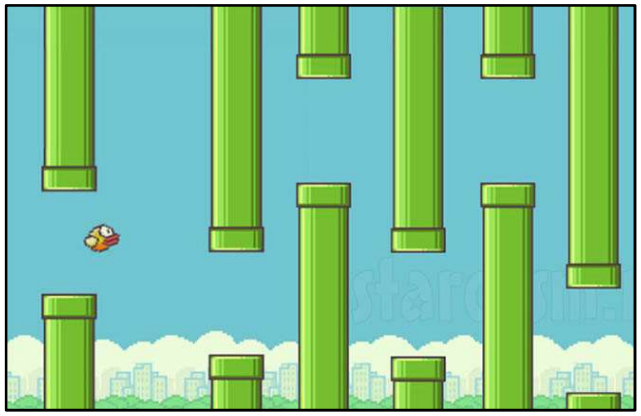
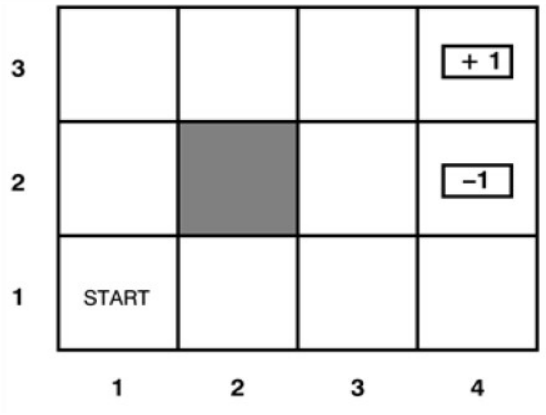
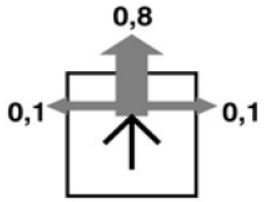
Megerősítéses tanulás

Q-tanulás







Ágensek

- Az új szemléletű, viselkedésalapú megközelítés szerint: a **mesterséges intelligencia** célja az, hogy a feladatmegoldást olyan ágensekkel végeztesse el, amelyek az intelligens viselkedés bizonyos vonásaival rendelkeznek.
- Egy **ágens** lehet bármely dolog, amely érzékelői segítségével észleli környezetét, majd megfelelő döntéseket hozva tevékenységével visszahat rá.

Erős definíció

Az ágens egy olyan rendszer, amely a következő tulajdonságokkal rendelkezik:

- **Beágyazottság**: a környezetbe ágyazottak, abból kiemelve nem tudnak funkcionálni (pl. egy ágens papírra nyomtatott programja nem ágens, mint ahogy egy vízbe dobott autóhegesztő robot sem az).
- **Reaktivitás**: az ágensek érzékelik környezetüket, valamint valós időben reagálnak, az abban bekövetkezett változásokra.
- **Autonómia**: az ágensek önállóan, emberek vagy mások direkt beavatkozása nélkül működnek és meghatározott mértékű kontrolljuk van a saját akcióik és belső állapotuk felett.
- **Helyzetfüggőség**: az ágensek helyzethez és szerephez kötötten ágensek csupán.

Erős definíció

- **Racionalitás**: az ágens a rendelkezésre álló számítási kapacitás és egyéb erőforrások mellett a lehető legjobb alternatívát választja.
- **Tanulás**: az ágens képes új ismereteket gyűjteni környezetéből és azt tárolni, felhasználni. A tanultak alapján viselkedését megváltoztathatja.
- **Alkalmazkodás**: képes tanulni a cselekedetei hatásából és a tanultakat felhasználva változtatni tud tervein, annak érdekében, hogy tevékenysége optimálisabb legyen.
- **Személyiség**: Számos alkalmazási területen szükség lehet mesterséges személyiség létrehozására. A személyiséggel bíró ágens identitással, speciális jegyekkel rendelkezik, melyek megkülönböztetik őt más ágensektől.

A tanuló ágens

- Egy **algoritmus tanul**, ha egy feladat megoldása során olyan változások következnek be a működésében, hogy később ugyanazt a feladatot vagy ahhoz hasonló más feladatotokat jobb eredménnyel, ill. hatékonysággal képes megoldani, mint korábban.

Mit jelent a megerősítéses tanulás

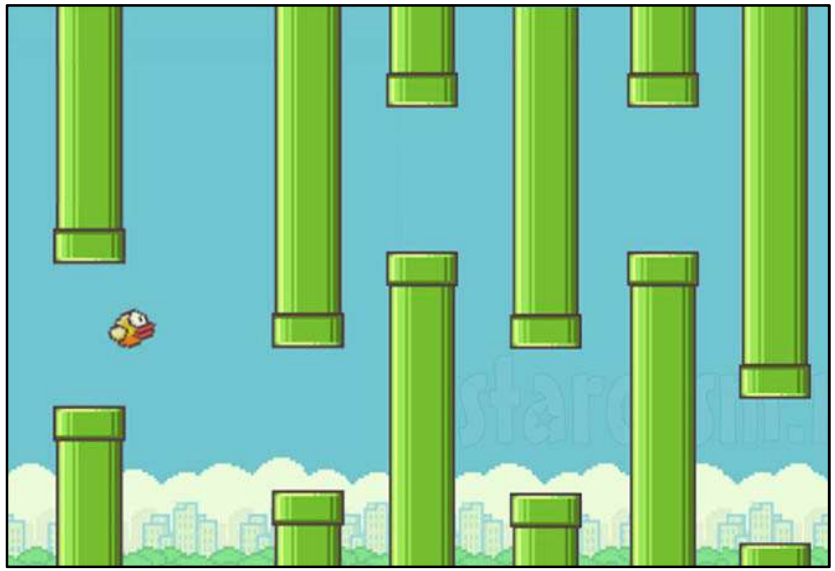
- Nincs tanító, aki felügyelné a rendszert.
- A rendszer nem kap példákat, és nem áll rendelkezésére hasznosságfüggvény sem.
- Modell felépítése: milyen lesz az állás, ha az ágens meglép/kiválaszt egy adott lépést, az ellenfél várhatóan mit fog válaszul lépni.
- Az ágensnek tudnia kell, hogy valami jó történt, amikor nyert, illetve valami rossz történt, amikor vesztett. Az ilyen típusú visszacsatolást nevezzük jutalomnak (reward) vagy megerősítésnek (reinforcement).

Fajtái

- **Passzív tanulás**
  az ágensnek rögzített stratégiája van, és az állapotok hasznosságának (vagy az állapot-cselekvés párok hasznosságának) megtanulása a feladat
- **Aktív tanulás**
  az ágensnek azt is meg kell tanulnia, hogy mit tegyen

Példa

- Az ágensnek minél tovább a levegőben kell maradnia úgy, hogy nem ér neki egyetlen egy csőnek sem. Az ágens folyamatosan halad előre, csak felfele tud ugrálni, és így kell a csövek között haladni.

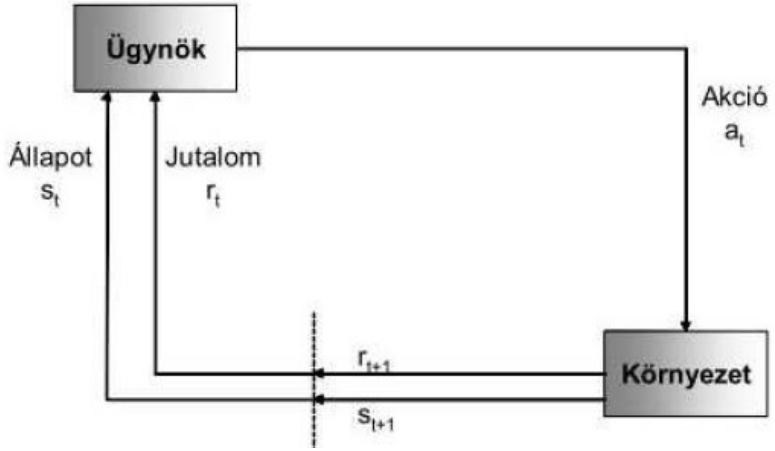


Ágens-környezet

- **Egyszerűen fogalmazva**: A megerősítéses tanulást és feladatát úgy fogalmazhatjuk meg, mint egy olyan módszert, amely kapcsolatok alapján, bizonyos célokra összpontosítva tanul.
- **Ágens-környezet modell**: Minden t időpillanatban az ágens megkapja a környezet $s_t \in S$ állapotleírását, ahol S a lehetséges állapotok (state) halmaza. Ennek alapján választ egy $a_t \in A(s_t)$ akciót, ahol $A(s_t)$ az s_t állapotban megengedett akciók halmaza.
- A következő lépésben a választott akció függvényeként kap egy $r_{t+1} \in R \subseteq \mathbf{R}$ jutalmat, és egy új s_{t+1} állapotba kerül.

Ágens-környezet

Minden egyes időpillanatban az ágens egy leképezést valósít meg az állapotleírások és az egyes akciók választási valószínűségei között.
A leképezést az **ágens politikájá**nak (policy) nevezzük, és Π_t -vel jelöljük, ahol $\Pi_t(s,a)\ s_t = s$ esetén $a_t = a$ választásának valószínűségét adja meg.
A megerősítéses tanulás különböző módszerei azt írják le, hogy az ágens hogyan változtatja a politikáját az idő előrehaladtával a tapasztalatai függvényében.
A cél az, hogy az ágens hosszú távon maximalizálja az összegyűjtött jutalmakat.



Q-tanulás jellemzői

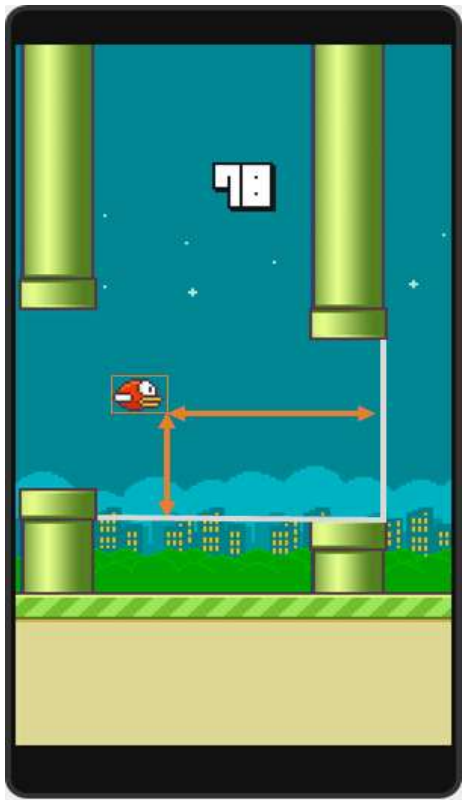
- A Q-tanulási algoritmus Markov döntési folyamat problémák esetén használható (a közvetlenül megelőző állapotot kell figyelembe venni).
- A Q-tanulás képes választani az azonnali jutalom és a jövőbeni jutalom között
- Az ágens minden lépésben meghatároz egy olyan a_t műveletet, amelynek végrehajtása után egy s_t állapotból egy s_{t+1} állapotba kerül a rendszer és az ágens kap egy $r(s_t,a_t)$ megerősítést (jutalmat)
- A tanulás célja, hogy megtaláljuk a műveletek egy olyan sorrendjét, amely maximalizálja a jövőbeli megerősítések összegét, így biztosítva az utat a kezdőponttól a végpontig
- Szabály:

Q(state, action) = R(state, action) + γ * Max[Q(next state, all actions)]

0 <= **γ** < 1 (tanulási ráta, gamma)

A feladat megközelítése

- **Az állapottér**
  - Függőleges távolság az alsó csőtől
  - Vízszintes távolság a következő csőtől
  - Élet (életben maradt vagy halott)
- **Az akciók**
  - Minden állapotból az alábbi lehetőségeket választhatjuk:
    - Klikkelés(Ugrás)
    - Semmit nem csinálunk
- **Jutalom**, jutalomfüggvény értéke:
  - +1, ha az ágens (madár) életben marad
  - -1000, ha az ágens (madár) meghal.



A tanulási folyamat ennél a játéknál

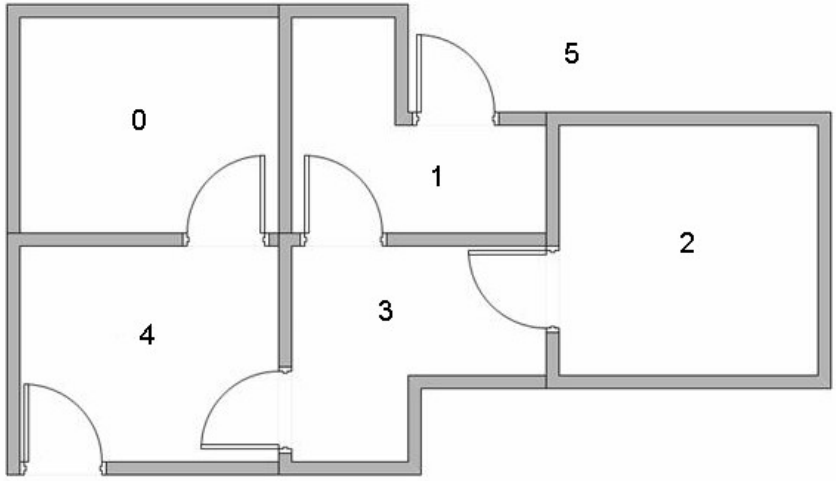
A Q tömb kezdetben 0 értékekkel van inicializálva és mindig a legjobb akciót fogja választani. Az akció maximalizálja a várt jutalmat.

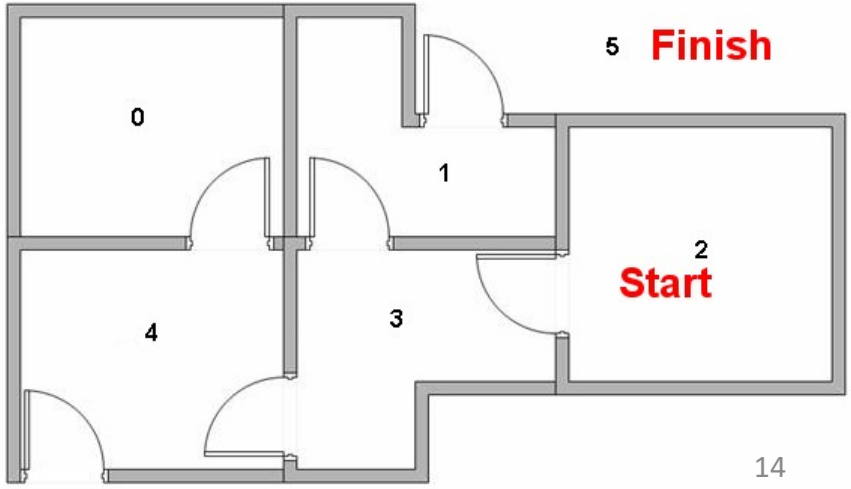
1. lépés: Megfigyeljük a madarat (ágenst), hogy melyik állapotban van, és a kiválasztott akció megpróbálja maximalizálni a várt jutalmat.
   Az ágens a következő állapotba kerül a művelet után: s'
2. lépés: Megfigyeljük az új állapotot, s' -t, ami jutalommal jár.
   +1, ha él még a madár, -1000, ha meghal.
3. lépés: Q függvény frissítése: $Q[s,a] \leftarrow Q[s,a] + \alpha\,(r + \gamma(\max\{Q[s,a]\})$
   Az α-t 0,7-nek érdemes választani, mert determinisztikus (tanulás mértéke).
   γ értéke 0,9 (jövőbeli jutalmak súlyozása).
5. lépés: Az aktuális állapotot s' re állítani, és elölről kell kezdeni a lépéseket.

Videó a fejlődésről: https://www.youtube.com/watch?time_continue=194&v=OJw4HTWvGdY

Alkalmazzuk az algoritmust egy másik feladaton

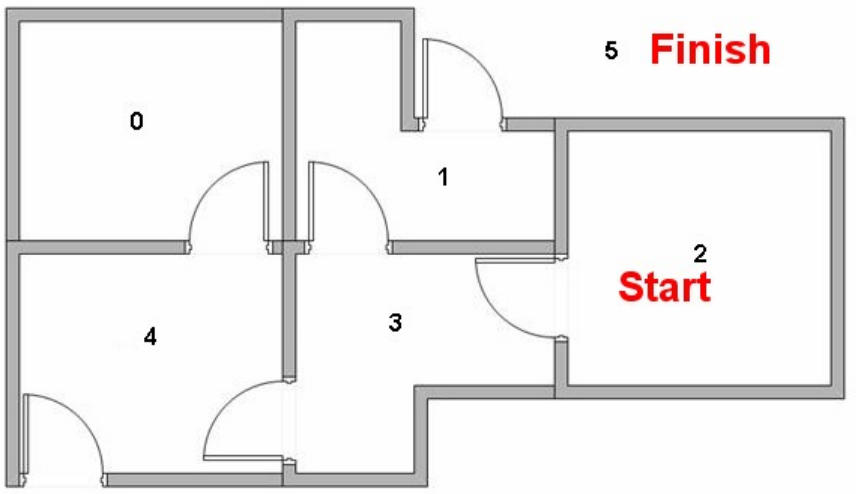
- Tegyük fel, hogy egy épületben 5 szoba van, amelyek ajtókkal vannak összekötve.
- Minden szobát 0-tól 4-ig megszámozunk.
- Az épület külső részét egy nagy egységnek tekintjük, ez lesz az 5.
- Az 1. és a 4. szobába be lehet jutni kívülről, azaz az 5. szobából.

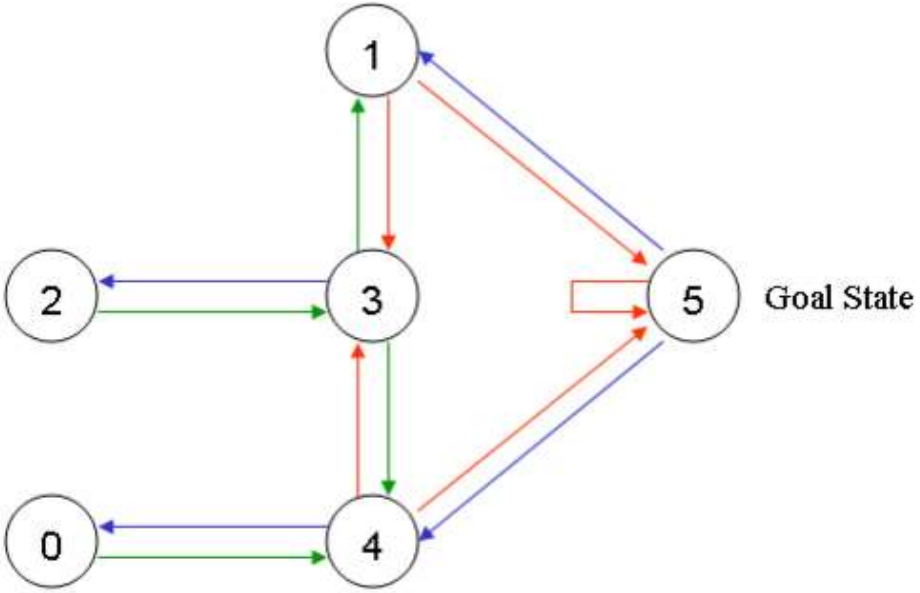




A feladat ábrázolása gráf segítségével

- A szobák sorszámai a csomópontok sorszámai.
- Az élek jelzik azt, hogy melyik szobából melyik másik szobába lehet eljutni közvetlenül.



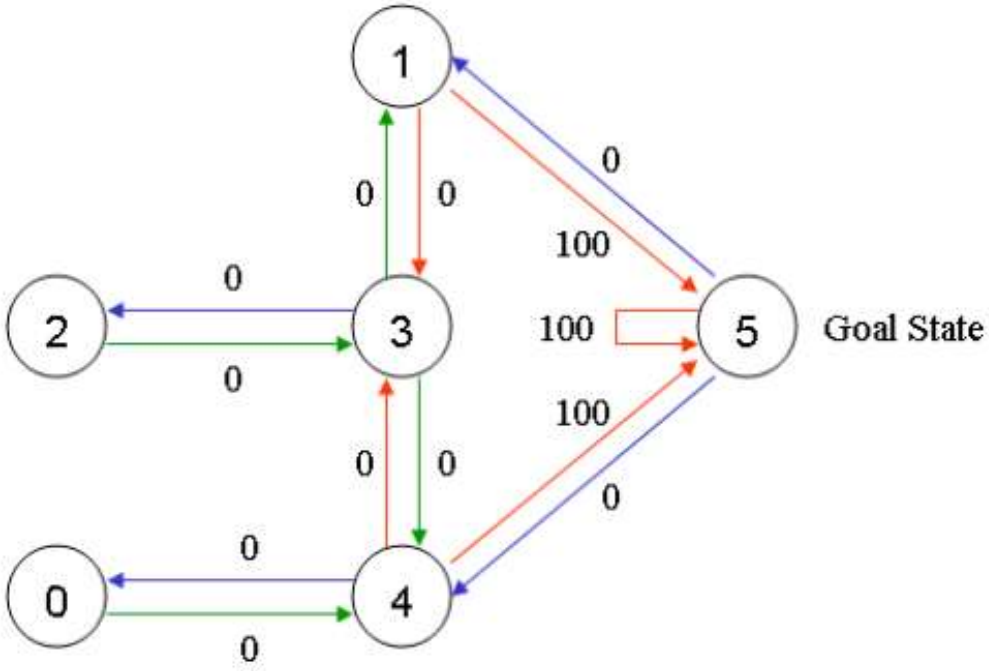


Cél: valamely szobából kijutni az épületen kívülre, azaz az 5. számú helyre

- Elhelyezünk egy ágenst valamelyik helyiségben.
- Az ágensnek innen kell kijutnia.
- Ahhoz, hogy az 5. szobát célként állítsuk be, minden egyes ajtóhoz **jutalom érték**et rendelünk (azaz a csomópontok közötti kapcsolatot ellátjuk egy értékkel).
- Azon ajtókhoz, amelyek közvetlenül a célhoz vezetnek, nagyobb jutalmat rendelünk, pl. 100-at.
- Más ajtókhoz, amelyek közvetlenül nem kapcsolódnak a célhelyiséghez, nulla jutalmat rendelünk.
- Mivel az ajtók kétirányúak (0 vezet 4-hez, és 4 vezet vissza 0-hoz), minden csomóponthoz (szobához) két nyilat rendelünk.

Élek súlyozása

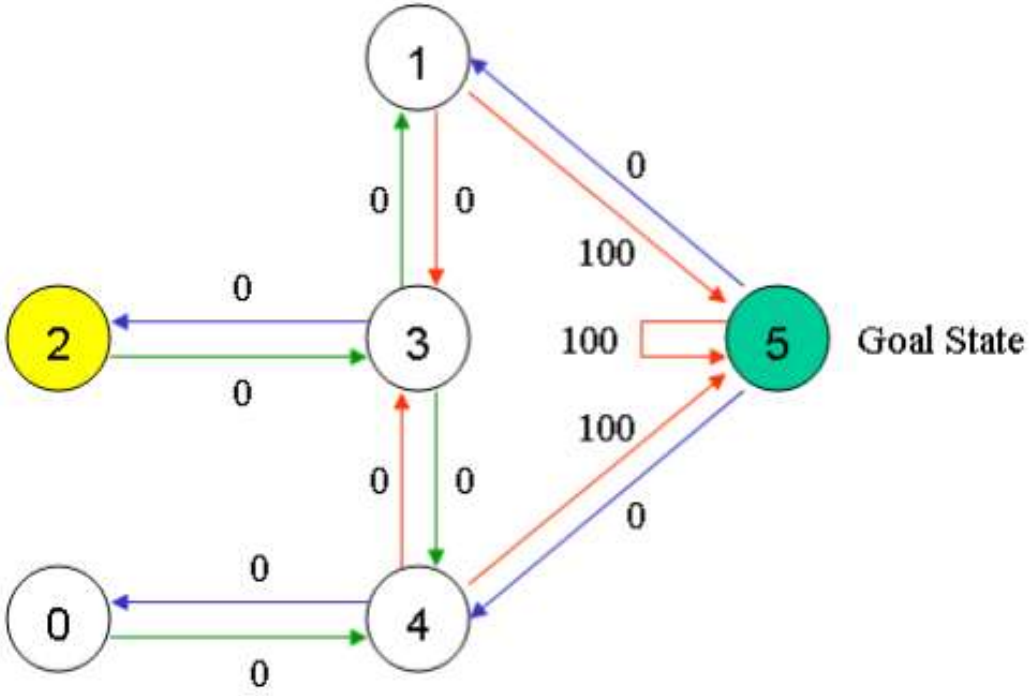
- Azokhoz az élekhez rendelünk 100-at, amelyek a cél állapotba visznek.



- A Q-tanulás során a legnagyobb jutalom megszerzésével szeretnénk eljutni a célba.

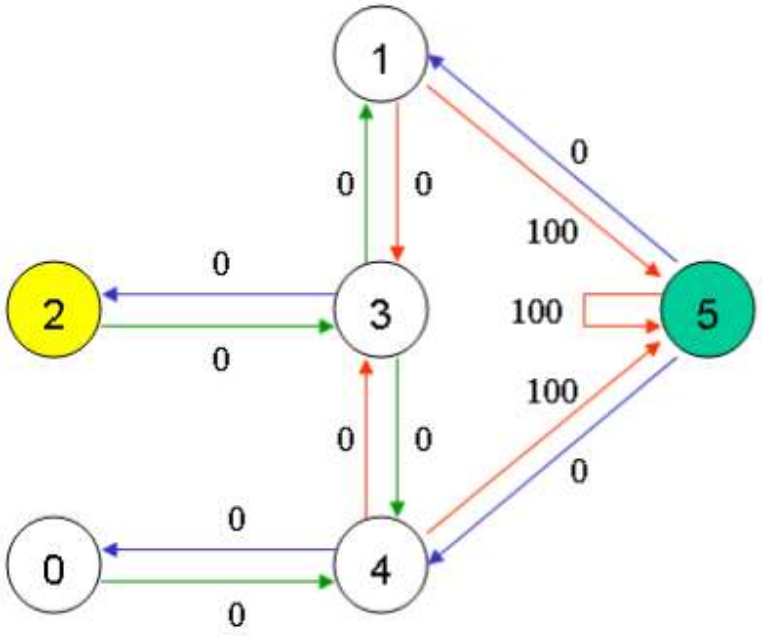
Állapot és akció (művelet)

- Minden szobát, beleértve a külteret, is állapotnak nevezzük.
- Az ágens mozgása az egyik szobából a másikba egy akció.
- A diagramon egy állapot egy csomópont, az akciókat a nyilak reprezentálják.



- Tegyük fel, hogy az ágens a 2. állapotban van. A 2. állapotból az ágens a 3. állapotba léphet, mert a 2. állapot csatlakozik a 3. állapothoz.
- A 2. állapotból azonban az ágens nem léphet közvetlenül az 1. állapotba, mert nincs közvetlenül ajtóval összekötve az 1. szoba és a 2. (nincsenek nyilak a gráfban).
- A 3. állapotból az 1. vagy a 4. állapotba vagy a 2. állapotba mehet az ágens.
- Ha az ágens a 4. állapotban van, akkor a három lehetséges művelet a 0., 5. vagy 3. állapotba jutás.
- Ha az ágens az 1. állapotban van, akkor az 5. vagy 3. állapotba mehet.
- A 0. állapotból csak vissza a 4. állapotba juthatunk.

- Az állapotdiagram alapján a jutalom értékeket a jutalomtáblázatba, "R mátrix" rögzíthetjük.
- A táblázatban szereplő -1 értékek azt fejezik ki, hogy nincs kapcsolat a csomópontok között. Például a 0. állapot nem léphet az 1. állapotba.



$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

- Egy **Q mátrix**ot adunk hozzá az ágenshez, amely megmutatja azt, hogy **az ágens hogyan tanul a tapasztalatok révén**.
- A Q mátrix sorai az ágens aktuális állapotát, az oszlopok a következő állapothoz vezető lehetséges műveleteket (a csomópontok közötti kapcsolatokat) jelentik.
- Az ágens először nem tud semmit, azaz a Q mátrixot nullára inicializáljuk:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- A Q-tanulás általános szabálya a következő:
  Q (állapot, akció) = R (állapot, akció) + γ * max [Q (következő állapot, minden művelet)]

- Virtuális ágensünk tapasztalatokon keresztül tanul, tanár nélkül (ezt felügyelet nélküli tanulásnak nevezzük).
- Az ágens a kezdő állapottól kutat a továbbhaladás érdekében, amíg el nem éri a célt. Minden lehetséges feltárást epizódnak nevezünk.

A Q-tanulás algoritmusának általános működése:

1. Állítsuk be a **γ** paramétert és a környezeti jutalmakat az **R** mátrixban.
2. Inicializáljuk a **Q** mátrixot nullára.
3. Minden epizód esetében:

   Válasszunk egy véletlenszerű kezdeti állapotot.

   DO WHILE amíg a célállapotot el nem értük

   a. Válasszunk egyet az aktuális állapothoz tartozó összes lehetséges akció/művelet közül.
   b. Ezzel a lehetséges művelettel fontoljuk meg a következő állapotba lépést.
   c. Állítsuk elő maximális **Q** értéket ehhez az állapothoz az összes lehetséges művelet megvizsgálásával.
   d. Számítsuk ki a **Q** mátrix értékeit: **Q (állapot, akció) = R (állapot, akció) + γ * Max [Q (következő állapot, minden művelet)]**
   e. Állítsuk be a következő állapotot aktuális állapotként.

   END DO

- Minden epizód egyenértékű egy edzéssel. Minden képzésen az ágens feltárja a környezetet (R mátrix), megkapja a jutalmat (ha van ilyen), amíg el nem éri a célállapotot.
- Az edzés célja a Q mátrix által képviselt ágens gondolkodásának a fejlesztése. Több edzés eredményesebben optimalizálja a Q mátrixot. Felhasználásával az ágens megtalálhatja az optimális utat a célállapotba.

A Q mátrix használatához az ágens egyszerűen nyomon követi az állapotok sorozatát, a kezdeti állapottól a célállapotig.
Az algoritmus megtalálja azokat a műveleteket, amelyek a jelenlegi állapotban a Q mátrixban rögzített legnagyobb jutalomértékekkel rendelkeznek.
Algoritmus a Q mátrix felhasználására:

1. Állítsa be az aktuális állapotot = kezdeti állapot.
2. Az aktuális állapotból keresse meg a legmagasabb Q értékű műveletet.
3. Állítsa be az aktuális állapotot = következő állapot.
4. Ismételje meg a 2. és 3. lépést, amíg az aktuális állapot = célállapot.

A fenti algoritmus visszaadja az állapotok sorrendjét a kezdeti állapotból a célállapotba.

Példa:

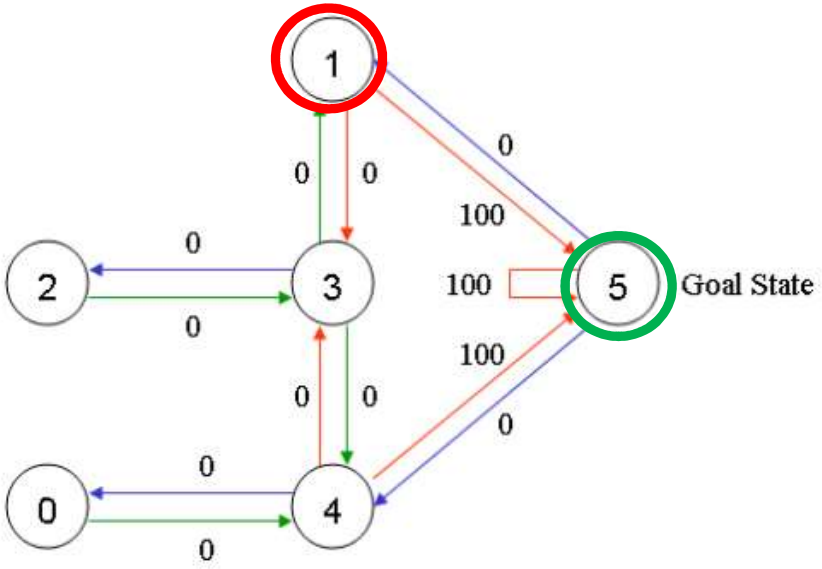
- A Q-tanulás algoritmus működésének megértéséhez lépésről lépésre áttekintünk néhány epizódot.
  Először beállítjuk, hogy a γ = 0,8 (tanulási paraméter), és a kezdeti állapotot az 1. helyiségre állítjuk (innen indulunk).

  Inicializálja a Q mátrixot nulla értékekkel:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

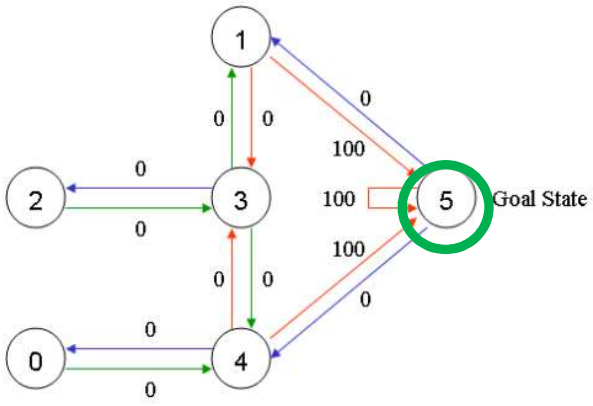
- Nézzük meg az R mátrix második sorát (1. állapot).
- Az aktuális 1. állapotban kétféle művelet közül választhatunk: lépjen a 3. állapotba, vagy lépjen az 5. állapotba.
- Véletlenszerű kiválasztással az 5-ösre lépünk.

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$



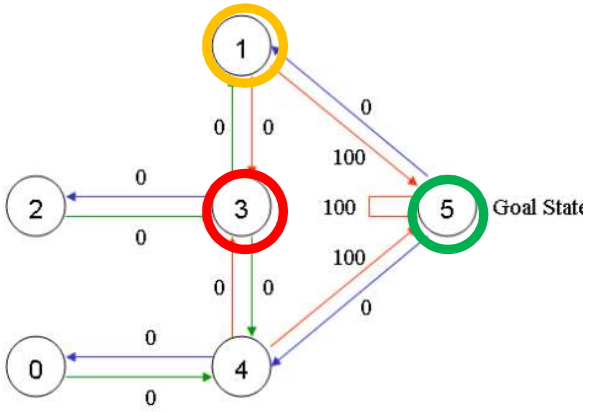
- Most képzeljük el, mi történne, ha az ágens 5. állapotba lépne. Nézzük meg az R jutalmazási mátrix 2. sorát.
- 3 lehetséges művelet lehetséges az 5-ből továbblépni: az 1., 4. vagy 5. állapotba léphetnénk.
- Q(state, action) = R(state, action) + γ * Max[Q(next state, all actions)]
- Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100
- Mivel a Q mátrixban minden érték 0, Q (5, 1), Q (5, 4), Q (5, 5) is nulla. A Q (1, 5) kiszámításának eredménye 100, az R (1,5) jutalom érték miatt.

A következő, 5. állapot az aktuális állapot lesz. Mivel az 5 célállapot, egy epizódot befejeztük. Ágensünk agyában most egy frissített Q mátrix található:



$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} & & & \text{Action} & & \end{array} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array} \qquad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- A következő epizódhoz egy véletlenszerűen kiválasztott kezdeti állapotból indulunk. Ezúttal a 3. állapotot használjuk kezdeti állapotként.
- Nézzük meg az R mátrix negyedik sorát.
- 3 lehetséges művelet jöhet szóba: az 1., 2. vagy 4. állapotba léphetünk. Véletlenszerű kiválasztással választjuk, hogy az 1. állapotba kerülünk.
- Ezután kiszámoljuk a Q értéket:
  Q(state, action) = R(state, action) + γ * Max[Q(next state, all actions)]
- Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 3), Q(1, 5)] = 0 + 0.8 * Max(0, 100) = 80
  Mivel Q(1, 3) = 0 and Q(1, 5) = 100.
- Az alábbi Q mátrixot kapjuk:



$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

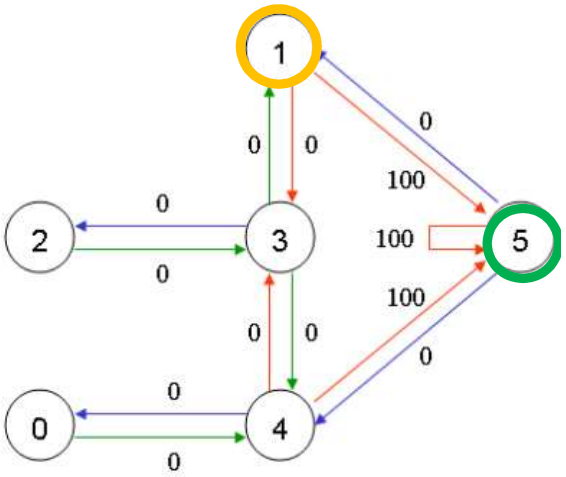
Action

State

Q

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- A következő, az 1-es állapot az aktuális állapot lesz. Megismételjük a Q tanulási algoritmus belső hurkot, mert az 1. állapot nem a célállapot.
- Ha az új ciklust az 1. állapottal indítjuk 2 lehetséges művelet van: lépjünk a 3. állapotba vagy az 5. állapotba. Véletlen választással legyen az 5. állapot.
- Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100
- A Q(1,5) értéke nem fog megváltozni, marad 100.
- Mivel az 5 célállapot, egy epizódot befejeztük. Ágensünk agyában ismét egy frissített Q mátrix található:



$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} \text{Action} & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Q

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Ha az ágens további epizódokon keresztül tanul, akkor elérheti a Q mátrix konvergencia értékeit, pl.:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

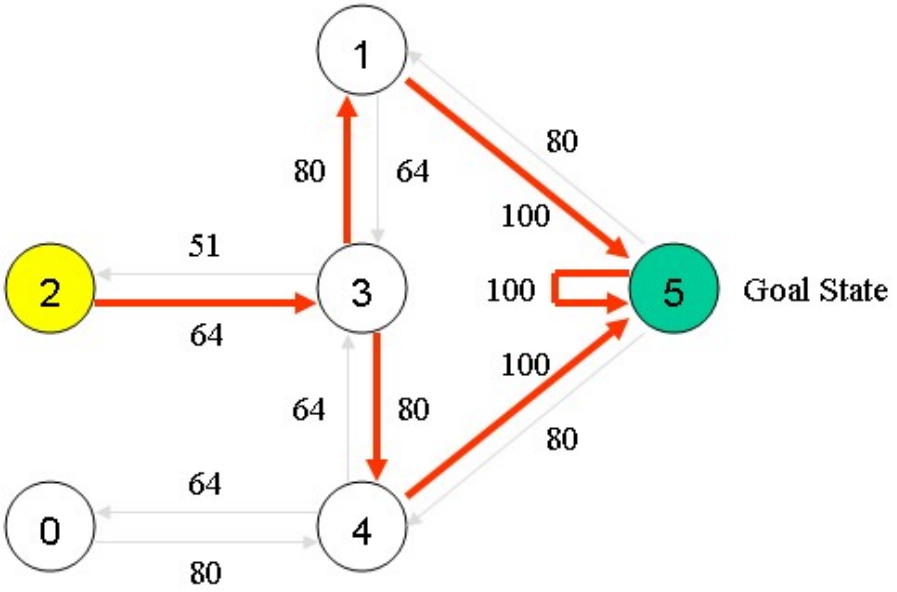
A Q mátrixot ezután normalizálhatjuk (pl. százalék értékre konvertálhatjuk), az összes nem nulla bejegyzést elosztjuk (ebben az esetben ez 5).

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array} \qquad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

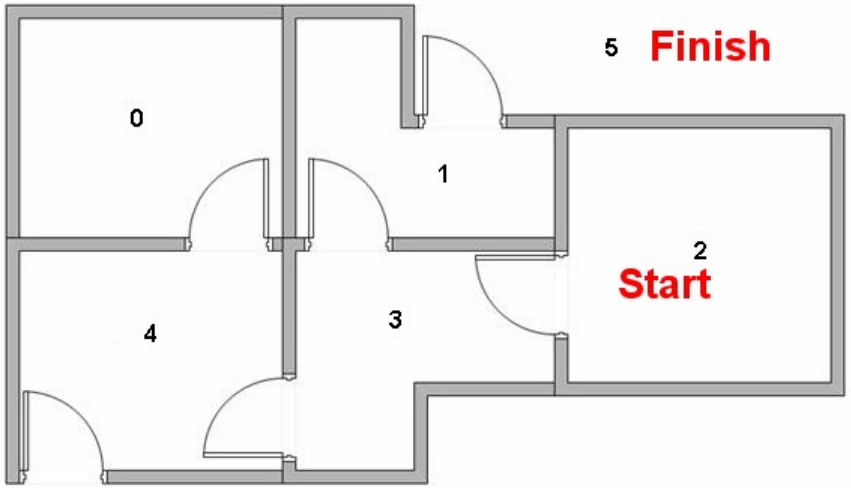
- Amint a Q mátrix elég közel áll a konvergencia állapotához, tudjuk, hogy az ágens megtanulta a célállapothoz vezető optimális útvonalat.
- Az állapotok legjobb szekvenciáinak nyomon követése, az egyes állapotokban a legmagasabb értékkel rendelkező élek követése.

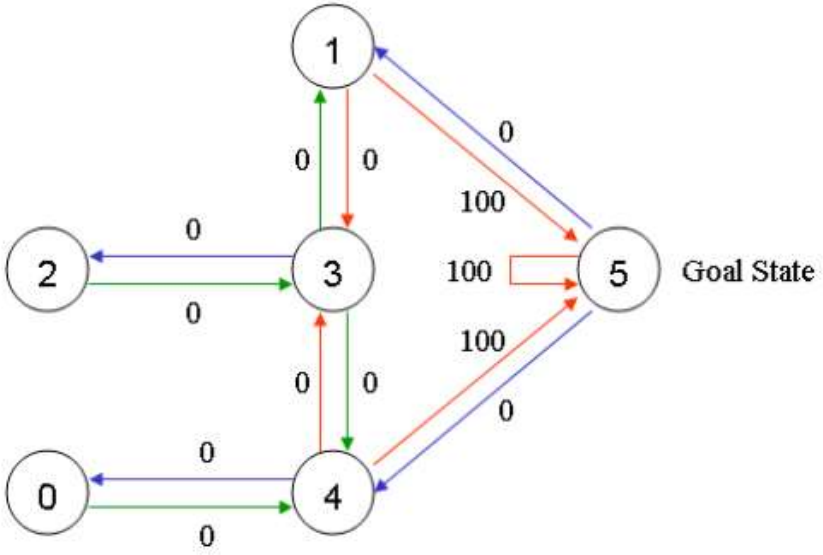
- Példánkban most két ilyen útvonal került megtalálásra a 2. állapotból (2. szobából) indulva a Q mátrixot felhasználva:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$



Hogyan változik a Q mátrix, ha a 0. szobából szeretnénk kijutni a kültérbe? A gamma=0,8.





$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Action

State

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Megerősítéses tanulás az ütemezésben – folyamatütemezés, probléma felvetés

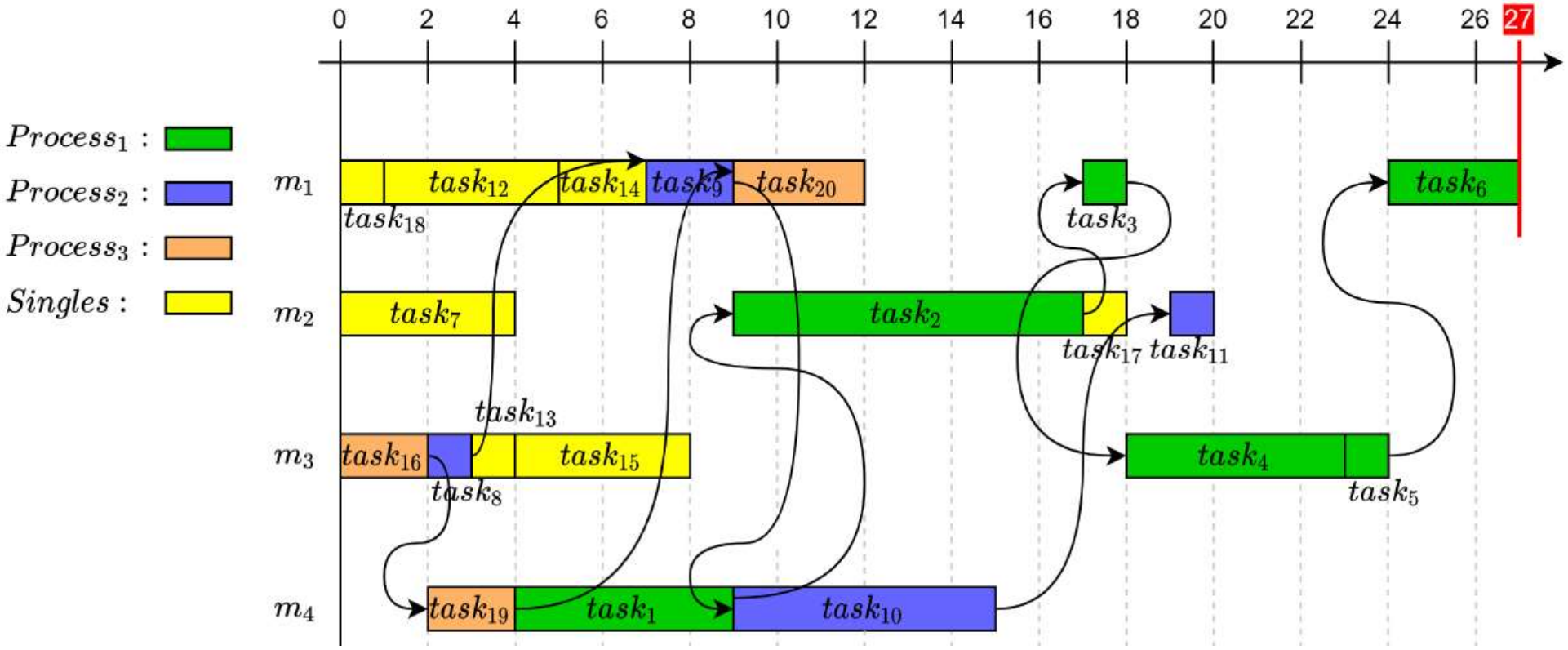
1. Hivatali ügyintézés (pl. adóhivatal, bevándorlási hivatal, önkormányzati hivatal stb.), dokumentumok feldolgozása:
   - Az első típusú dokumentumok feldolgozása egyszerű, azt bármilyen hivatali dolgozó képes elvégezni; a második típusú dokumentum feldolgozását már csak a megfelelő képzettséggel rendelkező dolgozók tudják elvégezni.
   - A cél a dokumentumok feldolgozási idejének a minimalizálása.
2. Építkezési munkálatok (pl. alapozás, falak felhúzása, mérnöki feladatok stb.):
   - Ezen részfeladatok között megelőzési relációk vannak. Például a falak felhúzása előtt az alapnak kell elkészülnie. Az erőforrások az építkezésen dolgozó emberek.
   - A cél pedig az, hogy a dolgozókat vagy azoknak a csoportjait úgy osszuk szét a feladatok között, hogy egyrészt azt az adott dolgozó vagy dolgozók képesek legyenek elvégezni, másrészt pedig a munka ideje minimális legyen.

Független gépek ütemezése megelőzési relációk figyelembevételével

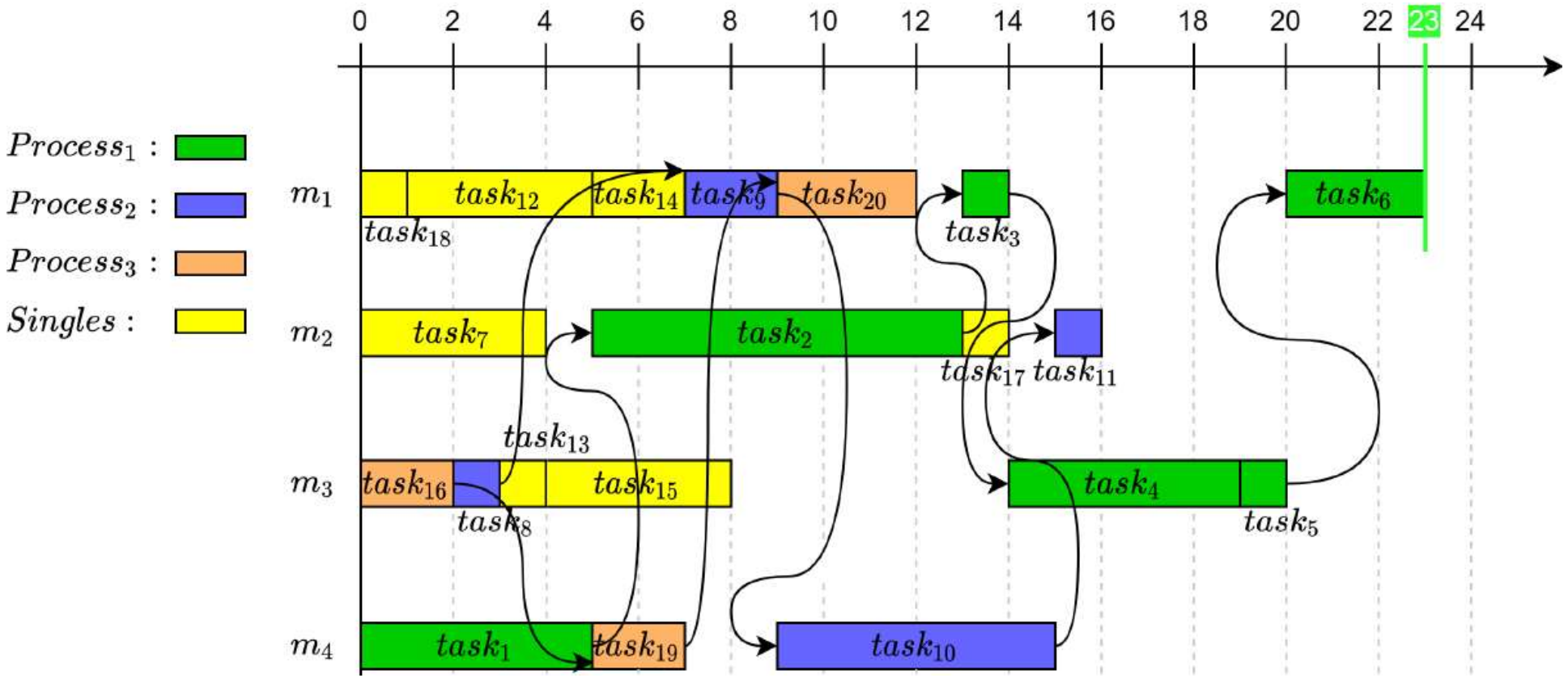
- $\mathcal{T} = \{task_1, task_2, task_3, task_4, \ldots, task_{20}\}$ az összes tevékenység halmaza
- $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$ az összes erőforrás halmaza
- Egy erőforrás egyszerre csak egy tevékenységet hajthat végre és a végrehajtás nem megszakítható.
- Az egyes tevékenységek végrehajtási ideje az erőforrásoktól függ.
- Az ütemezés célja a teljes átfutási idő minimalizálása.

- Diszjunk utak:
  - $task_1 \rightarrow task_2 \rightarrow task_3 \rightarrow task_4 \rightarrow task_5 \rightarrow task_6$
  - $task_8 \rightarrow task_9 \rightarrow task_{10} \rightarrow task_{11}$
  - $task_{16} \rightarrow task_{19} \rightarrow task_{20}$
- Izolált pontok:
  - $task_7, task_{12}, task_{13}, task_{14}, task_{15}, task_{17}, task_{18}$

Összesen 20 tevékenység van, amelyek végrehajtására 4 gép áll rendelkezésre - egy lehetséges ütemezés



Összesen 20 tevékenység van, amelyek végrehajtására 4 gép áll rendelkezésre - optimális ütemezés



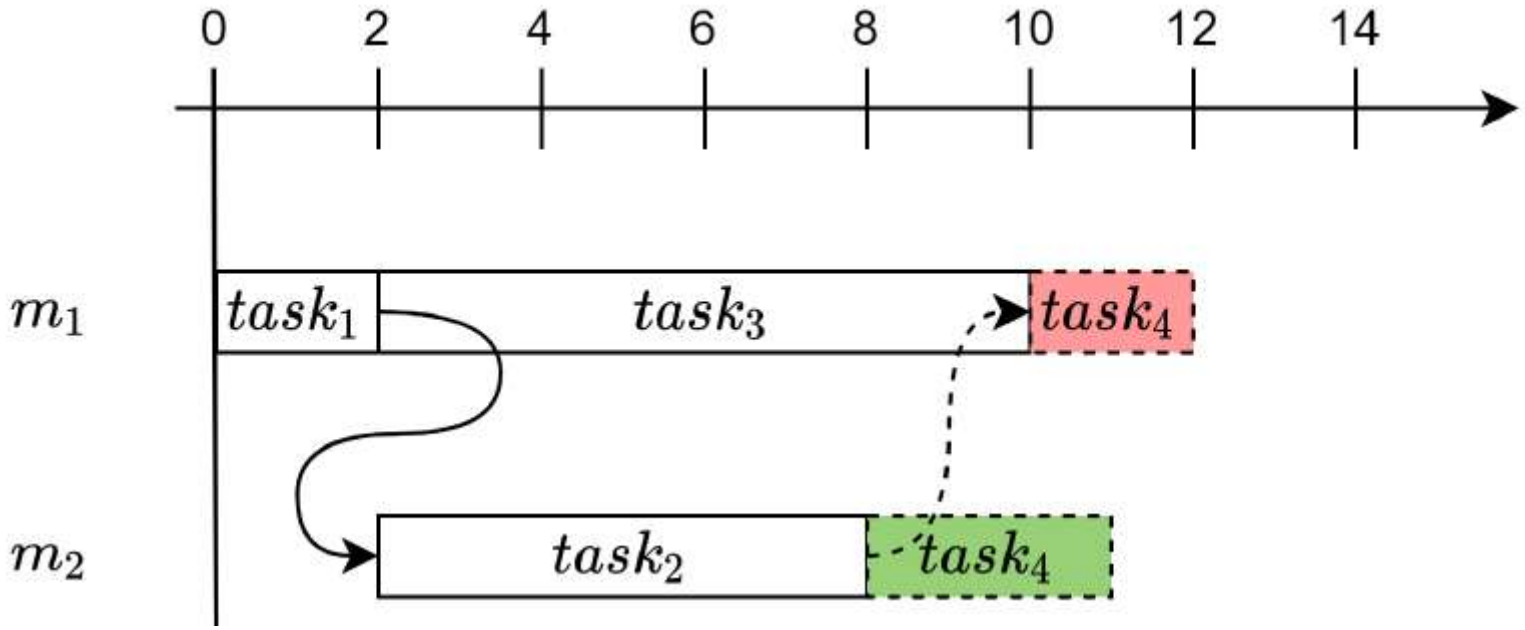
Javaslat a megoldásra

Két komponens használata:

- Az egyik egy mohó algoritmus, amely az ütemezést végzi a tevékenységeke egy megadott sorrendjében.
- A másik a Q-tanulással támogatott komponens, amelynek feladata a permutációnak az előállítása.
- A cél az, hogy olyan permutációt találjon az algoritmus, amely szerint mohón ütemezve az átfutási idő minimális lesz.

Mohó algoritmus

- A soron következő tevékenységhez mindig azt a gépet rendeli hozzá, amellyel az addig elért átfutási idő a legkisebb mértékben növekszik, a megelőzési relációkat is figyelembe véve.



- Önmagában nem elegendő csak a végrehajtási idők alapján való mohó ütemezés!

A megoldó algoritmus lehetséges lépései

1. A tevékenységek egy permutációjának meghatározása a Q-tanulás segítségével.
2. A kapott sorrend alapján a mohó ütemezés elvégzése.
3. Az átfutási idő kiszámítása az ütemezés alapján.
4. A jutalom értékének kiszámítása, amely az aktuális átfutási idő és az eddigi legjobb átfutási idő alapján történik.
5. A $Q_{t+1}(S_t, A_t) = (1 - \alpha_t)Q_t(S_t, A_t) + \alpha_t(R_t + \gamma_t \max_a Q_t(S_{t+1}, a))$ szabály alkalmazása a Q-értékek frissítéséhez.