Szoftverfejlesztő eszközök a tudományos munkában

Dr. Sergyán Szabolcs

EMBERI ERŐFORRÁS TÁMOGATÁSKEZELŐ

MINISZTERELNÖKSÉG
CSALÁDOKÉRT FELELŐS TÁRCA NÉLKÜLI MINISZTER

Nemzeti Tehetség Program

Routing

Szoftverfejlesztés a javából

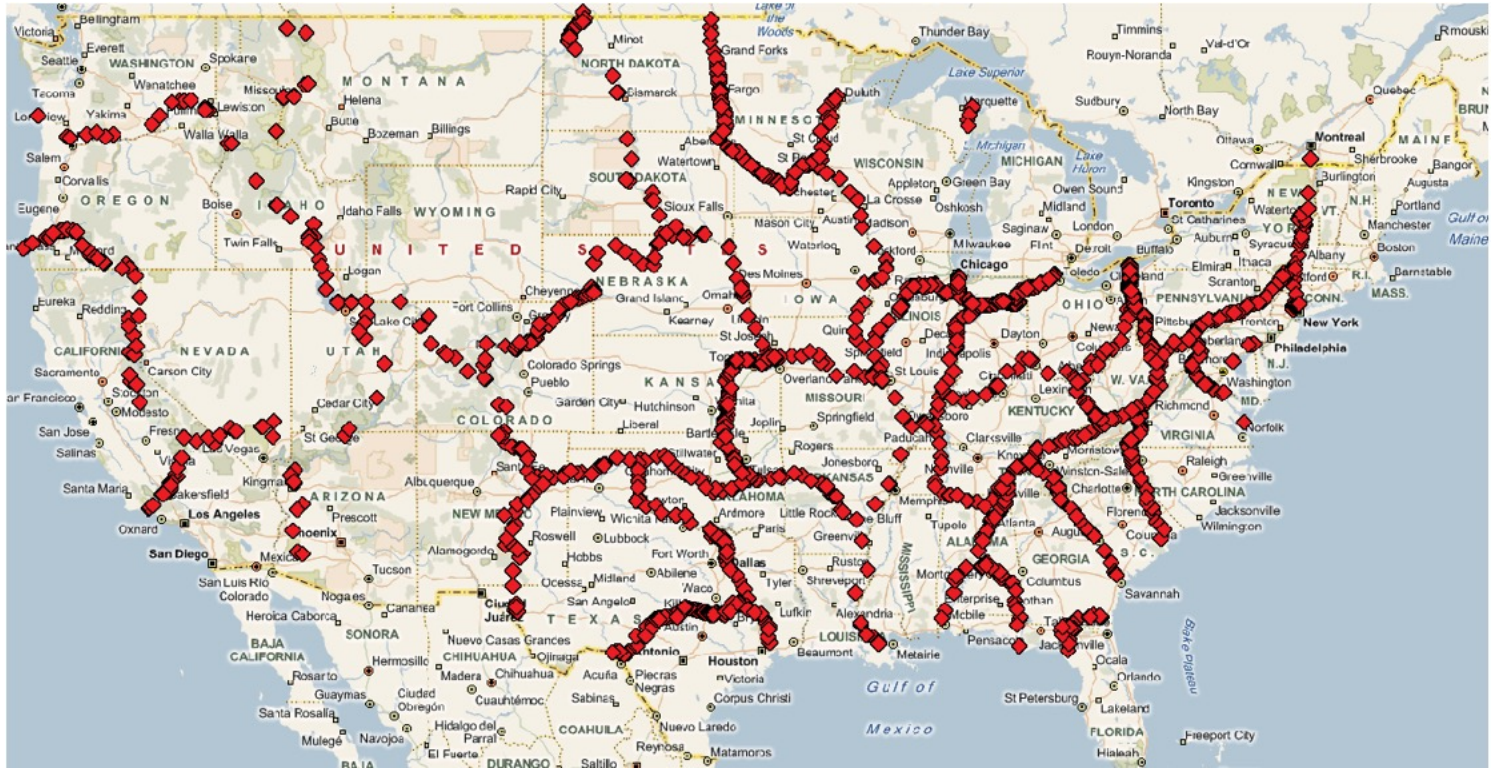
Dijkstra

```
függvény Dijkstra(G, start)
  ciklus ∀x ∈ G.Csúcsok
    d[x] ← ∞;  π[x] ← ∅
    S ⇐ x
  ciklus vége
  d[start] ← 0
  ciklus amíg S ≠ ∅
    u ← S.MinKivesz(d)
    ciklus ∀x ∈ u.Szomszédok
      ha d[u] + u.Súly(x) < d[x] akkor
        d[x] ← d[u] + u.Súly(x)
        π[x] ← u
      elágazás vége
    ciklus vége
  ciklus vége
  vissza (π, d)
függvény vége
```

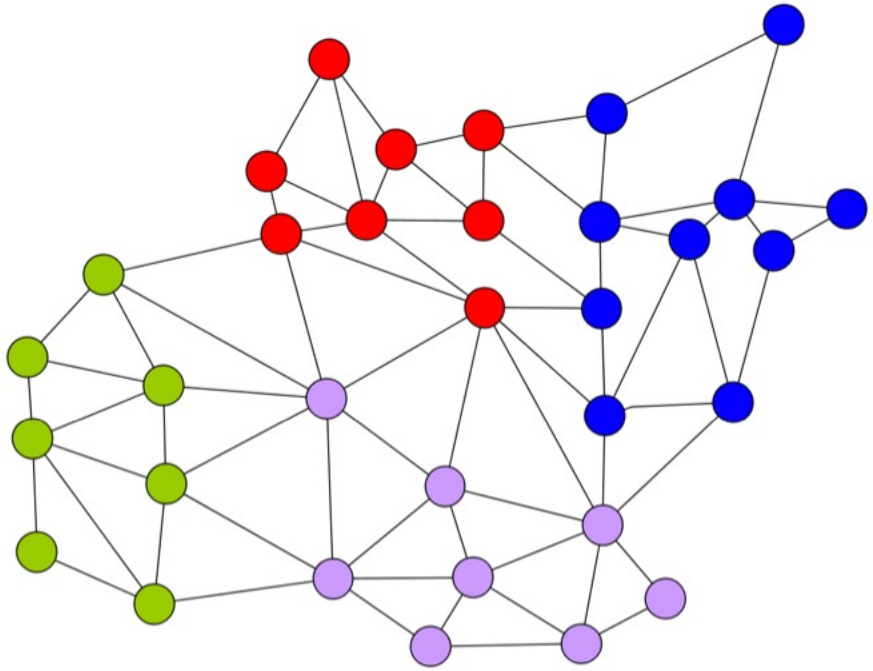
- Nem csúcsokon tervezünk (junction)
- Indulás szegmensről, érkezés szegmensre
- Szegmens a lényeg
- Viszont a junction is fontos, mert ott tudjuk figyelembe venni a kanyarodási tiltásokat

- Iszonyú sok szegmensen kell végigtervezni
- Cél: 1.000 km-es út max 250 millisec

Gráf partícionálás



Gráf partícionálás



- Lokális gráf, globális gráf
- Egy partíciónak kb. 50 border junction-je van
- Partícionként kb. 10.000 lokál szegmens
- A globál gráfnak vannak inner szegmensei (bármely két border között)
- A partíciókat bridge-k kötik össze
- A globál éleken már „gyorsan” tudunk haladni

Avoid

- Ferry, motorway, toll, ...
- Lokál gráfon egyszerű
- Globál gráfon viszont, ha egy inner szegmens megsért egy avoid-ot, akkor keresni kell olyan inner szegmens-t, ami megfelel az elvárásnak
- Két border junction között több inner szegmenst is tárolni kell
- Ezeket cost szerint rendezzük (növekvő sorrendbe)
- Dijkstránál, ha a kis costú szegmens megsérti az elvárt avoid-ot, akkor megvizsgáljuk a nagyobb costút is.

A*

- Dijkstra minden irányban növeli a keresési teret a source szegmensből indulva.
- Célunk viszont, hogy lehetőleg a target felé tartsunk.
- Ne csak azt nézzük, hogy mekkora costtal lehet eljutni az adott szegmensig, hanem becsüljük meg, hogy onnan mekkora costtal lehet eljutni a targetig.
- Heurisztika: például a vizsgált szegmens és a target légvonalbeli távolsága (valójában tunnel distance kell).

Turn penalty

- Kereszteződéseken áthaladásnál kell kanyarodási „büntetés” (akár az egyenes áthaladáshoz is).
- Ezt nem fogjuk megkapni adatként, nekünk kell szabályrendszert alkotni rá.
- A szabályt viszont nem akarjuk mindig újra és újra kiszámolni, ezért érdemes minden szegmens párra előre letárolni. (Iszonyú mennyiségű adat!)
- Jól tömöríthető, mert csak néhány penalty fordul elő.

Traffic applier

- Szolgáltatók adnak traffic információkat.
- Nem a gráf szegmenseire adják, hanem a térkép ún. linkjeire.
- Adott időközönkét rá kell tenni a trafficot a szegmensekre.

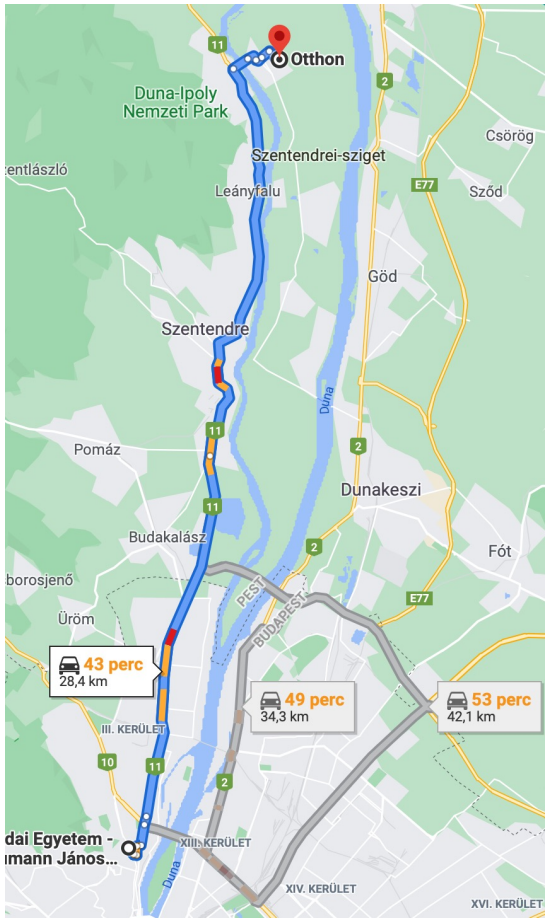
Partition updater

- A globális szegmenseken is kell trafficot aktualizálni.
- Viszont ilyenkor újra kell számolni az összes előre legyártott inner szegmenst.

Time domain

- Vannak olyan utak, amik bizonyos idő tartományokban nem járhatók.
- Vannak olyan kanyarodások, amikre ugyanez igaz.
- Sima útkeresésnél egyszerű a helyzet, mert tudjuk, hogy mikor járunk majd egy adott szegmensnél.
- Viszont a globális szegmensekkel itt is gond van.

Alternatív utak



- Jelentős mértékben térjen el az optimális úttól
- Ne legyen sokkal hosszabb (időben/távban)
- A letervezése öt alternatívával ne legyen sokkal lassabb az optimális út tervezésénél.
  (1.000 km-en kb. 750 millisec)

Térkép -> Gráf

- Hogyan lesz gráfunk?
- Open street map olcsón segít (van más is persze)
- Megjelenítésben is segít
- Viszont kell egy oda-vissza mapping a térkép adatok és a gráf(ok) között.

Routing client <-> routing server

- Routing request (mi legyen benne?)
- Routing response (és ebben?)
- API design

Fejlesztő eszközök, infrastruktúra

- Nyelv
- Tesztelés
- Verzió kezelés
- Dokumentálás
- Csapatmunka szervezés