Ágensek

1. Kérdés: Sorolja fel a környezetet meghatározó tényezőket és ezek alapján definiálja a lehetséges környezeteket.

1. Válasz:

teljesen/részben megfigyelhető:

**Teljesen megfigyelhető**: az ágens minden pillanatban pontosan látja a környezet állapotát.

**Részlegesen megfigyelhető**: az ágens csak korlátozott vagy zajos információhoz jut.

determinisztikus/sztochasztikus: Ha a környezet következő állapotát egyértelműen meghatározza az aktuális állapot és az ágens cselekedete, akkor a környezetet determinisztikusnak nevezzük. Stratégiai a környezet, ha más ágensek cselekvéseit leszámítva determinisztikus.

epizódszerű/sorozatszerű: Epizódszerű környezetben az ágens tapasztalata atomi epizódokra bontható; minden egyes epizód az ágens észleléseiből és cselekedetéből áll. A következő epizód független az előzőtől. (Pl. futószalag)

statikus/dinamikus: Ha a környezet megváltozhat, amíg az ágens gondolkodik, akkor a környezet dinamikus. Ha a környezet időben nem változik, de az ágens teljesítménymértéke igen, akkor szemidinamikus a környezet.

diszkrét/folytonos:

**Diszkrét:** a lehetséges állapotok és akciók száma véges.

**Folytonos:** a környezetben a változók folytonos értékeket vehetnek fel.

egy/többágenses: Hány ágens található a környezetben? Több ágens esetén lehet versengő vagy kooperatív

2. kérdés: Mi a racionális ágens (és mi nem)?

2. válasz: A racionális ágens azt a műveletet választja, mely maximalizálja a várható teljesítménymértéket adott érzékeléssorozathoz (**a rendelkezésére álló információból a legjobb döntést hozza**)

3. kérdés: Mitől függ egy ágens racionalitása (adott pillanatban)?

3. válasz: teljesítménymérték, környezet, beavatkozók, érzékelők

4. kérdés: Mi a különbség és a hasonlóság az ágensprogram és ágensfüggvény között?

4. válasz: az ágensfüggvény minden lehetséges körülmény esetén megadja az ágens cselekvéseit (ez egy matematikai formula), az ágensprogram implementálja az ágensfüggvényt

5. kérdés: Mi a specialitása az egyszerű reflex ágensek?

Nem tárolják a múltbeli észleléseket, és nem tudják követni a világ állapotát → csak a jelen pillanat számít. Az aktuális **észlelésre (percept)** közvetlenül egy **akciót** választanak egy előre megadott szabály („ha–akkor”) alapján.6. kérdés: Mi a specialitása a modellalapú reflexszerű ágensek?

6. válasz: Nemcsak az aktuális észlelésre reagálnak, hanem **nyilvántartják a világ állapotát** a korábbi perceptumok alapján.

7. kérdés: Mi a specialitása a célorientált ágensnek?

7. válasz: **Nem csak reagál**, hanem **célokat** is figyelembe vesz a döntéseiben. Nem elégszik meg a „helyes reakcióval”, hanem megvizsgálja, hogy az adott akció mennyire segít **elérni a kitűzött célt**.

8. kérdés: Mi a specialitása a hasznosságorientált ágensnek?

8. válasz: **Nemcsak célokat** vesz figyelembe, hanem **hasznosságot** (utility) is rendel a különböző kimenetekhez. Így képes **megkülönböztetni** több, ugyanarra a célra vezető út közül melyik a **jobb**

9. kérdés: Mi a specialitása a tanuló ágensnek?

9. válasz: **Képes javítani a teljesítményét az idő során** a **tapasztalat alapján**. Nem csak a pillanatnyi észlelések, a modell vagy a hasznosság alapján dönt, hanem **folyamatosan tanul a korábbi tapasztalatokból**.



Keresések és nem informált keresések

1. kérdés: Mi a különbség az online és az offline keresések között?

1. válasz:

**Offline problémamegoldás**

* A rendszer **minden szükséges információt előre ismer** a környezetről.
* A döntéseket és a tervet **előre, teljes tudás birtokában** készíti el.



**Online problémamegoldás**

* Az ügynök **nem rendelkezik minden információval a kezdetekkor**.
* **Menet közben, a környezetből gyűjti az adatokat**, és azok alapján módosítja a tervét.

2. kérdés: Milyen komponensei vannak egy jól definiált problémának keresési feladatok esetén?

2. válasz: kezdeti állapot, állapotátmenet függvény, célteszt, útköltség

3. kérdés: Keresési feladat esetén hogyan adjuk meg a cselekvéseket?

**3. válasz: A cselekvéseket egy függvénnyel (Actions), az állapotátmenetet pedig egy másik függvénnyel (Result) adjuk meg, így formálisan definiáljuk, hogyan mozog az ágens az állapottérben.**

4. kérdés: Mi a kapcsolat az állapottér és keresési fa között?

4. válasz: Az **állapottér = gráf**, a **keresési fa = ennek egy része**, amit a kereső algoritmus épít a kezdőállapottól kiindulva.

5. kérdés: Mit kell tárolni a keresési fa egy csúcsában?

5. válasz: szülő, gyerekek, mélység, útköltség

6. kérdés: Mi a különbség egy állapot és egy keresési fa csúcsa között?

6. válasz: az állapotnak nincs szülője, gyereke, mélysége, útköltsége

7. kérdés: Adja meg az általános fakeresési algoritmust!

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

8. kérdés: Adja meg az általános gráfkeresési algoritmust.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

9. kérdés: Mit jelent az, hogy vak (nem informált) keresés?

9. válasz: **Csak a problémastruktúrát és a csomópontokat ismeri**, de semmit sem tud arról, hogy melyik irány lehet ígéretesebb.

10. kérdés: Mi a perem? Adott adatszerkezetet peremként használva milyen keresési módszereket kapunk?

10. válasz: az **aktuálisan felfedezett, de még ki nem vizsgált csomópontok halmaza**.  
Széllességi bejárásnál sor van, mélységinél verem.

11. kérés: Adja meg a lényeges különbségeket a különféle nem informált keresési módszerek között!

11. válasz: **1. Szélességi keresés (BFS)**

* **Perem típusa:** FIFO sor
* **Működés:** Szintenként halad, először a gyökér gyerekeit vizsgálja, majd azok gyerekeit, és így tovább.
* **Optimalitás:** Igen, ha minden lépés költsége azonos, a legrövidebb utat találja meg.



* **Memóriaigény:** Nagy, mert minden szint összes csomópontját tárolni kell.
* **Alkalmazás:** Rövid út, kis állapottér.

**2. Egyenletes költségű keresés (Uniform Cost Search, UCS)**

* **Perem típusa:** Prioritásos sor a csomópontok költsége alapján
* **Működés:** Mindig a legolcsóbb csomópontot választja ki következőnek.
* **Optimalitás:** Igen, mindig a legkisebb költségű megoldást találja meg.
* **Memóriaigény:** Nagy, sok csomópontot kell tárolni.
* **Alkalmazás:** Lépcsők vagy lépések eltérő költsége esetén ideális.

**3. Mélységi keresés (DFS)**

* **Perem típusa:** LIFO verem
* **Működés:** Egy ágon mélyen halad, amíg el nem ér egy levelet vagy megoldást, majd visszalép.
* **Optimalitás:** Nem garantált a legrövidebb út.
* **Memóriaigény:** Kisebb, csak az aktuális ágat tárolja.
* **Alkalmazás:** Nagy vagy végtelen állapottér esetén előnyös, de végtelen ágon elakadhat.

**4. Mélységkorlátozott keresés (DFS + mélységlimit)**

* **Perem típusa:** LIFO verem, maximális mélységgel
* **Működés:** DFS, de nem lépi túl a megadott mélységet.
* **Optimalitás:** Nem garantált, de megelőzi a végtelen ágakba való elakadást.
* **Memóriaigény:** Kisebb, csak az aktuális ágat kell tárolni.



* **Alkalmazás:** Végtelen fák vagy mélység korlátozása esetén.

**5. Iteratívan mélyülő mélységi keresés (IDDFS)**

* **Perem típusa:** DFS ismételt futtatással növekvő maximális mélységgel



* **Működés:** Többször futtatja a DFS-t, először mélység 0-tól, majd 1-től, 2-től, …
* **Optimalitás:** Igen, megtalálja a legrövidebb utat, de memóriaigénye kisebb, mint BFS-é.
* **Memóriaigény:** Kisebb, mint BFS, csak az aktuális ágat kell tárolni.
* **Alkalmazás:** Nagy vagy végtelen fák, ahol a legrövidebb út megtalálása fontos.

12. kérdés: Melyek a teljes fakeresési módszerek (milyen feltételek mellett)?

12. válasz:

Szélességi: teljes ha b véges

Egyenletes költégű: teljes ha az élköltéget pozitivak és nagyobbak eps-nal

mélységi-korlátozott: teljes ha mélység limiten belül

iterativan mélyülő mélységi: mindig teljes

13. kérdés: A valós életben a szélességi keresés vagy az iterált mélységi keresés ad gyorsabban eredményt?  
13. válasz: Az iteratívan mélyülő mélységi keresés lineáris memóriát használ így rendszerint sokkal gyorsabban végez

14 kérdés: Mely fakeresési módszerek optimálisak?/ Melyek optimális gráfkeresési módszerek?

14. válasz: szélességi, egyenletes költségű, iteratívan mélyülő mélységi

15. kérdés: Mikor szükséges gráfkeresést használni fakeresés helyett?

15. válasz: A **gráfkeresést** akkor szükséges használni, amikor a probléma **állapottérében ismétlődő vagy ciklikus állapotok** fordulhatnak elő.



**Informált keresések**



**1.** kérdés: Melyek a legjobbat először keresés-család jellemzői?

1. válasz: nem teljes mivel beragadhat viszont azzá tehető, idő és tárbonyolultsága O(bm) és nem optimális. Egy jó heurisztika felgyorsíthatja, minden csúcsot a memóriában tart. kivánatosság maximalizálást végez kivánatossági heurisztika függvény alapján

2. kérdés: Milyen problémái vannak a mohó legjobbat először keresésnek?

2. válasz: Nem teljes, mivel beragadhat, és nem optimális. Nagy tár és időbonyolultság jellemzi.

3. kérdés: Melyek az A\* keresés jellemzői?

3. válasz: Olyan mint mohó legjobbat először, de a hátralévő út heurisztika mellett a csúcsig megtett utat is számba veszi, optimális eredményt ad. Optimális, teljes (ha nincs végtelen csúcs), időbonyolultsága exponenciális, tárbonyolultság szerint meg minden csúcs a memóriában van

4. kérdés: Mit jelent az elfogadható heurisztika?

4. válasz: **soha nem becsüli túl a cél elérésének valódi költségét**, mindig legfeljebb annyit mond, amennyi a valódi költség.

5. kérdés: Mit garantál az elfogadható heurisztika fakeresés esetén?

5. válasz: Ha a heurisztika elfogadható, akkor az A\* algoritmus fakeresés esetén garantáltan megtalálja az optimális megoldást.

6. kérdés: Mit garantál az elfogadható heurisztika gráfkeresés esetén?  
6. válasz: Ha a heurisztika elfogadható ÉS konzisztens (monoton), akkor az A\* gráfkeresés garantáltan megtalálja az optimális megoldást.

A képen szöveg, Betűtípus, képernyőkép, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.7. kérdés: Mit jelent a konzisztens heurisztika?  
7. válasz:

**Konzisztens heurisztika:** olyan heurisztika, amely   
minden élre teljesíti h(n) <= c(n,a,n’) + h(n). Ez biztosítja, hogy az f(n) értékek monoton nem csökkennek, és **garantálja az A\* gráfkeresés optimalitását**.

8. kérdés: Mit garantál a konzisztens heurisztika fakeresés esetén?

8. válasz: Konzisztens heurisztika fakeresés esetén nem ad többletgaranciát, mert az A\* már elfogadható h(n) mellett is optimális.

9. kérdés: Mit garantál a konzisztens heurisztika gráfkeresés esetén?

9. válasz: Konzisztens heurisztika esetén a A\* gráfkeresés optimális

10. kérdés: Miért fontos, ha egy heurisztika dominál egy másik heurisztikát?

10. válasz: Ha egy heurisztika dominál egy másikat, akkor az A\* kevesebb csomópontot bont ki, miközben az optimális megoldást megtartja. A domináló heurisztika hatékonyabb.

11. kérdés: Miért lehetnek érdekesek a relaxált problémák keresésnél?

11. válasz: **A relaxált problémák érdekesek, mert megoldásukból elfogadható heurisztikát készíthetünk, ami gyorsítja az A\* keresést és segít az optimális út becslésében.**

12. kérdés: A lokális algoritmusok miben hasonlítanak és miben térnek el a fa- és gráfkeresésektől?

12. válasz:  
**Hasonlóság:** mindkettő állapottérben dolgozik, állapotokat vizsgál, heurisztikát használhat.

**Különbség:**

* Lokális keresés **egy állapottal halad**, gyakran lokális optimumot talál, **memóriaigénye kicsi**.
* Fa-/gráfkeresés **teljes útvonalat vizsgál**, optimális megoldást találhat, **magas memóriaigényű**.

13. kérdés: Adja meg a hegymászó algoritmust

13. válasz:

**Kiinduló állapot kiválasztása**

* A probléma kezdőállapotát tekintjük az aktuális állapotnak.

**Ismétlés (loop)**

* Megnézzük az **aktuális állapot szomszédait** (lehetséges következő állapotok).
* Ha nincs több szomszéd, **visszatérünk az aktuális állapottal** (lokális maximum).

**Legjobb szomszéd kiválasztása**

* Kiválasztjuk a **legnagyobb értékű szomszédot** a heurisztikai függvény vagy érték alapján.

**Hasonlítás az aktuális állapottal**

* Ha a legjobb szomszéd értéke **nem jobb**, mint az aktuális állapoté → **lokális maximumot találtunk**, visszatérünk.

**Lépés a szomszédra**

* Ha a legjobb szomszéd jobb, akkor **lépünk rá**, és az most lesz az új aktuális állapot.
* Visszatérünk a 2. lépéshez.

14. kérdés: Milyen problémái vannak a hegymászó algoritmusnak?

14. válasz: Beragadhat az algoritmus a lapos területeken, nem garantálja a globális maximumot.

15. kérdés: Mi a lényege a genetikus algoritmusnak?

15. válasz: **A genetikus algoritmus egy evolúciós módszer, amely a legjobb megoldásokat kombinálja és mutálja, hogy iteratívan egyre jobb megoldásokat találjon.**

16. kérdés: Mi a lényege a szimulált hűtésnek?

16. válasz: A szimulált hűtés egy optimalizáló algoritmus, amely véletlenszerű lépéseket engedélyez rosszabb állapotokba a keresés elején, majd fokozatosan „lehűtve” egyre jobb megoldások felé konvergál.

17. kérdés: Mi a lényege a minimális konfliktusok algoritmusnak?

**17. válasz: A minimális konfliktusok algoritmus lokális keresésen alapul, konfliktusban lévő változókat módosít úgy, hogy a konfliktusok száma minimalizálódjon, amíg a feltételek teljesülnek.**



1. kérdés: Mi a különbség egy általános fakeresési valamint egy kényszer-kielégítési probléma között?

1. válasz:

* Általános keresési probléma
  + Az állapot egy fekete doboz
  + Az állapotot bármilyen adatstruktúra ábrázolhatja
  + Csak az állapotátmenetek, heurisztika és célállapot legyen implementálva
* Kényszerkielégítési probléma
  + Az állapotot Di tartományból származó Xi változókkal definiáljuk
  + A célteszt kényszerek halmaza, mely mindegyike a változók egy részhalmazát és megfelelő értékeket tartalmazzák

2. kérdés: Mivel lehet megadni egy kényszer-kielégítési feladatot?

2. válasz:

változók: WA, NT, Q, NSW, V, SA, T

tartományok: Di = {piros, zöld, kék}

kényszerek: szomszédos tartomány nem lehet ugyanolyan színű: WA ≠ NT

3. kérdés: Mit jelent a kemény és puha kényszer?

3. válasz: A **kemény**/valódi kényszer az, amelynek nem teljesülése meghiúsítja a megoldást. Míg a **puha** kényszer esetén egy megoldást még nem kell elvetnünk.

4. kérdés: Hogyan lehet megoldani egy kényszer-kielégítési problémát?

4. válasz:

* + Standard (inkrementális módszer)
  + Visszalépéses keresés

Az állapotokat a már értékkel rendelkező változók adják meg

* *kezdeti állapot*: egyik változónak sincs értéke
* *rákövetkező függvény*: rendeljünk egy olyan értéket valamely még értékkel nem rendelkező változóhoz, mely nem okoz konfliktust
  + ha nincs ilyen érték, akkor sikertelen a keresés
* *célteszt*: az értékelés teljes, minden változó kapott értéket

5. válasz: Milyen kényszerek lehetnek egy kényszer-kielégítési problémában?

* **Abszolút - kemény kényszerek**
  + Unáris kényszer egy változót tartalmaz: SA ≠ zöld
  + Bináris kényszer két változót tartalmaz: SA ≠ WA
  + **magasabb rendű kényszer legalább három változót tartalmaz**
  + pl. betűrejtvény: S+M+X3=O+10X4
* **Preferencia – puha kényszerek:** a piros jobb mint a zöld
  + Rendszerint minden értékeléshez költség kapcsolódik → kényszert tartalmazó optimalizációs probléma

6. feladat: Mit tartalmaz a kényszergráf, és mit jelölnek egyes alkotórészei?

6. válasz: **kényszergráf:** a csúcsok a változók, az élek a kényszereket jelölik

7. kérdés: Hogyan működik a visszalépéses keresés egy kényszer-kielégítési probléma esetén?

7. válasz: A **visszalépéses keresés (backtracking search)** egy **mélységi keresésen alapuló** módszer, amelyet gyakran használnak **kényszer-kielégítési problémák (CSP)** megoldására.

A folyamat lényege, hogy:

1. **Sorban értékeket rendelünk a változókhoz**, a kényszerek figyelembevételével.
2. Minden lépésben csak olyan értéket választunk, amely **nem sérti meg az eddig hozzárendelt változókra vonatkozó kényszereket**.
3. Ha egy változóhoz **már nem adható megengedett érték**, akkor **visszalépünk (backtrack)** az előző változóhoz, és másik értéket próbálunk.
4. A keresés addig folytatódik, amíg **minden változóhoz találunk megfelelő értéket** (megoldás), vagy **minden lehetőséget kimerítünk** (nincs megoldás).

8. kérdés: Milyen heurisztikákkal lehet gyorsítani a visszalépéses keresésen? Ismertesse az egyes módszerek lényegét!

8. válasz:

* Legkevesebb fennmaradó érték (MRV)
  + Válasszuk azt a változót, melynek a legkevesebb „megengedett” értékkel rendelkezik. (Legkorlátozottabb változó).
  + Gyorsan rátalál a hibákra, megnyesi a keresési fát
* Fokszám heurisztika
  + Holtverseny esetén dönt a legkevesebb fennmaradó érték heurisztika változóiról: Válasszuk azt a változót, amely legtöbbször szerepel a hozzárendeletlen változókra vonatkozó kényszerekben (azaz **legtöbb kényszert érint**).
* Legkevésbé korlátozó érték
  + Milyen sorrendben vizsgáljuk meg az értékeket? Válasszuk azt az értéket, mely a legkevesebb értéket zárja ki a hozzárendeletlen változóknál. válasszuk **azt az értéket**, amely **a legkevesebb tiltást** okozza a többiek tartományában.

9. kérdés Mi a különbség a előrenéző ellenőrzés (forward checking) és a kényszerek terjesztése (constaint propagation) között?

**Forward checking (előrenéző ellenőrzés)**

* Minden alkalommal, amikor **egy változónak értéket adunk**, a rendszer **csak az ezzel közvetlenül összefüggő (szomszédos)** változók domainjeit ellenőrzi.
* Azokból **eltávolítja azokat az értékeket**, amelyek **már nem kompatibilisek** az új hozzárendeléssel.
* Ha valamelyik szomszéd domainje **üressé válik**, akkor **visszalépés** történik. **Tehát csak “egy lépést néz előre”**, és **nem terjeszti tovább a változás hatását** más változókra. Egy lépés előre néz — csak a közvetlen hatást vizsgálja.

**Constraint propagation (kényszerek terjesztése)**

* Itt **nemcsak a közvetlen szomszédokat** vizsgáljuk, hanem **a kényszer következményeit is továbbterjesztjük** a hálóban.
* Például ha egy változó domainje szűkül, ez **további domain-szűkítéseket** válthat ki más, közvetetten kapcsolt változóknál is.
* A folyamat addig tart, amíg **több szűkítés már nem lehetséges** → az egész rendszer **konzisztens állapotba kerül** (pl. **ív-konzisztencia**, **AC-3 algoritmus**).  
  Tehát **a hatások láncreakció-szerűen terjednek**.

10. kérdés: Hogyan használható az él-konzisztencia? Milyen korlátai vannak?

10. válasz: Az **él-konzisztencia** a **kényszerek terjesztésének (constraint propagation)** legegyszerűbb formája.  
Azt mondjuk, hogy egy **X → Y él konzisztens**, ha **X minden lehetséges értékéhez** van legalább egy **Y-beli érték**, amely **kielégíti az X és Y közti kényszert**. (kétirányú)

Az él-konzisztencia biztosítása során (pl. **AC-3 algoritmussal**):

* minden élt (X,Y) megvizsgálunk,
* eltávolítjuk X domainjéből azokat az értékeket, amelyekhez Y-ban nincs kompatibilis érték,
* ha X domainje változik, a szomszédos éleket újra meg kell vizsgálni,
* az algoritmus addig fut, amíg minden él konzisztens nem lesz.

Az él-konzisztenciát **érdemes minden értékadás után lefuttatni**, mert **gyorsan felismeri az inkonzisztens hozzárendeléseket**, és **csökkenti a keresési tér méretét**.

**Milyen korlátai vannak?**

* Csak **bináris kényszereket** (kétváltozós összefüggéseket) tud kezelni.
* Az élkonzisztencia **nem garantálja**, hogy a probléma ténylegesen megoldható — csak azt, hogy nincs közvetlen ellentmondás a változók között.
* **Számításigényes** lehet (pl. az AC-3 algoritmus bonyolultsága O(n²·d³)), bár hatékonyabb változatai is léteznek (pl. AC-4).
* Nagyobb, összetett kényszerhálózatok esetén **nem veszi figyelembe a magasabb rendű (három vagy több változós) kényszereket**, így továbbra is szükség lehet visszalépéses keresésre.

Röviden: Az él-konzisztencia a kényszerek terjesztésének egy formája, mely biztosítja, hogy minden X változó minden értékéhez legyen olyan Y-beli érték, ami kielégíti az X–Y közti kényszert. Az AC-3 algoritmus segítségével fenntartható, és gyorsan kiszűri az inkonzisztenciákat. Korlátai: csak bináris kényszereket kezel, nem garantál megoldást, és nagy hálóknál számításigényes lehet.