Problémamegoldás kereséssel

Bevezetés

- Problémamegoldó ágens
- Kívánt állapotba vezető cselekvéseket keres
- Probléma megfogalmazása
- Megoldás megfogalmazása
- Keresési algoritmusok
- Nem informált keresés: csak a probléma definícióját ismerik (mai óra)
- Informált keresés: egyéb információval is rendelkeznek (merre lehet a megoldás) – jövő óra

Problémamegoldás

- **Cél megfogalmazása: mi a célállapot?**
- **Problémamegfogalmazás: milyen cselekvések és állapotok számítanak?**
- **Keresés: több ismeretlen értékű lehetőségből kiválasztjuk, mit tegyünk, ha megvizsgáljuk, a lehetséges cselekvéssorok milyen állapotokba vezetnek**
- **Keresés -> megoldás**
- **Végrehajtás**

Feltételezések

- **Statikus környezet**
- **Teljesen megfigyelhető környezet**
- **Diszkrét környezet**
- **Determinisztikus környezet**

Jól definiált probléma

- **Kiinduló állapot**
- **Állapottér: kezdeti állapot + állapotátmenet-függvény:**
  - **Állapot -> cselekvés, utódállapot**
  - **Gráf**
  - **Út**
- **Célteszt: egy adott állapot célállapot-e**
- **Útköltség: minden útnak van költsége**

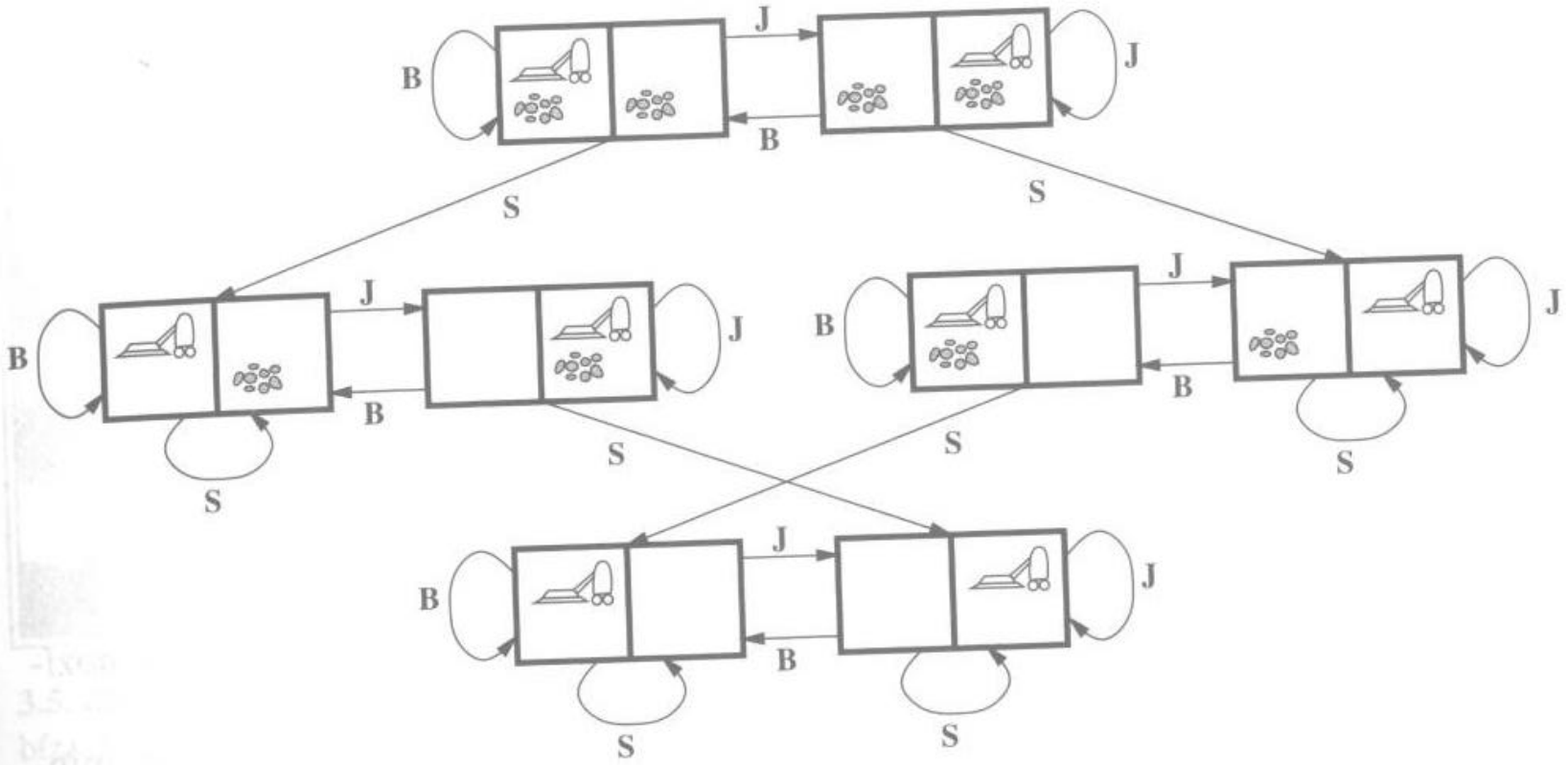
Probléma megoldásának hatékonysága

- **Megoldás: kiinduló állapotból a célállapotba vezető út**
- **Teljesség: talál-e megoldást, ha létezik megoldás?**
- **Optimalitás: optimális megoldást talál-e (alacsony költség)**
- **Keresési költség: idő, memória**
- **Komplexitás:**
  - **b: elágazási tényező**
  - **d: legsekélyebb célállapot mélysége**
  - **m: utak maximális hossza**

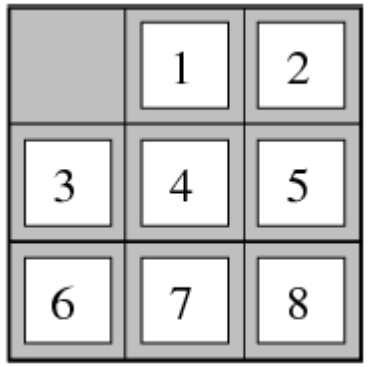
Porszívóvilág

- **Állapotok: 2 hely, mindegyik lehet piszkos/tiszta: n 2^n állapot (8)**
- **Kezdeti állapot: bármelyik**
- **Állapotátmenet-fv.: bal, jobb, szív cselekvésekből generált állapotok**
- **Célteszt: minden szoba tiszta-e**
- **Útköltség: minden lépés legyen 1 költségű**

Állapottér



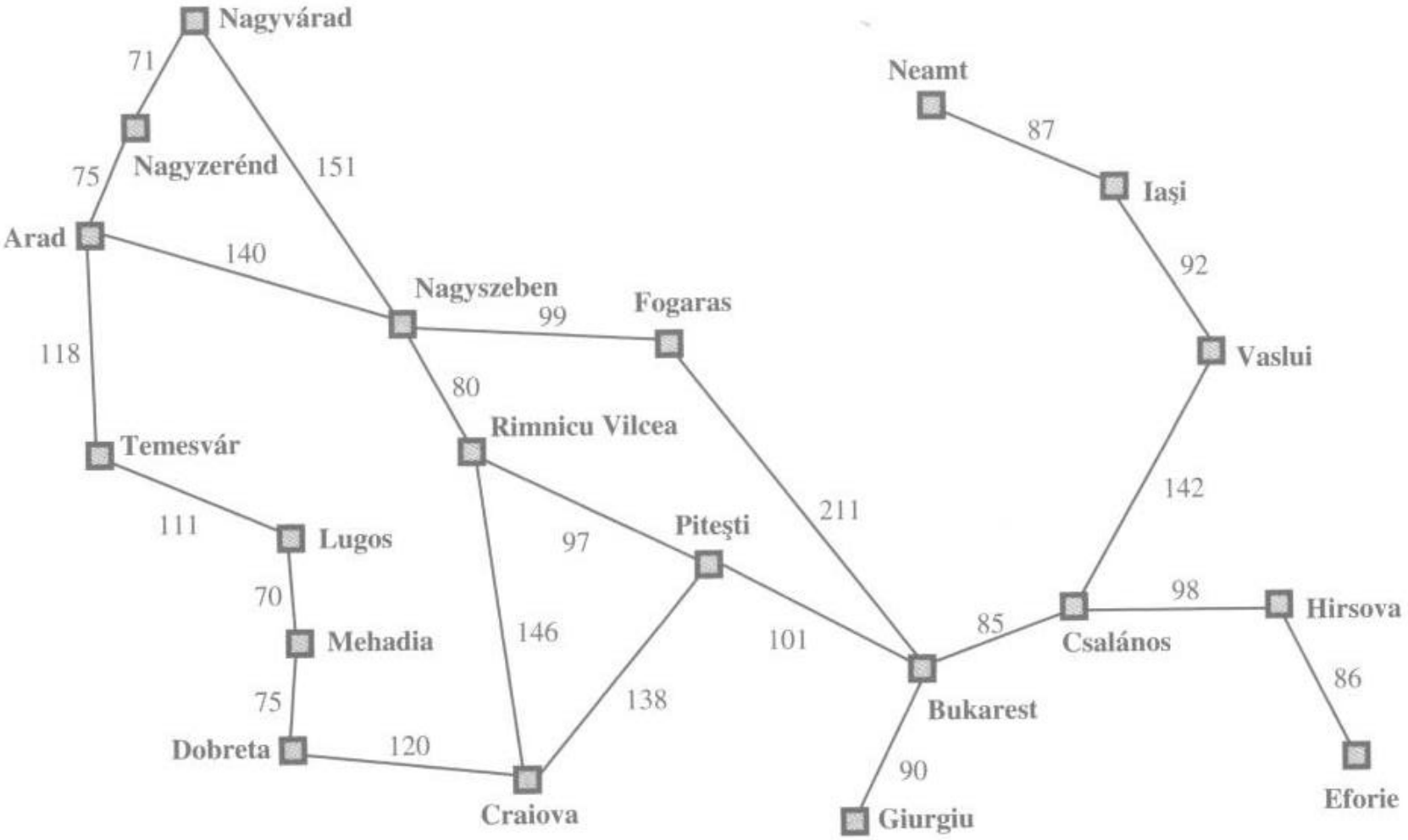
Kirakó



- **Állapotok: célállapotból csúsztatással elérhető konfigurációk**
- **Kezdeti állapot: bármelyik lehet**
- **Állapotátmenet-fv.: üres megy balra, jobbra, fel, le cselekvésekből generált legális állapotok**
- **Célteszt: célállapotban vagyunk-e?**
- **Útköltség: minden lépés 1**

utatócsoport

Útvonaltervezés: Arad-Bukarest



A keresési fa

- **adott egy állapot: merre lehet továbbindulni? → az állapot kifejtése, generálás**
- **állapottér → keresési fa**
- **a keresési fa csomópontjai:**
  - **az állapottérnek az adott csomóponthoz tartozó állapota**
  - **a szülőcsomópont a keresési fában**
  - **a csomópontot generáló operátor**
  - **a gyökércsomóponttól való távolság (mélység)**
  - **útköltség**

Általános keresési algoritmus

1. Legyen L kezdeti állapotokat tartalmazó lista;
2. Ha L üres, → VÉGE; nincs megoldás
3. Kiválasztunk egy csomópontot, legyen ennek neve : n;
4. Ha n a cél, akkor kész, a hozzávezető úttal megadja a megoldást → VÉGE; van megoldás
5. töröld n-t L-ből;
6. állítsd elő n gyermekeit;
7. jegyezd fel a hozzájuk vezető utat;
8. add a gyermekeket L-hez;
9. menj 2-re;

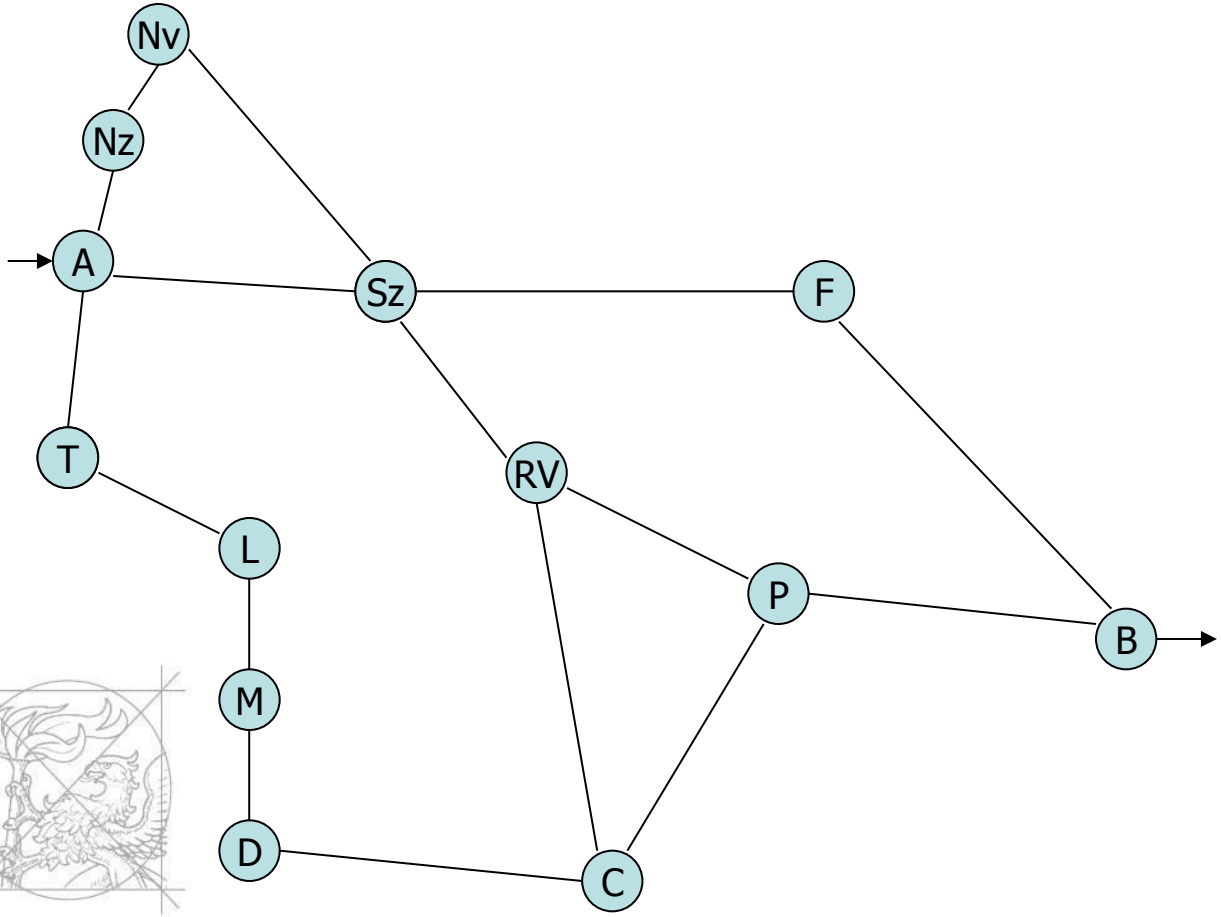
Keresési stratégiák

- véletlenszerű keresés
- vak keresés (nem informált keresés)
- heurisztikus keresés (informált keresés)

A keresési stratégiák tulajdonságai:

- teljesség
- időigény
- tárigény
- optimalitás

A probléma

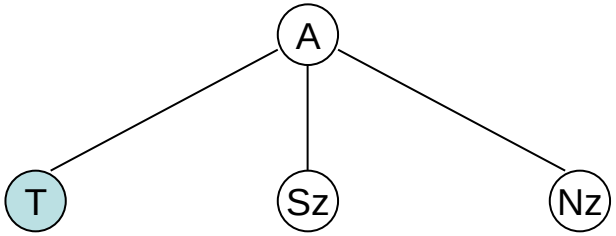


Szélességi keresés

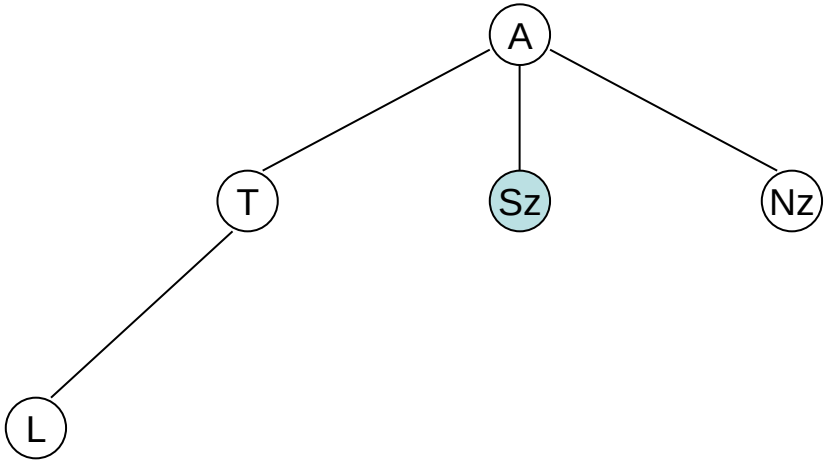
1. Legyen L kezdeti állapotokat tartalmazó lista;
2. Ha L üres, → VÉGE; nincs megoldás
3. Kiválasztjuk az első csomópontot, legyen ennek neve: n;
4. Ha n a cél, akkor kész, a hozzávezető úttal megadja a megoldást → VÉGE; van megoldás
5. töröld n-t L-ből;
6. állítsd elő n gyermekeit;
7. jegyezd fel a hozzájuk vezető utat;
8. add a gyermekeket L VÉGÉHEZ;
9. menj 2-re;

L_0: A

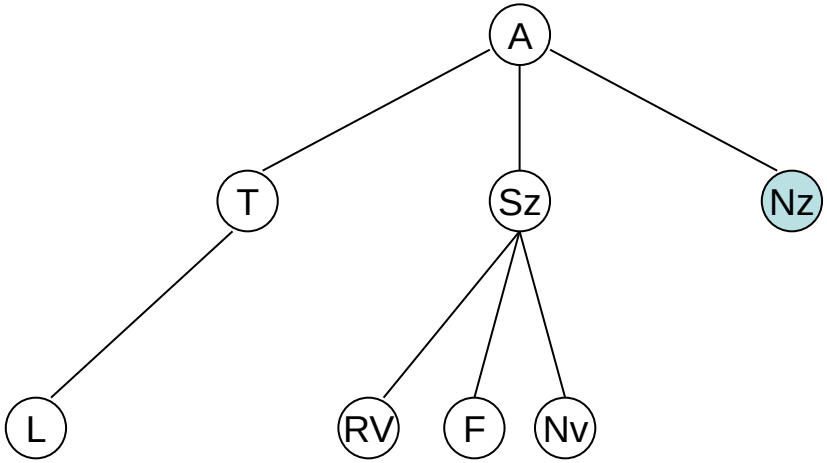
L_1: T Sz Nz



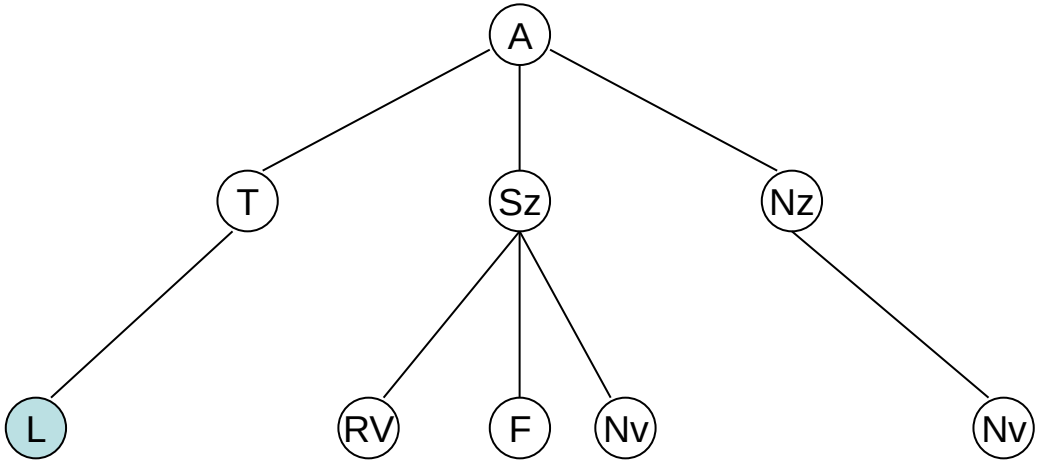
L_2: Sz Nz L



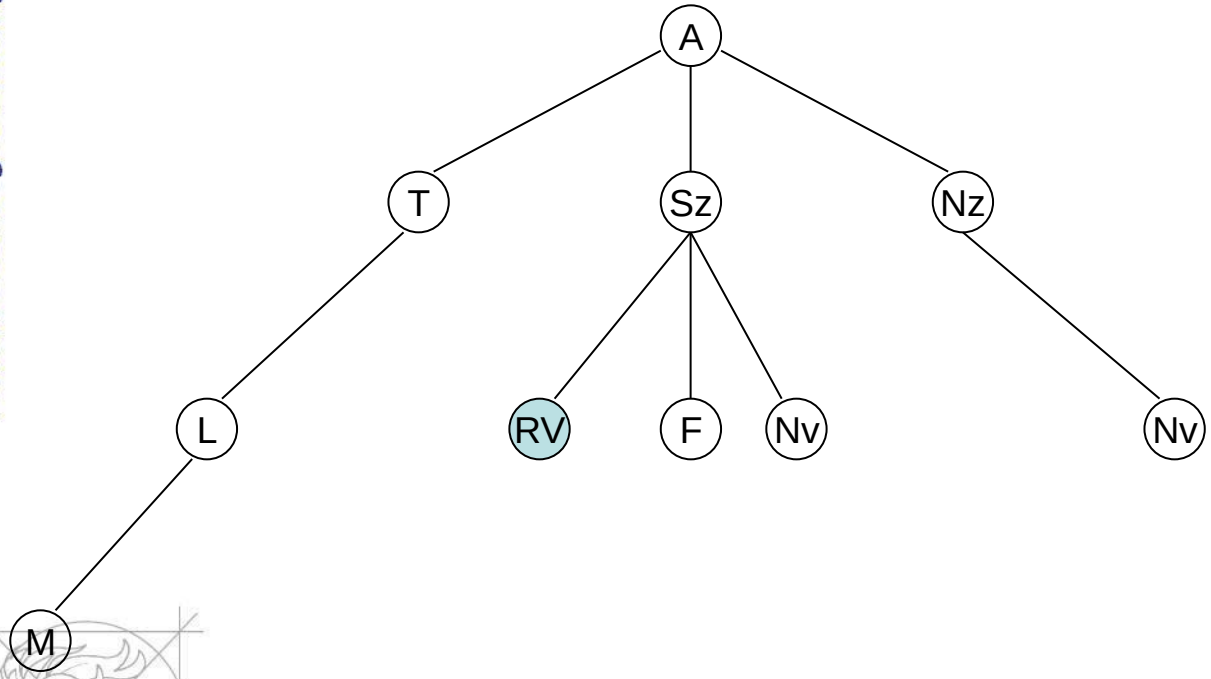
L_3: Nz L RV F Nv



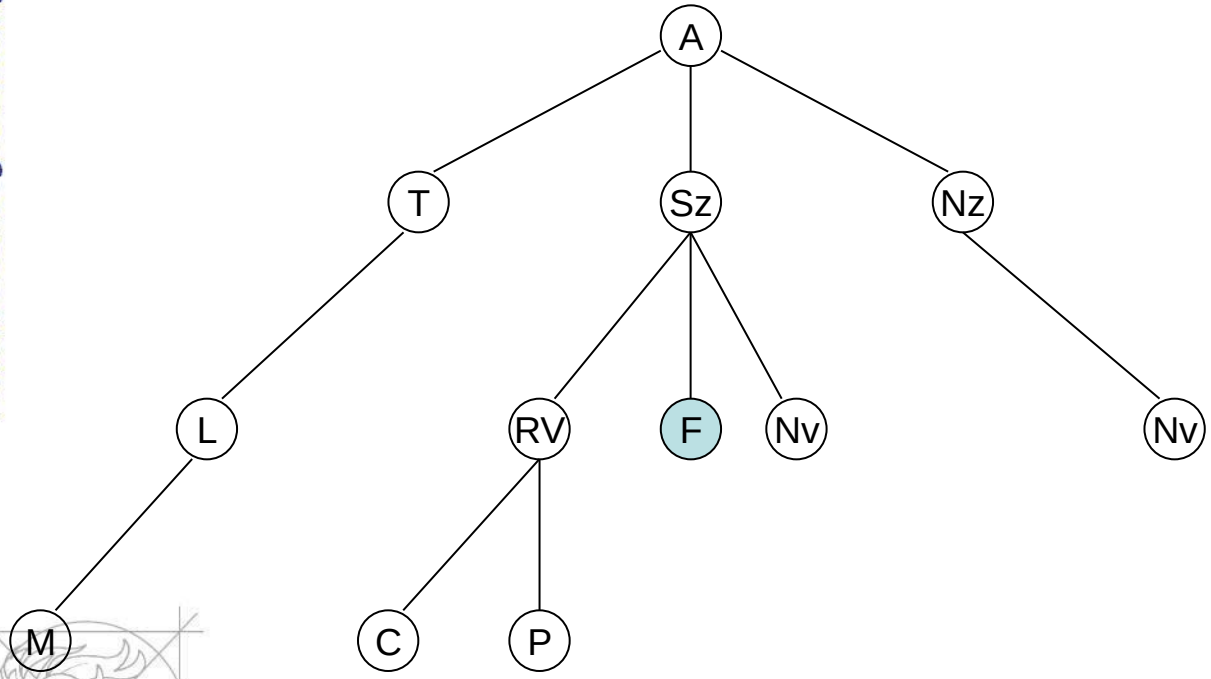
L_4: L RV F Nv Nv



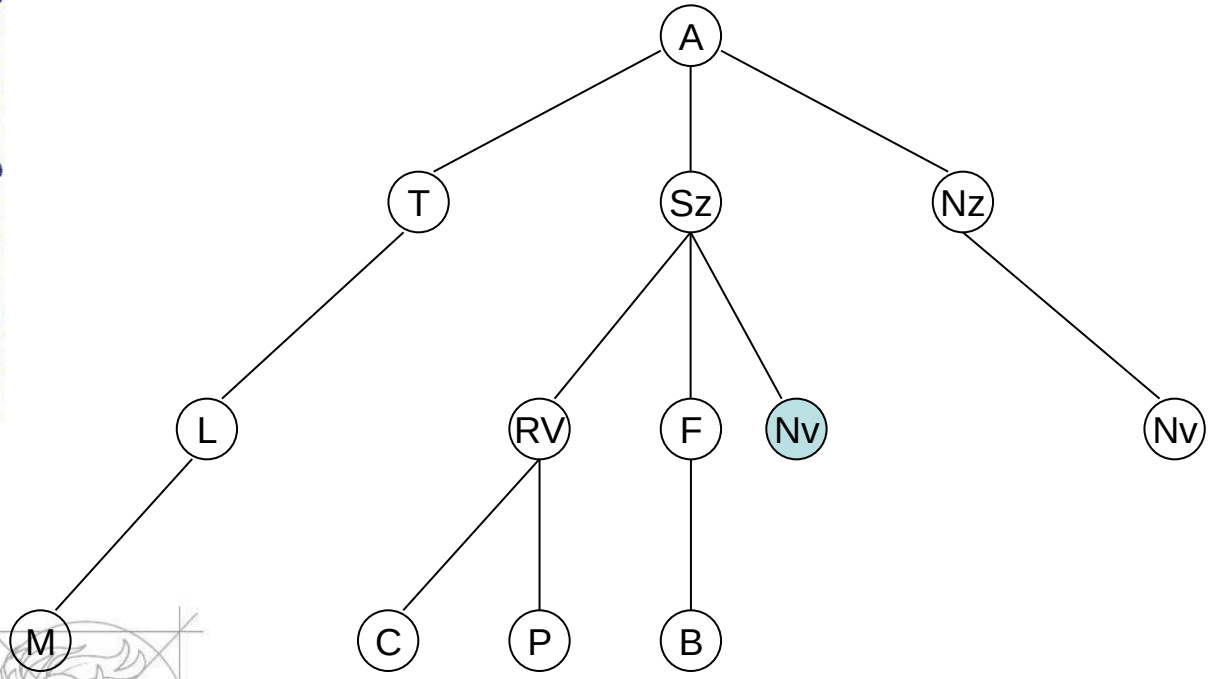
L_5: RV F Nv Nv M



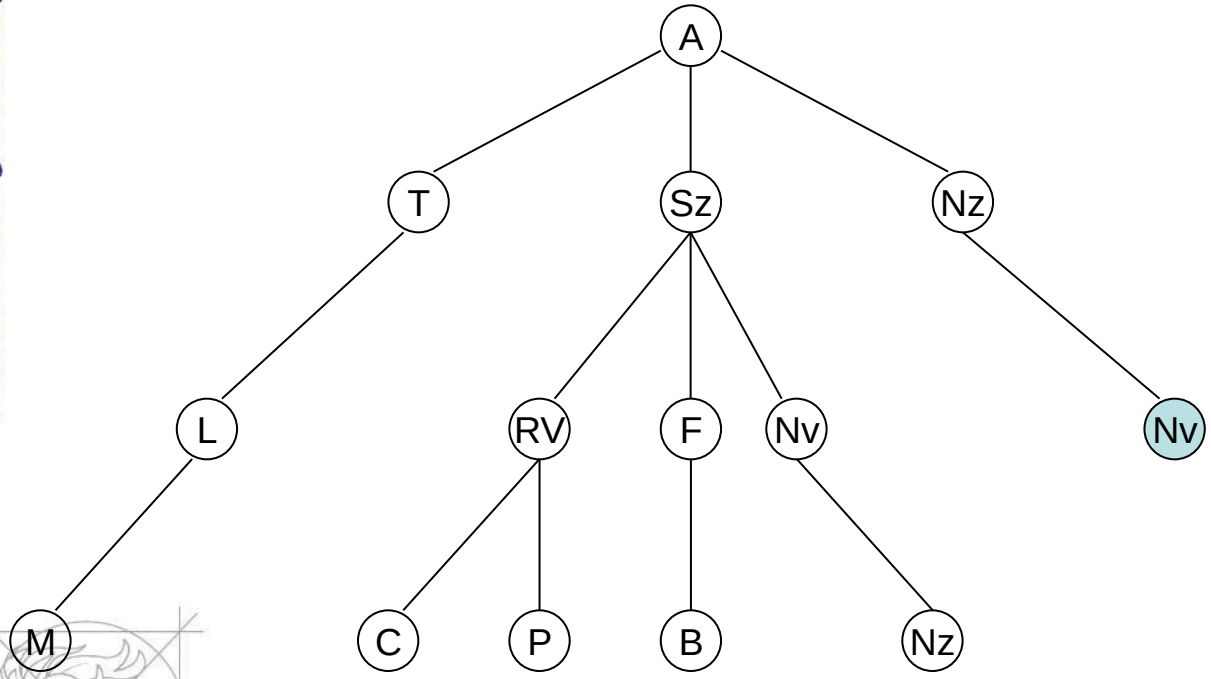
L_6: F Nv Nv M C P



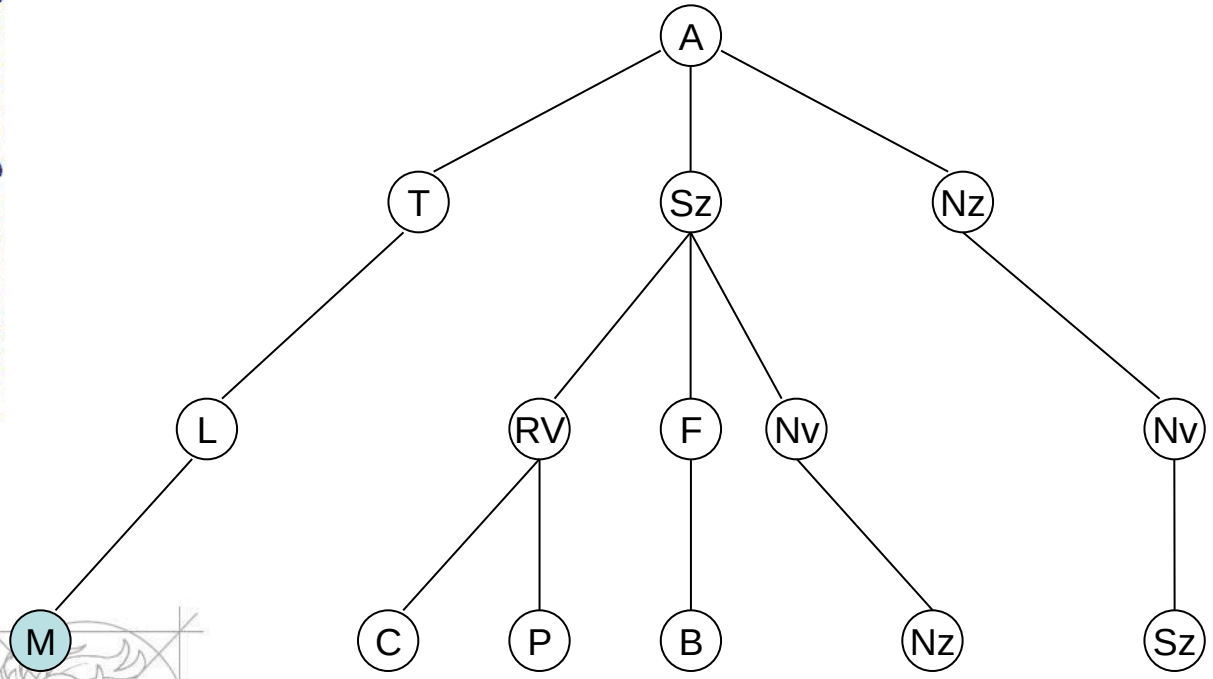
L_7: Nv Nv M C P B



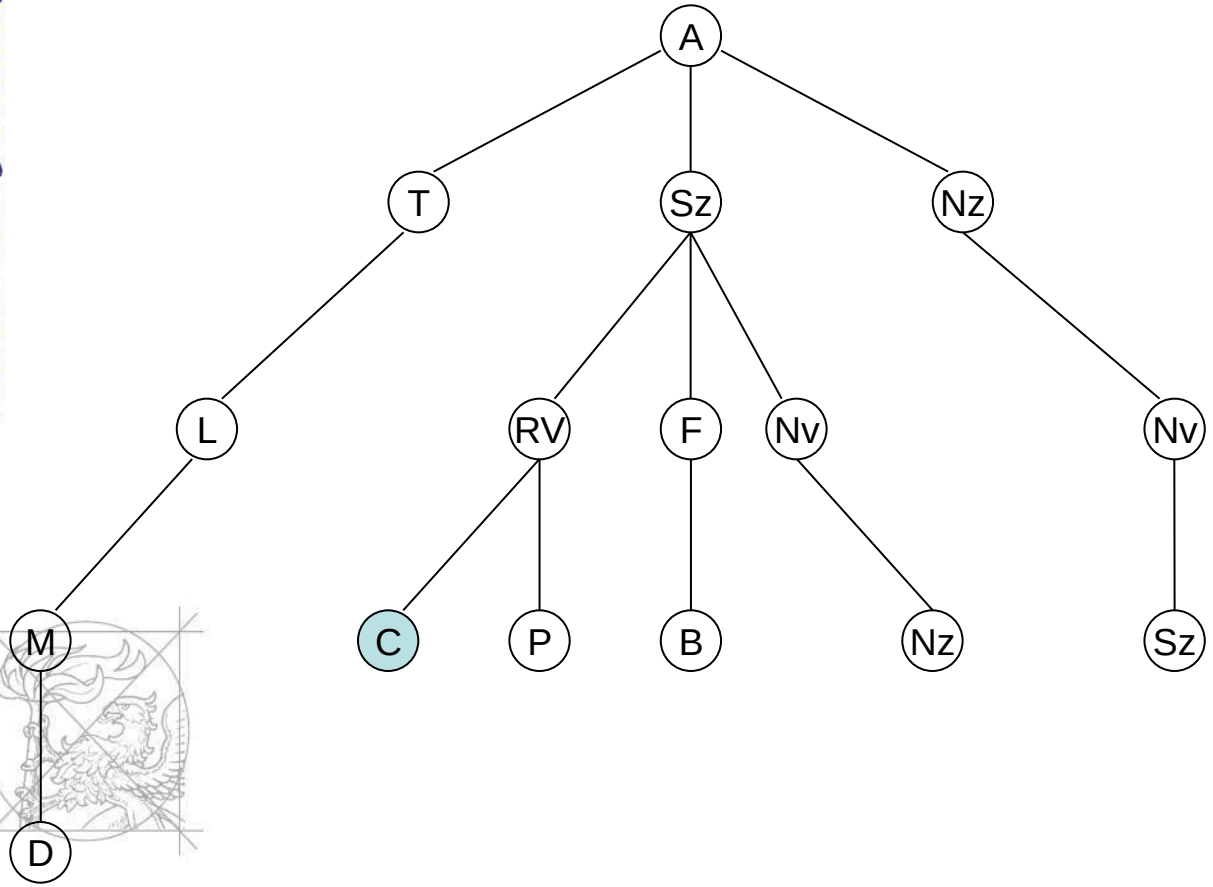
L_8: Nv M C P B Nz



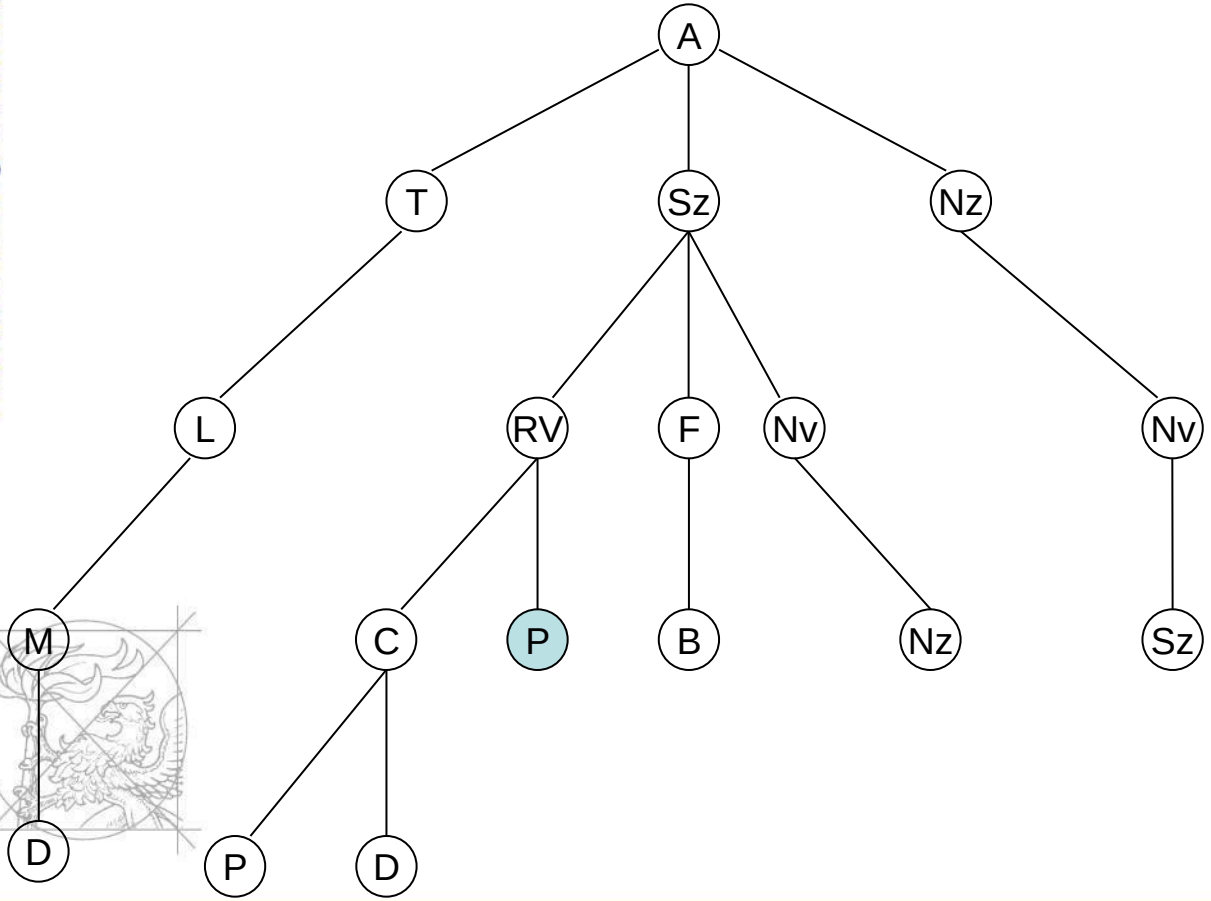
L_9: M C P B Nz Sz



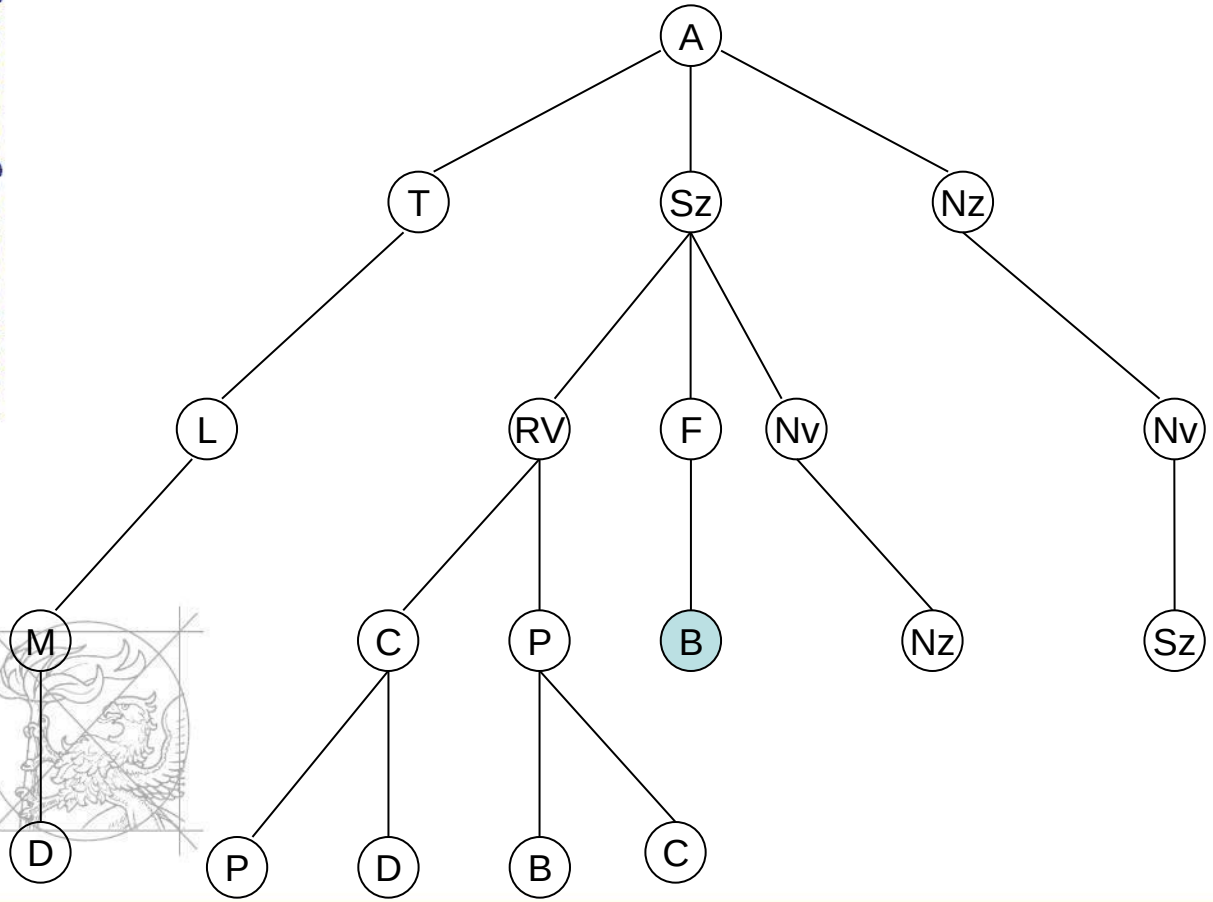
L_{10}: C P B Nz Sz D



L_{11}: P B Nz Sz D P D



L_{12}: B Nz Sz D P D B C



A szélességi keresés keresési költsége 10-es keresési fában (b=10)

Mélység (d)	*Csomópont*	*Időigény*	*Memória*
0	1	0,001 másodperc	100 byte
2	111	0,1 másodperc	11 kbyte
4	11111	11 másodperc	1 Mb
6	10^6	18 perc	111 Mb
8	10^8	31 óra	11 Gb
10	10^{10}	128 nap	1 Tb
12	10^{12}	35 év	111 Tb

Szélességi keresés

- mindig talál megoldást
- nagy keresési idő (b^d)
- nagy memóriaigény (b^d)
- Optimális, ha minden lépés ugyanolyan költségű

Egyenletes költségű keresés

útiköltség figyelembevétele!

1. **Legyen L kezdeti állapotokat tartalmazó lista;**
2. **Ha L üres, → VÉGE; nincs megoldás**
3. **Kiválasztjuk az első csomópontot, legyen ennek neve: n;**
4. **Ha n a cél, akkor kész, a hozzávezető úttal megadja a megoldást → VÉGE; van megoldás**
5. **töröld n-t L-ből;**
6. **állítsd elő n gyermekeit;**
7. **jegyezd fel a hozzájuk vezető utat;**
8. **add a gyermekeket L-hez;**
9. **rendezd L-t útiköltség szerint, az olcsóbb legyen elöl;**
10. **menj 2-re;**

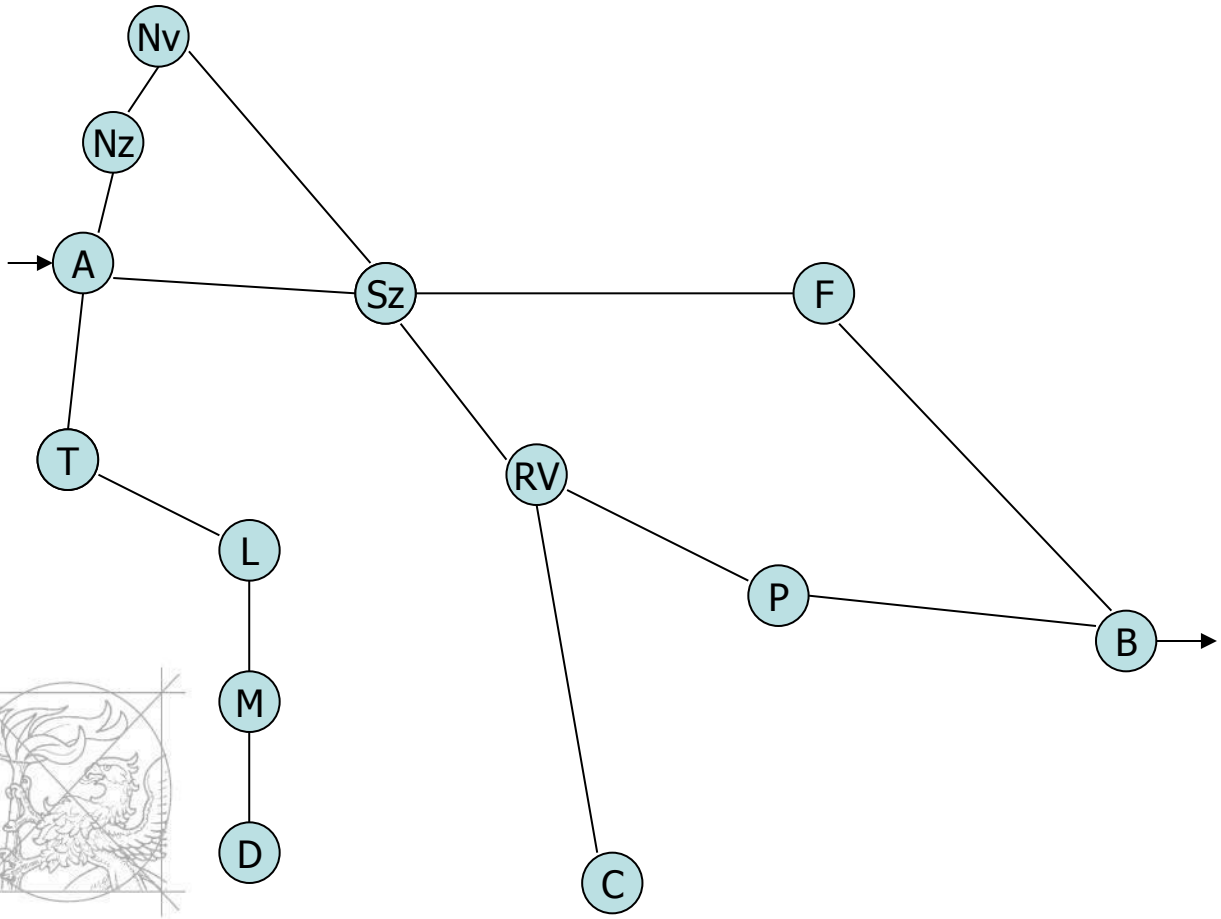
Egyenletes költségű keresés

- biztosan talál megoldást
- optimális (ha az útiköltség csak nőhet az út folyamán)
- nagy keresési költség (b^d)

Mélységi keresés

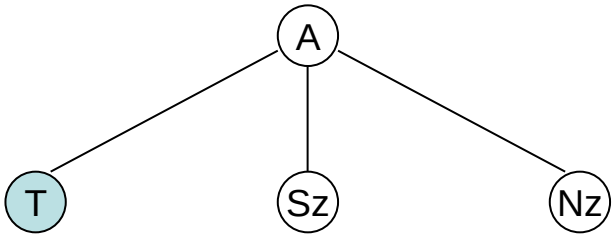
1. Legyen L kezdeti állapotokat tartalmazó lista;
2. Ha L üres, → VÉGE; nincs megoldás
3. Kiválasztjuk az első csomópontot, legyen ennek neve: n;
4. Ha n a cél, akkor kész, a hozzávezető úttal megadja a megoldást → VÉGE; van megoldás
5. töröld n-t L-ből;
6. állítsd elő n gyermekeit;
7. jegyezd fel a hozzájuk vezető utat;
8. add a gyermekeket L ELEJÉHEZ;
9. menj 2-re;

A módosított probléma

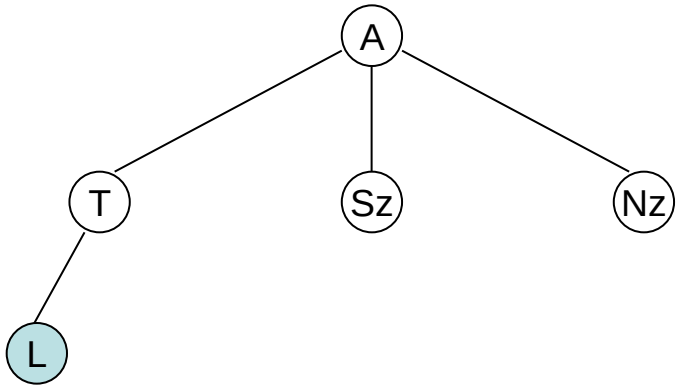


L_0: A

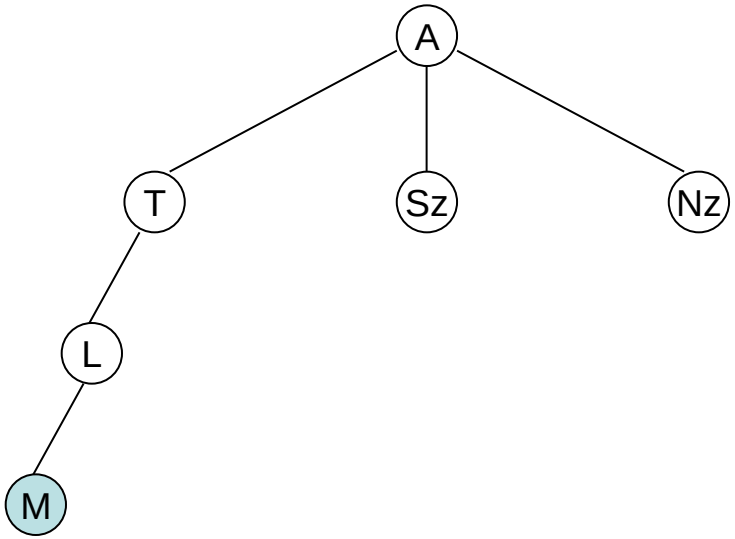
L_1: T Sz Nz



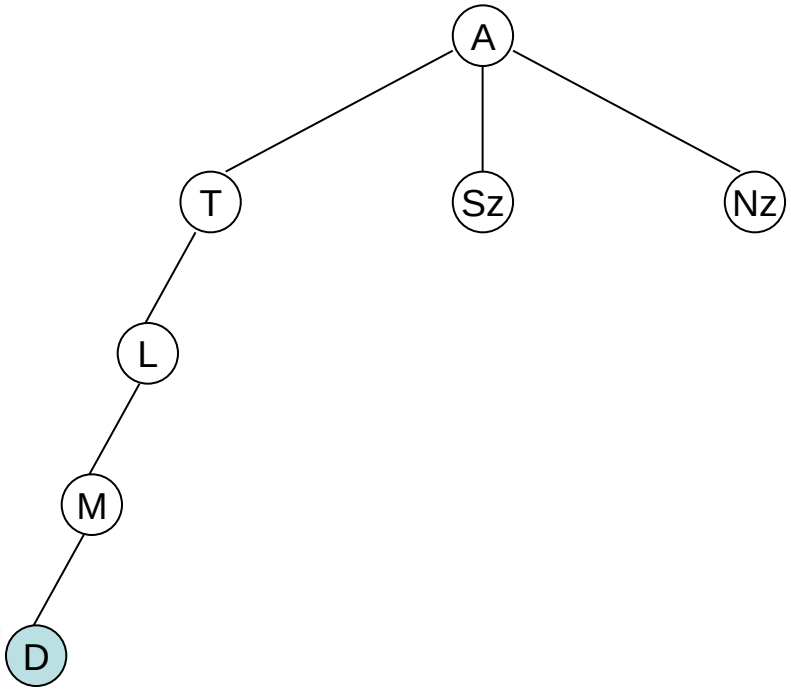
L_2: L Sz Nz



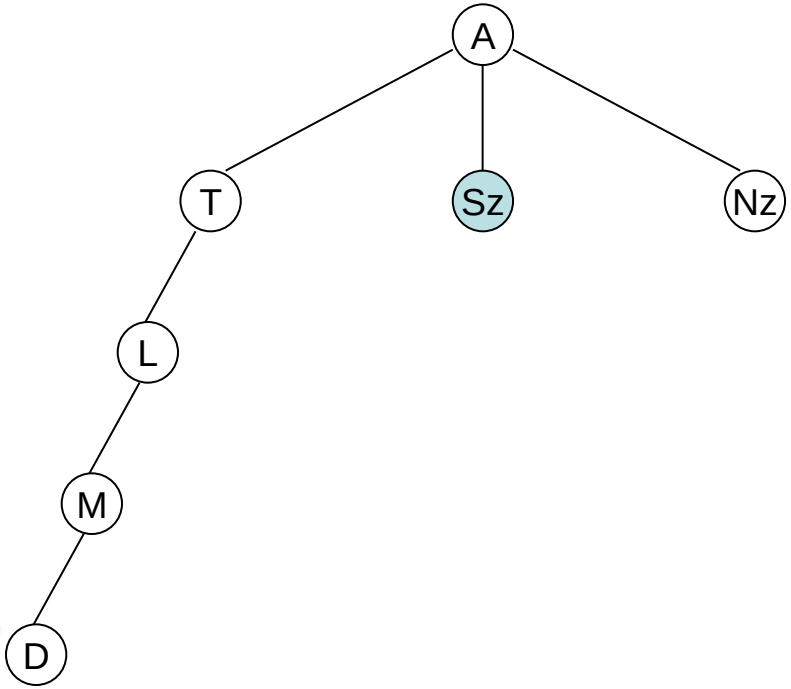
L_3: M Sz Nz



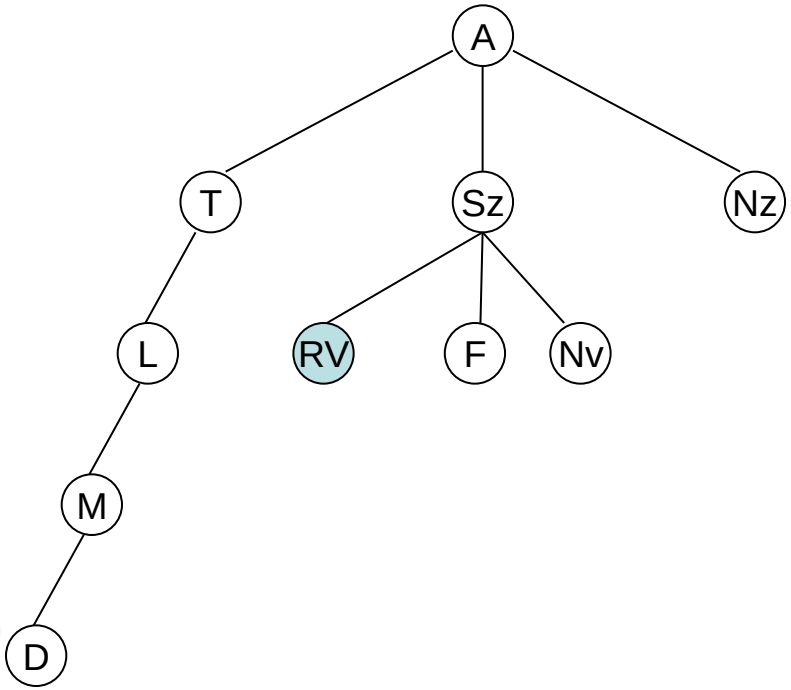
L_4: D Sz Nz



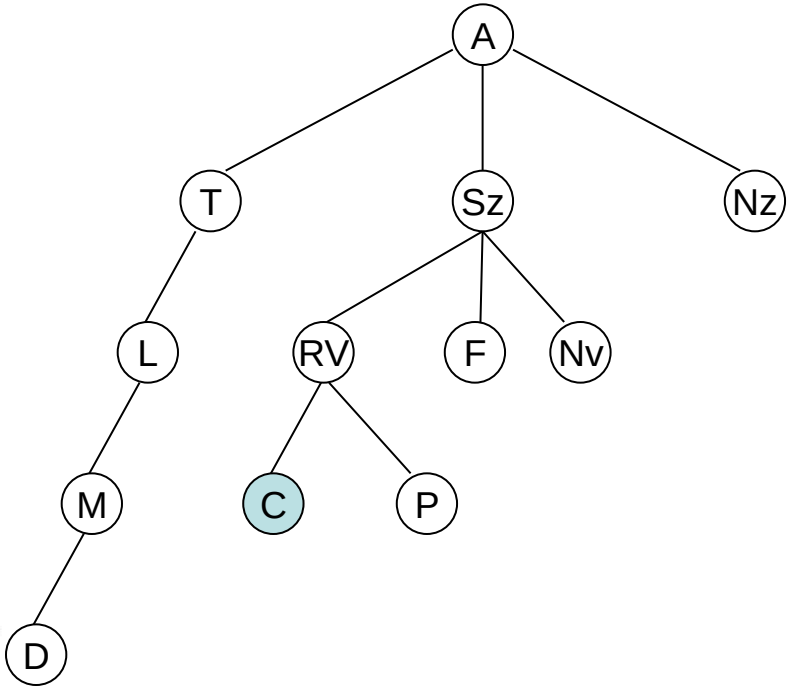
L_5: Sz Nz



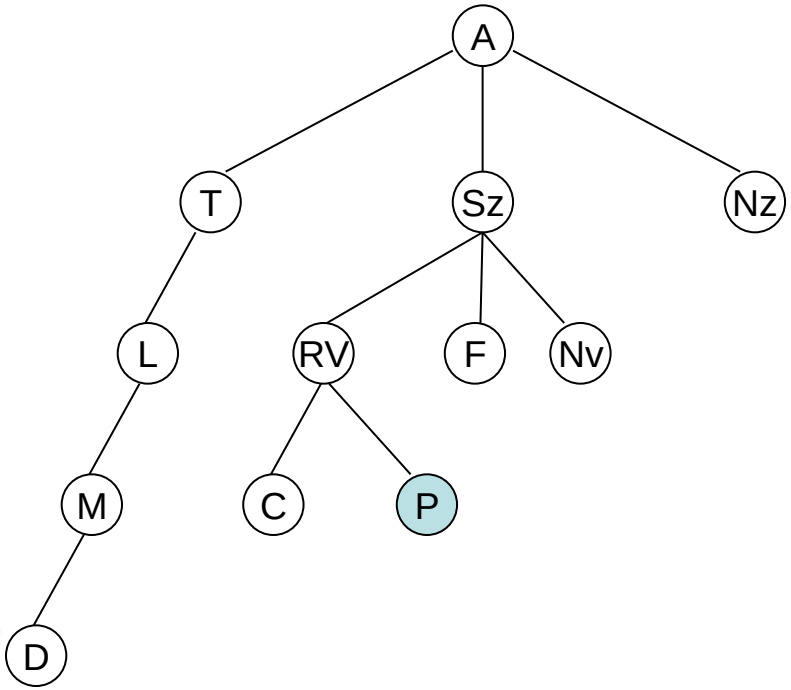
L_6: RV F Nv Sz Nz



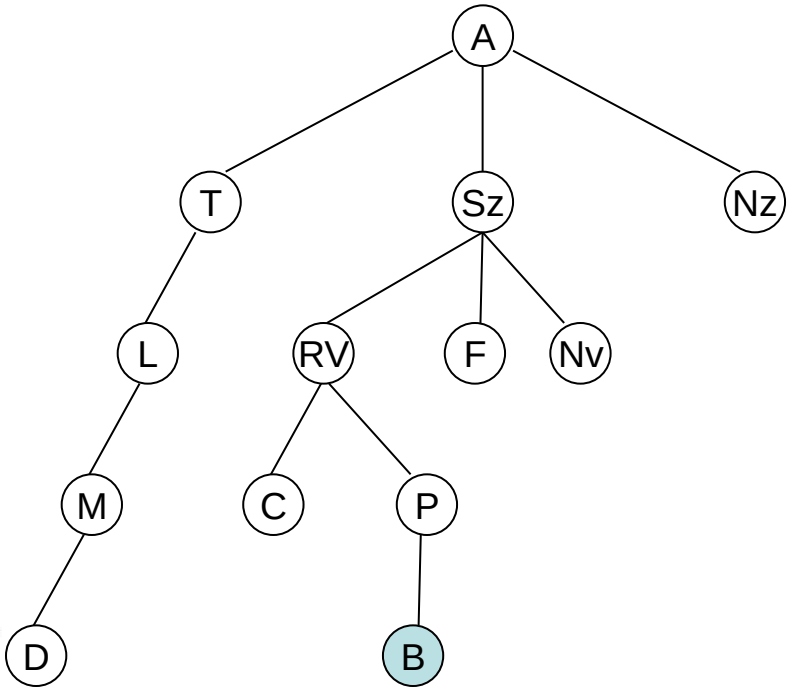
L_7: C P F Nv Sz Nz



L_8: P F Nv Sz Nz



L_9: B F Nv Sz Nz



Mélységi keresés

- kis memóriaigényű (b*d)
- nagy keresési idő (b^d)
- nem biztos, hogy talál megoldást (végtelen mélységű ágban elakad)
- nem optimális

Mélységkorlátozott keresés

1. Legyen L kezdeti állapotokat tartalmazó lista;
2. Ha L üres, → VÉGE; nincs megoldás
3. Kiválasztjuk az első csomópontot, legyen ennek neve: n;
4. Ha n a cél, akkor kész, a hozzávezető úttal megadja a megoldást → VÉGE; van megoldás
5. töröld n-t L-ből;
6. HA n MÉLYSÉGE NEM NAGYOBB, MINT K, állítsd elő n gyermekeit;
7. jegyezd fel a hozzájuk vezető utat;
8. add a gyermekeket L elejéhez;
9. menj 2-re;

Mélységkorlátozott keresés

- kis memóriaigényű (b*d)
- nagy keresési idő (b^d)
- biztosan talál megoldást
- nem optimális

Iteratívan mélyülő keresés

1. Legyen L kezdeti állapotokat tartalmazó lista, K = 0;
2. Ha L üres, → VÉGE; nincs megoldás
3. Kiválasztjuk az első csomópontot, legyen ennek neve: n;
4. Ha n a cél, akkor kész, a hozzávezető úttal megadja a megoldást → VÉGE; van megoldás
5. töröld n-t L-ből;
6. ha n mélysége nem nagyobb, mint K, állítsd elő n gyermekeit;
7. jegyezd fel a hozzájuk vezető utat;
8. add a gyermekeket L elejéhez;
9. K = K + 1;
10. menj 2-re;

Iteratívan mélyülő keresés

- sok állapotot többször is kifejt
- keresési idő: $2*(b^d)$
- kis memóriaigény (b*d)

Kétirányú keresés

- **Kezdeti állapotból és a célállapotból is indítunk keresést**
- **Ha a két keresés találkozik -> VÉGE**
- **Idő: $b^{d/2}$**
- **Memória: $b^{d/2}$**
- **Optimális és teljes, ha mindkét irányban szélességi keresés**