10. Tétel: A mesterséges intelligencia alapjai

Függvények leírása és számítógépes ábrázolása.

Függvény segítségével jeleníthető meg egy ábra: minden x-hez tartozik egy y érték. A grafikon elméleti, ezt csak közelíthetjük, mivel a gyakorlatban behelyettesítünk, és az adott értékeket ábrázoljuk, majd ezeket összekötjük. A gyakorlati függvény tehát szakaszokból áll. A behelyettesített értékek számától, sűrűségétől függ, hogy hány ponton lesz törött a vonal. Számítógép annyi értéket helyettesít, hogy legalább 1 px-t ugorjon (ami függ a képernyő felbontásától is), így görbének látszik, viszont mindig töröttvonal marad.

1. **Polinomiális függvények**

   $\mathbf{a_n \times x^n + a_{n-1} \times x^{n-1} + ... + a_1 \times x^1 + a_0 \times x^0}$

   Polinom: x polinomját valamilyen hatványon megszorozzuk (n:fokszám, a:együtthatók, x:változó), konstans függvény: 0-adfokú polinom.

   f(x)=ax+b 1.fokú polinom, ahol **a:**meredekség, **b:**hol metszi **f(x)**-et (vagy **y** tengelyt) (mivel a értéke nem tud elég nagy lenni, ezért függőlegest nem tudunk ábrázolni.
   Elképzeléshez kitalálni függvényt:
   -ha 2 pont van megadva, éppen egyértelmű
   - 1 ponttal aluldeterminált
   - több, mint 2 pont esetén ha 1 egyenesre esnek, könnyű helyzet, ha nem, akkor egyik ponton sem megy át, hanem minden pontot közelít a legkisebb eltéréssel.

   $f(x)=ax^2+bx+c$ 2.fokú polinom
   Elképzeléshez függvényt találni:
   - próbálkozás különböző a, b és c értékekkel
   - kideríteni hol a csúcspont
   - megadott minimum 3 ponttal egyenletrendszer megoldása

   Hullám ábrázolása esetén is célszerűbb polinomiális függvény használata szögfüggvény helyett, mert az x-hez tartozó f(x) értékek keresése szögfüggvény esetén időigényes, lassú, és apró változtatástól is összeomlik.

   $f(x)=ax^3+bx^2+cx+d$ 3.fokú polinom
   Megtalálása próbálkozással vagy minimum 4 pont megadásával (4 ismeretlen miatt).
   Ábrázolható 2 parabolával is, de úgy törés lesz a csatlakozási ponton.

2. **Implicit függvények**

   Ha nincs egyértelmű hozzárendelés, tehát x-hez több y érték tartozik, pl egy kör, vagy egy függőleges esetén
   Eddig: $x \rightarrow f(x)$, ahol $R \rightarrow R$
   Implicit esetben: $x,y \rightarrow F(x,y)$, ahol $R \times R \rightarrow R$
   $F(x,y) = 0$
   Minden eddigi függvény átalakítható ilyenné:
   explicit: $y= 3x^2+5x+1$ $y=f(x)$
   implicit: $0=3x^2+5x+1-y$ $0=f(x)-y$
   tehát pl: $x^2+y^2=9$
   $x^2+y^2-9=0$
   $F(x,y)=0$

3. Vektor alapján függvény ábrázolás

Ezek a paraméteres vektorfüggvények, vagy paraméterekkel megadott vektor görbék. Mozgó helyvektor végpontjai rajzolják a görbét, tehát idő függvényében vektort írunk le. Értelmezési tartomány: idő.

$r(t)$: $[a,b] \rightarrow V^2$, ahol V^2 a szabad vektorok halmaza, a t paraméter valós érték, $[a,b]$ intervallum az ábrázolás idejének kezdő és végpontja közötti intervallum, a függvény értéke pedig a vektor.

2 függvényként is lehet kezelni:
Van (x,y) koordinátám
x(t) és y(t) is függvénye a t-nek
$x(t)$: $[a,b] \rightarrow R$, $y(t)$: $[a,b] \rightarrow R$
A vektorfüggvény tehát koordináta függvények együttese. Minden t (azaz $[a,b]$ intervallumon értelmezett idő) értékre x és y koordinátákat ad.

Ez a módszer egyesíti az eddigiek előnyeit:
- könnyű kirajzolás (pl y=f(x) esetén)
- a görbe maga alá is görbülhet, bármilyen alakú lehet (F(x,y)=0 esetén)

Görbék leírása és számítógépes ábrázolása.

Implicit típusú görbék

Behelyettesítés esetén bonyolult egyenletet kapunk (Pl $4x^5-5x^3y^7+\sin(x)+\cos(y^2)-7=0$)
Ábrázolás módja: (0,0) behelyettesítése, pontonkénti (pixelenkénti) kiértékelés: ha az eredmény rajta van a vonalon, akkor 0-val egyenlő, egyébként nincs rajta.
Minden pixelt be kell helyettesíteni. Lassú kiértékelés, és nem egyértelmű, akár a szoftver is elbizonytalanodhat, ezért ritkán használjuk computer grafikában.
Tulajdonságai:
-az (x_0,y_0) koordinátájú pont akkor és csakis akkor illeszkedik a görbére, ha $F(x_0,y_0)=0$
- ha $F(x_0,y_0)>0$, akkor az görbe fölött helyezkedik el (fel és jobbra)
- ha $F(x_0,y_0)<0$, akkor az görbe alatt helyezkedik el (le és balra)
Értékek behelyettesítése a távolságot is mutatja (minél nagyobb az érték, annál távolabb van a pont)

Polinommal megadott görbe

elsőfokú polinom: $3x-4y-7 = 0$, egyenes → elsőrendű görbe
másodfokú polinom: $x^2+y^2-9=0$, kör→ másodrendű görbe
$x^2-y=0$, parabola → másodrendű görbe
n-edfokú polinom: $x^n-3y^n+...=0$ → n-edrendű görbe
Csak polinommal megadott lehet valahanyadrendű görbe vagy algebrai görbe (mivel a függvény is algebrai, nem analitikus. Pl. sin(x) analitikus lenne)
Egy n-edrendű és egy m-edrendű görbének legfeljebb n×m darab látható metszéspontja lehet (ezen mindkét görbe átmegy)

Vektorral megadott görbe:

Mozgó helyvektor végpontjai rajzolják a görbét, tehát idő függvényében vektort írunk le. (Továbbiak: Függvények, 3.pont) Ezek a legáltalánosabban használt görberajzoló függvények.

Térbeli görbe rajzolására alkalmatlanok az implicit és az explicit függvények.
Csak vektorfüggvénnyel rajzolhatóak.
r(t): [a,b]→V^3, ahol V^3 a térbeli vektorok halmaza
x(t): [a,b]→ R, y(t): [a,b]→ R, z(t): [a,b]→ R

Görbe alapján írjunk képletet
Egyenes esetén: r(t): $x(t)=a_1t+a_0$
$y(t)=b_1t+b_0$
Keressük a_0, a_1, b_0, b_1 számokat
Legyen t=0 pillanat az (1,6) pontnál, ez r(0).
Legyen t=1 pillanat az (5,4) pontnál, ez r(1).
t=0 => r(0)
x=1 $x(0)=a_1\times0+a_0=1$ => $a_0=1$
y=6 $y(0)=b_1\times0+b_0=6$ => $b_0=6$
t=1 => r(1)
x=1 $x(1)=a_1\times1+a_0=5=a_1+1=5$ => $a_1=4$
y=6 $y(1)=b_1\times1+b_0=4= b_1+6=4$ => $b_1=-2$
Az egyenes leírása tehát: r(t): $x(t)=4t+1$
$y(t)=-2t+6$
Algoritmikusan: $x(t)=(x_1-x_0)\times t+x_0$
$y(t)=(y_1-y_0)\times t+y_0$
Ahol x0, x1, y0, y1 az egér által meghatározott pontok.
Három pontra illesztett görbe: r(t): $x(t)=a_2t^2+a_1t+a_0$
$y(t)=b_2t^2+b_1t+b_0$
Keressük a_0, a_1,a_2, b_0, b_1, b_2 számokat
Legyen t=0 pillanat az (1,6) pontnál, ez r(0).
Legyen t=0.7 pillanat az (5,2) pontnál, ez r(0.7).
Legyen t=1 pillanat az (7,4) pontnál, ez r(1).
t=0 => r(0)
$x(0)=a_20^2+a_10+a_0=1$ => $a_0=1$
$y(0)=b_20^2+b_10+b_0=6$ => $b_0=6$
t=0.7 => r(0.7)
$x(0.7)=a_21^2+a_11+a_0=5= a_20.7^2+a_10.7+1=5$
$y(0.7)=b_21^2+b_11+b_0=2= b_20.7^2+b_10.7+6=2$ t=1 => r(1)
$x(1)= a_21^2+a_11+a_0=7$
$y(1)= b_21^2+b_11+b_0=4$
t=0.7 és t=1 alapján egyenetrendszer, ami a maradék ismeretleneket megadja

Pontok alapján történő ábrázolás
n darab pontra fektetek görbét, tehát megadok n db (x_i, y_i) koordinátájú pontot. Vektoros ábrázoláshoz megadom a t_i értékeit úgy, hogy a t_i görbepont éppen az (x_i,y_i) pontra essen. Ennek a görbének az egyenletét keressük.
Egyenlet megoldásához szükség van az egyenletek fokszámának meghatározására: n db ponthoz n-1-ed fokú polinomra van szükség.
Görbe: r(t)
$x(t)=a_{n-1}t^{n-1}+a_{n-2}t^{n-2}+...+a_1t+a_0$
$y(t)=b_{n-1}t^{n-1}+b_{n-2}t^{n-2}+...+b_1t+b_0$
egyenletek pontonként, ahol ti-t ismerem:

$r(t_1)$

$x(t_1)=x_1=a_{n-1}t_1^{n-1}+a_{n-2}t_1^{n-2}+...+a_1t_1+a_0$

$y(t_1)=y_1=b_{n-1}t_1^{n-1}+b_{n-2}t_1^{n-2}+...+b_1t_1+b_0$

$r(t_2)$

$x(t_2)=x_2=a_{n-1}t_2^{n-1}+a_{n-2}t_2^{n-2}+...+a_1t_2+a_0$

$y(t_2)=y_2=b_{n-1}t_2^{n-1}+b_{n-2}t_2^{n-2}+...+b_1t_2+b_0$

$r(t_n)$

$x(t_n)=x_n=a_{n-1}t_n^{n-1}+a_{n-2}t_n^{n-2}+...+a_1t_n+a_0$

$y(t_n)=y_n=b_{n-1}t_n^{n-1}+b_{n-2}t_n^{n-2}+...+b_1t_n+b_0$

n pont esetén (n-1)×2 egyenlet, melyek megoldása behelyettesítéssel nem megvalósítható. Megoldása **Lagrange-interpoláció**val (megadott pontokon átmenő görbe egyenlete):
a_0, a_1, …, a_{n-1}, és b_0, b_1,…, b_{n-1} skalárok, melyekkel a görbe felírható így:

$r(t)$

$x(t)=a_{n-1}t^{n-1}+a_{n-2}t^{n-2}+...+a_1t+a_0$

$y(t)=b_{n-1}t^{n-1}+b_{n-2}t^{n-2}+...+b_1t+b_0$

Megoldása Gauss eliminációval (átló alatt kinullázom). N db ismeretlen és n db független egyenlet esetén kapok csak egyértelmű megoldást (jól meghatározott). Kevesebb egyenlet esetén végtelen sok megoldás (alulhatározott), több esetén nem tudom az összes egyenletet igazzá tenni, tehát nincs megoldás(túlhatározott). Mivel ez polinom, ezért gyorsan és pontosan dolgozhatok vele, viszont paraméterezése nehézkes, emellett oszcillál, tehát ha a görbe közepén egy egyenes van, akkor a pontok az egyenes körül oszcillálva hullámot adnak.

<u>Módszerek paraméterek hozzárendelésére a pontokhoz:</u>

1. Uniform módszer: [0,1] intervallumot egyenlő részekre osztjuk. Maga alá görbülő görbékre nem alkalmazható.
2. Húrhossz szerinti paraméterezés: intervallum felosztása a pontok távolságának arányában.

Hermite-ív:

Ha a görbém egy része egyenes, akkor tudnom kell, hogy adott ponton merre tartson a görbe. Ehhez az adott pont érintőjére van szükség, tehát pontonként függvényre és deriváltra is szükség van. Mivel 2 megadott pont esetén 4 feltétel teljesül (2 pont és 2 vektor), 3-adfokú polinomra van szükség 2 pontra fektetett görbe esetén.
1-1 harmadfokú polinommal megadott koordináta függvényt keresünk az alábbi alakban:

$r(t)$

$x(t)$

$y(t)$

Ehhez adottak:

$P_0(p_0x, p_0y)$ $v_0(v_0x, v_0y)$

$P_1(p_1x, p_1y)$ $v_1(v_1x, v_1y)$

Feltételek:

$r(0)= P_0$	→	$x(0)= p_0x$	$y(0)= p_0y$
$r(1)= P_1$	→	$x(1)= p_1x$	$y(1)= p_1y$
$r'(0)= v_0$	→	$x'(0)= v_0x$	$y'(0)= v_0y$
$r'(1)= v_1$	→	$x'(1)= v_1x$	$y'(1)= v_1y$

Polinomiális függvényeket keresünk, koordináta függvényenként 4 ismeretlenünk van,

tehát összesen 4 egyenletet keresünk.

r(t)

$x(t)=a_3t^3+a_2t^2+a_1t+a_0$

$y(t)=b_3t^3+b_2t^2+b_1t+b_0$

r'(t)

$x'(t)= a_3\times3t^2+a_2\times2t+a_1$

$y'(t)=b_3\times3t^2+b_2\times2t+b_1$

x a 0 helyen vegye fel az általunk megadott koordinátát:

$x(0)= p_0x \rightarrow \mathbf{a_0= p_0x}$

$x(1)= p_1x \rightarrow a_3+a_2+a_1+a_0=p_1x$

$x'(0)= v_0x \rightarrow \mathbf{a_1= p_1x}$

$x'(1)= v_1x \rightarrow 3a_3+2a_2+a_1=v_1x$

Megoldás:

r(t)

$x(t)=(-2p_1x+2p_0x+v_0x+v_1x)\times t^3+(3p_1x-3p_0x-2v_0x-v_1x)\times t^2+v_0x\times t+p_0x$

$y(t)=(-2p_1y+2p_0y+v_0y+v_1y)\times t^3+(3p_1y-3p_0y-2v_0y-v_1y)\times t^2+v_0y\times t+p_0y$

Átalakítva pontokra és vektorokra:

$r(t)=(-2p_1+2p_0+v_0+v_1)\times t^3+(3p_1-3p_0-2v_0-v_1)\times t^2+v_0\times t+p_0$

$r(t)=(2t^3-3t^2+1)p_0+(-2t^3+3t^2)p_1+(t^3-2t^2+t)v_0+(t^3-t^2)v_1$

$r(t)=H_0(t)\times p_0+H_1(t)\times p_1+H_2(t)\times v_0+H_3(t)\times v_1$

Computer grafikában minden görbe polinom, paraméteres előállítású, ahol geometriai adatokat (pontok és érintők) szorzunk meg polinomokkal.

Hermite-ív problémái: hosszú vektor esetén visszahúzódás, és nem lehet megjósolni mekkora vektornál torzul el a görbe.

Bézier görbe

Ennek a továbbfejlesztése a **Bézier görbe**, ahol vektorok helyett újabb pontokkal kontrollálom a görbét, ezek a kontroll pontok. 4 pont határoz meg egy töröttvonalat, mivel a 2 kiegészítő pont is a körülbelüli haladást mutatja.

Hermite-ívhez képest:

$v_0=3(P_1-P_0)$

$v_1=3(P_3-P_2)$

$r(t)=(2t^3-3t^2+1)p_0+(-2t^3+3t^2)p_3+(t^3-2t^2+t)\times3(p_1-p_0)+(t^3-t^2)\times3(p_3-p_2)$

$r(t)=(^3_0)t^0\ (1-t)^3p_0+(^3_1)t^1\ (1-t)^2p_1+(^3_2)t^2\ (1-t)^1p_2+(^3_3)t^3\ (1-t)^0p_3$

Bézier görbe n+1 darab pontra n-edfokú polinomokkal:

$r(t)=(^n_0)t^0\ (1-t)^np_0+(^n_1)t^1\ (1-t)^{n-1}p_1+...+(^n_{n-1})t^{n-1}\ (1-t)^1p_{n-1}+(^n_n)t^n\ (1-t)^np_n$

$$r(t)=\sum_{i=0}^{n}\binom{n}{i}t^i(1-t)^{n-i}\times p_i$$

Felületek leírása és számítógépes ábrázolása.

1. **explicit** előállítás z=f(x,y)
   minden (x,y) pontra ad egy z értéket. R×R→R
   x,y síkon megkeresem a pontot, majd z irányába tolom.
   1. összes x-et behelyettesítem, y=0
   2. összes y-t behelyettesítem, x=0
   3. a többi pontot is behelyettesítem, kiszámolom

   <u>Drótháló megjelenítés:</u>
   Minden pontot ábrázolni kell, mert a köztes pontokat nem határozzák meg a szélsők.
   Így a felületi görbéket ábrázolom, nem magát a felületet
   Előny: könnyű ábrázolás
   Hátrány: maga alá görbülő felületet nem tud ábrázolni
2. **implicit** F(x,y,z) = 0
   1 (x,y) értékhez több z érték társul, tehát egymás alá görbül a felület.
   A pont 3 koordinátája egy nagy egyenletben egyesül. Azok a pontok vannak a felületen, ahol kiértékelés során 0-t kapunk.
   Előnye: maga alá görbülő felületet is tud ábrázolni
   Hátránya: pontonkénti kiértékelés
   Ha adott az F(x,y,z) n-edfokú polinom, akkor az F(x,y,z)=0 egyenletet kielégítő pontok összességét n-edrendű (vagy algebrai) felületnek nevezzük.
   Pl n=1: ax+by+cz+d=0 egy sík
   n=2: $ax^2+by^2+cz^2+dxy+exz+fyz+gx+hy+jz+k = 0$, ahol dxy,exz és fyz is másodfokú tagok, teház ha nincs x^2,y^2,z^2, akkor is másodfokú.
   Metszéspontok: felület+görbe, felület+felület.
   Egy n-edrendű felületnek és egy m-edrendű görbének n×m látható metszéspontja lehet.
   Tehát egy felület egyenletének fokszámát eldönthetem úgy, hogy egyenessel metszem, és a metszéspontok száma megadja a felület egyenletének fokszámát.
   Egy n-edrendű és egy m-edrendű felület metszésvonala egy m×n-edrendű görbe.
2. **paraméteres** megadás
   r: [a,b]×[c,d] (a két időintervallum Descartes szorzata)→V^3
   x: [a,b]×[c,d] →R, y: [a,b]×[c,d] →R, z: [a,b]×[c,d] →R
   Míg a görbék ábrázolása során pontokat kötök össze, itt paraméter vonalakkal rácsozom be
   Lépései ([a,b] intervallumot az u tengelyen, [c,d] intervallumot a v tengelyen ábrázolva):
   1. u értéke fix, a v értéke pedig c-ről d-re egységenként nő. (1. paraméter állandó, 2. változó) Eredményei pontok, amelyeket összekötve görbét kapok
   2. új, de szintén fix u érték, míg a-ból b-be nem érünk (az intervallumokon minél kisebb a lépték, annál simább a felület, annál pontosabb a végeredmény)
   3. az első 2 lépést a v értékein is megismételni
   Előnyök, hátrányok (implicit vs paraméteres)

	IMPLICIT	PARAMÉTERES
MEGJELENÍTÉS	-	+
KOORDINÁTA	+, behelyettesítés után	–, 3 egyenletből álló
ILLESZKEDÉSE	0 eredmény rajta van	2 ismeretlenes
A FELÜLETRE	nem 0, nincs	egyenletrendszer

   Előfordul, hogy az egyik implicit, a másik paraméteres formában jó.
   Felület érzékeltetése: rácsvonalakkal, színezéssel, szintvonalakkal

Problémák reprezentálása állapottéren:

A mesterséges intelligencia problémáinak megoldása a probléma megfogalmazásával kezdődik: a problémát leírjuk, reprezentáljuk. Az egyik legelterjedtebb reprezentációs technika az **állapottér-reprezentáció** (*state space representation*).

Legyen adott egy probléma, amit jelöljünk *p*-vel.

- Megkeressük *p* világának legalább egy, de véges sok – a probléma megoldása során fontosnak vélt – meghatározóját. *(pl.: objektum, pozíció, méret, hőmérséklet, szín, stb.)*
  Tegyük fel, hogy *m* ilyen jellemzőt találtunk.
- Minden egyes jellemző *p* világát különböző értékekkel jellemzi. *(pl.: szín: fekete/fehér; hőmérséklet: [−20°, 40°], stb.)*

Ha a megadott jellemzők épp rendre a $h_1,\ldots,h_m$ értékekkel rendelkeznek azt mondjuk, hogy *p* világa a $(h_1,\ldots,h_m)$ érték *m*-essel leírt **állapotban** *(state)* van. A világunk állapotainak halmaza az **állapottér** *(statespace)*.

Jelölje az *i*-edik jellemző által felvehető értékek halmazát H_i $(i=1,\ldots,m)$. Ekkor *p* állapotai elemei a $H_1 \times\ldots\times H_m$ halmaznak. Azokat a feltételeket, amelyek meghatározzák, hogy ebből a halmazból mely érték *m*-esek állapotok, **kényszerfeltételeknek** nevezzük. Az állapottér tehát az érték halmazok Descartes-szorzatának a kényszerfeltételekkel kijelölt részhalmaza:

$$A = \{\, a \mid a \in H_1 \times\ldots\times H_m \text{ és kényszerfeltétel}(a) \,\}$$

Az *A* állapottér azon állapotát, amit a probléma világa jellemzőinek kezdőértékei határoznak meg, **kezdőállapotnak** *(initial state)* nevezzük és *kezdő*-vel jelöljük.
A kezdőállapotból kiindulva a probléma világának sorban előálló állapotait rendre meg szeretnénk változtatni, míg végül valamely számunkra meg felelő ún. **célállapotba** *(goal state)* jutunk.
Jelölje $C \subseteq A$ a **célállapotok halmazát**. Megadása kétféleképpen történhet:

- felsorolással: $C = \{\, c_1,\ldots,c_l \mid c_i \in A,\ i=1,\ldots,l,\ l \geq 1 \,\}$
- célfeltételek megadásával: $C = \{\, c \mid c \in A \text{ és célfeltételek}(c) \,\}$

Általában $C \subset A$, hiszen *kezdő* $\notin C$, különben nincs megoldandó feladat.

Hogy célállapotba juthassunk, meg kell tudnunk változtatni bizonyos állapotokat. Az állapotváltozásokat leíró leképezéseket **operátoroknak** *(operator)* nevezzük. Nem minden operátor alkalmazható feltétlenül minden állapotra, ezért megszoktuk adni az operátorok értelmezési tartományát az **operátor alkalmazási előfeltételek** segítségével. Jelöljön az operátorok *O* véges halmazából *o* egy operátort. Ekkor:

$$Dom(o)= \{\, a \mid a \in A \text{ és } o\text{-alkalmazásának-előfeltétele}(a) \,\} \text{ és}$$
$$Rng(o)= \{\, o(a) \mid a \in Dom(o) \text{ és } o(a) \in A \}.$$

Legyen p egy probléma. Azt mondjuk, hogy a p problémát **állapottér-reprezentáltuk**, ha megadtuk az $\langle A, kezdő, C, O\rangle$ négyest, azaz:

- az $A \neq \emptyset$ halmazt, a probléma állapotterét,
- a $kezdő \in A$ kezdőállapotot,
- a célállapotok $C \subset A$ halmazát és
- az operátorok $O \neq \emptyset$ véges halmazát.

Jelölése: $p = \langle A, kezdő, C, O\rangle$

Legyen p egy probléma, $\langle A, kezdő, C, O\rangle$, p egy állapottér-reprezentációja és legyenek $a, a' \in A$. Az a állapotból az a' állapot **közvetlenül elérhető**, ha van olyan $o \in O$ operátor, hogy *o-alkalmazásának-előfeltétele(a)* és $o(a)=a'$.
Jelölése: $a \Rightarrow a'$, ha fontos, hogy o állítja elő a-ból a'-t: $a \underset{o}{\Rightarrow} a'$

Az a állapotból az a' állapot **elérhető**, ha vagy $a=a'$, vagy van olyan $a_1,\ldots,a_k \in A$ $(k \geq 2)$ (véges) állapotsorozat, hogy $a = a_1$, $a' = a_k$ és $a_i \Rightarrow a_{i+1}$ minden $1 \leq i \leq k-1$ esetén. Jelölése: $a \overset{*}{\Rightarrow} a'$.

Legyen $p = \langle A, kezdő, C, O\rangle$. A p probléma **megoldható** ebben az állapottér-reprezentációban, ha van olyan $c \in C$ célállapot, hogy $kezdő \overset{*}{\Rightarrow} c$. Ha $kezdő \underset{o_1,\ldots,o_r}{\overset{*}{\Rightarrow}} c$, és $(r \geq 1)$, akkor az $o_1,\ldots,o_r$ operátorsorozat a probléma egy **megoldása**.

Jelölje *költség(o, a)* az o operátor a állapotra alkalmazásának a **költségét**.

Legyen $p = \langle A, kezdő, C, O\rangle$ és $o_1,\ldots,o_r \in O$ a probléma egy megoldása, azaz $kezdő \underset{o_1,\ldots,o_r}{\overset{*}{\Rightarrow}} c$ $(c \in C)$. Legyen rendre $a_i \underset{o_i}{\Rightarrow} a_{i+1}$ $(1 \leq i \leq r)$, ahol $a_1 = kezdő$ és $a_{r+1} = c$. Ekkor a megoldás költsége:

$$\sum_{i=1}^{r} költség(o_i, a_i)$$

Ha $költség(o, a) \equiv 1$, akkor a megoldásköltsége az alkalmazott operátorok száma.

Az állapottérgráf

Legyen a p probléma az $\langle A, kezdő, C, O\rangle$ állapottér-reprezentációval megadva. Ez a reprezentáció egy irányított gráfot határoz meg.

- Az A állapottér elemei (az állapotok) a gráf **csúcsai**. Vezessük be az $a \in A$ állapot által definiált csúcsra az n_a jelölést. Ekkor a gráf csúcsainak halmaza $N = \{n_a \mid a \in A\}$.
- A gráf csúcsai közül kitüntetett szerepet játszanak a kezdő állapotot szemléltető ún. **startcsúcs** (jele: $n_{kezdő}$ vagy s) és a célállapotokat szemléltető **terminális csúcsok**. A terminális csúcsok halmaza tehát: $T = \{n_c \mid c \in C\}$.
- Minden $a \in A$-t szemléltető csúcsból **irányított élt** húzunk az $a' \in A$-t szemléltető csúcsba, ha a-ból közvetlenül elérhető a', azaz a gráf irányított éleinek halmaza a következő: $E = \{(n_a, n_{a'}) \mid a, a' \in A \text{ és } a \Rightarrow a'\}$.

Az $\langle N, s, T, E\rangle$ irányított gráfot a p probléma $\langle A, kezdő, C, O\rangle$ állapottér-reprezentációjához tartozó **állapottérgráfjának** vagy **reprezentációs gráfjának** nevezzük.

Legyen $\langle N, s, T, E\rangle$ a p probléma $\langle A, kezdő, C, O\rangle$ állapottér-reprezentációjához tartozó állapottérgráfja. Pontosan akkor **vezet** az állapottérgráf n_a csúcsából az ettől különböző $n_{a'}$ csúcsba irányított út, ha az a állapotból az a' állapot elérhető.

Legyen $\langle N, s, T, E\rangle$ a p probléma $\langle A, kezdő, C, O\rangle$ állapottér-reprezentációjához tartozó állapottérgráfja. Pontosan akkor **megoldható** p, ha van az állapottérgráfban a startcsúcsból valamelyik terminális csúcsba vezető irányított út.

A p probléma $\langle A, kezdő, C, O\rangle$ állapottér-reprezentációjában vegyük figyelembe az operátorok alkalmazási költségeit. Rendeljünk ekkor minden $(n_a, n_{a'})$ élhez költséget: ha $a \underset{o}{\Rightarrow} a'$, akkor *költség(o, a)* legyen ezen **él költsége** *(jelölve: költség(n_a, $n_{a'}$))*.

Egy $(n_{a1}, n_{a2})(n_{a2}, n_{a3}),\ldots,(n_{ak-1}, n_{ak}) \in E$ $(k \geq 2)$ **irányított út költsége** a benne szereplő élek költségösszege:

$$\sum_{i=1}^{k-1} költség(n_{a_i}, n_{a_{i+1}})$$

Ha minden él költsége egységnyi, az irányított út költsége éppen az út éleinek a száma.

Egy állapottér-reprezentált probléma megoldásának sikerét jelentősen befolyásolja a reprezentációs gráf bonyolultsága:

- a csúcsok száma,
- az egy csúcsból kiinduló élek száma,
- a hurkok és körök száma és hossza.

Ezért célszerű minden lehetséges egyszerűsítést végrehajtani. A lehetséges egyszerűsítések:

- csúcsok számának csökkentése – ügyes reprezentációval az állapottér kisebb méretű lehet;
- egy csúcsból kiinduló élek számának csökkentése – az operátorok értelmezési tartományának alkalmas megválasztásával;
- fává alakítás – a reprezentációs gráfban a hurkok, illetve körök „kiegyenesítésével"

A megoldás keresése visszalépéssel felvezető:

Gyakran előfordul, hogy egy problémát úgy próbálunk megoldani, hogy több külön-külön megoldandó részproblémára bontjuk. Ha a részproblémákat megoldjuk, az eredeti probléma megoldását is megkapjuk. A részproblémák megoldását további részek megoldására vezetjük vissza, egészen addig, amíg csupa olyan problémához nem jutunk, amelyeket egyszerűségüknél fogva már könnyedén meg tudunk oldani. A probléma megoldásnak ezt a módját **problémaredukciónak** nevezzük.

Problémaredukciós reprezentáció

Először is le kell írni az eredeti problémát, jelöljük ezt most p-vel.

- Egy probléma részproblémákra bontása során a nyert részek az eredeti problémához hasonló, de annál egyszerűbb problémák. Jelöljük az így nyert problémahalmazt P-vel. Természetesen $p \in P$.
- P problémáinak összegyűjtése során törekszünk arra, hogy legyenek közöttük olyanok, melyeket meg tudunk oldani, vagy ismerjük a megoldásukat. Ezek a problémák az ún. **egyszerű problémák**. Az egyszerű problémák halmazát E-vel jelöljük. $E \subset P$, hiszen $p \notin E$, különben nincs megoldandó feladat.

- Meg kell még adni a problémákat egyszerűsítő, illetve részekre bontó **redukciós operátorokat**. Egy redukciós operátor egy problémához azokat a (rész)problémákat rendeli hozzá, melyek egyenkénti megoldásával a problémamegoldása is előáll. Jelöljön a redukciós operátorok R véges halmazából r egy operátort, ekkor:

$$Dom(r) = \{\ q \mid q \in P \setminus E \text{ és } r\text{-alkalmazásának-előfeltétele}(q)\ \}$$
$$Rng(r) = \{\ r(q) = \{q_1,\ldots,q_m\} \mid q \in Dom(r) \text{ és } q_1,\ldots,q_m \in P\}.$$

Tehát egy redukciós operátor egy-egy problémához P egy-egy részhalmazát rendeli, így értékkészlete P hatvány halmazának valamely részhalmaza.

Legyen p egy probléma. Azt mondjuk, hogy a p problémát **problémaredukciós reprezentációval** írtuk le, ha megadtuk a $\langle P, p, E, R\rangle$ négyest, azaz

- a megoldandó $p \in P$ problémát,
- a $P \neq \emptyset$ halmazt, a p problémához hasonló problémák halmazát,
- az egyszerű problémák $E \subset P$ halmazát és
- a redukciós operátorok $R \neq \emptyset$ véges halmazát.

Jelölése: $\langle P, p, E, R\rangle$

Legyen a p probléma a $\langle P, p, E, R\rangle$ reprezentációval leírva és legyenek

$$Q = \{p_1, p_2,\ldots,p_i,\ldots,p_n\} \subseteq P$$
$$Q' = \{p_1, p_2,\ldots,p_{i-1}, q_1, q_2,\ldots,q_m, p_{i+1},\ldots,p_n\} \subseteq P$$

egy-egy problémahalmaz ($n \geq 1$, $m \geq 1$). Azt mondjuk, hogy a Q problémahalmaz **egy lépésben** vagy **közvetlenül redukálható** a Q' problémahalmazzá, ha van olyan $r \in R$ redukciós operátor, melyre $p_i \in Dom(r)$, és $r(p_i) = \{q_1, q_2,\ldots,q_m\}$.
Ennek jelölése: $Q \lessdot Q'$, illetve ha fontos, hogy az r redukciós operátor segítségével állítottuk elő Q-ból a Q'-t, akkor $Q \lessdot_r Q'$

Legyen a p probléma reprezentációja $\langle P, p, E, R\rangle$ és $Q, Q' \subseteq P$. A Q-ból a Q' **redukálható**, ha van olyan $P_1,\ldots,P_k \subseteq P$ $(k \geq 2)$ véges problémahalmaz-sorozat, hogy

$$P_1 = Q,\ P_k = Q' \text{ és}$$
$$P_i \lessdot P_{i+1} \text{ minden } 1 \leq i \leq k-1 \text{ esetén.}$$

Jelölése: $Q \lessdot^* Q'$.

Nyilvánvaló, hogy ha $P_i \lessdot P_{i+1}$ minden $1 \leq i \leq k-1$ esetén, akkor van olyan $r_1, r_2,\ldots,r_{k-1}$ redukciós operátorsorozat, hogy $P_i \lessdot_{ri} P_{i+1}$ $(1 \leq i \leq k-1)$. Ilyenkor azt mondjuk, hogy a Q probléma halmazt a Q' problémahalmazzá az $r_1, r_2,\ldots,r_{k-1}$ redukciós operátorsorozat segítségével redukáltuk. Jelölve: $Q \lessdot^*_{r1,\ldots,rk-1} Q$.

Legyen a p probléma problémaredukciós reprezentációja $\langle P, p, E, R\rangle$. A p probléma **megoldható** ebben a reprezentációban, ha $\{p\}$ csupa egyszerű problémából álló problémahalmazzá redukálható, azaz $\{p\} \lessdot^*_{r1,\ldots,rl} Q \subseteq E$. Ekkor az $r_1,\ldots,r_l$ redukciós operátorsorozatot tekinthetjük a probléma **megoldásának**.

Jelölje *költség(r,q)* ≥ *0* az *r* redukciós operátor *q* **problémára való alkalmazásának a költségét**, és ha *q* ∈ *E*, akkor *c(q)* ≥ *0* pedig a *q* egyszerű probléma közvetlen megoldásának költségét.

A ⟨*P*, *p*, *E*, *R*⟩ problémaredukciós reprezentációban *g(q)* a *q* ∈ *P* probléma megoldásának **minimális költsége**, ha

$$g(q)=\begin{cases} c(q) & ha\ q \in E, \\ \infty & ha\ q \notin E, de \\ & \neg\exists r\big(r \in R\ \wedge q \epsilon\ Dom(r)\big) \\ \min\{költség(r,q) + \sum_{q_i \in r(q)} g(q_i)\} & egyébként. \end{cases}$$

A részproblémák párhuzamos megoldása esetén lehetőségünk van a legrövidebb idő alatt elő állítható megoldás megkeresésére. Ekkor *költség(r, q)* az *r* redukciós operátor *q* problémára való alkalmazásának a **végrehajtási idejét**, *c(q)* pedig a *q* egyszerű probléma közvetlen megoldásának idejét jelenti.

A ⟨*P*, *p*, *E*, *R*⟩ problémaredukciós reprezentációban *t(q)* a *q*∈*P* probléma megoldásának minimális ideje,ha

$$t(q)=\begin{cases} c(q) & ha\ q \in E, \\ \infty & ha\ q \notin E, de \\ & \neg\exists r\big(r \in R\ \wedge q \epsilon\ Dom(r)\big) \\ \min\{költség(r,q) + max_{q_i \in r(q)} t(q_i)\} & egyébként \end{cases}$$

A megoldás keresése visszalépéssel:

Legyen a *p* probléma a ⟨*P*, *p*, *E*, *R*⟩ reprezentációval megadva. Ez a reprezentáció is egy irányított gráfot, ún. **ÉS/VAGY gráfot** határoz meg.

- A *P* problémahalmaz elemei (a problémák) a **gráf csúcsai**. Vezessük be az *q* ∈ *P* probléma által definiált csúcsra az n_q jelölést. Ekkor a gráf csúcsainak halmaza:
  $$N = \{n_q \mid q \in P\}.$$
- A gráf csúcsai közül kitüntetett szerepet játszanak a *p* problémát szemléltető ún. **startcsúcs** (jele: n_p vagy *s*)
- és az egyszerű problémákat szemléltető **terminális csúcsok**. A terminális csúcsok halmaza tehát: $T = \{n_e \mid e \in E\}$
- Egy *q* ∈ *P* problémát szemléltető csúcsból **irányított éleket** húzunk az $q_1,\ldots,q_m \in P$ problémákat szemléltető $n_{q1},\ldots,n_{qm}$ csúcsokba, amikor $\{q\} \not\leftarrow \{q_1, q_2,\ldots,q_m\}$. Ezek az élek összetartozónak tekinthetők: egy **ÉS élköteget** vagy **hiperélt** alkotnak. A gráf hiperéleinek halmaza tehát a következő:
  $$E = \{(n_q, \{n_{q1}, n_{q2},\ldots,n_{qm}\}) \mid q, q_1, q_2,\ldots,q_m \in P \text{ és } \{q\} \not\leftarrow \{n_{q1}, n_{q2},\ldots,n_{qm}\}\}.$$

Azt mondjuk, hogy az ⟨*N*, *s*, *T*, *E*⟩ irányított ÉS/VAGY gráf a *p* probléma ⟨*P*, *p*, *E*, *R*⟩ problémaredukciós reprezentációjához tartozó **reprezentációs gráfja**.

Visszalépéses megoldáskeresés ÉS/VAGY fák esetén

Legyen ⟨*P*, *p*, *E*, *R*⟩ a *p* probléma problémaredukciós reprezentációja.

Egy visszalépéses megoldáskereső **adatbázisa** a reprezentációs gráf egy a startcsúcsból induló hiperútját tartalmazza. Ezt az utat **aktuális hiperútnak** nevezzük.

Az adatbázis az aktuális hiperút csúcsait és e csúcsokból kiinduló bizonyos hiperéleket (explicit vagy implicit módon) nyilvántartó csomópontokból épül fel. A keresés megkezdésekor az adatbázis egyetlenegy - a *p* kezdő – problémát tartalmazó csomópontból áll.

Egy csomópont az alábbi információkat tartalmazza:

- egy $q \in P$ problémát;
- arra a csomópontra mutatót, mely a szülőproblémát (azt a problémát, melyre redukciós operátort alkalmazva előállt q) tartalmazza;
- q első részproblémáját (q első ÉS gyermekét) nyilvántartó csomópontra mutatót;
- q szülőjének q-t követő részproblémáját (q következő ÉS testvérét) nyilvántartó csomópontra mutatót;
- azt a redukciós operátort, melyet q-ra aktuálisan alkalmaztunk;
- q-ra a keresés során már alkalmazott (vagy még alkalmazható) redukciós operátorok halmazát

A visszalépéses megoldáskeresők **műveleteit** egyrészt a redukciós operátorokból származtatjuk, továbbá alkalmazhatjuk a visszalépést.

- Az $r \in R$ redukciós operátorból nyert művelet
  - alkalmazási előfeltétele: a kiválasztott levél csomópontban található problémára alkalmazható r, de még sikertelenül nem alkalmaztuk rá.
  - hatása:
    - előállítjuk a részproblémákat, s felfűzzük az akt. hiperútra az adott csúcshoz, mint szülőhöz (szülőnél egy mutató fog az 1. gyermekre mutatni).
    - Szülőnél operátor törlése (mert azt már alkalmaztuk).
    - Gyermekeknél létrejön a csomópont a megfelelő információkkal.
- A visszalépés
  - alkalmazási előfeltétele: van csomópont az adatbázisban.
  - hatása: Kitöröljük a szülő gyermekeit: a csúcsban van egy visszamutató a szülőre, ahonnan elérhető az 1. gyermek. Az első gyermekből van egy mutató a következő testvérre, ezáltal a szülő gyermekeit be tudjuk járni, s ki tudjuk őket törölni.

Az induló adatbázis létrehozása után kezdi el a **vezérlő** a keresést.

- Ha elfogytak a csomópontok az adatbázisból → az adott reprezentációban nincs megoldás.
- Ha van nem egyszerű problémát tartalmazó levélcsomópont az adatbázisban, akkor a vezérlő választ egyet.
  - A kiválasztott problémára választ egy még sikertelenül ki nem próbált redukciós operátort, és alkalmazza.
  - Ha ilyen nincs, visszalép.
- Ha a hiperút minden levél csomópontja egyszerű problémát tartalmaz → előállt egy megoldás.

Szisztematikus és heurisztikus gráfkereső eljárások: szélességi, mélységi és az A algoritmus.

Az állapottérgráfban keressük a megoldást: a start csúcsból valamely terminális csúcsba vezető utat. Az állapottérgráfot implicit módon – az állapottér-reprezentáció megadásával – adjuk meg a megoldást kereső rendszereknek. Ezek a keresés során addig és úgy építik a gráfot, amíg megoldást nem találnak, vagy amíg valamilyen ok miatt kudarcot nem vallanak a kereséssel.
A megoldást kereső rendszerek **felépítése**:

- Az **adatbázis** az állapottérgráfnak a keresés során előállított része, amit kiegészíthetünk a hatékony kereséshez szükséges bizonyos információkkal.
- A **műveletek** módosítják az adatbázist, azaz az állapottérgráf adatbázisbeli részéből az állapottérgráf egy újabb (további) részét állítják elő. A rendszer alkalmazhat:
    - állapottér-reprezentációs operátorokból származtatott műveleteket,
    - „technikai" műveleteket (pl. visszalépést).

  A műveleteknek is vannak végrehajtási feltételeik.
- A **vezérlő** irányítja a keresést. Megmondja, hogy a megoldás keresés folyamán az adatbázisra, annak mely részén, mikor, melyik a végrehajtási feltételeknek eleget tevő művelet hajtódjon végre. Figyeli azt is, hogy befejeződhet-e a keresés, azaz
    - meg van-e a problémamegoldása,
    - vagy kiderült, hogy nem megoldható a probléma.

A megoldást kereső rendszerek **osztályozhatók**:

- **Módosítható**-e valahogy egy már alkalmazott művelet hatása?
    - nem: nem módosítható megoldáskeresők
    - igen: módosítható megoldáskeresők
        - visszalépéses (backtracking) keresők
        - keresőfával keresők
- Használunk-e valamiféle **speciális tudást** a keresés során?
    - nem: irányítatlan (vak, szisztematikus) megoldáskeresők
    - igen: heurisztikus megoldáskeresők
        - „A **heurisztika** (heurisztikus szabály, módszer) olyan ökölszabály, stratégia, trükk, egyszerűsítés, vagy egyéb eszköz, amely drasztikusan korlátozza a megoldás keresését nagyméretű reprezentációs gráfokban."
- Milyen **irányú** a keresés?
    - **előrehaladó (forward)** vagy adatvezérelt kereső rendszer: a kezdő állapotból kiindulva keresünk célállapotba vezető utat.
    - **visszafelé haladó (backward)** vagy célvezérelt kereső rendszer: a célállapotból kiindulva – visszafelé haladva – próbáljuk rekonstruálni a kezdőállapotból odavezető utat.
    - **kétirányú (bidirectional)** kereső rendszer: mindkét irányból elindul, s valahol találkozik

A megoldáskereső rendszerek **értékelési szempontjai:**

- **Teljesség (completeness):** A rendszer minden olyan esetben megtalálja- e a megoldást, amennyiben az létezik?
- **Optimalitás (optimality):** Több megoldás létezése esetén a rendszer az optimális

megoldást találja-e meg?
- **Időigény (time complexity):** Mennyi ideig tart egy megoldás megtalálása?
- **Tárigény (space complexity):** Mekkora tároló területre van szükség a megoldás megtalálásához?

Keresőfával kereső rendszerek

Legyen $p = \langle A, kezdő, C, O \rangle$. A keresőfával kereső rendszerek **adatbázisa** a reprezentációs gráf már bejárt részét feszítő fa, az ún. **keresőfa**. A keresőfa csúcsait és a velük kapcsolatban lévő éleket (explicit vagy implicit módon) nyilvántartó csomópontok az alábbi információkat tartalmazzák:

- egy $a \in A$ állapotot;
- arra a csomópontra mutatót, mely a szülő állapotot tartalmazza;
- azt az operátort, melyet a szülő állapotra alkalmazva előállt a;
- státusz:
    - **zárt**, ha a utódait tartalmazó csomópontokat a keresés során már előállítottuk;
    - **nyílt**, egyébként.

Művelete a **kiterjesztés**: a keresőfát annak egy nyílt csomópontján keresztül kibővíti.

- alkalmazási előfeltétele: a keresőfában van nyílt csomópont.
- hatása:
    - alkalmazzuk az összes alkalmazható operátort a nyílt csomópont állapotára,
    - az előálló állapotok közül
        - amelyek még nem szerepeltek a keresőfa egyetlen csomópontjában sem, azokból a keresőfába felfűzött új nyílt csomópont készül,
        - amelyek már szerepeltek a keresőfa valamely csomópontjában, azok sorsa keresőfüggő.
    - a kiterjesztett csomópont zárttá válik.

A **vezérlő** megmondja, hogy melyik nyílt csomópont legyen a következő lépésben kiterjesztve

- Ha a kiválasztott nyílt csomópont állapota teljesíti a célfeltételeket, a keresőfában a szülőre mutatók mentén elő tudunk állítani egy megoldást is.Nincs megoldás, ha egyetlenegy nyílt csomópont sincs a keresőfában.

Ugyanazon probléma megoldásának keresése esetén lényeges eltérés lehet

1. a választás módjában. A vezérlő választhat
    - irányítatlanul, szisztematikusan
        - a csomópontok keresőgráfbeli mélysége alapján (szélességi és mélységi keresők)
        - a csomópontok állapotait előállító költség alapján (optimális kereső)
    - heurisztikusan (best-first algoritmus, A algoritmusok)
2. abban, hogy mi történik, ha a keresőgráf egy csúcsához a keresés során újabb odavezetőutat tár fel a vezérlő.
3. a célfeltételek vizsgálatának időpontjában.

Szélességi és mélységi keresők

Szélességi kereső:

- Nem informált kereső, azaz a probléma definícióján kívül nem rendelkezik más információval
- Gyökércsomópont kifejtésével indít az algoritmus, majd az összes gyökércsomópontból generált csomópont kifejtésével és így tovább
- A keresési stratégia minden adott mélységű csomópontot hamarabb fejt ki, mielőtt bármelyik egy szinttel lejjebbi csomópontot kifejtené

A komplexitás kifejezői:

1. elágazási tényező (b)
2. a legsekélyebb célállapot mélysége (d)
3. maximális mélység (m)
4. idő: a keresés közben generált csomópontok száma
5. tár: a memóriában maximálisan tárolt csomópontok száma

A szélességi kereső értékelése

Teljesség: A vezérlő,

- ha van megoldás, tetszőleges reprezentációs gráfban véges sok keresőlépés után előállít egy megoldást, ha az elágazási tényező (b) véges, akkor teljes, tehát az algoritmus garantáltan megtalál egy megoldást, ha az létezik
- ha nincs az adott reprezentációban megoldás, akkor véges gráf esetén azt a nyílt csomópontok elfogyásával felismeri.

Optimalitás: Ha van megoldás, tetszőleges reprezentációs gráfban a vezérlő a legrövidebb megoldást állítja elő. Általában nem optimális, de pl., ha a költség a mélység nemcsökkenő függvénye, akkor igen

Tárigény: Nagy az adatbázis. Legyen a reprezentációs fa minden csúcsának b gyermeke, és l hosszúságú a legrövidebb megoldás. Ekkor a keresőgráf csomópontjainak száma a keresés végére (a legrosszabb esetben): $1 + b + b^2 + b^3 + \ldots + b^{d+1} - d \approx O(b^{d+1})$.

Időigény: $1+b+b2+\cdots+bd + (bd+1-b) = O(bd+1)$

Mélységi kereső:

- Nem informált kereső, azaz a probléma definícióján kívül nem rendelkezik más információval
- Mindig a keresési fa aktuális peremében a legmélyebben fekvő csomópontot fejti ki
- A keresés azonnal a fa legmélyebb pontjára jut el
- A legmélyebb csomópontok kifejtésüket követően kikerülnek a peremből, és a keresés visszalép ahhoz a következő legmélyebben fekvő csomóponthoz, aminek még vannak ki nem fejtett követői

A mélységi kereső értékelése

Teljesség: A vezérlő véges reprezentációs gráfban,

- ha van megoldás, véges sok keresőlépés után előállít egy megoldást,
- ha nincs a problémának az adott reprezentációban megoldása, akkor azt a nyílt csomópontok elfogyásával felismeri.
- Nem teljes, pl., ha a bal oldali részfa korlátlanul mély, és nem tartalmazza a megoldást. Csak véges körmentes gráfban teljes

Optimalitás: Nem optimális, pl., mélyebben fekvő megoldást talál

A nyílt csomópontokat gyakran

Időigény: 1+ b + b2+···+bm = O(bm)
Tárigény: 1+b*m = O(b*m) A gyökértől egy levélig terjedő utat kell tárolnia + az út minden egyes csomópontja melletti kifejtetlen csomópontokat
Ha egy kifejtett csomópont összes leszármazottja meg lett vizsgálva, akkor a csomópont törölhető a memóriából
a szélességi kereső **sorban**, a mélységi kereső **veremben** tartja nyilván, melyből mélységi szám szerint ezek épp megfelelő sorrendben kerülnek ki.

Az A algoritmus

Informált kereső, mert a probléma definícióján túl más információval is rendelkezik, nevezetesen heurisztikával és útköltséggel (a gyökércsomópontból a kérdéses csomópontig vezető út költsége)

- **Heurisztika (hasznosságfüggvény)**
  Egy költségbecslés, ami segít az algoritmusnak eldönteni, hogy melyik csomópontot fejtse ki hamarabb. A jó heurisztika nagyban felgyorsíthatja a keresést. A heurisztika nem mindig pontos, DE elvárjuk tőle, hogy célállapotban pontos legyen, ahol 0 az értéke.
- Az A* kereső a csomópontokat úgy értékeli ki, hogy figyelembe veszi
  - az aktuális csomópontig megtett út költségét, valamint
  - az adott csomóponttól a célig vezető út becsült költségét

1. A keresőgráf minden csomópontjában megbecsüljük a rajta keresztülhaladó megoldás költségét. Ez egyrészt a csomópontig vezető nyilvántartott út költsége, amihez hozzászámítjuk a célig hátralevő út becsült költségét:
   $$összköltség(m) = útköltség(m) + heurisztika(m),$$ azaz
   $$összköltség(m) = útköltség(n) + költség(n, m) + heurisztika(m),$$
   ha $(n,m) \in E$. Ha *összköltség*(m)*-mel jelöljük az m csúcson keresztül célba jutás optimális költségét, akkor minden m csúcsra
   $$összköltség^*(m) \approx összköltség(m).$$
   Kiterjesztésre az A algoritmus vezérlője a **legkisebb összköltségű** nyílt csomópontot választja ki.
2. Ha a vezérlő a keresőgráf egy csúcsához a keresés során **újabb odavezető utat** tár fel, azaz az n csomópont kiterjesztéskor előállt állapot szerepel már a keresőgráf m csomópontjában, és az
   $$útköltség(n) + költség(n, m) < útköltség(m),$$
   ekkor az új kisebb költségű utat tároljuk, a régit „elfelejtjük”.
   - Ha m nyílt volt, más teendő nincs.
   - Ha m zárt volt, a keresőfa m-ből induló részének csomópontjaiban az *útköltséget* frissíteni kell, ami problémát okoz:
     - külön eljárást írunk a frissítésre;
     - az A algoritmussal frissíttetjük a részgráfot;
     - megelőzzük a probléma kialakulását.

Az A algoritmus értékelése
Teljesség: A vezérlő,

- ha van megoldás, tetszőleges reprezentációs gráfban véges sok keresőlépés után előállít egy megoldást, az A* algoritmus teljes, tehát az algoritmus garantáltan megtalál egy megoldást, ha az létezik
- ha nincs a problémának az adott reprezentációban megoldása, akkor véges gráf esetén azt a nyílt csomópontok elfogyásával felismeri.

Optimalitás: Nincs garancia az optimális megoldás előállítására. De ha minden $a \in A$ esetén $h(a)$

$\leq h^*(a)$, ahol $h^*(a)$ az a állapotból célba jutás optimális költsége, akkor az **A** algoritmus az optimális megoldást állítja elő, ha van megoldás. Ez az **A*** algoritmus. Gráfkeresést alkalmazva, véges fában, konzisztens heurisztikával A* optimális

Legyen P és Q két A^* algoritmus! Azt mondjuk, hogy P *jobban informált*, mint Q, ha célállapotot tartalmazó csomópontok kivételével bármely n csomópontra

$$heurisztika_P(n) > heurisztika_Q(n)$$

teljesül, ahol $heurisztika_P$ és $heurisztika_Q$ a P és Q algoritmusok heurisztikus függvényei. (Más szóval: a P algoritmus alulról pontosabban becsli a hátralévő út költségét bármely csúcsban.)

A monoton A algoritmus

Azt mondjuk, hogy egy h heurisztikus függvény kielégíti a **monoton megszorítás** feltételét, ha értéke bármely él mentén legföljebb az illető él költségével csökken, azaz minden $(n,m) \in E$ él esetén $h(n) - h(m) \leq koltseg(n, m)$.
Monoton A algoritmusnak nevezzük azt az **A** algoritmust, amelynek heurisztikus függvénye monoton megszorításos.

Kétszemélyes teljes információjú játékok leírása és reprezentálásuk:

Egy játék leírásához meg kell adni

- a játék lehetséges állásait (helyzeteit),
- a játékosok számát,
- hogyan következnek lépni az egyes játékosok (pl. egy időben vagy felváltva egymás után),
- egy-egy állásban a játékosoknak milyen lehetséges lépései (lehetőségei) vannak,
- a játékosok milyen – a játékkal kapcsolatos – információval rendelkeznek a játék folyamán,
- van-e a véletlennek szerepe a játékban és hol,
- milyen állásban kezdődik és mikor ér véget a játék,
- és az egyes játékosok mikor, mennyit nyernek, illetve veszítenek.

Osztályozás

- a játékosok száma szerint: pl. kétszemélyes játékok
- a játszma állásból állásba vivő lépések sorozata: diszkrét játékok
- az állásokban véges sok lehetséges lépése van minden játékosnak és a játszmák véges sok lépés után véget érnek: véges játékok
- a játékosok a játékkal kapcsolatos összes információval rendelkeznek a játék folyamán: teljes információjú játékok;
- nincs a véletlennek szerepe a játékban: determinisztikus játékok
- a játékosok nyereségeinek és veszteségeinek összege 0: zérusösszegű játékok.

A továbbiakban játék alatt kétszemélyes, diszkrét, véges, teljes információjú, determinisztikus, zérusösszegű stratégiai játékot fogunk érteni.

A játékok reprezentációja

Jelölje a két játékost A és B, a játékállások halmazát H. A játékot az $a_0 \in H$ kezdőállásban kezdje $J_0 \in \{A, B\}$. Tegyük fel, hogy a játékosok a játék során felváltva lépnek, és ismerjük az egyes állásokban megtehető lépéseket: $\{l \mid l: H \rightarrow H\}$. Az l lépés egy a állásban akkor tehető meg, ha *l-megtételének-előfeltétele(a).* A játék az a állásban véget ér, ha *végállás(a).* A szabályok leírják, itt ki a nyerő játékos*: nyer:* $\{a \mid végállás(a)\} \rightarrow \{jön_a, volt_a\}$, ahol $jön_a$ lenne az a állásban soron következő

játékos ($jön_a \neq volt_a$). E játék állapottér-reprezentációja az az $\langle A, kezdő, C, O\rangle$ négyes, ahol

- $A = \{(a, J) \mid a \in H, J \in \{A, B\}, J \text{ következik lépni}\}$,
- $kezdő = (a_0, J_0)$
- $V = \{(a, J) \mid végállás(a), J \text{ nyer, ha nyer}(a) = jön_a\}$
- $O = \{o_l \mid o_l(a, J) = (l(a), I), I, J \in \{A,B\}, I \neq J\}$

A játék állapottér-reprezentációját szemléltető gráf a **játékgráf.** „Egyenesítsük ki" a játékgráfot fává.

A játékfában

- páros szinteken lévő állásokban a kezdő játékos, páratlan szinteken lévőkben pedig az ellenfele léphet;
- egy állást annyi különböző csúcs szemléltet, ahány különböző módon a játék során a kezdőállásból eljuthatunk hozzá;
- véges hosszúságúak az utak, hisz véges játékokkal foglalkozunk.

Ha a játék során *kezdő* állapotból a játékosok valamelyik $v \in V$ állapotba érnek, azaz $kezdő \overset{o_1,\ldots,o_r}{\Rightarrow} v\ (r \geq 1)$, azt mondjuk lejátszottak egy **játszmát**. A játszmákat a játékfában a startcsúcsból a levélelemekbe vezető utak szemléltetik. Egy játék játékfája a játék összes lehetséges játszmáját szemlélteti a startcsúcsból induló, a különböző levelekben végződő útjaival.

Az $\langle N, HE\rangle$ párt **ÉS/VAGY gráfnak** nevezzük:

- N nemüres halmaz, a gráf csúcsainak halmaza,
- $HE \subseteq \{(n, M) \in N \times 2^N \mid 0 \neq |M| < \infty\}$ pedig az irányított **hiperélek** halmaza.

Az ÉS/VAGY gráfban a gráf hiperéleinek egy olyan

$$(n_1, \{n_{11}, n_{12}, \ldots, n_{1k1}\}),$$
$$(n_2, \{n_{21}, n_{22}, \ldots, n_{2k2}\}),$$
$$\vdots$$
$$(n_r, \{n_{r1}, n_{r2}, \ldots, n_{rkr}\})$$

sorozata, ahol

$$\forall i \forall j \left(\neg(i = j) \supset \neg(n_i = n_j)\right) \wedge$$
$$\wedge \forall i \left((i > 1) \supset \exists j \left((i > j) \wedge \left(n_i \in \{n_{j_1}, n_{j_2}, \ldots, n_{jk_j}\}\right)\right)\right)$$

a gráf egy **hiperútja.**

Nyerő stratégia:

Stratégiai játékok azok a játékok, melyekben játékosoknak a játék kimenetelére (ellenőrizhető módon) van befolyásuk. Ilyen játékok pl. a sakk, a bridzs, a póker, az üzleti „játékok"mint két vállalat konkurencia harca, harci „játékok".

A stratégia

A *J* játékos **stratégiája** egy olyan S_J: $\{(a, J) \mid (a, J) \in A\} \rightarrow O$ döntési terv, amely *J* számára előírja, hogy a játék során előforduló azon állásokban, melyekben *J* következik lépni, a megtehető lépései közül melyiket lépje meg.

A J játékos stratégiáinak szemléltetése a játékfában

Alakítsuk át a játékfát ÉS/VAGY fává *J* játékos szempontjából: *J* lépéseit szemléltető élek mindegyike egy élből álló hiperél marad (VAGY élek), ellenfelének egy-egy állásból megtehető lépéseit szemléltető élköteg egy-egy hiperél lesz (ÉS élek). Ebben az ÉS/VAGY gráfban *J* stratégiáit a startcsúcsból kiinduló olyan hiperutak szemléltethetik, melyek levelei az eredeti játékgráfnak is levelei.

Tegyük fel, hogy a *J* játékos az S_J stratégiájával játszik. Ekkor csak az S_J-t szemléltető hiperutat alkotó közönséges utak által szemléltetett játszmák játszhatók le.

Tegyük fel, hogy az *A* játékos az S_A, a *B* játékos pedig az S_B stratégiájával játszik. A két stratégia egyértelműen meghatározza a lejátszható játszmát.

A *J* játékos stratégiáját *J* **nyerő stratégiájának** nevezzük, ha (az ellenfelének stratégia választásától függetlenül) minden a stratégia alkalmazása mellett lejátszható játszmában *J* nyer.

A *J* szempontjából átalakított ÉS/VAGY fában a *J* nyerő stratégiát szemléltető hiperút levélelemei mind **J-nyerő állások.**

Tétel: (Az általunk vizsgált) minden játék esetén valamelyik (de nyilván csak az egyik) játékos számára van nyerő stratégia.

Lépésajánló algoritmusok: a min-max módszer, az alfa-béta vágás.

Minimax algoritmus

Cél: a támogatott játékosnak, *J*-nek, egy adott állásban „elég jó" lépést ajánlani. Az algoritmus számára át kell adni

- a játék $\langle A, kezdő, C, O\rangle$ reprezentációját,
- *J* azon *a* állását, ahol lépni következik,
- az állások „jóságát" *J* szempontjából becslő $h_J : A \rightarrow R$ heurisztikát
- és egy mélységi *korlátot*.

Az algoritmus fő lépései:

1. A játékfa *(a, J)* állapotot szemléltető csúcsából kiinduló részének előállítása *korlát* mélységig.
2. A részfa leveleiben található állások jóságainak becslése a heurisztika segítségével: $jóság(n_b) = h_J(b)$.
3. Szintenként csökkenő sorrendben a részfa nem levél csúcsai jóságainak számítása: ha, az *n*

csúcs gyermekei rendre $n_1,\ldots, n_k$, akkor

$$jóság(n) \rightleftharpoons \begin{cases} \max\{jóság(n_1,\ldots,jóság(n_k)\}, & ha\ n\ szintje\ páros \\ \min\{jóság(n_1,\ldots,jóság(n_k)\}, & ha\ n\ szintje\ páratlan \end{cases}$$

Javaslat: az *a* állásból egy olyan lépést tegyen meg *J*, amelyik az n_a csúcs „jóság" értékével megegyező értékű gyermekébe vezet.

Tekintsünk egy játékfát, azon belül a játékfának egy részfáját. A részfa legyen mondjuk k = 3 mélységű, legyen bináris fa, (tehát minden szülőnek két gyermeke van). A támogatott játékosunk maximalizál, az ellenfél meg minimalizál. Levélelemektől szintenként felfelé haladva, először a k = 3-as szinten megvizsgáljuk a két levélelemet és a k=2-es szintre, a maximalizálónak, azaz a támogatott játékosnak beadjuk a nagyobbik értéket. Ez az érték ideiglenesen bekerül a k = 1-es szinten lévő minimalizálóhoz, azaz az ellenfélhez. Ezután visszamegyünk a k=3-as szinten a jobb oldali részfa levél elemeire, majd megvizsgáljuk őket. Itt is a nagyobbik érték kerül fel a kettes szinten lévő maximalizálóhoz. Ezután a minimalizáló megvizsgálja az ideiglenesen kapott értéket meg az épp most kapott értéket, majd a két érték közül kiválasztja a kisebbiket. Mivel bináris fáról beszélünk, így hasonlóan járunk el a gyökér jobb oldali részfájával is. Így a k = 1-es szinten két minimális értékünk van. Ezután a maximalizáló kiválasztja a nagyobbik értéket és ez bekerül a gyökércsomópontba. Az algoritmus arrafelé fogja ajánlani a lépést, amerről a gyökércsomópont értéke származik, mint levélelem.

Negamax algoritmus

Cél: a támogatott játékosnak, *J*-nek, egy adott állásban „elég jó" lépést ajánlani. Az algoritmus számára át kell adni

- a játék ⟨*A, kezdő, C, O*⟩ reprezentációját,
- *J* azon *a* állását, ahol lépni következik,
- az állások „jóságát" a soron következő játékos szempontjából becslő *h: A* → *R* heurisztikát
- és egy mélységi korlátot.

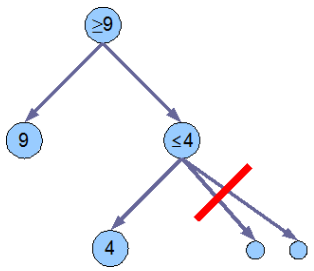
Az algoritmus fő lépései:

1. A játékfa *(a, J)* állapotot szemléltető csúcsából kiinduló részének előállítása korlát mélységig.
2. A részfa leveleiben található állások jóságainak becslése a heurisztika segítségével: $jóság(n_b) = h(b)$.
3. Szintenként csökkenő sorrendben a részfa nem levél csúcsai jóságainak számítása: ha az *n* csúcs gyermekei rendre $n_1,\ldots,n_k$, akkor

$$jóság(n) \rightleftharpoons \max\{-jóság(n_1,\ldots,-jóság(n_k)\}$$

Javaslat: az *a* állásból egy olyan lépést tegyen meg *J*, amelyik az n_a csúcs „ jóság" értékének −1-szeresével megegyező értékű gyermekébe vezet.

Alfa-Béta vágás



max

min

Alfa-vágás: Az aktuális csúcsban minimumot számolunk, a szülőjében maximumot. Az aktuális csúcs kiterjesztését félbeszakítjuk, ha a csúcsban eddig kiszámolt minimum kisebbé válik, mint a szülőben eddig kiszámolt maximum. A 28. ábrán látható egy ilyen szituáció: az aktuálisan kiterjesztendő csúcs a ≤ 4 feliratot viseli, hiszen már egy 4 súlyú gyermekét előállítottuk, vagyis az aktuális csúcs súlya max. 4(1) lesz. Mivel hasonló okból a szülőjének a súlya min. 9 lesz, így az aktuális

csúcs többi gyermekét már szükségtelen előállítanunk, hiszen az aktuális csúcs súlya innentől már teljesen irreleváns.

Béta-vágás: Ugyanaz, mint az alfa-vágás, csak most az aktuálisan kiterjesztett csúcsban maximumot, annak szülőjében pedig minimumot képzünk.