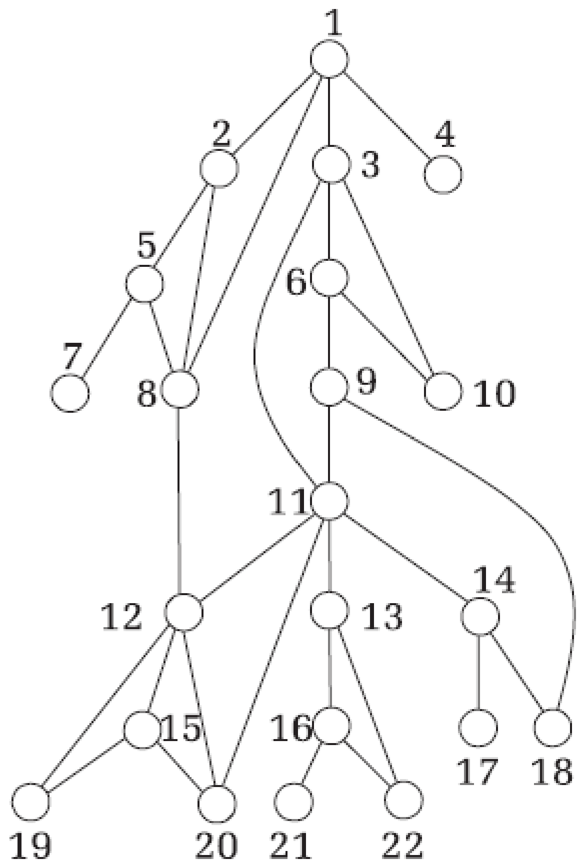
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17,



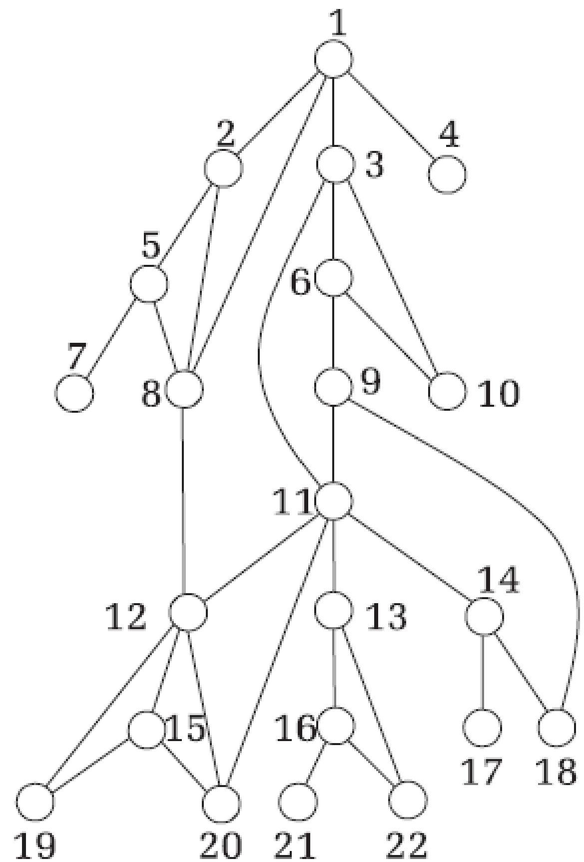
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17,



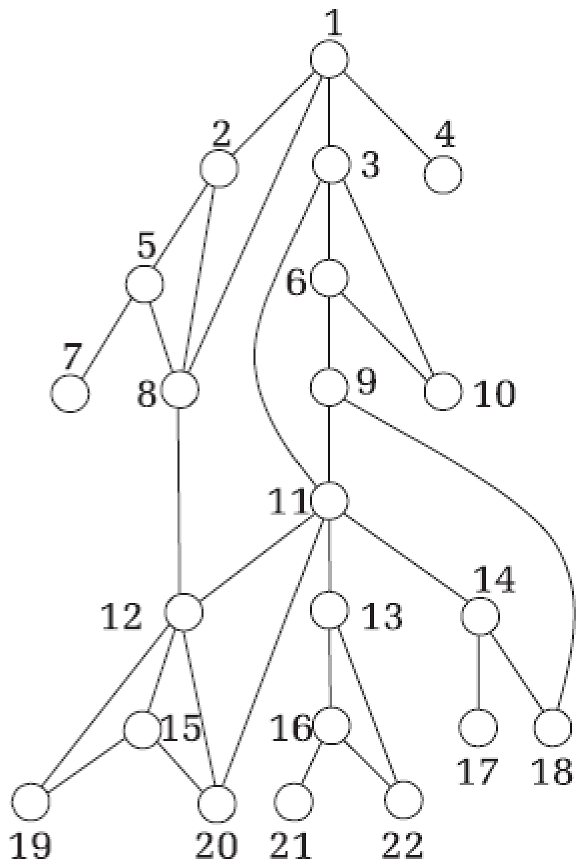
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21



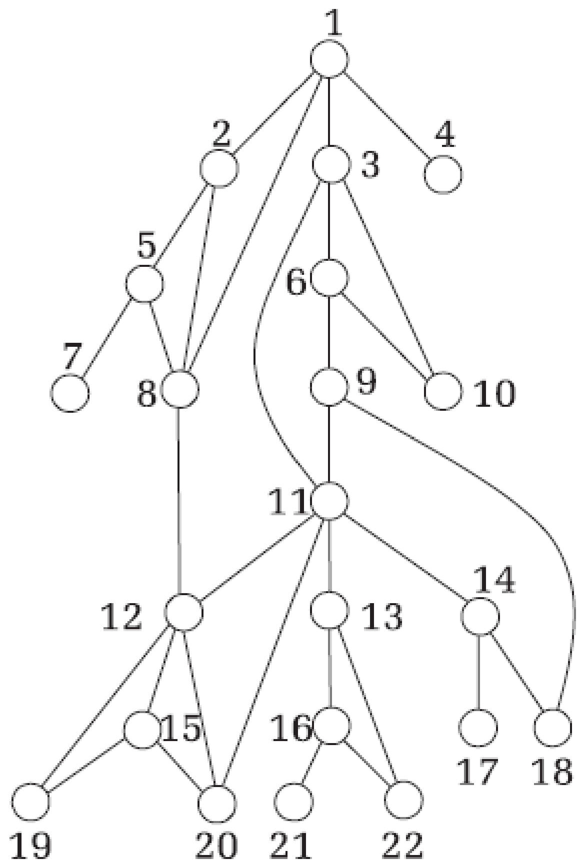
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21



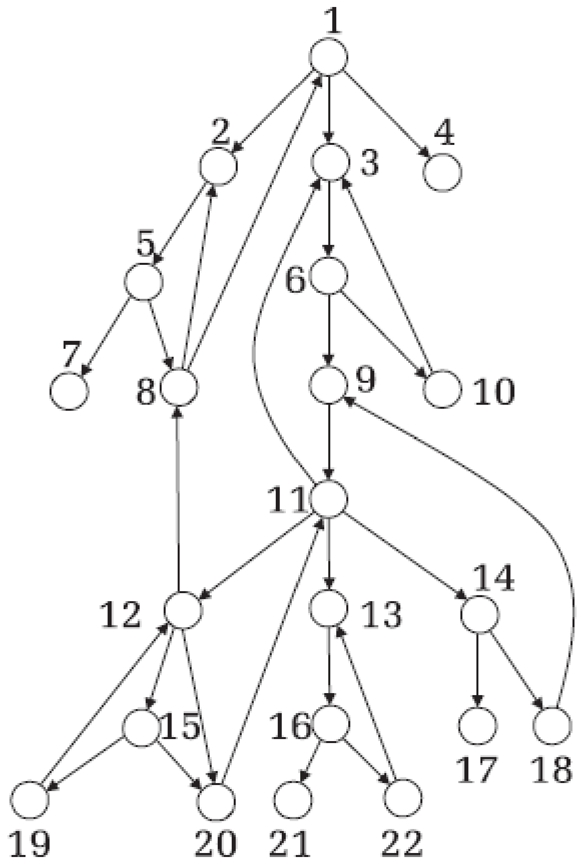
1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21

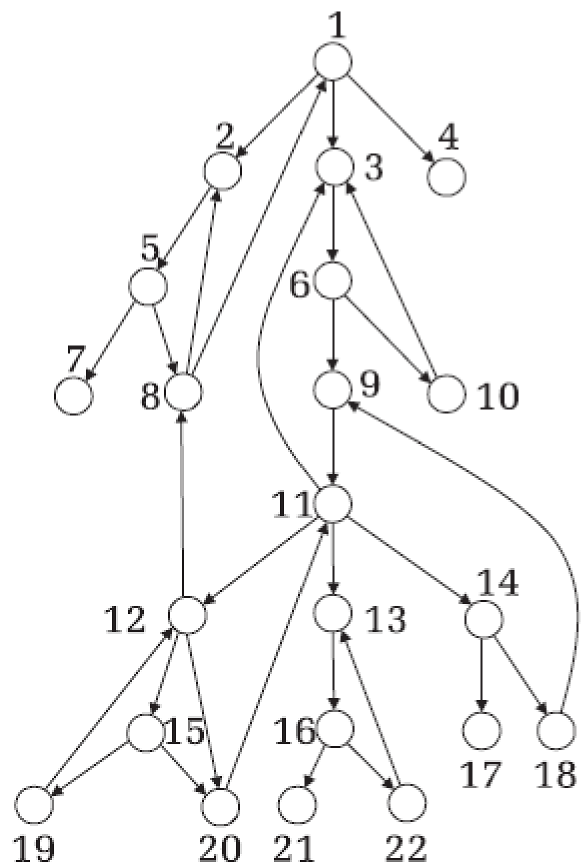


1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21



1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22



1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 16, 17, 18, 19, 21, 22

1) Szélességi bejárás (BFS – Breadth First Search)

<u>Algoritmus</u>: Egy sorszerkezetet (Q) használunk. Kezdetben a sor egyedül a kiindulási csúcspontot (s) tartalmazza. Minden lépésben először a sorelső pont (MÁSOL_SORELSŐ függvény) még meg nem látogatott szomszédait (a FEHÉR színűeket) betesszük a sor végére (BETESZ_SORVÉGÉRE eljárás), majd töröljük az illető pontot a sorból (TÖRÖL_SORELSŐ eljárás).

1) Szélességi bejárás (BFS – Breadth First Search)

Algoritmus: Egy sorszerkezetet (Q) használunk. Kezdetben a sor egyedül a kiindulási csúcspontot (s) tartalmazza. Minden lépésben először a sorelső pont (MÁSOL_SORELSŐ függvény) még meg nem látogatott szomszédait (a FEHÉR színűeket) betesszük a sor végére (BETESZ_SORVÉGÉRE eljárás), majd töröljük az illető pontot a sorból (TÖRÖL_SORELSŐ eljárás).

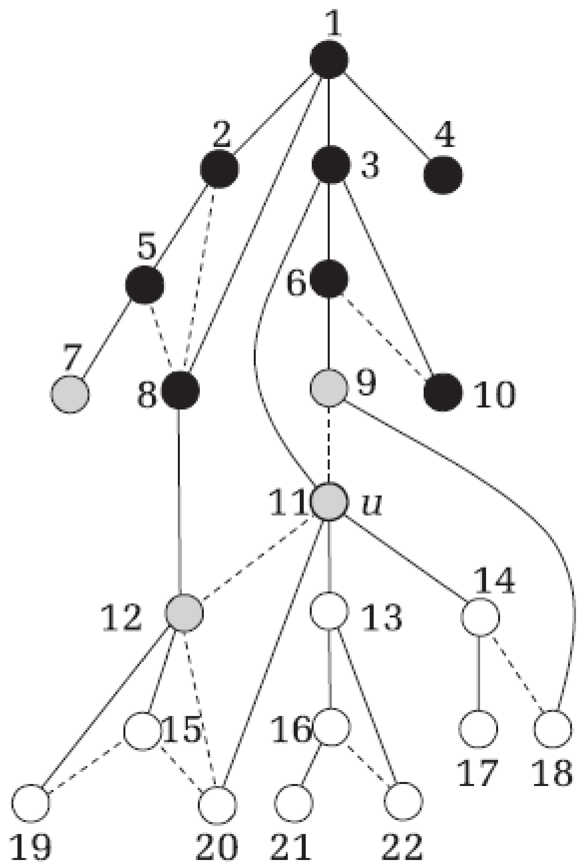
A bejárás alatt nyilvántartjuk, hogy minden egyes (fiú) csúcspontot mely (apa) csúcspont szomszédjaként értünk el (azaz melyik sorelső pont szomszédjaként került be az illető pont a Q sor végére). Ezt a nyilvántartást az `apa` tömb segítségével végezzük.

1) Szélességi bejárás (BFS – Breadth First Search)

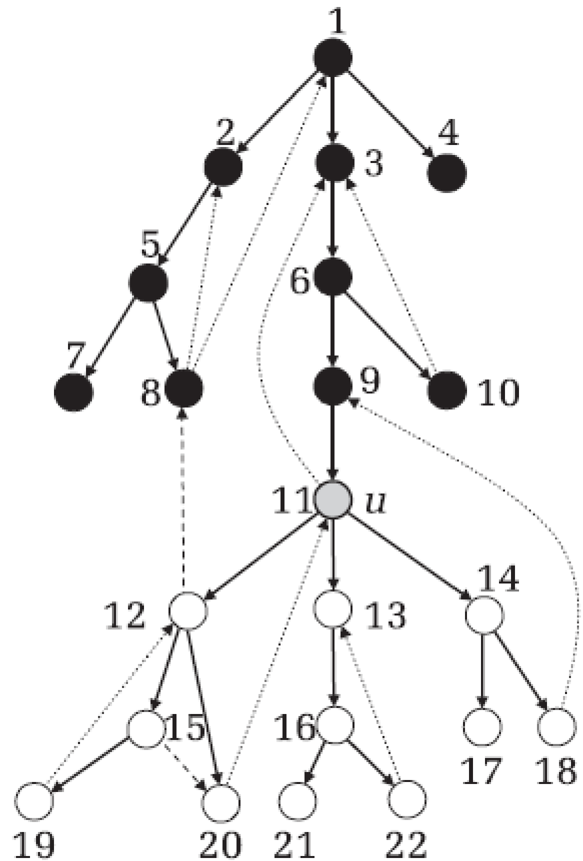
Algoritmus: Egy sorszerkezetet (Q) használunk. Kezdetben a sor egyedül a kiindulási csúcspontot (s) tartalmazza. Minden lépésben először a sorelső pont (MÁSOL_SORELSŐ függvény) még meg nem látogatott szomszédait (a FEHÉR színűeket) betesszük a sor végére (BETESZ_SORVÉGÉRE eljárás), majd töröljük az illető pontot a sorból (TÖRÖL_SORELSŐ eljárás).

A bejárás alatt nyilvántartjuk, hogy minden egyes (fiú) csúcspontot mely (apa) csúcspont szomszédjaként értünk el (azaz melyik sorelső pont szomszédjaként került be az illető pont a Q sor végére). Ezt a nyilvántartást az `apa` tömb segítségével végezzük.

A `szín` tömböt arra használjuk, hogy nyilvántartsuk a pontok bejárás alatti státusát: érintetlen pont (FEHÉR, még nem került be a Q sorba), elért pont (SZÜRKE, bent van a Q sorban), elhagyott pont (FEKETE, eltávolítottuk a Q sorból). A szélességi sorrendet a pontok elérési (szürkévé válási) sorrendje jelenti.

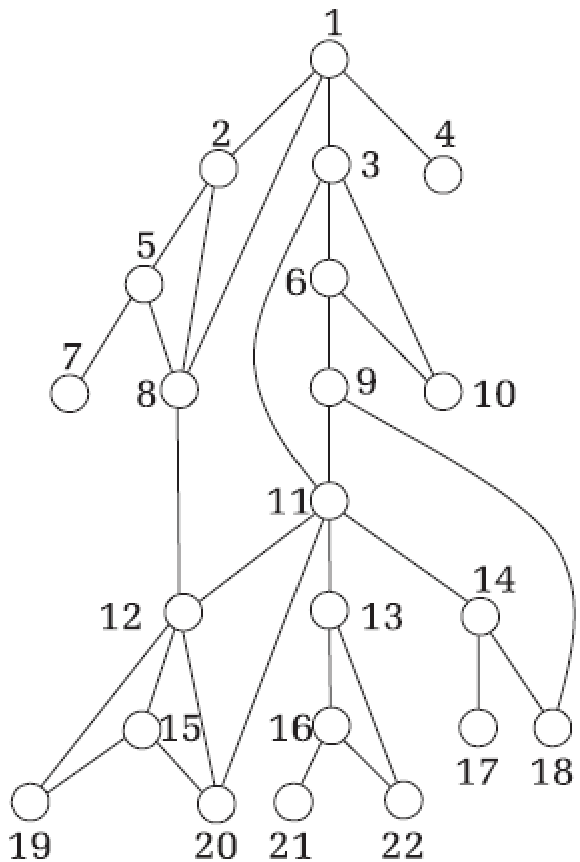


1	2	3	4	8	5	6	10	11	12	7	9	13	14	20	15	19	18	16	22	17	21



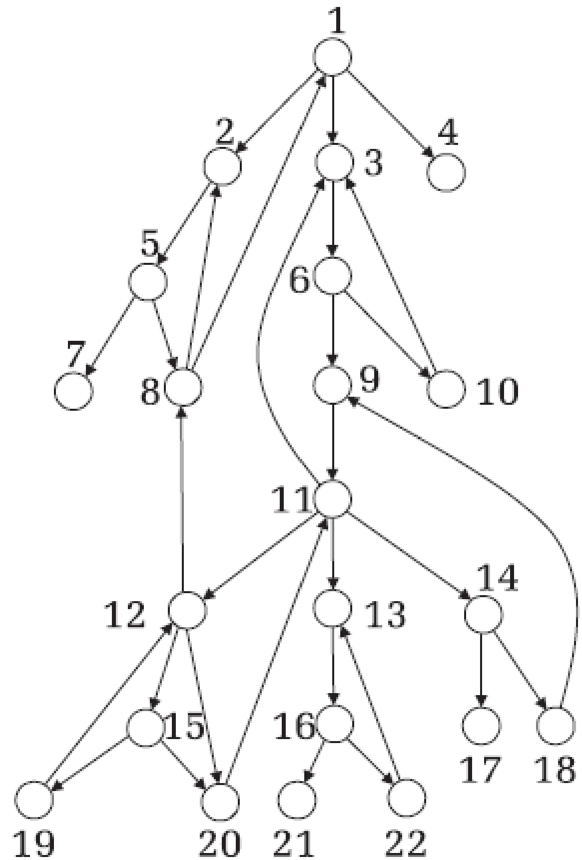
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	20	16	17	18	19	21	22

A `d` tömbben azt tároljuk, hogy milyen távolságra vannak az egyes csúcspontok a kiindulási csúcsponttól (az út hossza alatt az utat alkotó élek számát értjük). Minden (fiú) pont egy éllel távolabb esik s-től, mint az (apa) pontja.



1, 2, 3, 4, 8, 5, 6, 10, 11, 12, 7, 9, 13, 14, 20, 15, 19, 18, 16, 22, 17, 21

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
d	0	1	1	1	2	2	3	1	3	2	2	2	3	3	3	4	4	4	3	3	5	4
apa	0	1	1	1	2	3	5	1	6	3	3	8	11	11	12	13	14	9	12	11	16	13



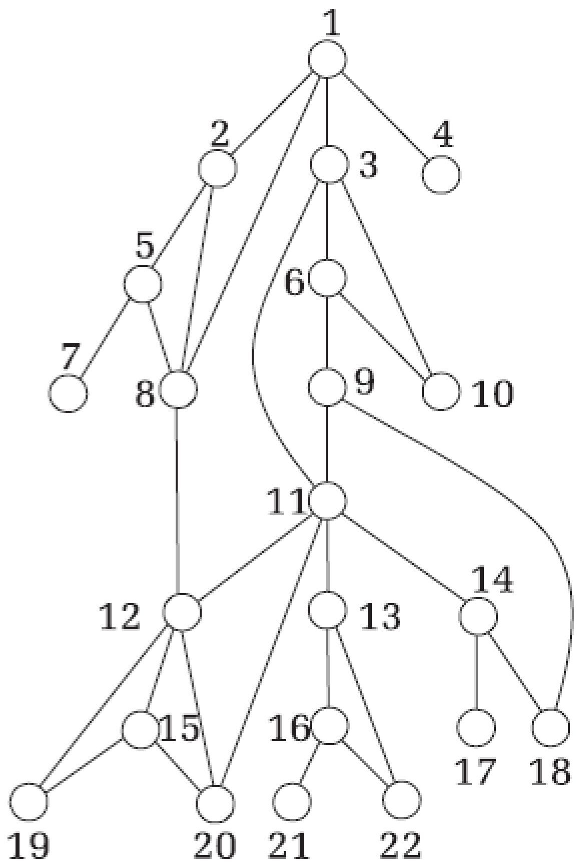
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 16, 17, 18, 19, 21, 22

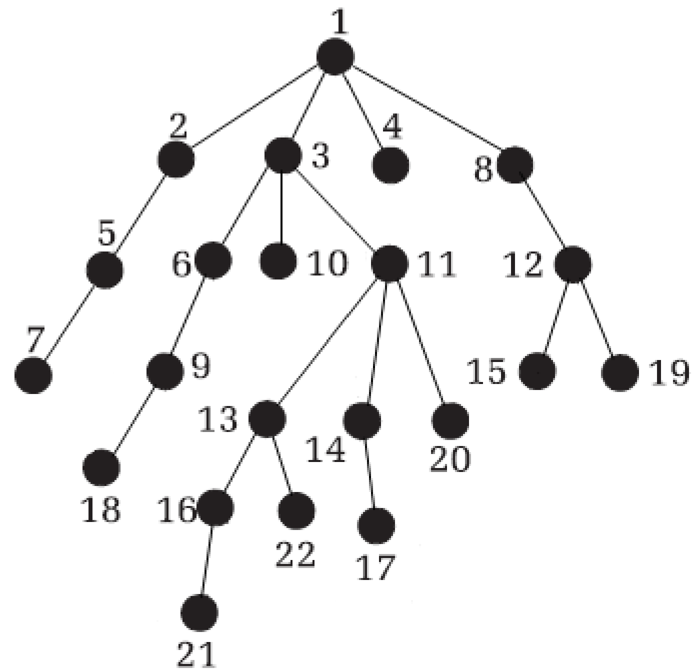
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
d	0	1	1	1	2	2	3	3	3	3	4	5	5	5	6	6	6	6	7	6	7	7
apa	0	1	1	1	2	3	5	5	6	6	9	11	11	11	12	13	14	14	15	12	16	16

```
eljárás SZÉLESSÉGI_BEJÁRÁS(G,s)
   minden u ∈ V(G)-{s} végezd
      szín[u] ← FEHÉR
      apa[u] ← 0
      d[u] ← ∞
   vége minden
   szín[s] ← SZÜRKE
   apa[s] ← 0
   d[s] ← 0
   Q ← {s}
   amíg Q ≠ ∅ végezd
      u ← MÁSOL_SORELSŐ(Q)
      kiír: u
      minden v ∈ Szomszéd(u) végezd
         ha szín[v] = FEHÉR akkor
            szín[v] ← SZÜRKE
            d[v] ← d[u] + 1
            apa[v] ← u
            BETESZ_SORVÉGÉRE(Q,v)
         vége ha
      vége minden
      TÖRÖL_SORELSŐ(Q)
      szín[u] ← FEKETE
   vége amíg
vége SZÉLESSÉGI_BEJÁRÁS
```

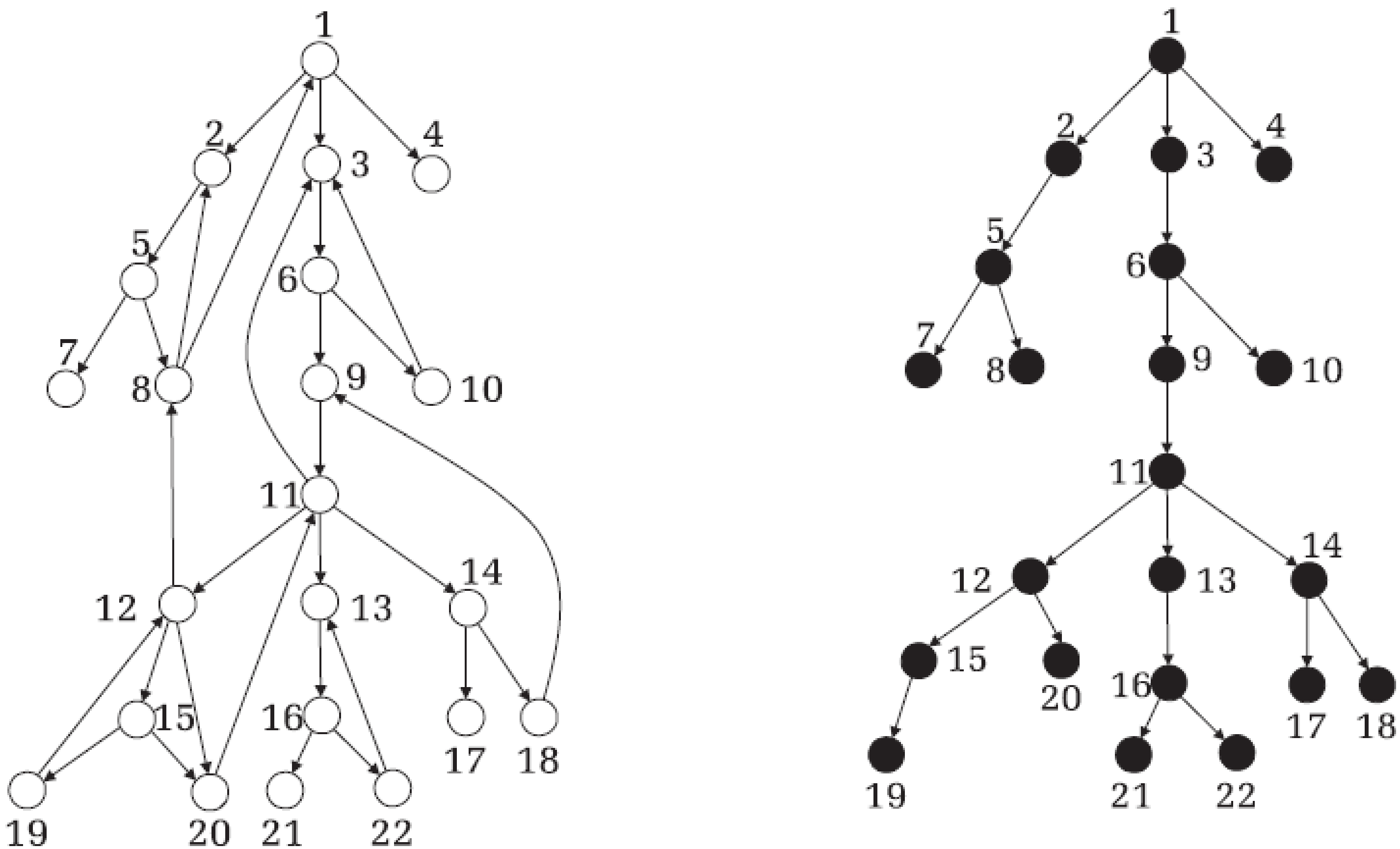
```
eljárás SZÉLESSÉGI_BEJÁRÁS(G,s)
   minden u ∈ V(G)-{s} végezd
      szín[u] ← FEHÉR
      apa[u] ← 0
      d[u] ← ∞
   vége minden
   szín[s] ← SZÜRKE
   apa[s] ← 0
   d[s] ← 0
   Q ← {s}
   amíg Q ≠ ∅ végezd
      u ← MÁSOL_SORELSŐ(Q)
      kiír: u
      minden v ∈ Szomszéd(u) végezd
         ha szín[v] = FEHÉR akkor
            szín[v] ← SZÜRKE
            d[v] ← d[u] + 1
            apa[v] ← u
            BETESZ_SORVÉGÉRE(Q,v)
         vége ha
      vége minden
      TÖRÖL_SORELSŐ(Q)
      szín[u] ← FEKETE
   vége amíg
vége SZÉLESSÉGI_BEJÁRÁS
```

A SZÉLESSÉGI_BEJÁRÁS algoritmus bonyolultsága $O(n+m)$.
(ha a gráf csúcslistával van tárolva)





Irányítatlan gráf szélességi fája



Irányított gráf szélességi fája

Megjegyzés:

Úgy is fogalmazhatnánk, hogy szélességi bejárás esetén mindig abból a szürke pontból lépünk tovább, amelyik legkorábban vált szürkévé. Mivel erre az elvre épül a sorszerkezet is, ezért logikus, hogy ezt használjuk, mint adatszerkezetet, a szélességi bejárás implementálásakor. Mivel csak fehér pontok irányába lépünk tovább, nyilvánvaló, hogy a bejárt élek (feszítő)fát alkotnak.

Megjegyzés:

Úgy is fogalmazhatnánk, hogy szélességi bejárás esetén mindig abból a szürke pontból lépünk tovább, amelyik legkorábban vált szürkévé. Mivel erre az elvre épül a sorszerkezet is, ezért logikus, hogy ezt használjuk, mint adatszerkezetet, a szélességi bejárás implementálásakor. Mivel csak fehér pontok irányába lépünk tovább, nyilvánvaló, hogy a bejárt élek (feszítő)fát alkotnak.

E fát nagyon gazdaságosan tárolja az `apa` tömb. Ennek az „ára” az, hogy a fa élei ellentétes irányítással kerülnek eltárolásra. Ezért van szükség rekurzív eljárásra, ha a gyökértől egy adott levélhez vezető utat az irányításának megfelelően szeretnénk kiírni.

Megjegyzés:

Úgy is fogalmazhatnánk, hogy szélességi bejárás esetén mindig abból a szürke pontból lépünk tovább, amelyik legkorábban vált szürkévé. Mivel erre az elvre épül a sorszerkezet is, ezért logikus, hogy ezt használjuk, mint adatszerkezetet, a szélességi bejárás implementálásakor. Mivel csak fehér pontok irányába lépünk tovább, nyilvánvaló, hogy a bejárt élek (feszítő)fát alkotnak.

E fát nagyon gazdaságosan tárolja az `apa` tömb. Ennek az „ára" az, hogy a fa élei ellentétes irányítással kerülnek eltárolásra. Ezért van szükség rekurzív eljárásra, ha a gyökértől egy adott levélhez vezető utat az irányításának megfelelően szeretnénk kiírni.

Az `apa` tömb alapján tehát egy rekurzív eljárással bármely csúcspontra meghatározható az s-ből hozzá vezető legrövidebb út.

```
eljárás Kiír_LRU(i,apa[])
   ha apa[i]≠0 akkor
      Kiír_LRU(apa[i],apa[])
   vége ha
   kíir: i
vége Kiír_LRU
```

Elkerülhetetlen pontok

Legyen G egy körmentes irányított gráf, amelynek van egy forrása és egy nyelője, és minden út elvezet a forrásból a nyelőbe. Egy pontot elkerülhetetlennek nevezünk, ha minden forrásból nyelőbe vezető út áthalad rajta.

Elkerülhetetlen pontok

Legyen G egy körmentes irányított gráf, amelynek van egy forrása és egy nyelője, és minden út elvezet a forrásból a nyelőbe. Egy pontot elkerülhetetlennek nevezünk, ha minden forrásból nyelőbe vezető út áthalad rajta.

Szélességi bejárással meghatározhatók egy – az előbbi feltételeknek eleget tevő – gráf elkerülhetetlen pontjai. Nem nehéz belátni, hogy ha a szélességi bejárás során mindig egy olyan szürke csúcspontból lépünk tovább, amelynek már minden be-szomszédja fekete, akkor azok az elkerülhetetlen pontok, amelyek egy adott pillanatban egymaguk maradnak a Q sorban.

Az élek osztályozása a szélességi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók (ha a gráf nem összefüggő, akkor a bejárt komponens éleire vonatkoztatjuk), minden él akkor kap besorolást, amikor a szélességi bejárás először érzékeli a létezését:

Az élek osztályozása a szélességi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók (ha a gráf nem összefüggő, akkor a bejárt komponens éleire vonatkoztatjuk), minden él akkor kap besorolást, amikor a szélességi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a szélességi fának.

Az élek osztályozása a szélességi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók (ha a gráf nem összefüggő, akkor a bejárt komponens éleire vonatkoztatjuk), minden él akkor kap besorolást, amikor a szélességi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a szélességi fának.

Visszamutató él – ha v őse u-nak a szélességi fában (és ha (v, u) él nem minősült már faélnek). Az ilyen él visszamutat az aktuális csúcspont valamelyik szürke ősére.

Az élek osztályozása a szélességi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók (ha a gráf nem összefüggő, akkor a bejárt komponens éleire vonatkoztatjuk), minden él akkor kap besorolást, amikor a szélességi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a szélességi fának.

Visszamutató él – ha v őse u-nak a szélességi fában (és ha (v, u) él nem minősült már faélnek). Az ilyen él visszamutat az aktuális csúcspont valamelyik szürke ősére.

Előremutató él – ha v utóda u-nak a szélességi fában (és ha (u, v) él nem minősült már faélnek). Az ilyen él előremutat az aktuális csúcspont valamelyik, már feketévé vált utódjára.

Az élek osztályozása a szélességi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók (ha a gráf nem összefüggő, akkor a bejárt komponens éleire vonatkoztatjuk), minden él akkor kap besorolást, amikor a szélességi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a szélességi fának.

Visszamutató él – ha v őse u-nak a szélességi fában (és ha (v, u) él nem minősült már faélnek). Az ilyen él visszamutat az aktuális csúcspont valamelyik szürke ősére.

Előremutató él – ha v utóda u-nak a szélességi fában (és ha (u, v) él nem minősült már faélnek). Az ilyen él előremutat az aktuális csúcspont valamelyik, már feketévé vált utódjára.

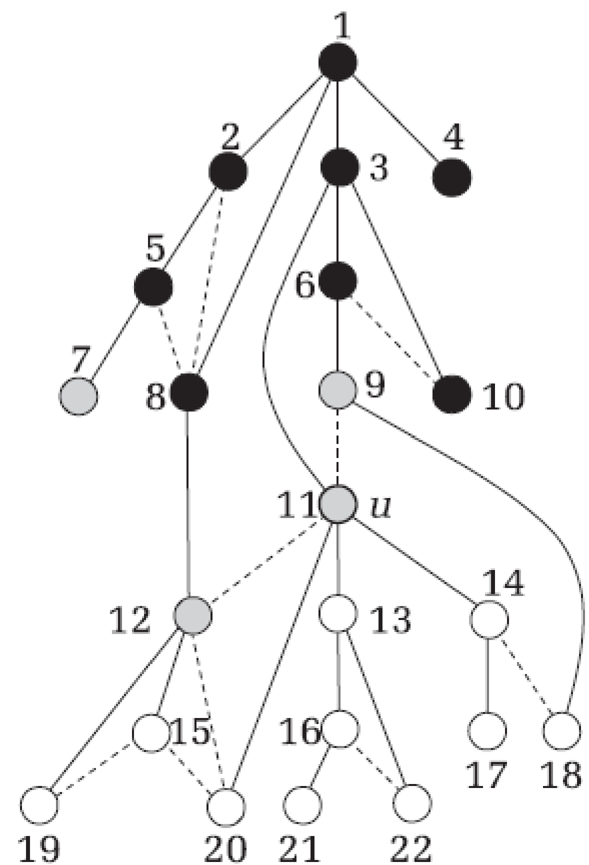
Keresztél – az összes többi él. Azokat az éleket kötik össze, amelyeknek végpontjai között nincs ős-utód, vagy utód-ős kapcsolat a szélességi fában.

Egy irányítatlan gráf szélességi bejárása nyomán belátható, hogy:

- Egyik él sem lehet visszamutató él, sem előremutató él.
- Minden (u, v) faélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u] + 1$
- Minden (u, v) keresztélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u]$ vagy $\mathtt{d}[v] = \mathtt{d}[u] + 1$

Egy irányítatlan gráf szélességi bejárása nyomán belátható, hogy:

- Egyik él sem lehet visszamutató él, sem előremutató él.
- Minden (u, v) faélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u] + 1$
- Minden (u, v) keresztélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u]$ vagy $\mathtt{d}[v] = \mathtt{d}[u] + 1$

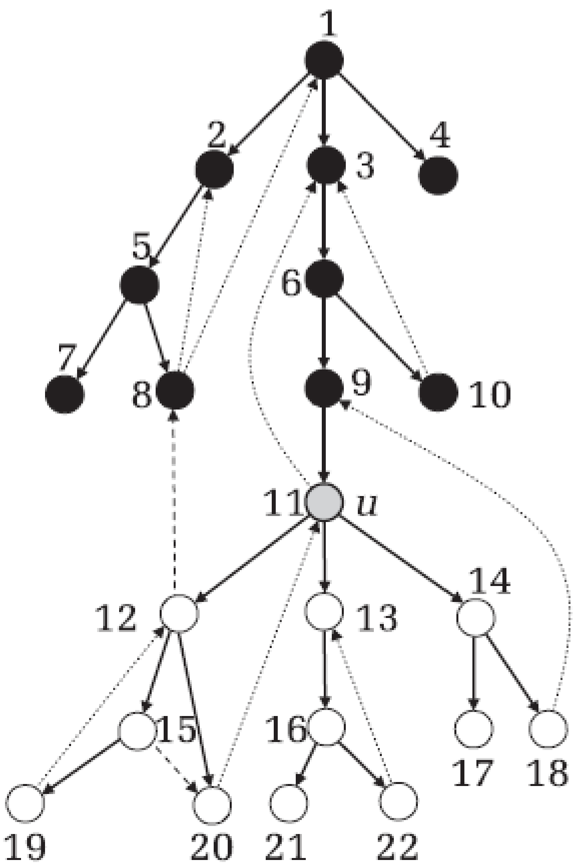


Egy irányított gráf szélességi bejárása nyomán belátható, hogy:

- Minden (u, v) faélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u] + 1$
- Minden (u, v) keresztélre teljesül: $\mathtt{d}[v] \le \mathtt{d}[u] + 1$
- Minden (u, v) visszamutató élre teljesül: $0 \le \mathtt{d}[v] \le \mathtt{d}[u]$
- Egyik él sem lehet előremutató él.

Egy irányított gráf szélességi bejárása nyomán belátható, hogy:

- Minden (u, v) faélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u] + 1$
- Minden (u, v) keresztélre teljesül: $\mathtt{d}[v] \leq \mathtt{d}[u] + 1$
- Minden (u, v) visszamutató élre teljesül: $0 \leq \mathtt{d}[v] \leq \mathtt{d}[u]$
- Egyik él sem lehet előremutató él.

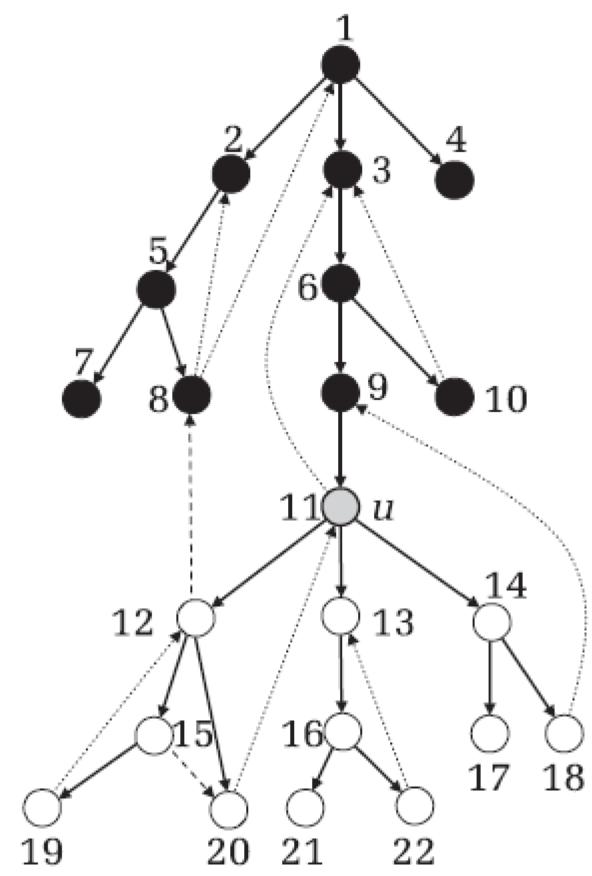


Egy irányított gráf szélességi bejárása nyomán belátható, hogy:

- Minden (u, v) faélre teljesül: $\mathtt{d}[v] = \mathtt{d}[u] + 1$
- Minden (u, v) keresztélre teljesül: $\mathtt{d}[v] \leq \mathtt{d}[u] + 1$
- Minden (u, v) visszamutató élre teljesül: $0 \leq \mathtt{d}[v] \leq \mathtt{d}[u]$
- Egyik él sem lehet előremutató él.

Megjegyzés:

Ha egy irányított gráfban töröljük az élek irányítását, akkor az ugyanabból a csúcspontból indított szélességi bejárás is átminősíti az éleket.



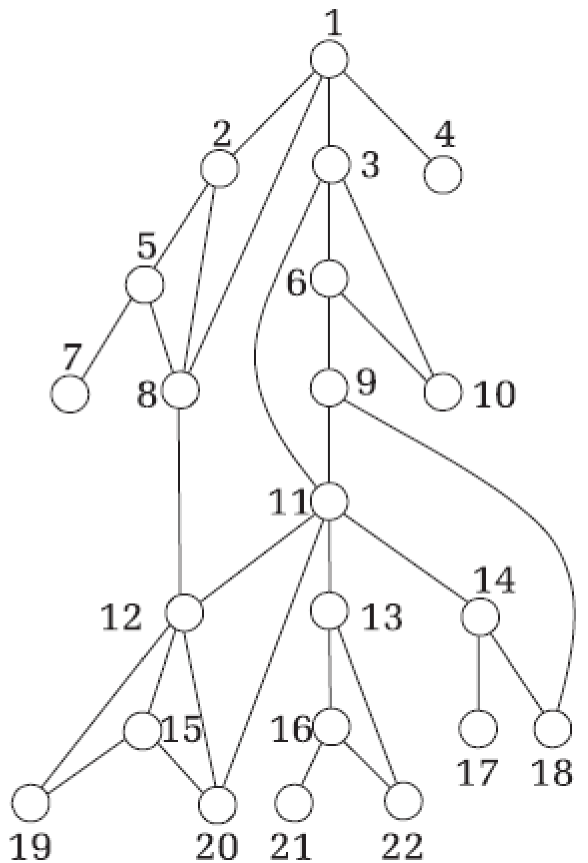
2) Mélységi bejárás (DFS – Depth First Search)

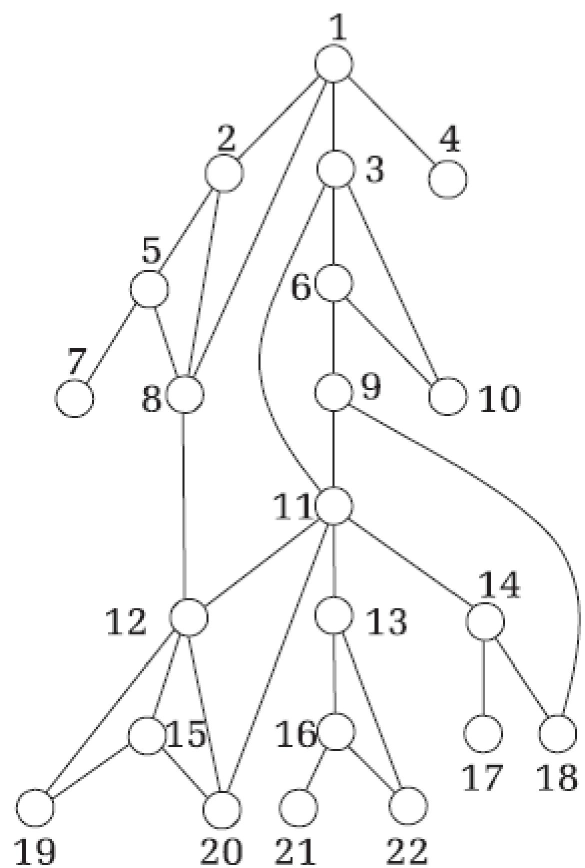
<u>Stratégia</u>: A gráf bejárását a kiindulási csúcspontból (s) indítjuk, színét szürkére változtatjuk. Az érintetlen (fehér) szomszédok közül a legkisebb azonosítójú csúcspontot látogatjuk meg, színét szürkére változtatjuk. Ezt egészen addig folytatjuk, amíg az aktuális csúcspontnak van fehér szomszédja. Ha már nincs több fehér szomszéd, akkor az adott pont színét feketére változtatjuk, s visszafelé haladunk ahhoz a csúcsponthoz, ahonnan idejöttünk. Itt folytatjuk, amit abbahagytunk, azaz megpróbálunk egy következő fehér szomszéd (ha létezik) irányába továbbmenni. Miután kimerítettünk minden lehetőséget (azaz sorra minden fehér szomszéd irányába elmentünk és visszajöttünk), akkor innen is visszafelé haladunk (a csúcspont színe feketére vált), ahhoz a ponthoz, ahonnan annak idején idejöttünk. A mélységi bejárás akkor ér véget, amikor a kiindulási csúcspontból haladnánk visszafelé (elhagyjuk a gráfot).

Vegyük észre, hogy egy csúcspont szürkévé válik, amint előrehaladva elérjük, és mindaddig szürke marad, míg visszafelé haladva el nem hagyjuk. Ekkor színe feketére változik.

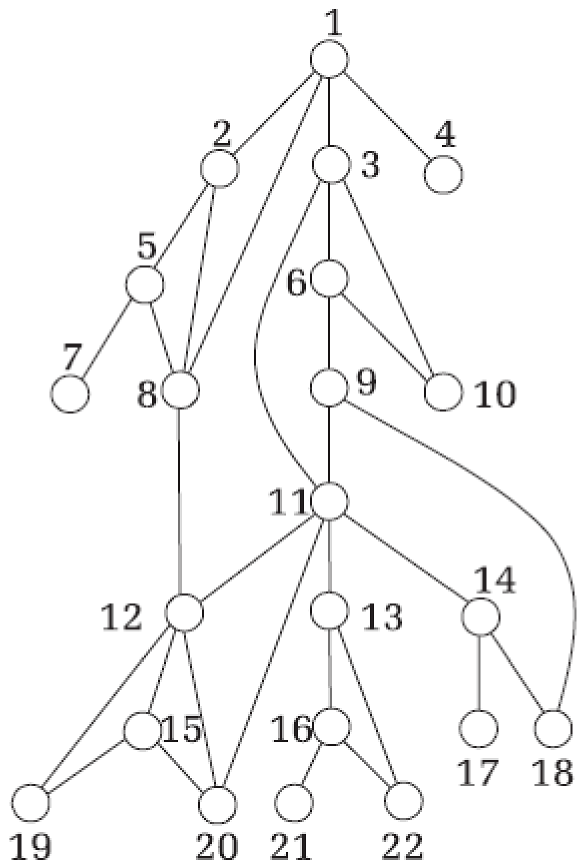
Vegyük észre, hogy egy csúcspont szürkévé válik, amint előrehaladva elérjük, és mindaddig szürke marad, míg visszafelé haladva el nem hagyjuk. Ekkor színe feketére változik.

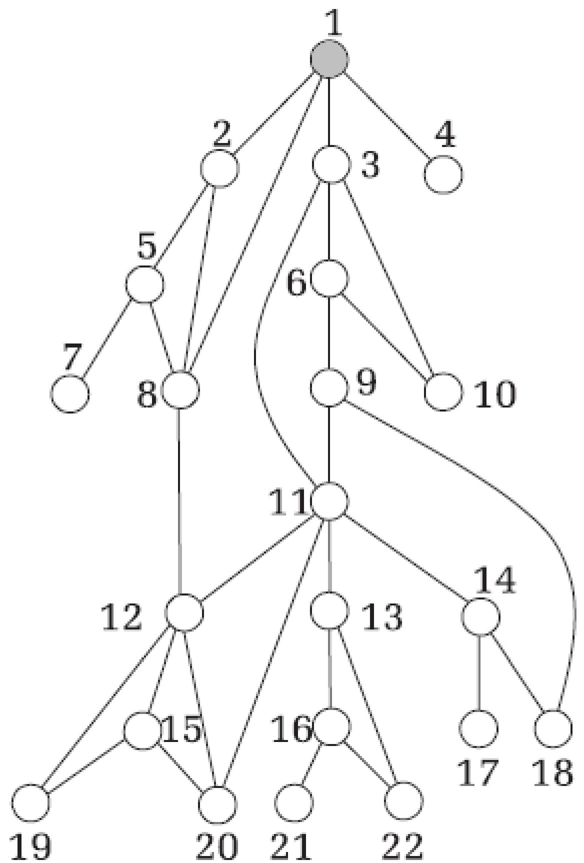
A gráf azon éleit, amelyeken áthaladtunk, faéleknek nevezzük. Minden faél egy apa-fiú kapcsolatot képvisel a gráf csúcspontjai között. Egy csúcspont azoknak a pontoknak apja, amelyek az ő fehér szomszédjaiként érhetők el. Az apa-ponttól előre haladva érkezünk egy adott fiú-ponthoz, a fiú-pontoktól viszont visszafelé haladva térünk vissza az apa-pontjukhoz. A faélek alkotta s gyökerű fát **mélységi fá**nak nevezzük.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

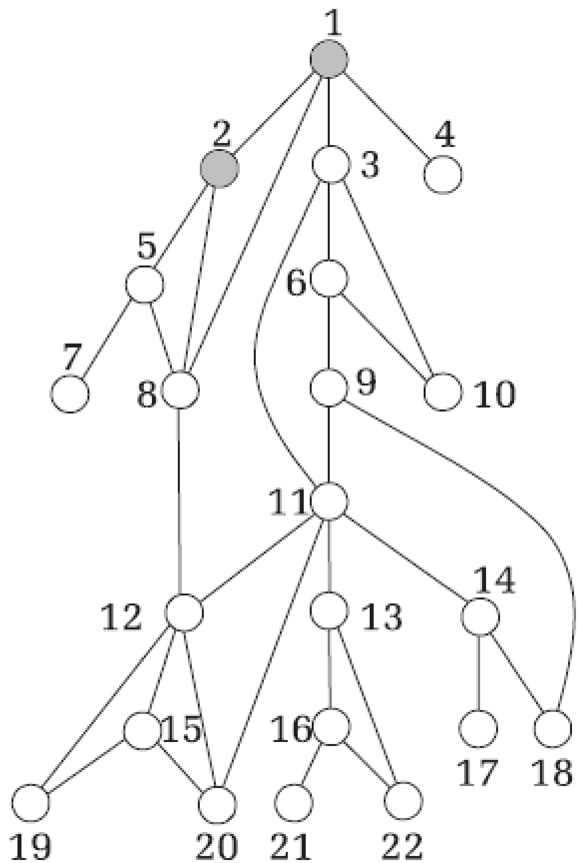


1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19, 4

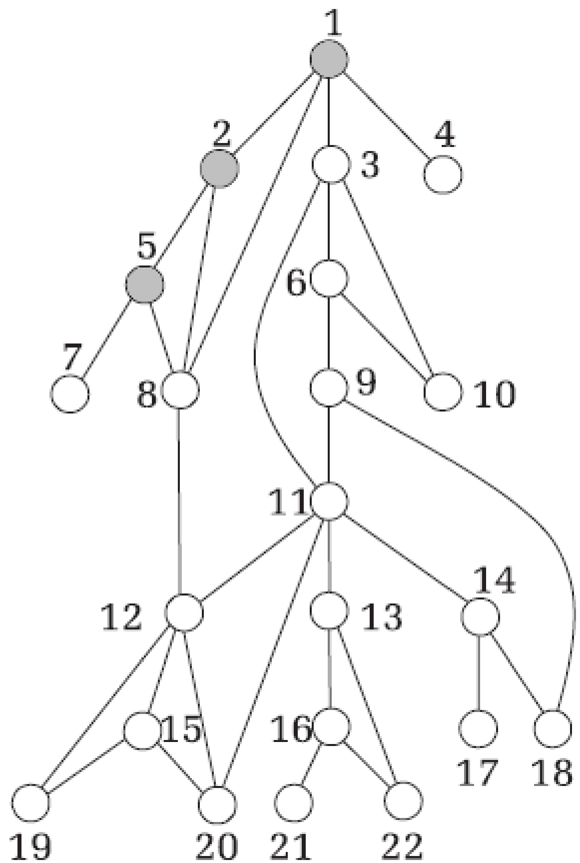

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22



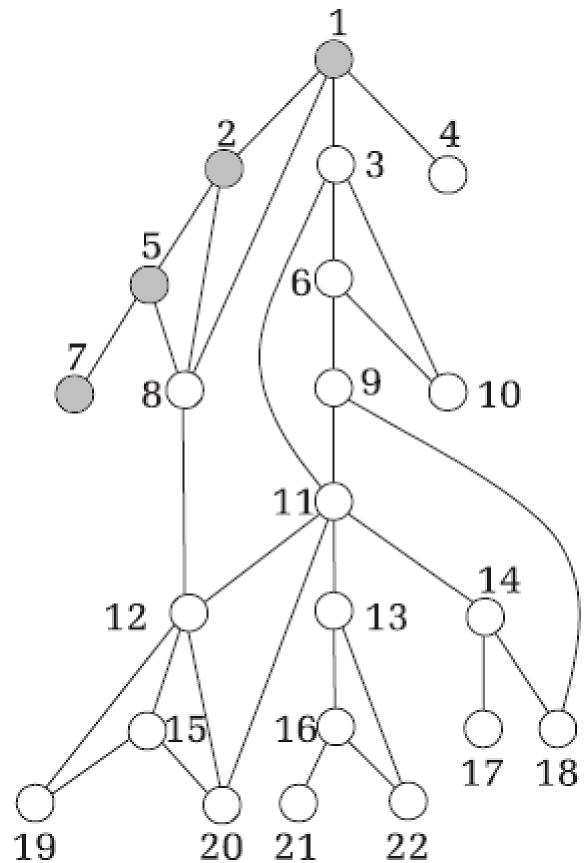
1,



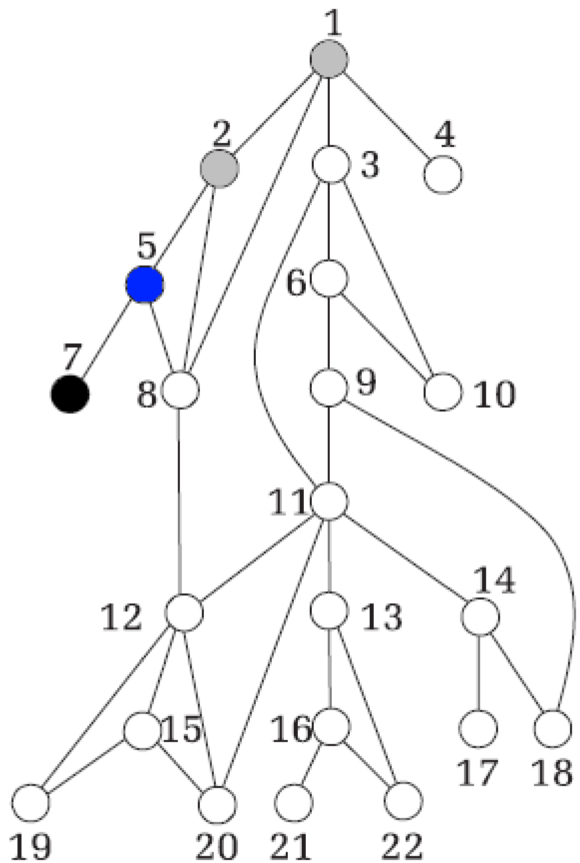
1, 2,



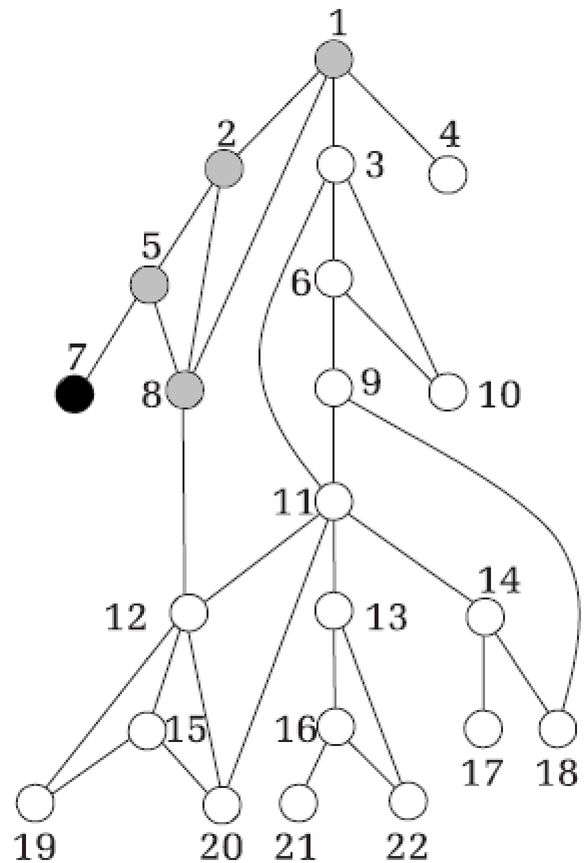
1, 2, 5,



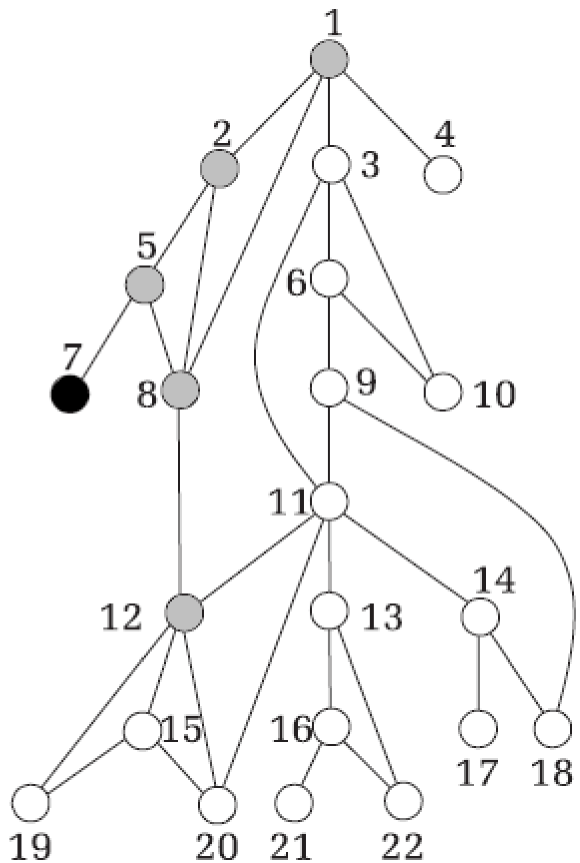
1, 2, 5, 7,



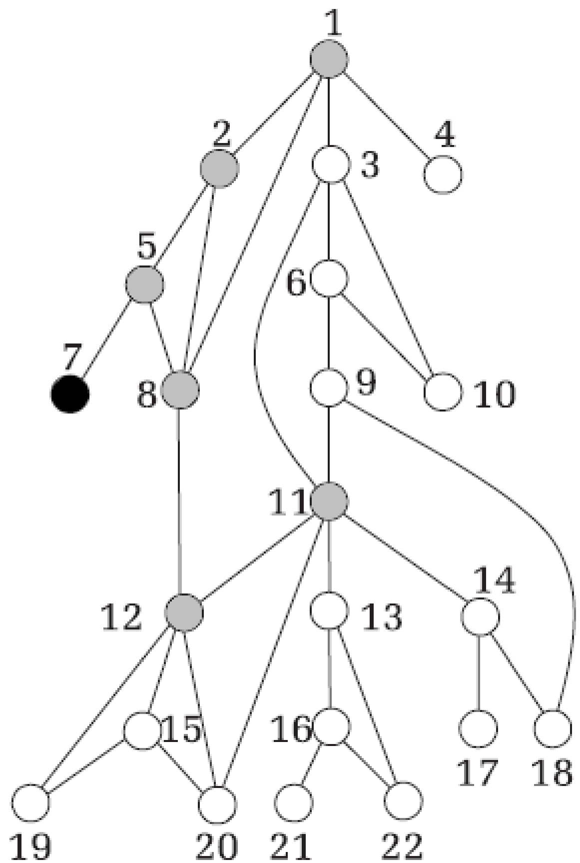
1, 2, 5, 7,



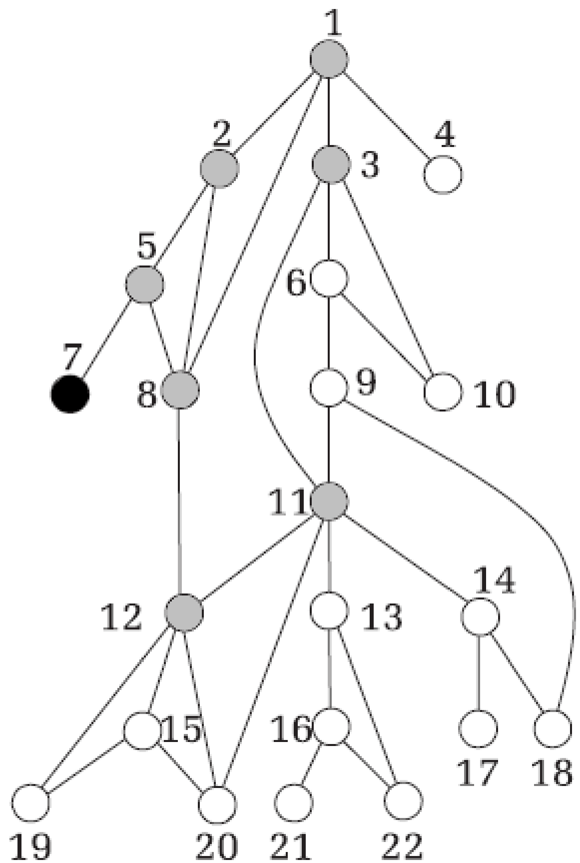
1, 2, 5, 7, 8,



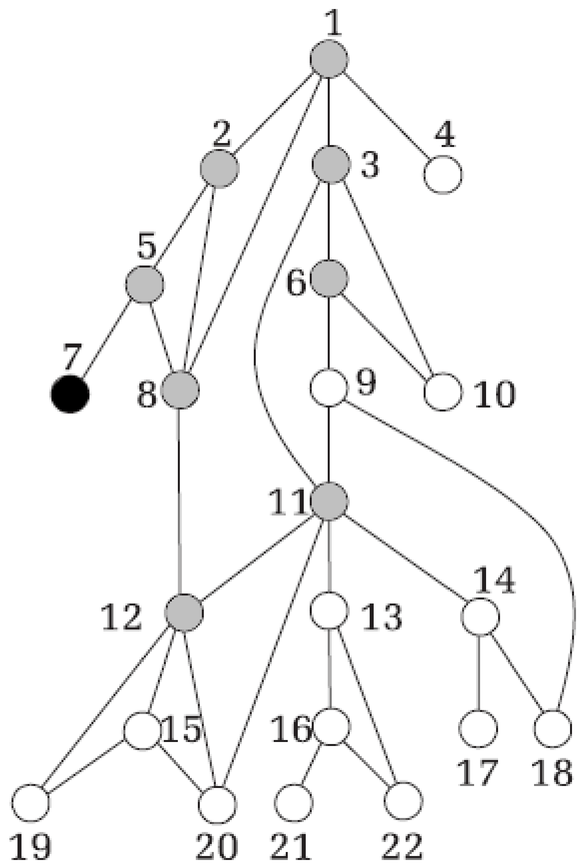
1, 2, 5, 7, 8, 12,



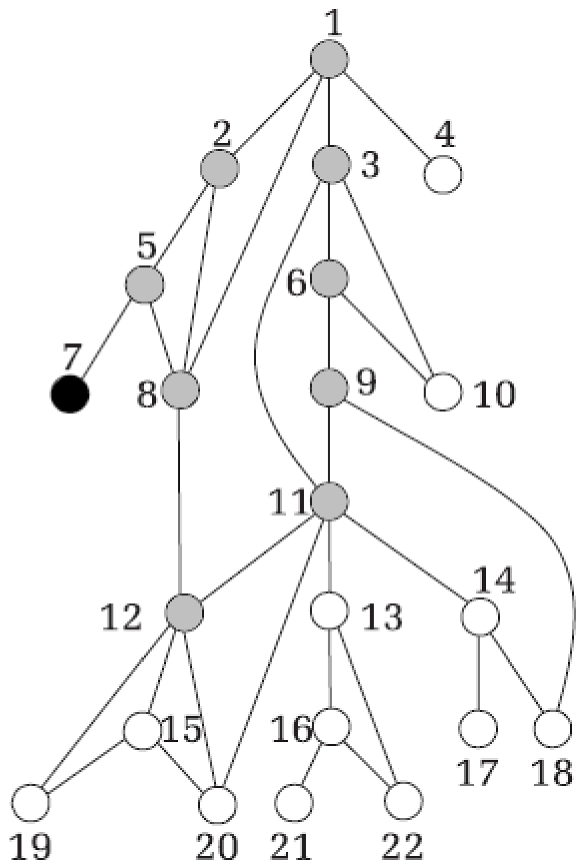
1, 2, 5, 7, 8, 12, 11,



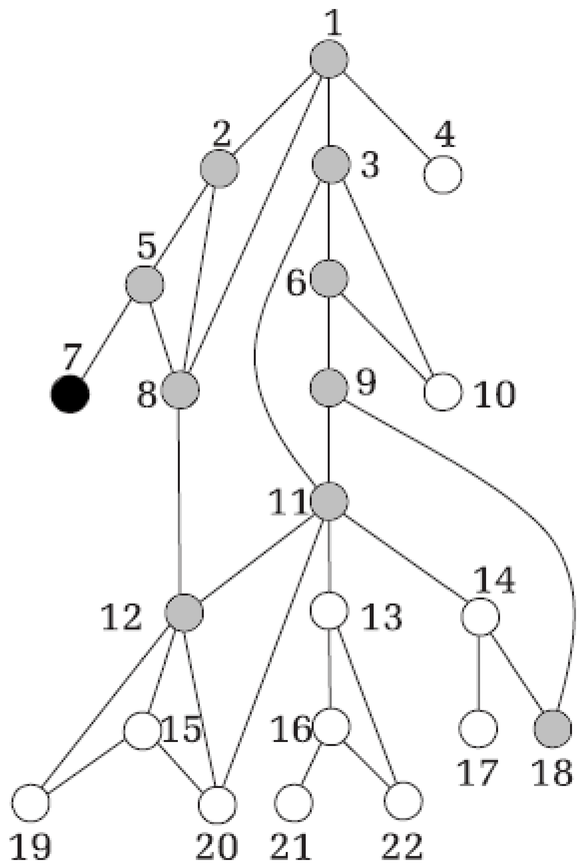
1, 2, 5, 7, 8, 12, 11, 3,



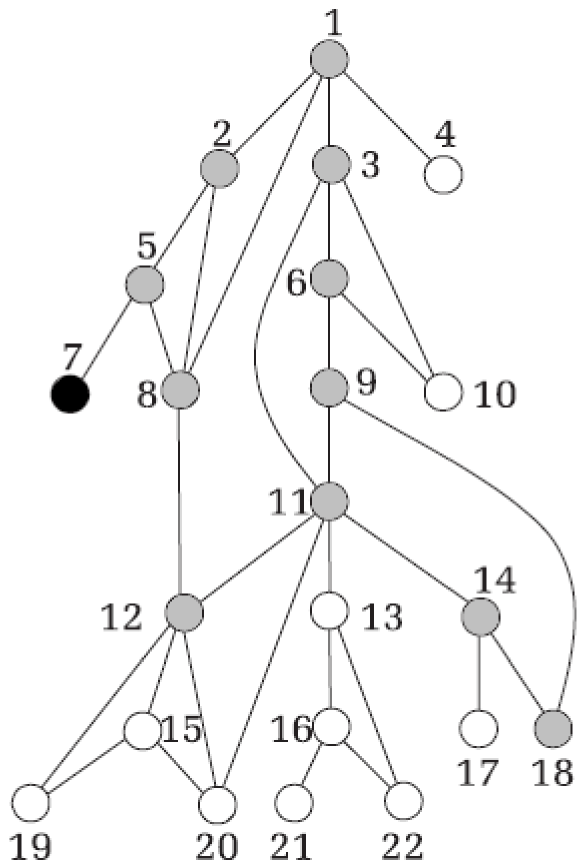
1, 2, 5, 7, 8, 12, 11, 3, 6,



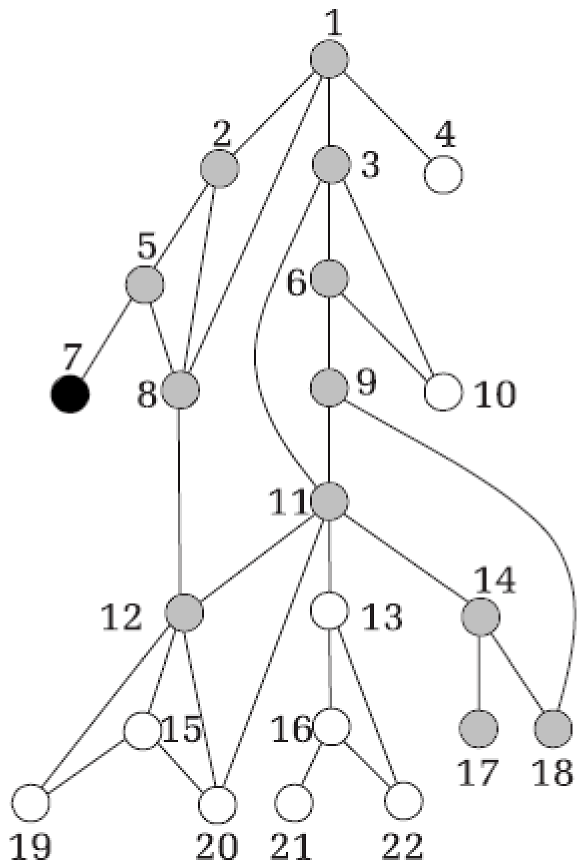
1, 2, 5, 7, 8, 12, 11, 3, 6, 9,



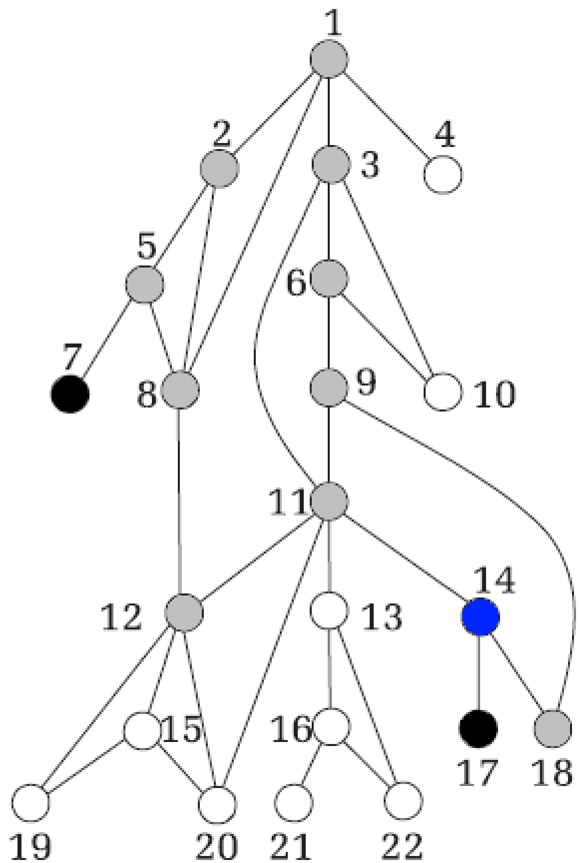
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18,



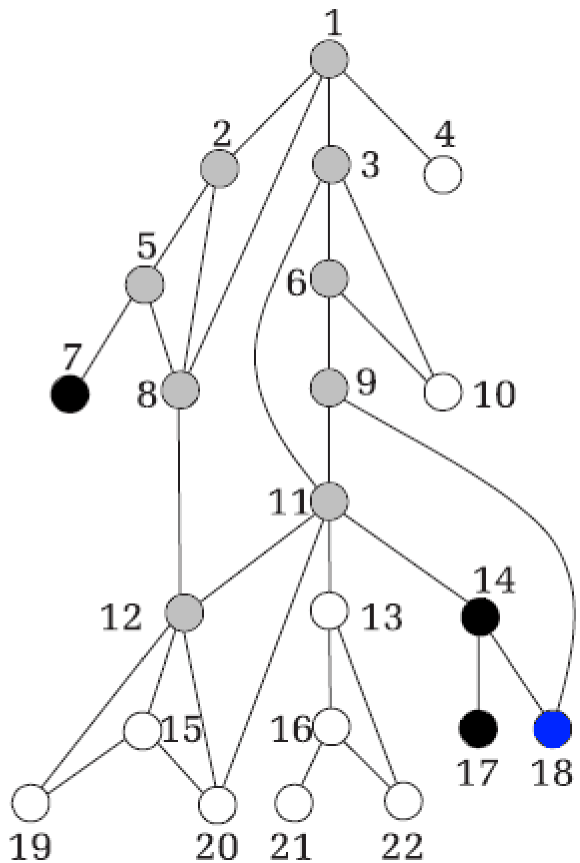
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14,



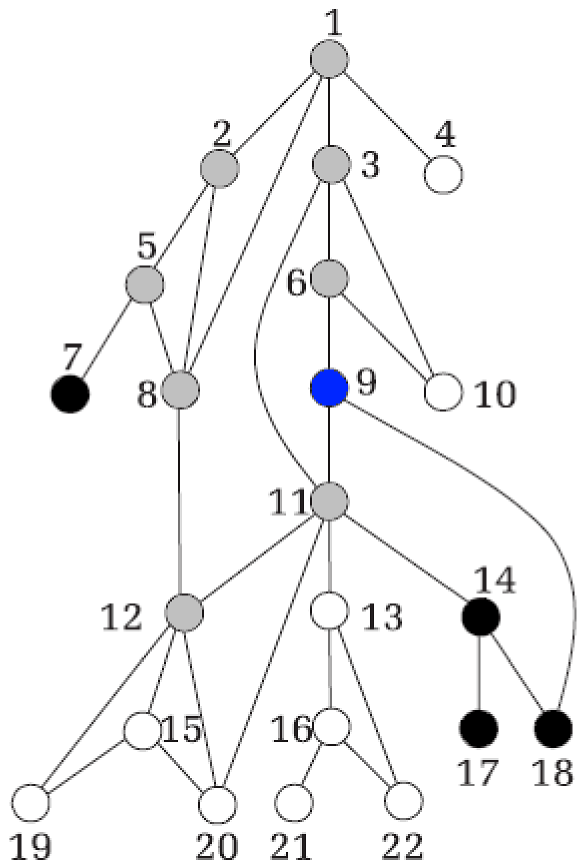
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17,



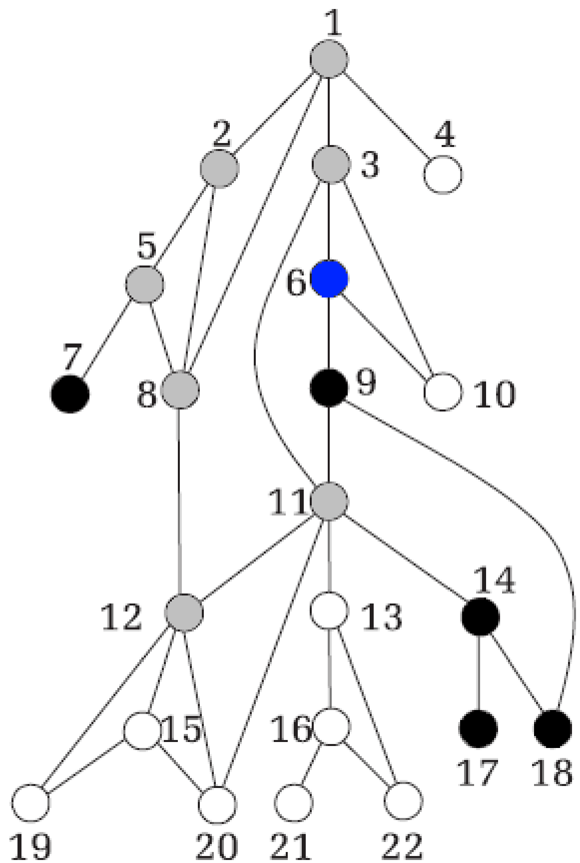
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17,



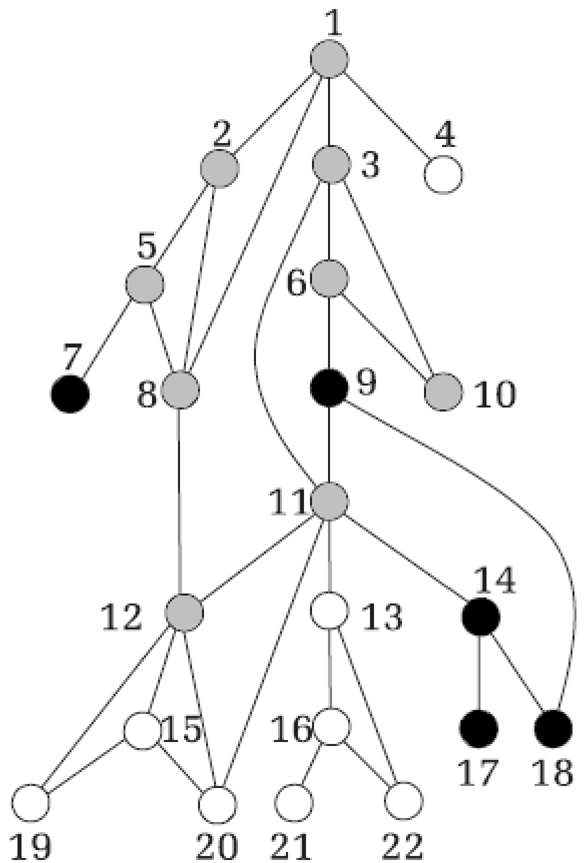
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17,



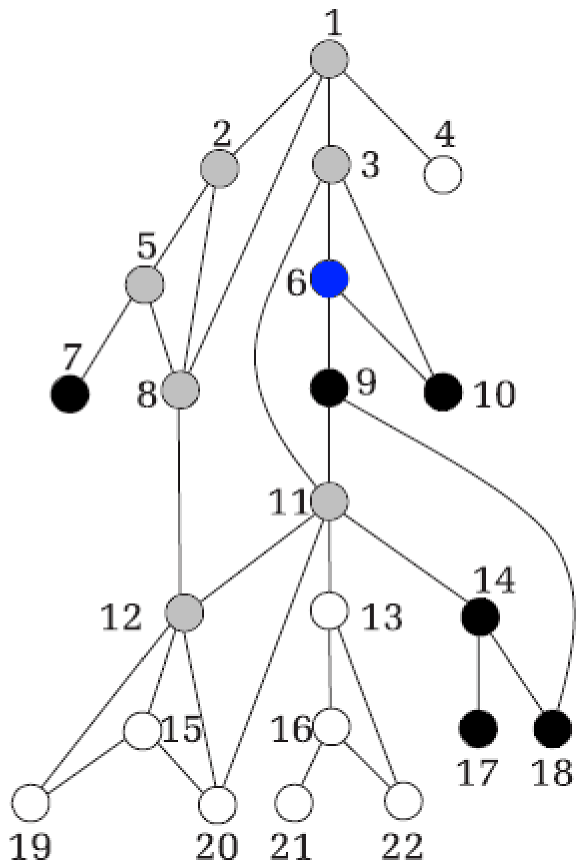
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17,



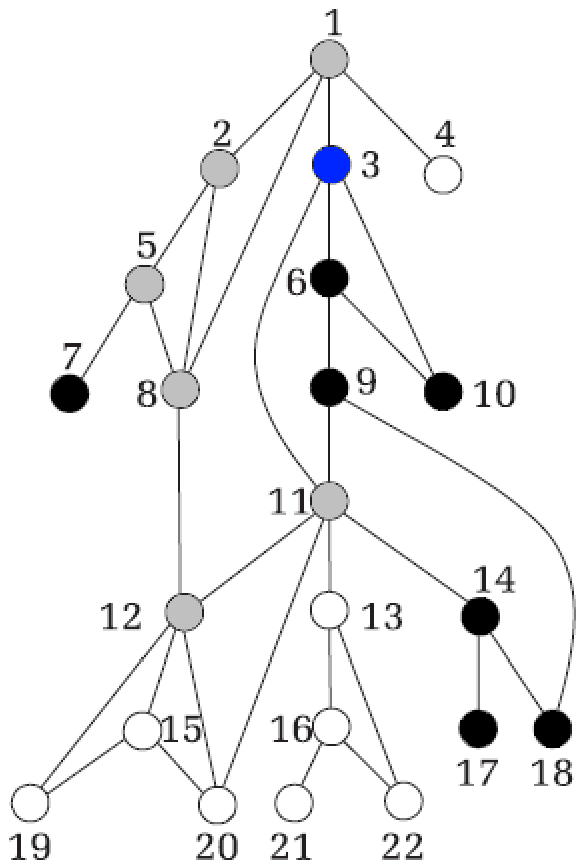
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17,



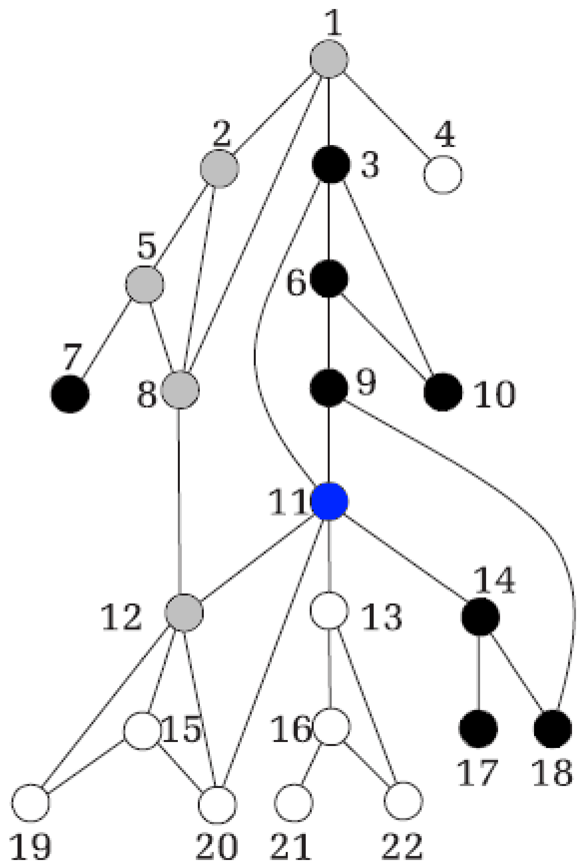
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10,



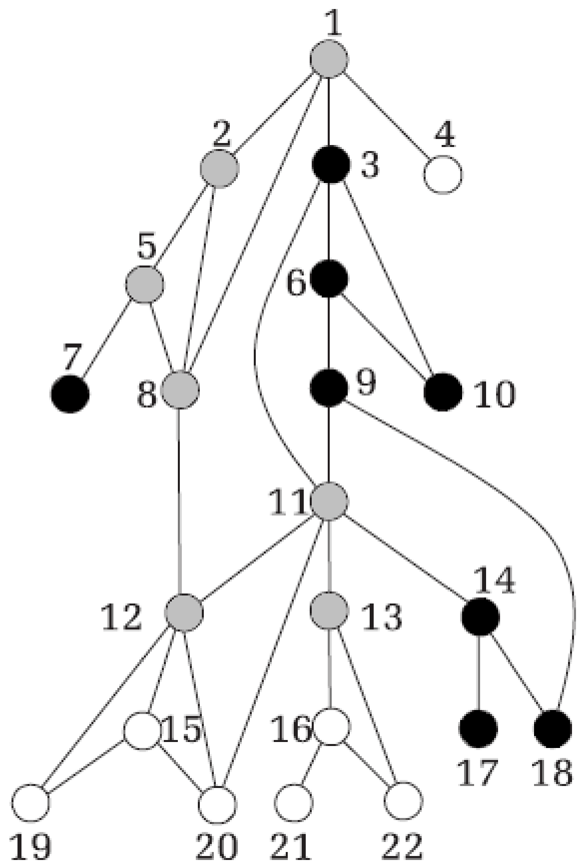
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10,



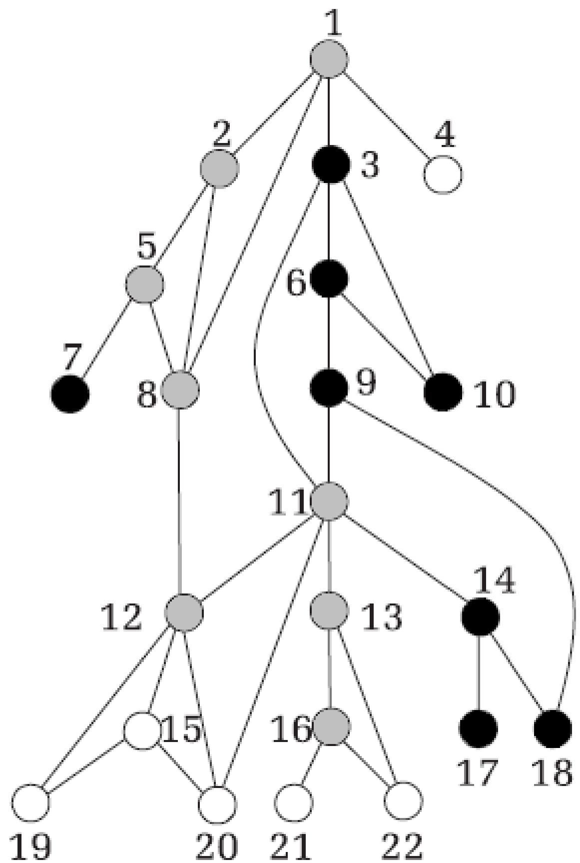
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10,



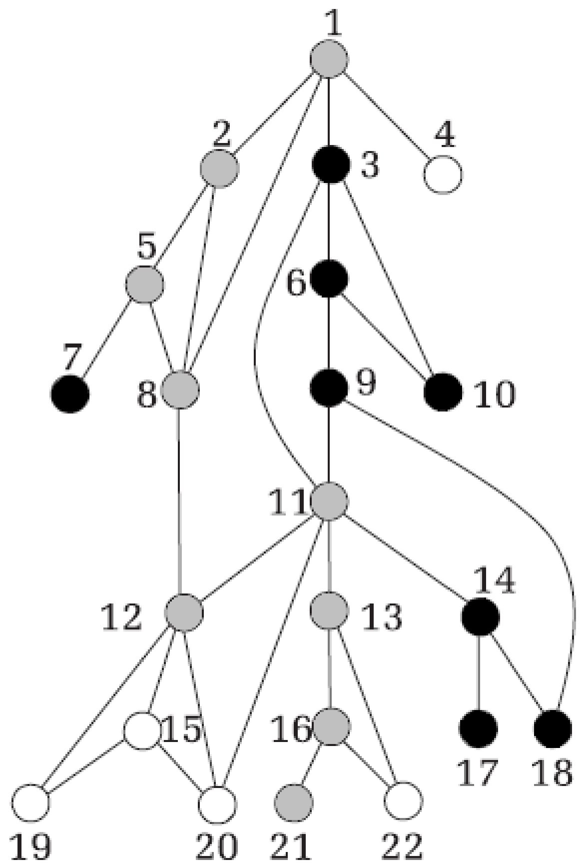
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10,



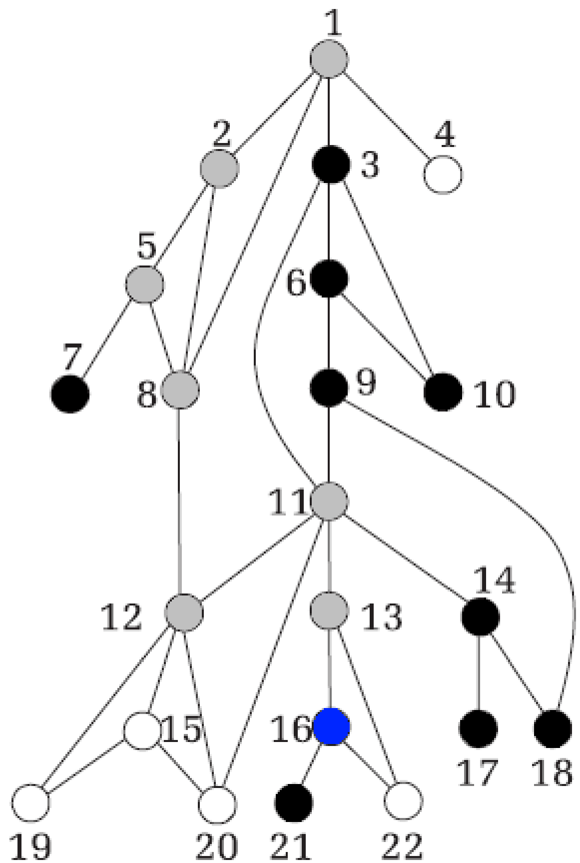
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13,



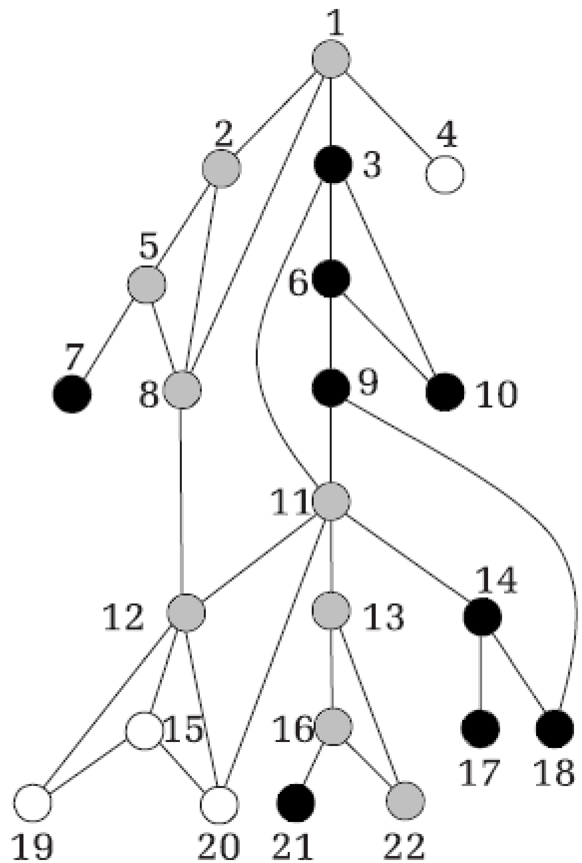
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16,



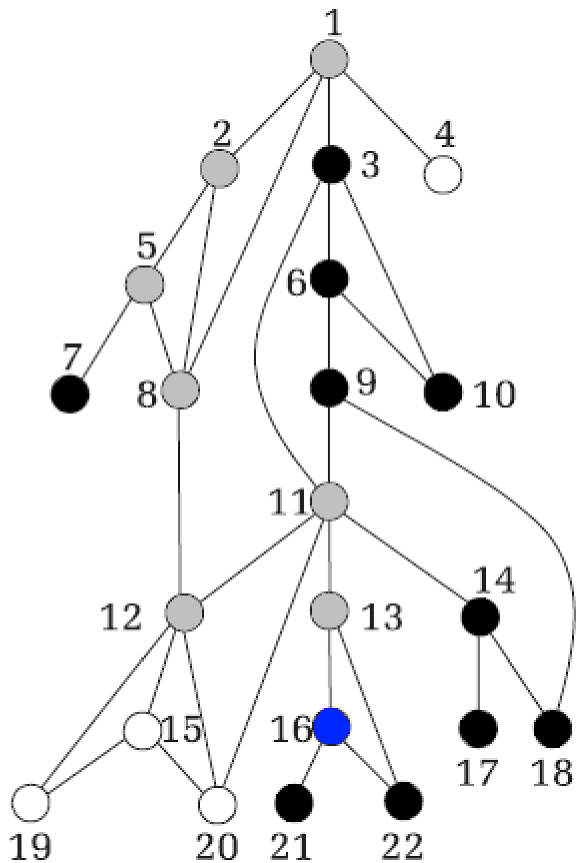
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21,



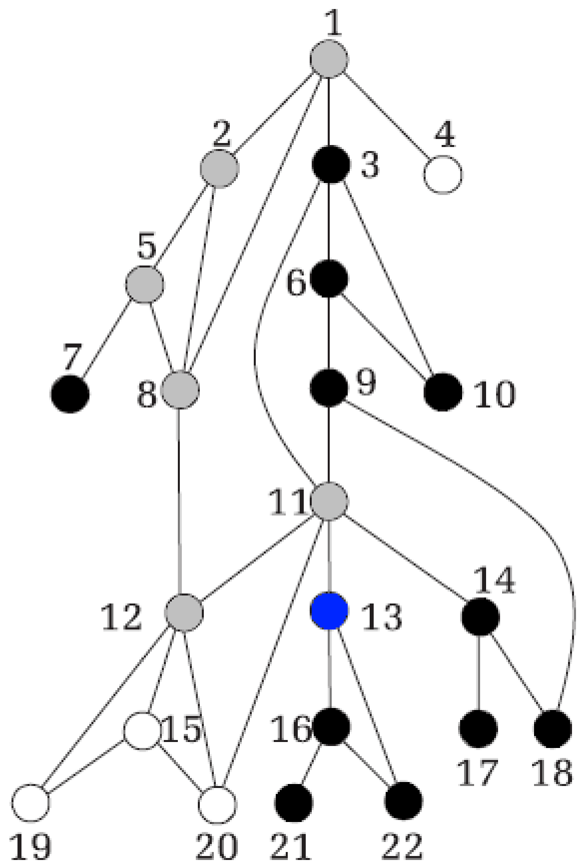
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21,



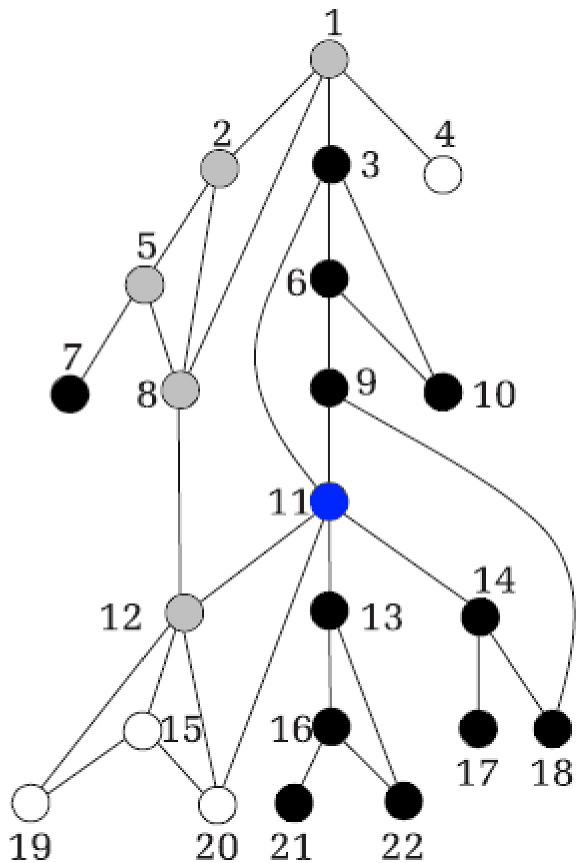
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22,



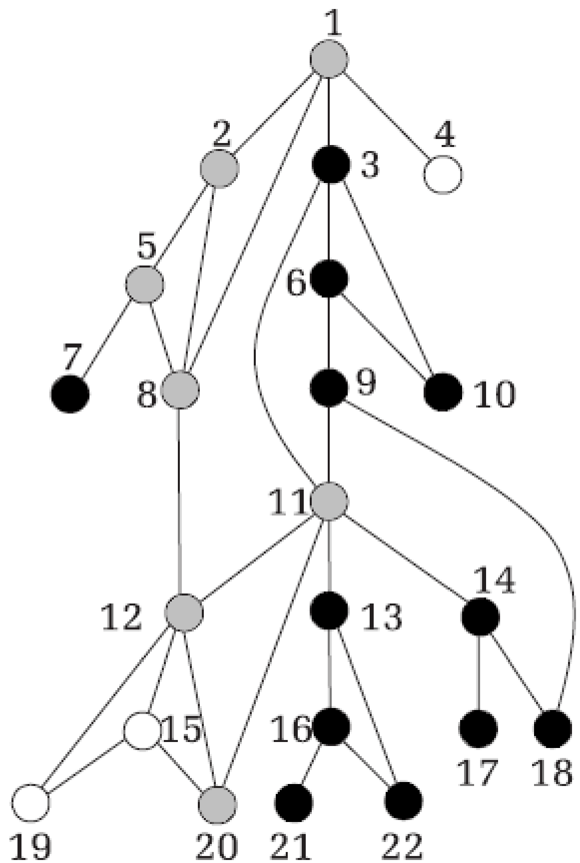
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22,



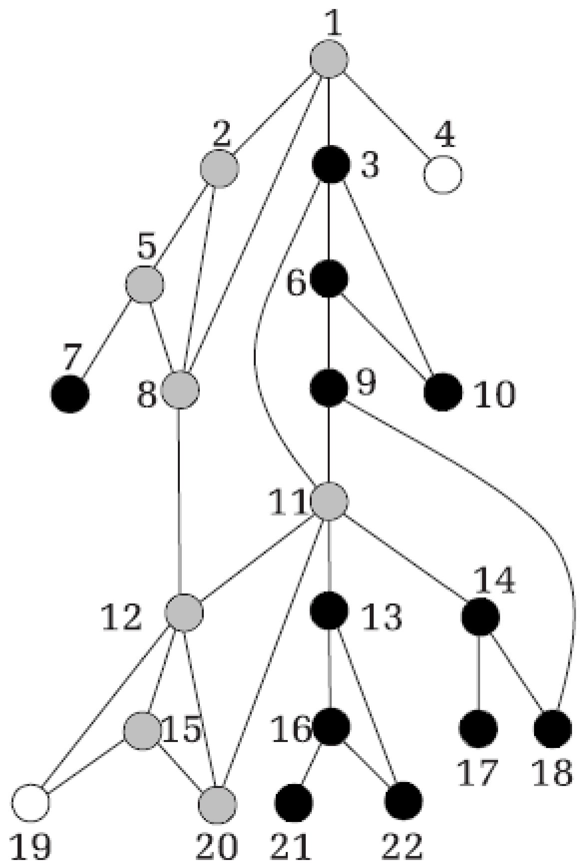
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22,



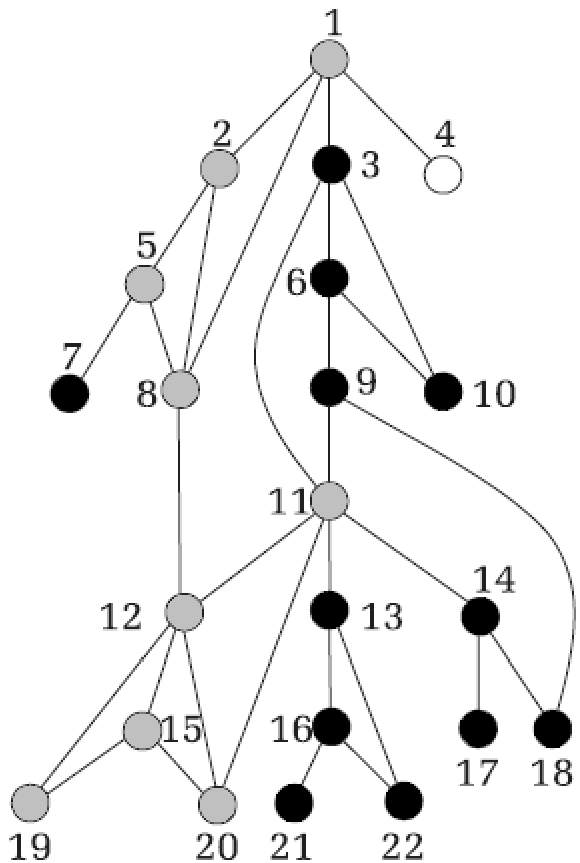
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22,



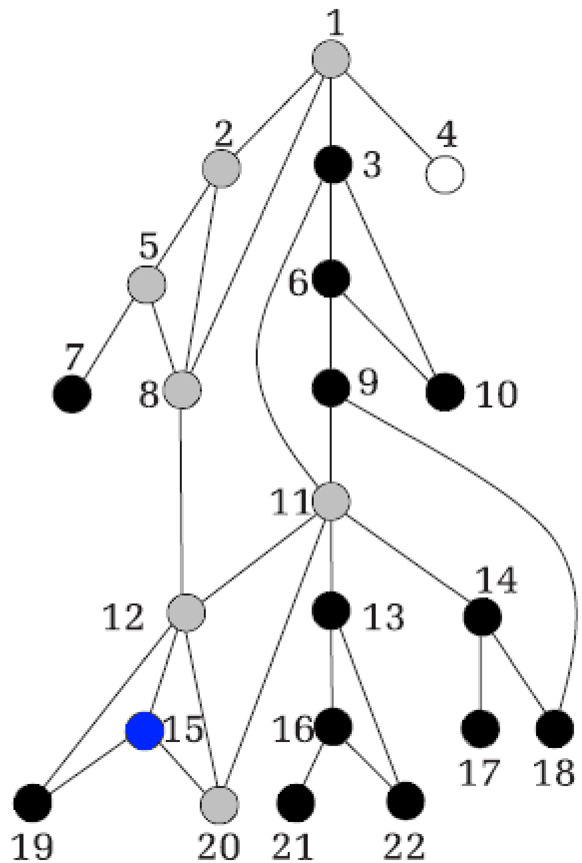
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20,



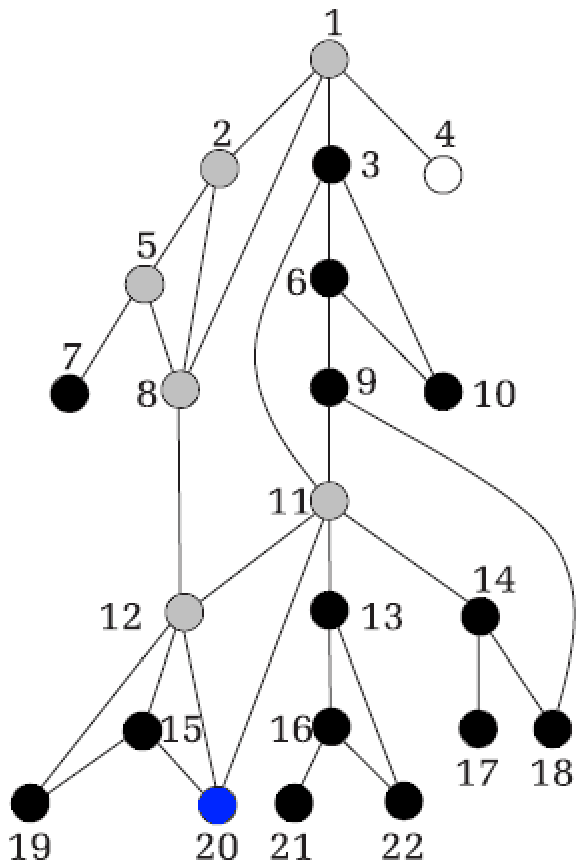
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15,



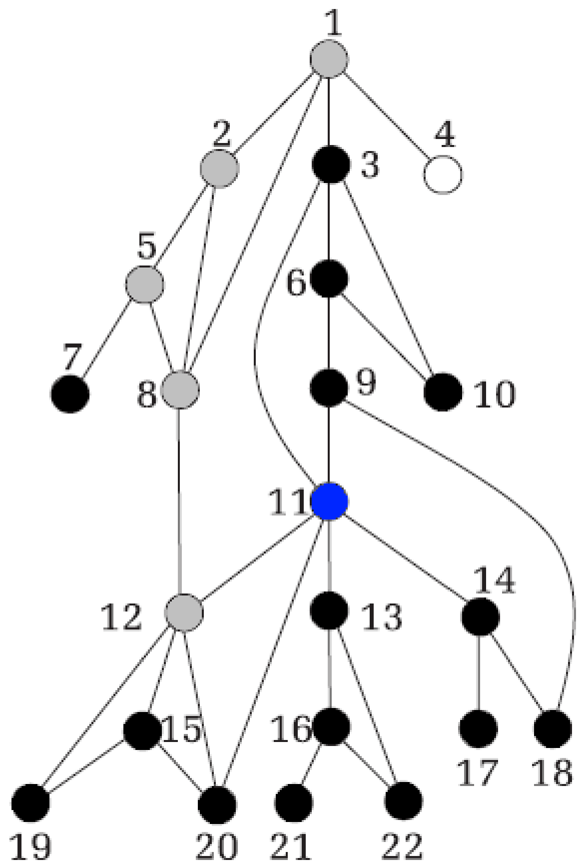
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



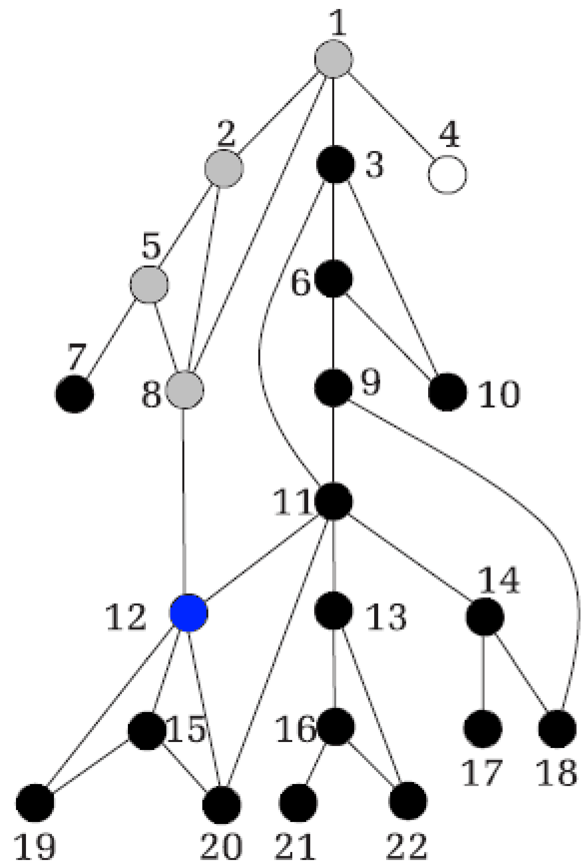
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



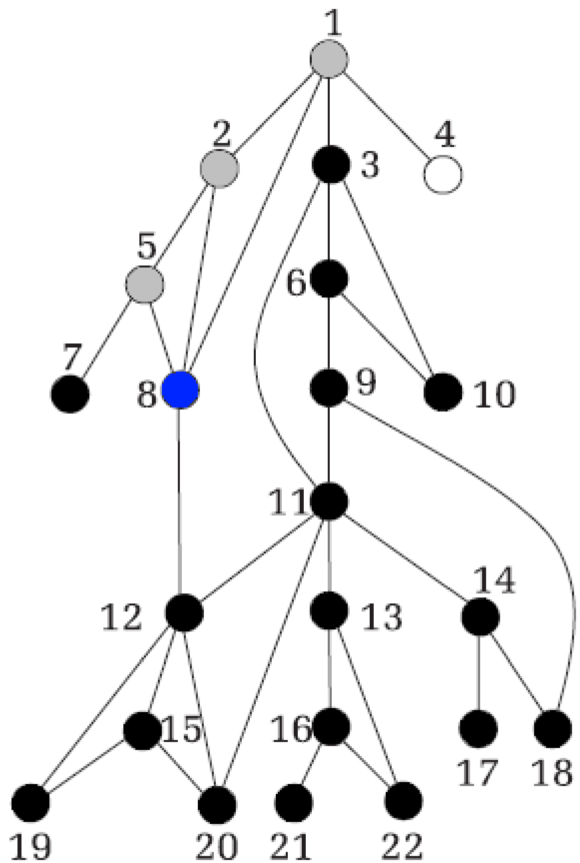
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



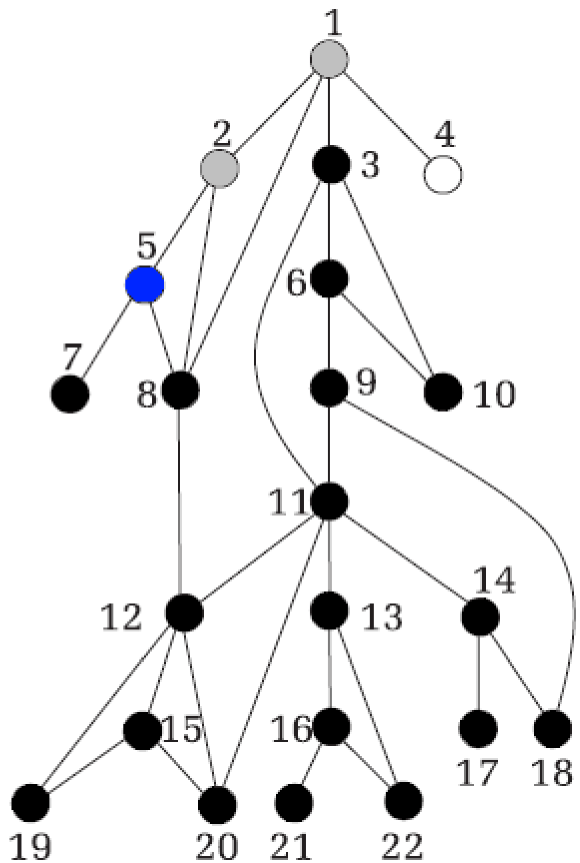
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



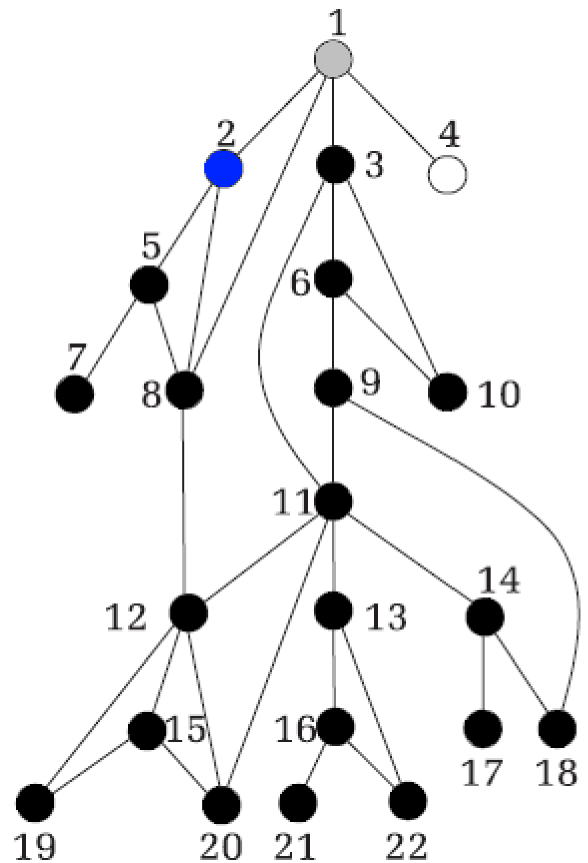
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



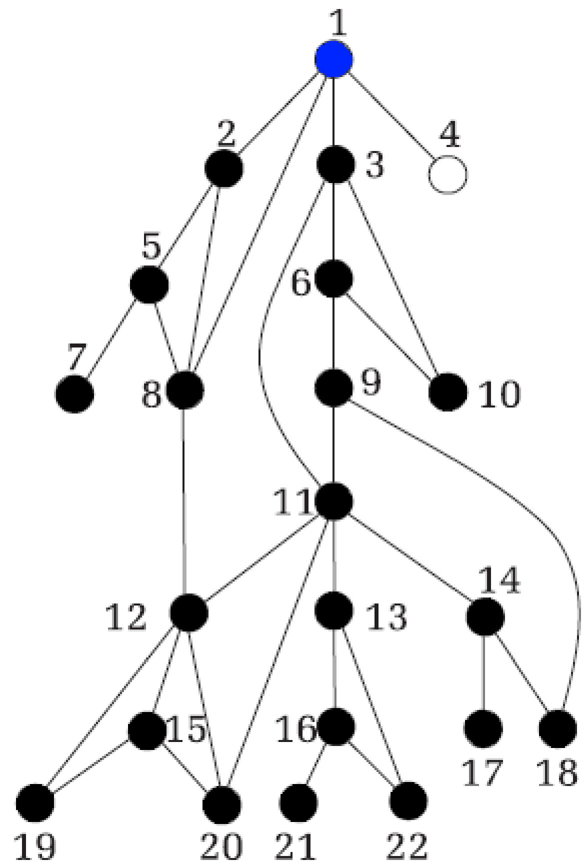
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



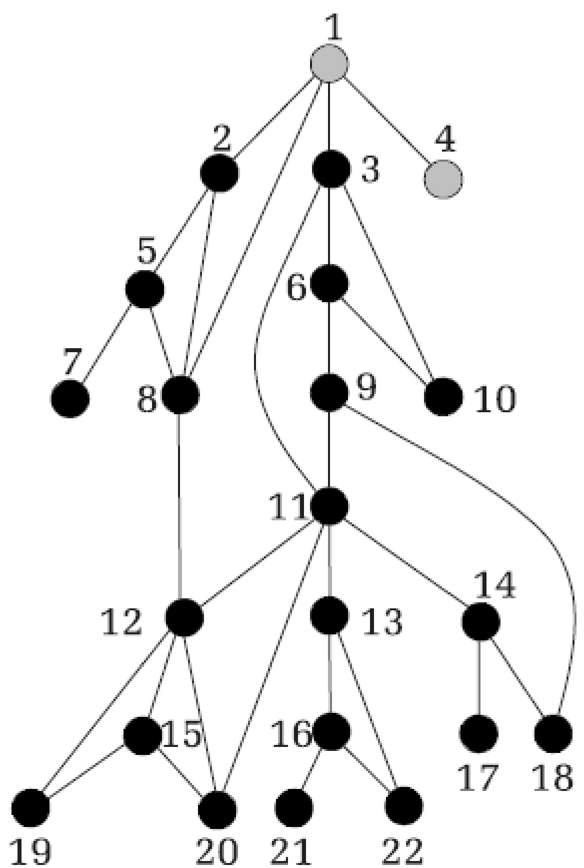
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



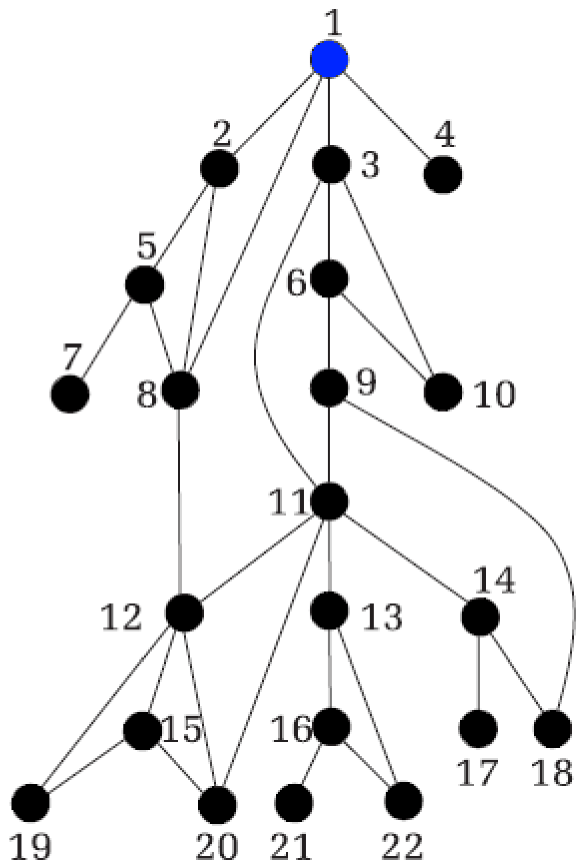
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



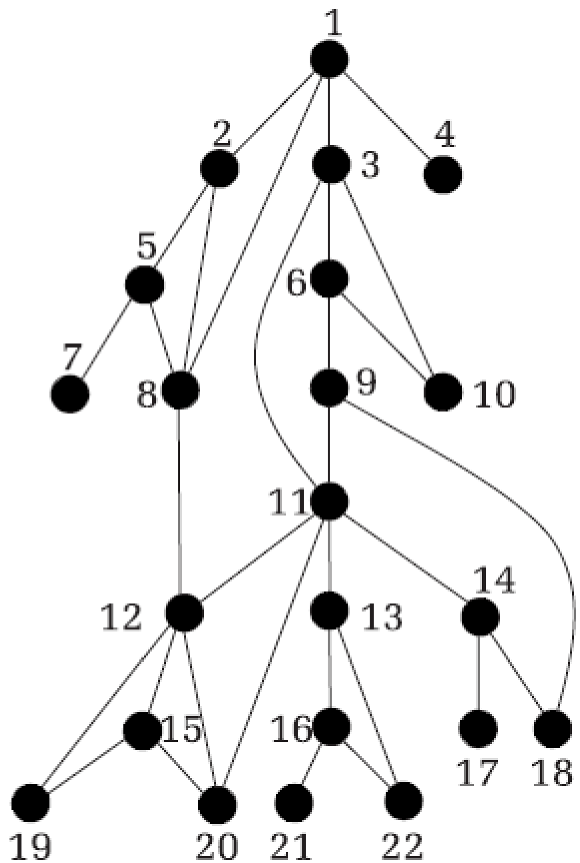
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19,



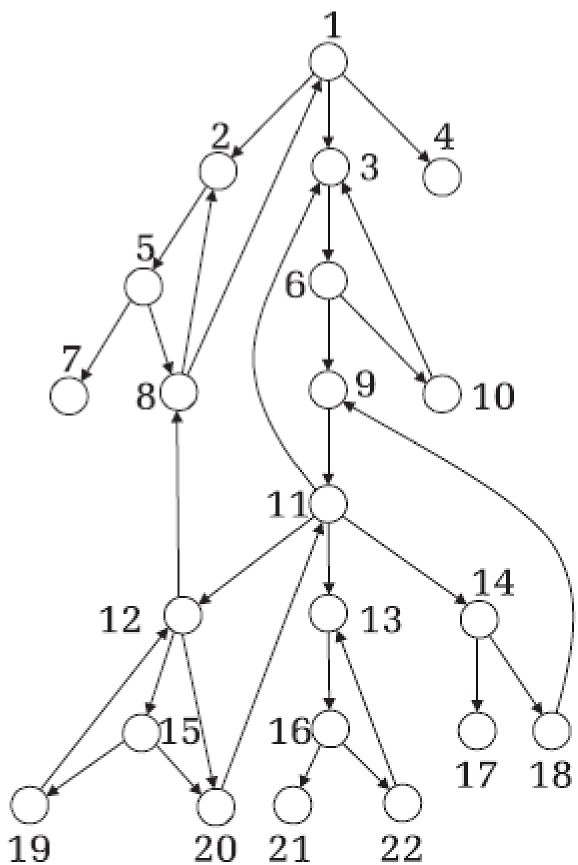
1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19, 4

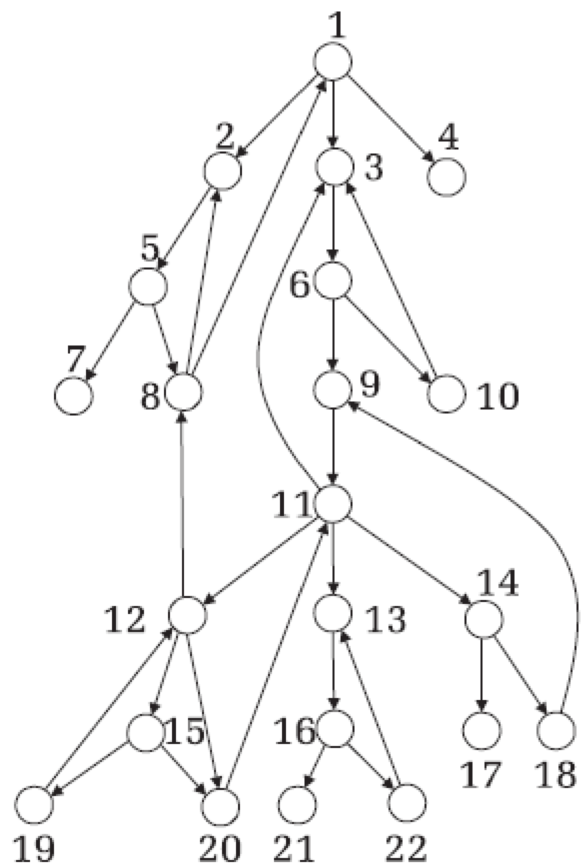


1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19, 4



1, 2, 5, 7, 8, 12, 11, 3, 6, 9, 18, 14, 17, 10, 13, 16, 21, 22, 20, 15, 19, 4


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22



1, 2, 5, 7, 8, 3, 6, 9, 11, 12, 15, 19, 20, 13, 16, 21, 22, 14, 17, 18, 10, 4

2) Mélységi bejárás (DFS – Depth First Search)

Algoritmus: A gráf mélységi bejárása megvalósítható rekurzióval. A MÉLYSÉGI_MENET rekurzív eljárás egyetlen összefüggő komponens mélységi bejárását valósítja meg. A MÉLYSÉGI_BEJÁRÁS eljárás második `minden` ciklusa minden egyes komponensre meghívja a MÉLYSÉGI_MENET eljárást. Tehát a mélységi bejárás (a szélességi bejárástól eltérően) minden komponens bejárását biztosítja.

2) Mélységi bejárás (DFS – Depth First Search)

<u>Algoritmus</u>: A gráf mélységi bejárása megvalósítható rekurzióval. A MÉLYSÉGI_MENET rekurzív eljárás egyetlen összefüggő komponens mélységi bejárását valósítja meg. A MÉLYSÉGI_BEJÁRÁS eljárás második `minden` ciklusa minden egyes komponensre meghívja a MÉLYSÉGI_MENET eljárást. Tehát a mélységi bejárás (a szélességi bejárástól eltérően) minden komponens bejárását biztosítja.

Az algoritmus meghatározza minden csúcspont esetén az elérési (amikor szürke színt kap) és elhagyási időpontokat (amikor fekete színt kap). Ennek érdekében egy globális időváltozót használunk. Minden csúcspont-színváltás alkalmával az időváltozó lép egyet. Az `elér` és `elhagy` tömbök lehetővé teszik a csúcspontok elérési és elhagyási sorrendjeinek felállítását.

2) Mélységi bejárás (DFS – Depth First Search)

Algoritmus: A gráf mélységi bejárása megvalósítható rekurzióval. A MÉLYSÉGI_MENET rekurzív eljárás egyetlen összefüggő komponens mélységi bejárását valósítja meg. A MÉLYSÉGI_BEJÁRÁS eljárás második `minden` ciklusa minden egyes komponensre meghívja a MÉLYSÉGI_MENET eljárást. Tehát a mélységi bejárás (a szélességi bejárástól eltérően) minden komponens bejárását biztosítja.

Az algoritmus meghatározza minden csúcspont esetén az elérési (amikor szürke színt kap) és elhagyási időpontokat (amikor fekete színt kap). Ennek érdekében egy globális időváltozót használunk. Minden csúcspont-színváltás alkalmával az időváltozó lép egyet. Az `elér` és `elhagy` tömbök lehetővé teszik a csúcspontok elérési és elhagyási sorrendjeinek felállítását.

A mélységi sorrendet a pontok elérési (szürkévé válási) sorrendje jelenti.

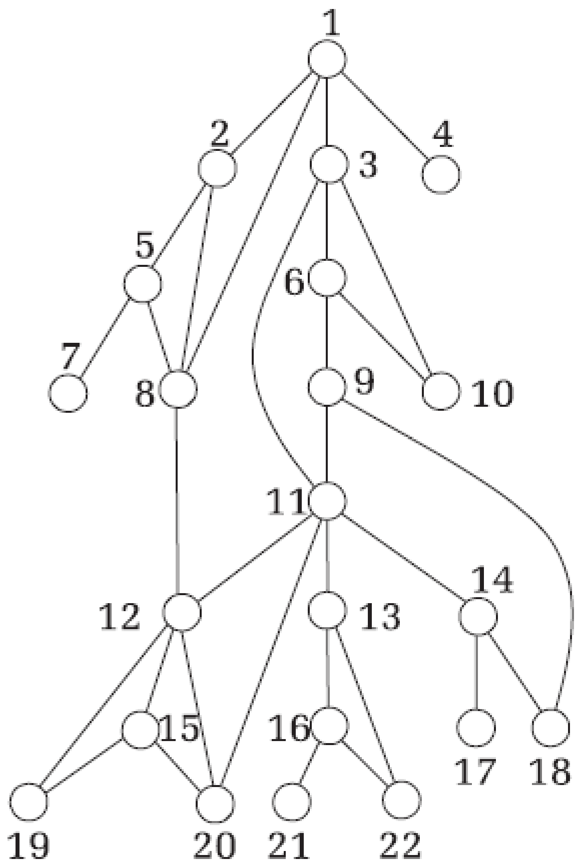
```
eljárás MÉLYSÉGI_BEJÁRÁS(G)
   minden u ∈ V(G) végezd
      szín[u] ← FEHÉR
      apa[u] ← 0
   vége minden
   idő ← 0
   minden u ∈ V(G) végezd
      ha szín[u] = FEHÉR akkor
         MÉLYSÉGI_MENET(G,u)
      vége ha
   vége minden
vége MÉLYSÉGI_BEJÁRÁS
```

```
eljárás MÉLYSÉGI_MENET(G,u)
   kiír: u
   szín[u] ← SZÜRKE
   idő ← idő + 1
   elér[u] ← idő
   minden v ∈ Szomszéd(u) végezd
      ha szín[v] = FEHÉR akkor
         apa[v] ← u
         MÉLYSÉGI_MENET(v)
   vége minden
   szín[u] ← FEKETE
   idő ← idő + 1
   elhagy[u] ← idő
vége MÉLYSÉGI_MENET
```

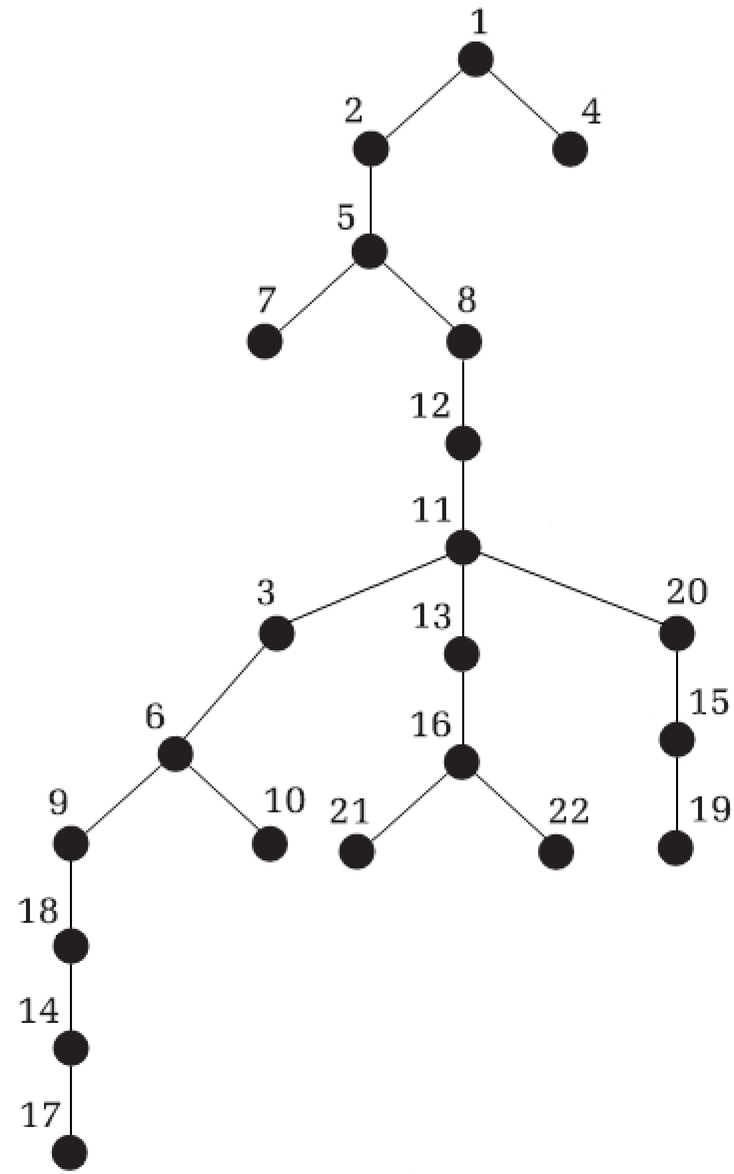
```
eljárás MÉLYSÉGI_BEJÁRÁS(G)
   minden u ∈ V(G) végezd
      szín[u] ← FEHÉR
      apa[u] ← 0
   vége minden
   idő ← 0
   minden u ∈ V(G) végezd
      ha szín[u] = FEHÉR akkor
         MÉLYSÉGI_MENET(G,u)
      vége ha
   vége minden
vége MÉLYSÉGI_BEJÁRÁS
```

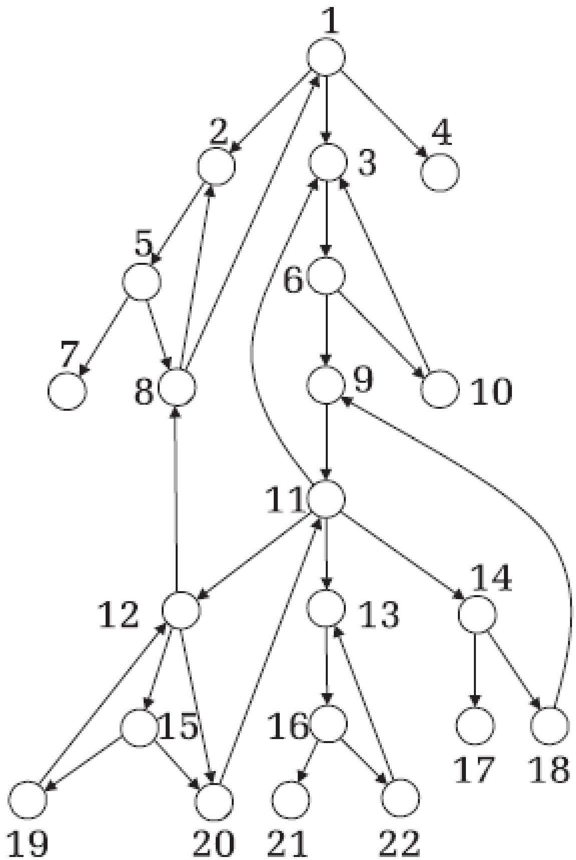
```
eljárás MÉLYSÉGI_MENET(G,u)
   kiír: u
   szín[u] ← SZÜRKE
   idő ← idő + 1
   elér[u] ← idő
   minden v ∈ Szomszéd(u) végezd
      ha szín[v] = FEHÉR akkor
         apa[v] ← u
         MÉLYSÉGI_MENET(v)
   vége minden
   szín[u] ← FEKETE
   idő ← idő + 1
   elhagy[u] ← idő
vége MÉLYSÉGI_MENET
```

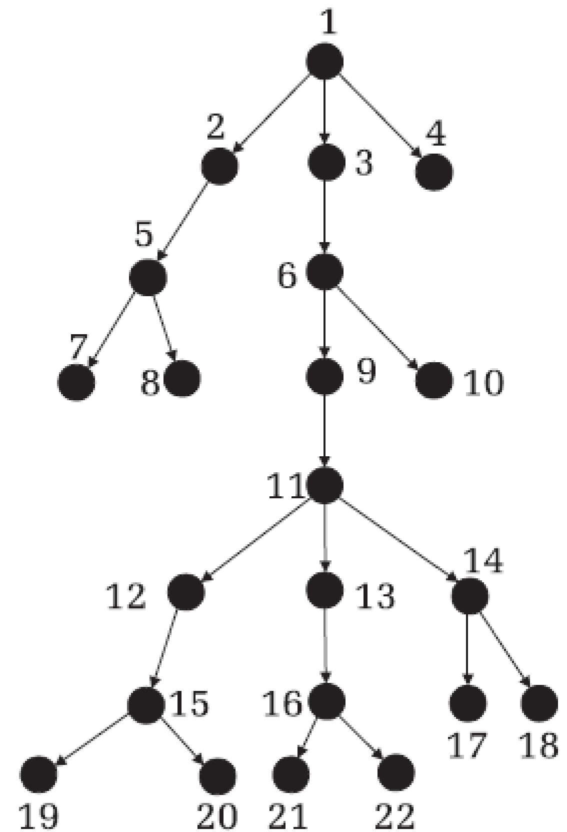
A MÉLYSÉGI_BEJÁRÁS algoritmus bonyolultsága $O(n+m)$.
(ha a gráf csúcslistával van tárolva)



Irányítatlan gráf mélységi fája





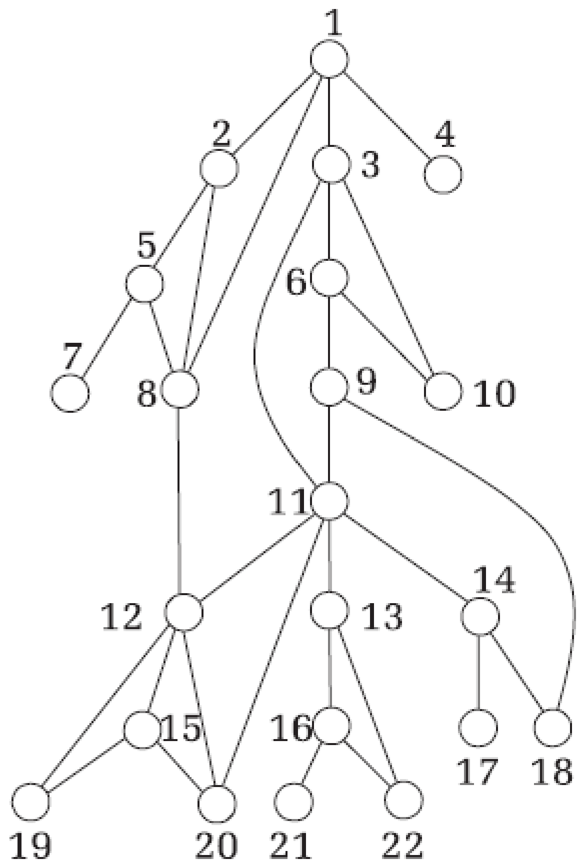


Irányított gráf mélységi fája

Megfigyelhető, hogy míg a szélességi fák általában szélesek és „alacsonyak“, addig a mélységi fák inkább mélyek és „karcsúak“. Például a K_n irányítatlan teljes gráf szélességi fája két szint mélységű és $(n - 1)$ pont széles, mélységi fája pedig n szintes egyenes fa.

Megfigyelhető, hogy míg a szélességi fák általában szélesek és „alacsonyak“, addig a mélységi fák inkább mélyek és „karcsúak“. Például a K_n irányítatlan teljes gráf szélességi fája két szint mélységű és $(n-1)$ pont széles, mélységi fája pedig n szintes egyenes fa.

A mélységi fát az algoritmus az `apa` tömbben kódolja.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Elér	1	2	9	42	3	10	4	6	11	19	8	7	23	13	32	24	14	12	33	31	25	27
Elhagy	44	41	22	43	40	21	5	39	18	20	37	38	30	16	35	29	15	17	34	36	26	18

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
apa	0	1	11	1	2	3	5	5	6	6	12	8	11	18	20	13	14	9	15	11	16	16

Megjegyzés:

Úgy is fogalmazhatnánk, hogy mélységi bejárás esetén mindig abból a szürke pontból lépünk tovább, amelyik legkésőbb vált szürkévé. Mivel erre az elvre épül a veremszerkezet is, ezért logikus, hogy ezt használjuk mint adatszerkezetet a mélységi bejárás implementálásánál. Mivel csak fehér pontok irányába lépünk tovább, nyilvánvaló, hogy a bejárt élek (feszítő)fát alkotnak.

Az élek osztályozása a mélységi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók, minden él akkor kap besorolást, amikor a mélységi bejárás először érzékeli a létezését:

Az élek osztályozása a mélységi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók, minden él akkor kap besorolást, amikor a mélységi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a mélységi fának.

Az élek osztályozása a mélységi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók, minden él akkor kap besorolást, amikor a mélységi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a mélységi fának.

Visszamutató él – ha v őse u-nak a mélységi fában (és ha (v, u) él nem minősült már faélnek).

Az élek osztályozása a mélységi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók, minden él akkor kap besorolást, amikor a mélységi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a mélységi fának.

Visszamutató él – ha v őse u-nak a mélységi fában (és ha (v, u) él nem minősült már faélnek).

Előremutató él – ha v utóda u-nak a mélységi fában (és ha (u, v) él nem minősült már faélnek).

Az élek osztályozása a mélységi bejárás során

Egy gráf (u, v) élei az alábbi módon osztályozhatók, minden él akkor kap besorolást, amikor a mélységi bejárás először érzékeli a létezését:

Faél – ha a v csúcspontot először az (u, v) él vizsgálata nyomán értük el. A faél olyan él, amely részévé vált a mélységi fának.

Visszamutató él – ha v őse u-nak a mélységi fában (és ha (v, u) él nem minősült már faélnek).

Előremutató él – ha v utóda u-nak a mélységi fában (és ha (u, v) él nem minősült már faélnek).

Keresztél – az összes többi él. Azokat az éleket kötik össze, amelyeknek végpontjai között nincs ős-utód, vagy utód-ős kapcsolat a mélységi fában.

Az (u, v) él besorolása meghatározható a v csúcs színe alapján:

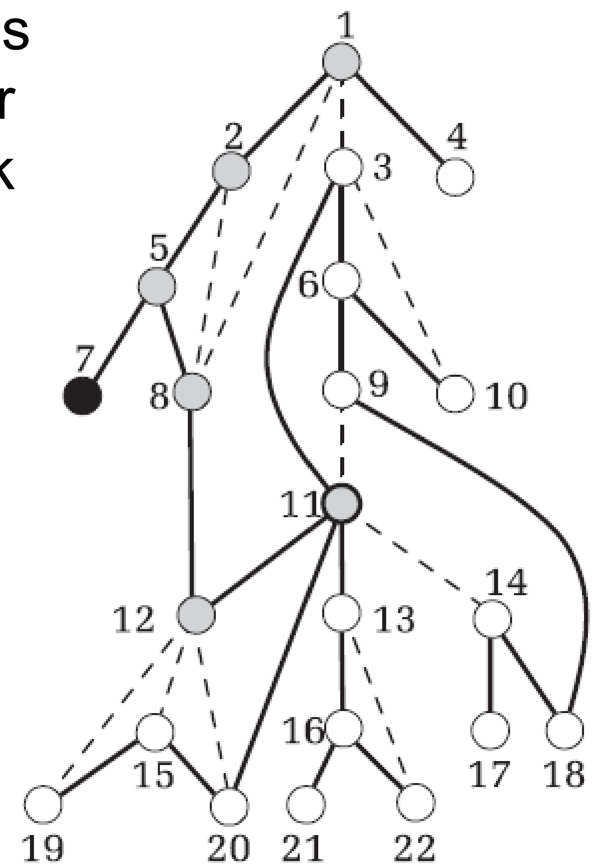
- FEHÉR – faél,
- SZÜRKE – visszamutató él,
- FEKETE – előre mutató él (ha $\texttt{elér}[u] < \texttt{elér}[v]$)
  vagy keresztél (ha $\texttt{elér}[u] > \texttt{elér}[v]$)

Egy irányítatlan gráf mélységi bejárása nyomán belátható, hogy:

- Egyik él sem lehet keresztél, sem előremutató él.

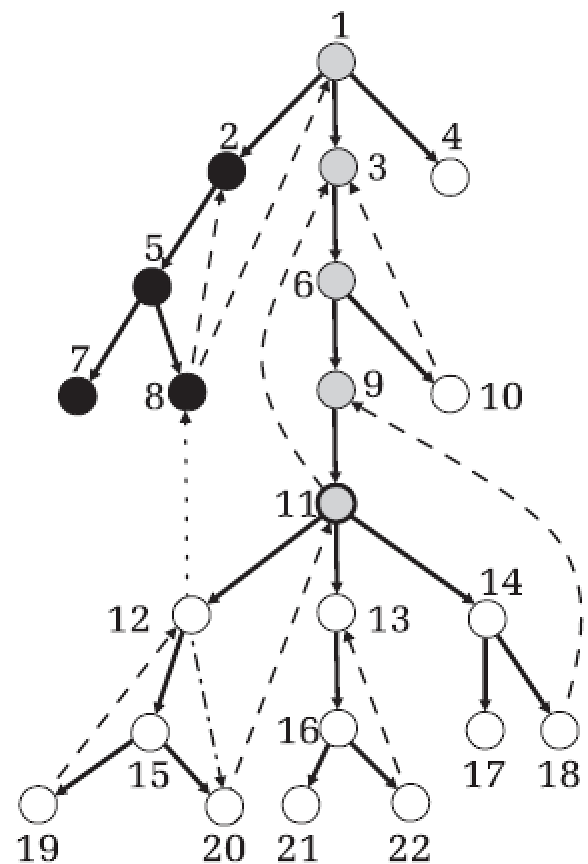
Egy irányítatlan gráf mélységi bejárása nyomán belátható, hogy:

- Egyik él sem lehet keresztél, sem előremutató él.
- A visszamutató éleket akkor érzékeli az algoritmus, amikor az aktuális pont (u) szomszédait (v) pásztázva valamelyiket szürkének találja. Kivéltelt képez az aktuális pont apa-pontja ($v = \texttt{apa}[u]$), amely bár szürke, az $(\texttt{apa}[u], u)$ él már faélnek minősült.



Egy irányított gráf mélységi bejárása nyomán belátható, hogy:

- A visszamutató éleket akkor érzékeli az algoritmus, amikor az aktuális pont (u) szomszédait (v) pásztázva valamelyiket szürkének találja.



Egy irányított gráf mélységi bejárása nyomán belátható, hogy:

- A visszamutató éleket akkor érzékeli az algoritmus, amikor az aktuális pont (u) szomszédait (v) pásztázva valamelyiket szürkének találja.

Megjegyzés:

Ha egy irányított gráfban töröljük az élek irányítását, akkor az ugyanabból a csúcspontból indított mélységi bejárás is átminősíti az éleket. A keresztélek rendszerint faélekké alakulnak át, az előre mutató élek pedig visszamutató élekké.