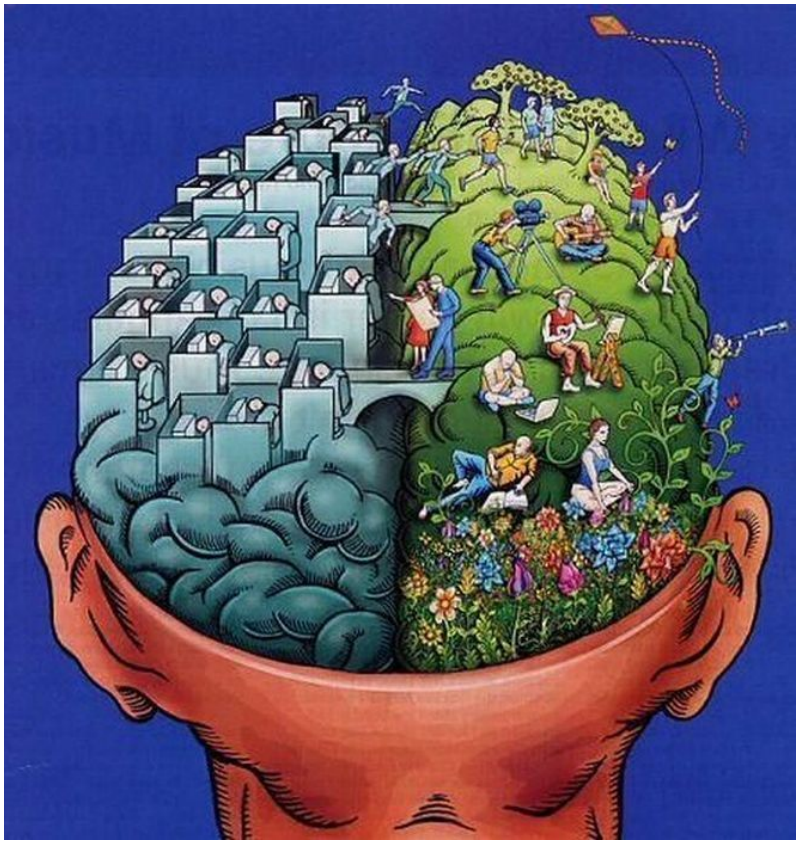
Mesterséges Intelligencia MI

Problémamegoldás kereséssel - általános problémák



Dobrowiecki Tadeusz
Eredics Péter, és mások

BME I.E. 437, 463-28-99
dobrowiecki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/tade

A most kezdődő anyagrész kérdései

– Mik a problémamegoldás lépései?
– Hogyan lehet problémákat jól definiálni?
– Hogyan mérhető a problémamegoldó hatékonyság?
– Milyen keresési stratégiák állnak rendelkezésre?
– Milyen tanulságok vonhatók le ezek komplexitásából?
– Mivel tudjuk mérsékelni a módszerek komplexitását?
– Hogyan terjeszthetők-e e módszerek nehezebb környezetekre?

Keressünk pénzt útvonaltervező szoftver fejlesztésével!
Manapság mindenki utazik, gépkocsival, repülővel, …
útvonalterv kell turistának, utazási iroda ügynökének,
robotnak nehéz terepen, víz alatt cikkázó UUV-nak, egy határsáv felett repdeső UAV-nak, ...
Egyre többen, egyre messzebbre akarnak menni,
gépkocsival egész kontinensek, repülővel már egész világ ...

A kigondolt (helyettünk intelligens) rendszer kap két helyszínt, tudja a mozgási (utazási) lehetőségeket, dolgozik, dolgozik, …
és kiad egy úttervet.

Sima ügy. Miért?
A probléma nem más, mint egy gráfban - melynek élei súlyozottak - élutat találni két csomópont között.
Gráfokat tanultunk, keresést tanultunk, optimális út megkeresését is tanultuk.

Pontosabban mit is tanultunk?

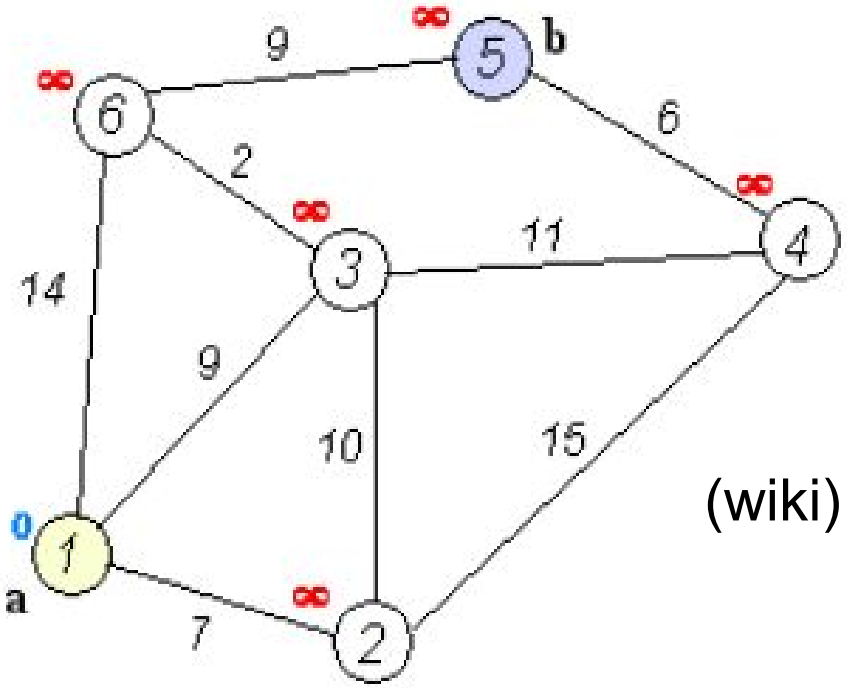
Előzmények:

gráf, csomópont, él, út, fa, útkeresés, keresési fa, mélységi keresés, szélességi keresés, élsúly, optimális út keresése, Dijkstra-algoritmus, komplexitás, időkomplexitás, tárkomplexitás, …

```
function FA-KERESÉS(probléma, stratégia) returns egy megoldás vagy kudarc
  a probléma kezdeti állapotából kiindulva inicializáld a keresési fát
  loop do
    if nincs kifejtendő csomópont then return kudarc
    a stratégiának megfelelően válassz ki kifejtésre egy levélcsomópontot
    if a csomópont célállapotot tartalmaz then return a hozzá tartozó megoldás
    else fejtsd ki a csomópontot és az eredményül kapott csomópontokat, és add a keresési fához
  end
```

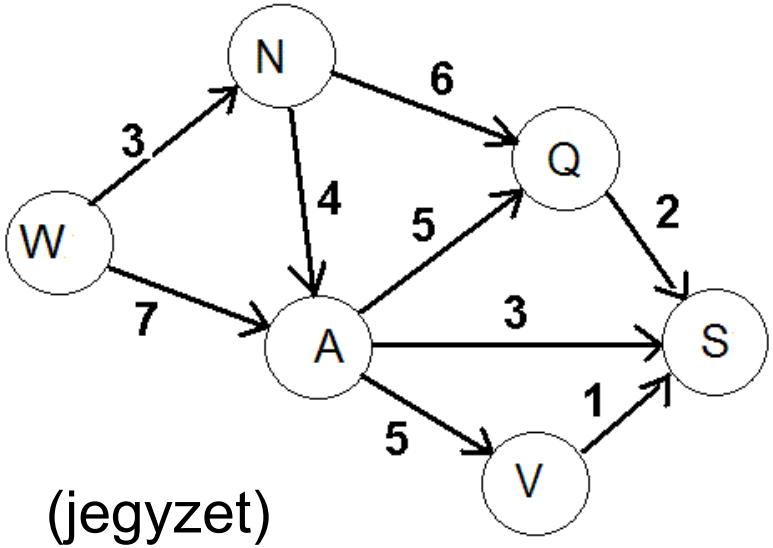
és most innen kezdjünk ...

Dijkstra-algoritmus



(wiki)

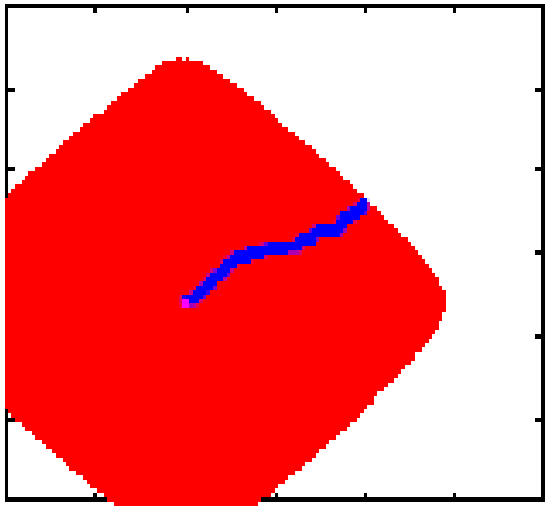
és még egy gráf …



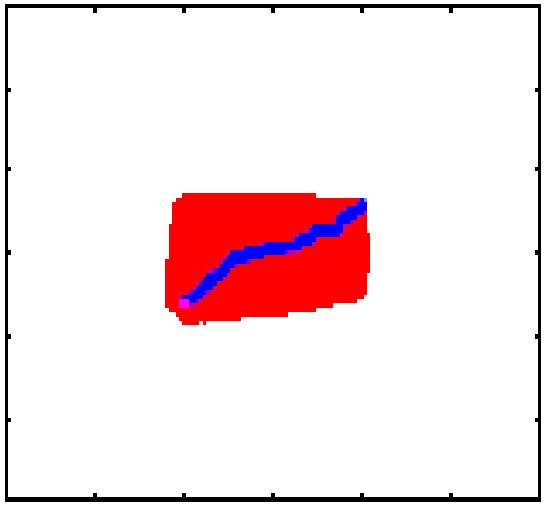
(jegyzet)

és most egy kicsit másképpen ...

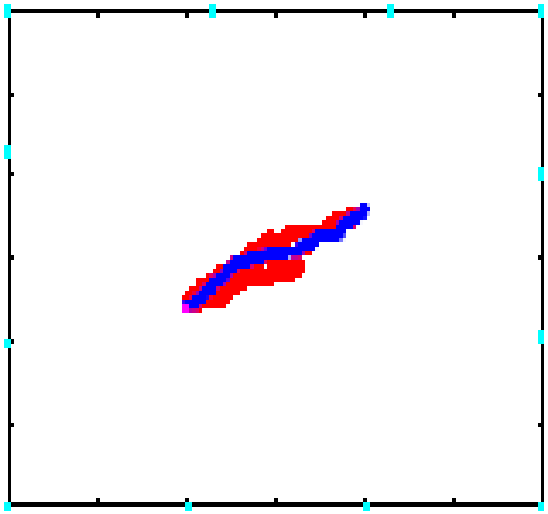
Mi a probléma?



Dijkstra



az egy másik



és még egy másik

Problémák: gráfméret - milliós csomópontok, sűrű kiszolgálás

Hatékonyságban különbségek, hogyan mérünk?

Komplexitás: idő-, tár- …, pl. mélységi keresés,
szélességi keresés, ...

Dijkstra mennyi? Ez sok, vagy kevés? Menni fog?

Bontás kisebb gráfokra, menni fog?

Egyedi, nem ismételt feladatnál?

És ha a kiszolgálások tömegesek, helyileg előre ismeretlenek?

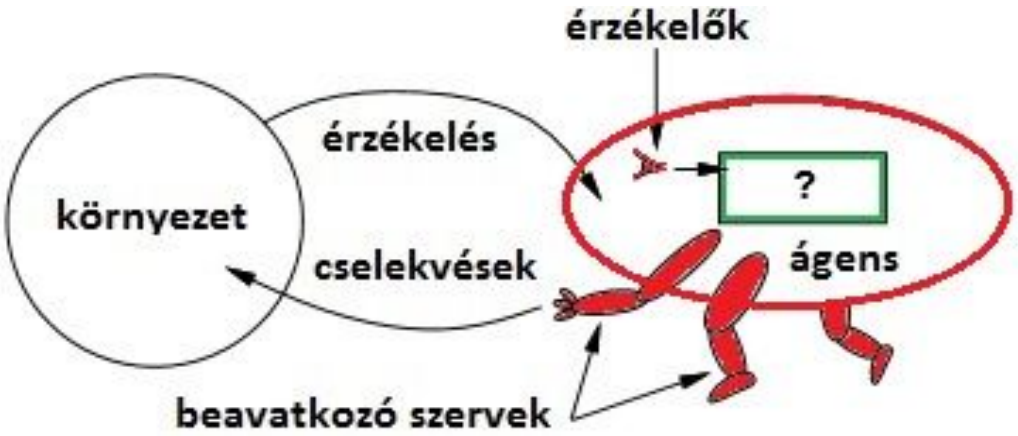
Ha ilyen sok kiszolgálás közben változnak az objektív körülmények (gráf alakja, élsúlyok, …)?

Ha a gráf „előre” nem adható meg (csak pici részletben a kiindulás környezetében)?

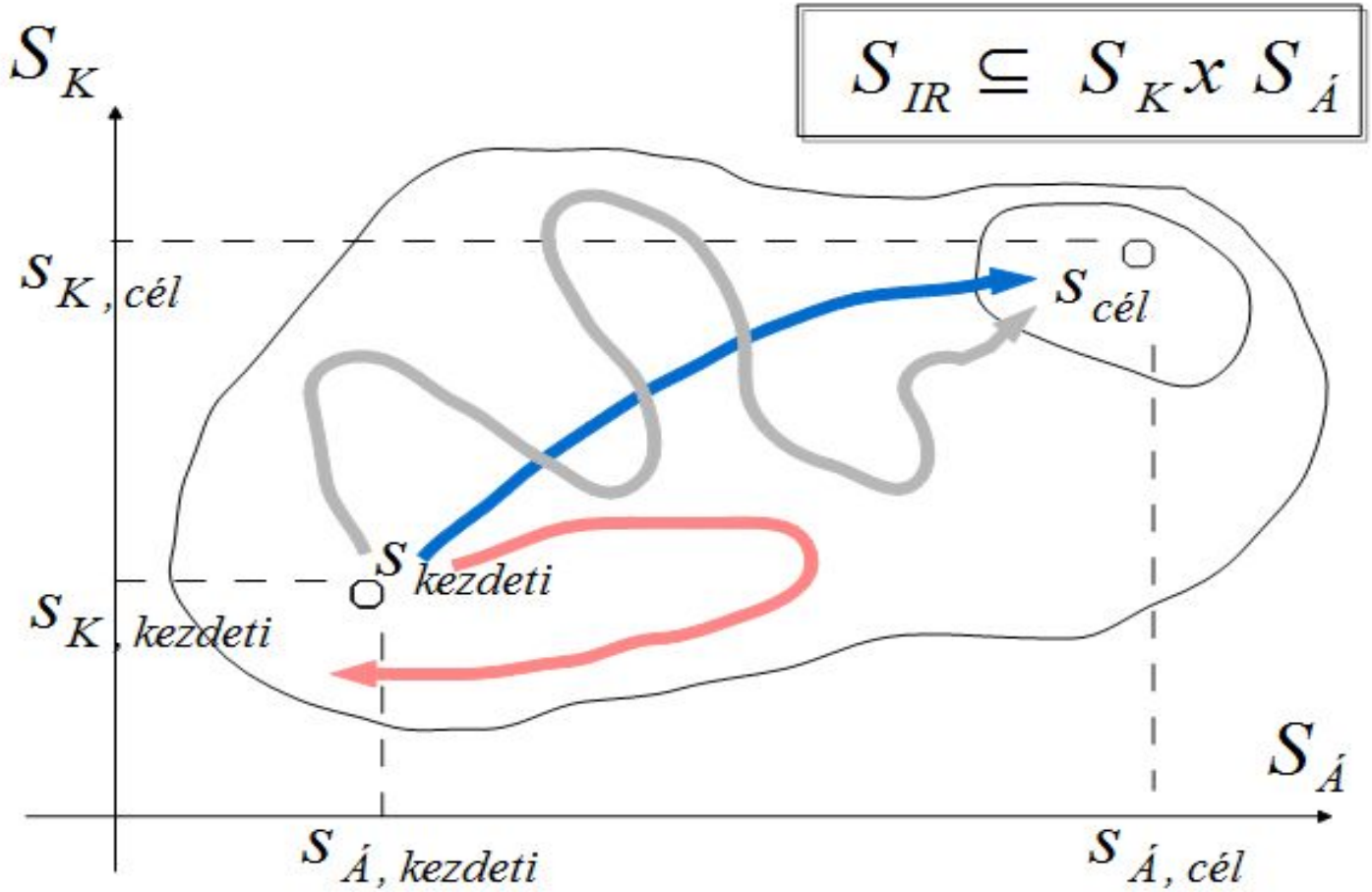
A tanult dolgokhoz képest nagyon sok kérdés, ami a módszerek tényleges használhatóságával kapcsolatos.

Fontos-e egyáltalán annyira a keresés, hogy vele foglalkozzunk?

Egy képzelt utazási iroda igénye még nem lehet elegendő érv.



$$S_{IR} \subseteq S_K x\, S_Á$$



A reflexszerű ágens tervezője keres tervezés közben.
Reflexszerű ágensnek már nem kell keresnie működés közben.

Célorientált ágens tervezője nem keres tervezés közben.
Célorientált ágens maga kénytelen keresni működés közben.

Absztrakció módja = a jól definiált probléma megalkotása

Probléma = információk gyűjteménye, alapvető elemei:
(hiedelmi) állapotok + legális cselekvések

Formális megközelítés:

(1) **kiinduló állapot** (ágensnek tudnia kell, hogy abban az állapotban van)
(2) ágens által rendelkezett **lehetséges cselekvések** halmaza
operátor = egy-egy cselekvés leírása, tipikusan

cselekvés: Ha egy állapot'-ban van,
AKKOR majd állapot”-ban lesz
a cselekvés alkalmazása után

„követő állapot” függvény = egy cselekvés egy adott állapotban való alkalmazásának hatására az ágens mely állapotba kerül?

ezek együtt (implicit módon adják meg):
a probléma állapottere: azon állapotok halmaza, amelyek a kiinduló állapotból valamilyen cselekvés sorozattal elérhetők.
(nehézség: az állapottér explicite ritkán adható meg)

Absztrakció módja = a jól definiált probléma megalkotása

állapottérben egy **út:** egy állapotból egy másik állapotba vezető cselekvéssorozat

(3) **célteszt**: az ágens el tudja dönteni, hogy az egy célállapot-e.

*a. a lehetséges célállapotok egy **explicit** halmaza*
- egyszerűen megnézi, hogy az ágens elérte-e ezek egyikét

*b. a cél valamilyen **absztrakt tulajdonsággal** van definiálva,*
pl. a sakkban az ún. „sakk-matt” helyzet

(4) **útköltség** függvény: az úthoz hozzárendel egy költséget

A keresési algoritmus kimenete - egy **megoldás**:
a kiinduló állapotból egy olyan állapotba vezető út,
amely teljesíti a cél tesztet.

A problémamegoldó hatékonyság mérése

Keresés:

1. egyáltalán **talál-e** megoldást.
2. a megtalált megoldás **jó megoldás-e**
   (egy alacsony útköltségű megoldás-e)?
3. mi a megoldás megtalálásához szükséges,
   az idő és a memória szükséglethez kapcsolódó **keresési költség?**

garantált + optimális + hatékony ????

A keresés **összköltsége**: **az útköltség + a keresési költség (!)**

avagy egy racionális ágens számára a probléma megoldásának költsége a probléma megoldásának számítási költsége ÉS a probléma megoldásának alkalmazási költsége EGYÜTTESEN

Példa: tili-toli (kirakó) játék, szélességi kereséssel

a szg. **Tianhe-1A**, 2.5 Pflops, 260 Tbyte mem, 2 Pbyte diszk
legyen 1 csp – 1 flop, 100 byte

Méret	Csomópont	Idő	Tár	Lépés
8-as (3 x 3)	10^5	400 nsec	10 Mbyte	< 31
15-ös (4 x 4)	10^{13}	4 msec	100 Tbyte	< 80
24-es (5 x 5)	10^{25}	100 év	1000 Ybyte	152 < < 208
35-ös (6 x 6)	10^{48}	10 Yév	????	????

T = 10^{12}, P = 10^{15}, Y = 10^{24},
Világűr kora kb. 14 Gév

Legrövidebb megoldás megkeresése:
NP teljes

(4 MW, 20 m$/év, 200 fő)

Mi kell? Tárból, időből levenni, a megoldásra garanciát adni
(a menyasszony legyen szép, okos, és gazdag). Felejtsük el!

De valamiből mindig le kell adni.
Levenni időigényből (időkomplexitásból), levenni tárból,
jó lenne levenni mindkettőből = kevesebb átnézett csomópont!

Mi a helyzet a mélységi és a szélességi kereséssel?

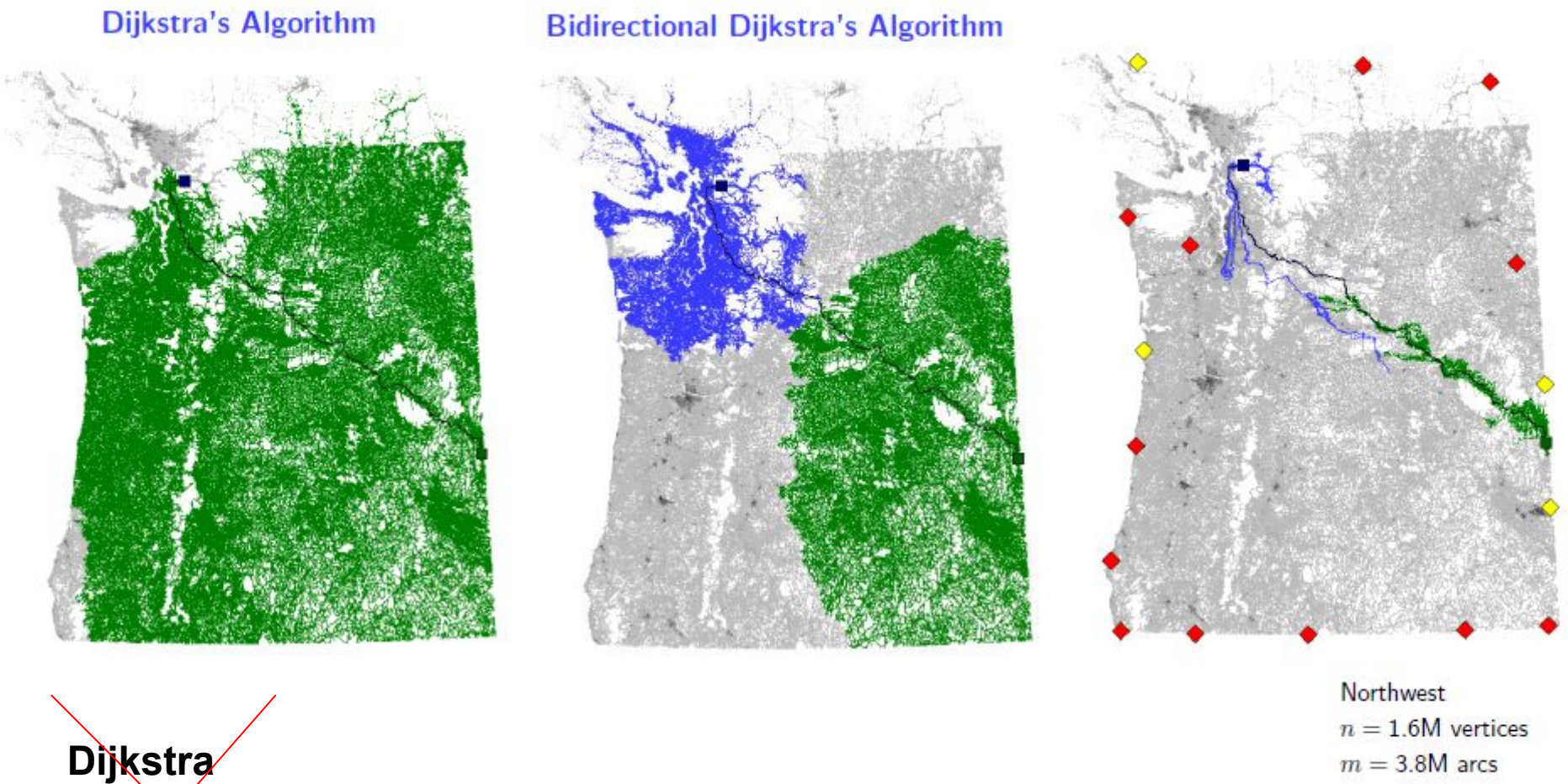
Garancia és gyorsaság – nem fog menni.

Ha garancia nagyon kell, meg kell enni a békát.

Garancia nélkül, csak ha jó a tapasztalat, vagy ha nagyon gyors
az eszköz, mert ha kudarcba fullad, még van idő megismételni.

Szólaltassuk a mérnöki kreativitást!

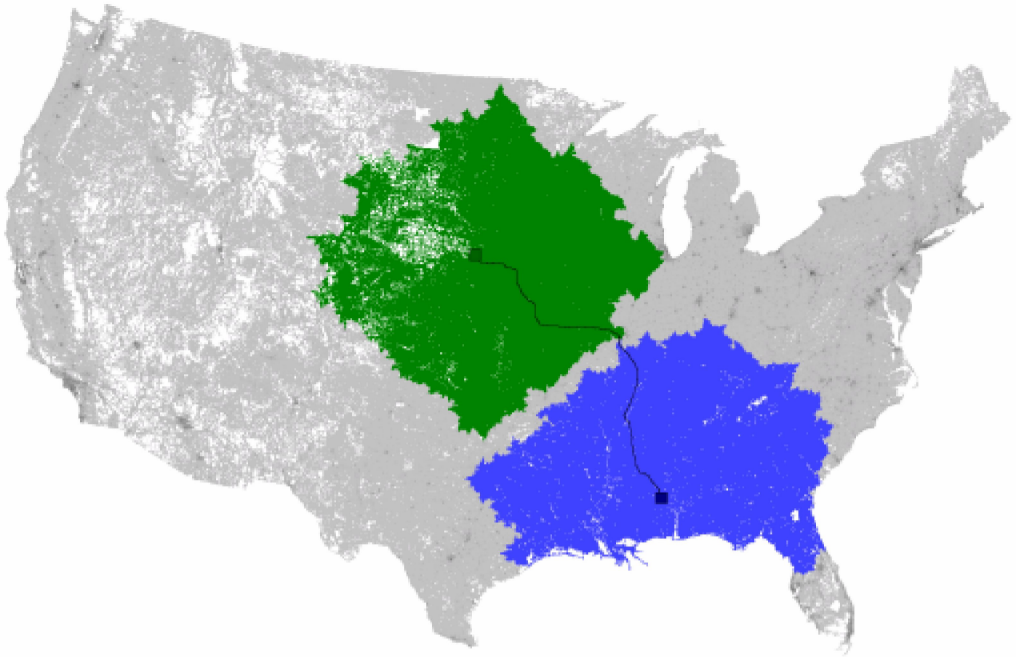
(1) **Büntessük a hátrakeresést.** (A*, stb.)
(2) **Keressünk mindkét irányból.** (kétirányú)
(3) **Ügyesen alkalmazzuk a mélységi keresést**. (iteratív)
(4) Alkalmazzuk a gráf előfeldolgozását (problémát alakítsuk át egyszerűbbre, de ennek van ára). (ALT landmarks)
(5) **Eleve tervezzük be a kicsi memóriát.** (EMA*)
(6) **Mondjunk le a visszalépésekről.** (hegymászás)
(7) **Engedélyezzük a nem a legjobb lépéseket is.** (nyaláb keresés)
(8) **Engedélyezzünk rossz lépéseket is.** (szimulált lehűtés)
(9) Tiltsunk egyes potenciálisan rossz lépéseket is. (tabú keresés)
(10) Folyamatosan profitáljunk az eddigi megoldásokból. (tanuló A*, stb.)
(11) **Randomizáljunk és ismételjünk.**
(12) **Maradjunk keresni az eredeti folytonos térben.**
...



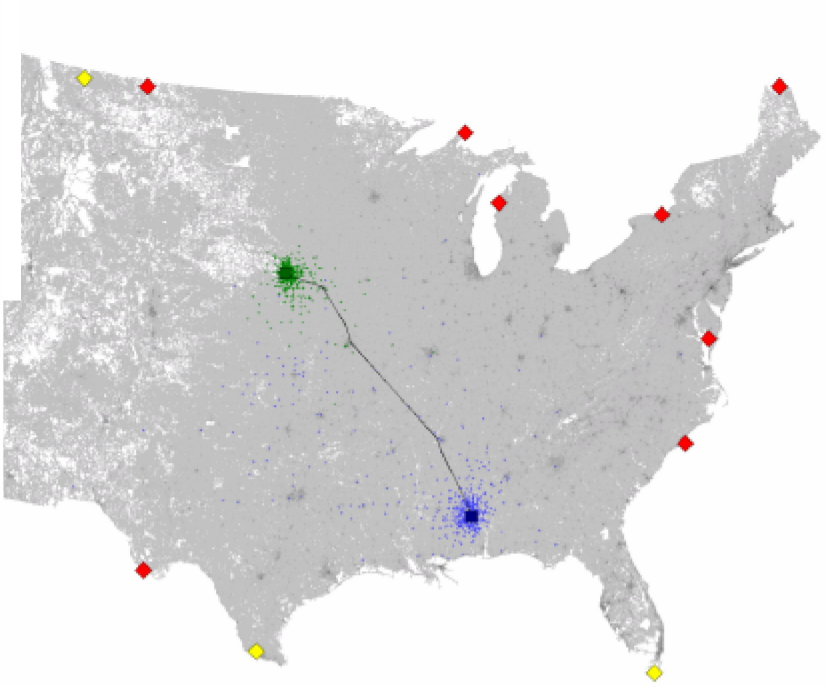
Dijkstra
Elő: -
Run: Dijkstra

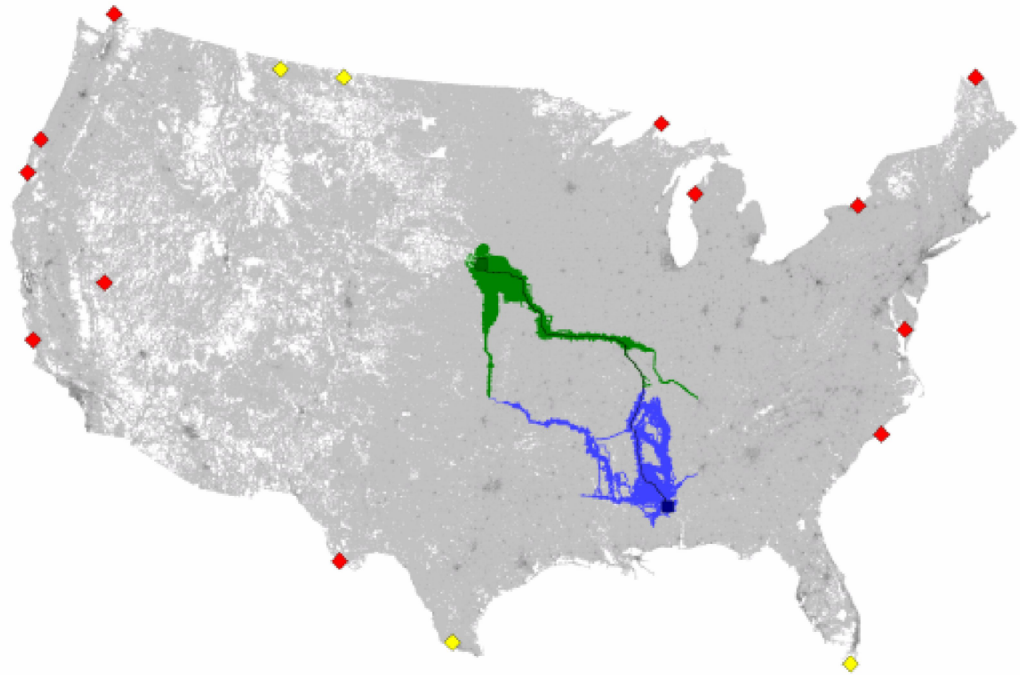
Teljes előfeldolgozás
Elő: n x n tábla, idő = (Dijkstra) = ca. 5 év
tábla = ca. 1Pbyte
Run: egyetlenegy kiolvasás

Ami kell
Elő: percek, max. órák, lineáris az adatban
Run: valós-időben



24M csúcs, 58M él





USA, travel times, random pairs

METHOD	PREPROCESSING minutes	PREPROCESSING MB	QUERY scans	QUERY ms
Dijkstra	—	536	11 808 864	5440.49
ALT(16)	18	2563	187 968	295.44
RE	28	893	2 405	1.77
REAL(16)	45	3032	592	0.80
REAL(64,16)	114	1579	538	0.86