Backtracking vs. Divide et impera Hamilton és Euler gráfok

6. ELŐADÁS

Sapientia-EMTE

2020-21

Bemelegítő

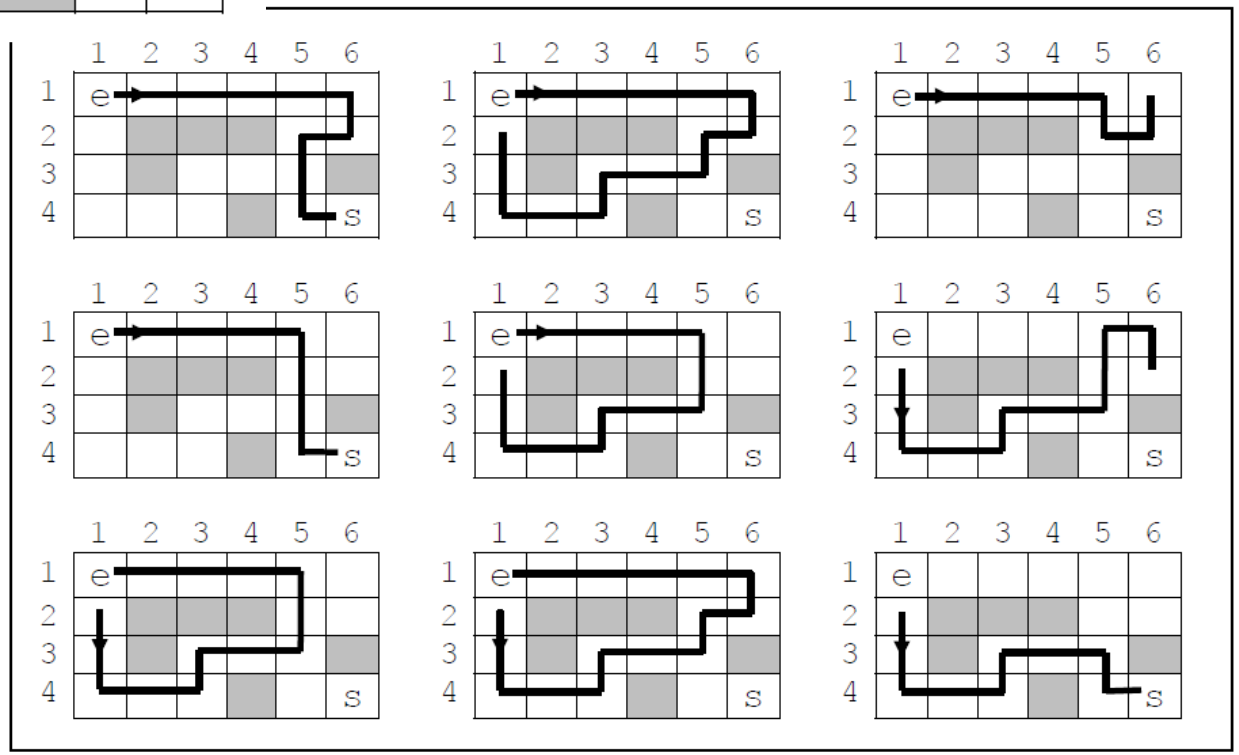
4. feladat:

A 2021 olyan négyjegyű szám, melyben az első kétszámjegyből álló kétjegyű szám és az utolsó kétszámjegyből álló kétjegyű szám egymás utáni természetes számok. Írd le az ilyen tulajdonságú négyjegyű természetes számokat 2021-ig!

Egér–sajt probléma (1)

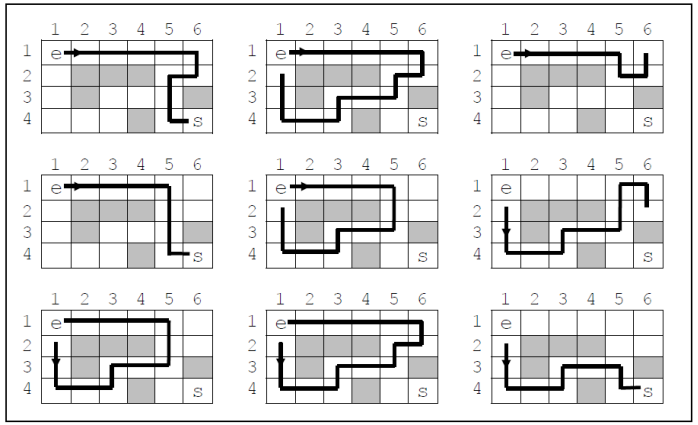


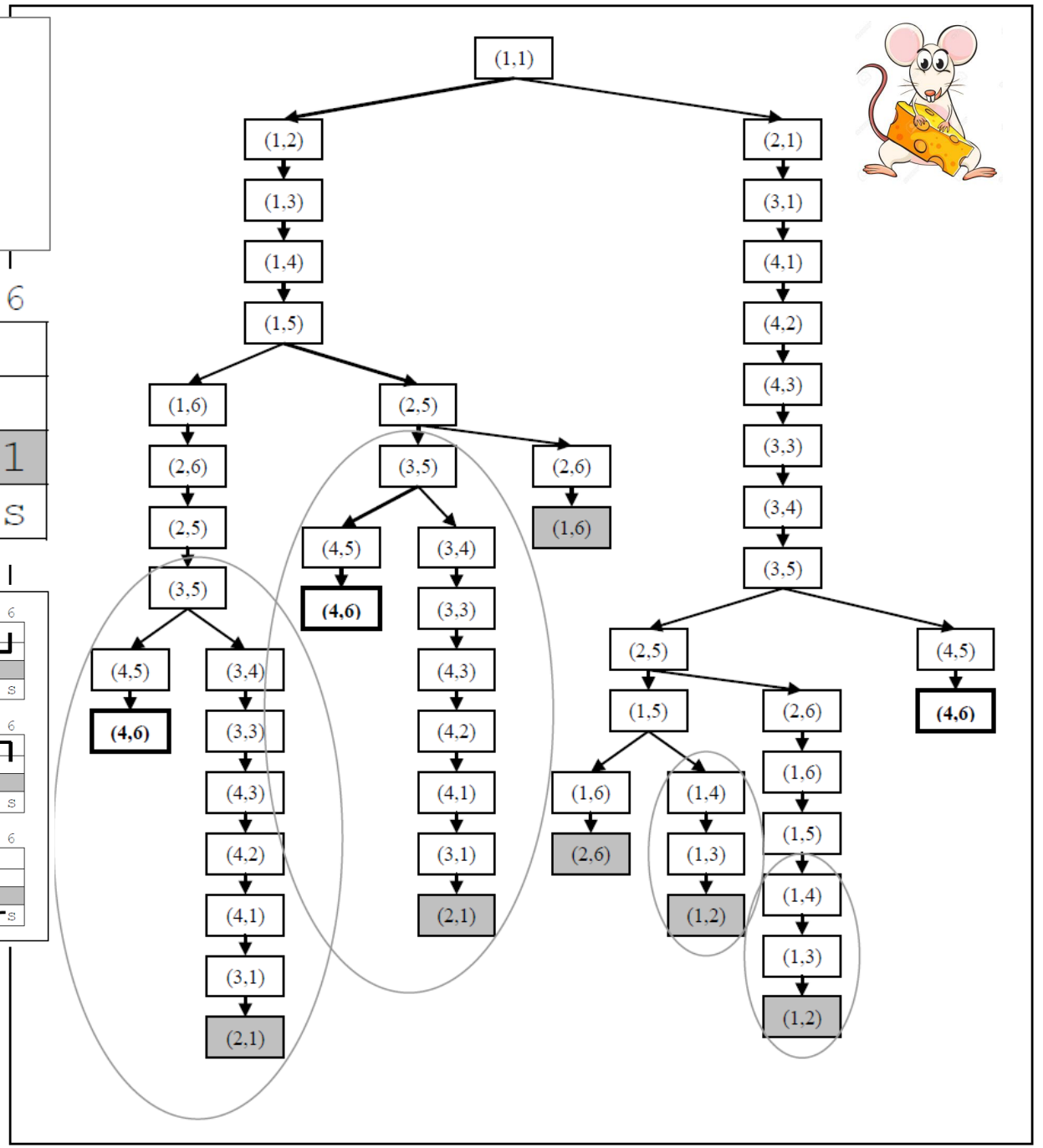
	1	2	3	4	5	6
1	e					
2		1	1	1		
3		1				1
4				1		s



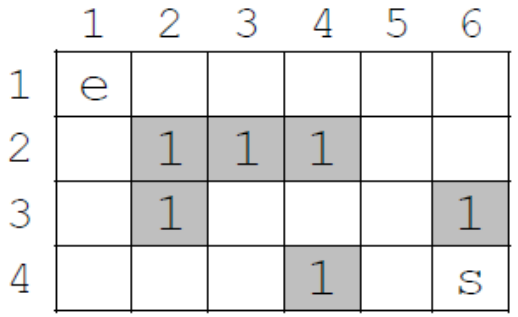
Egér–sajt (2)

	1	2	3	4	5	6
1	e					
2		1	1	1		
3		1				1
4				1		s





BackTracking-egér (1)



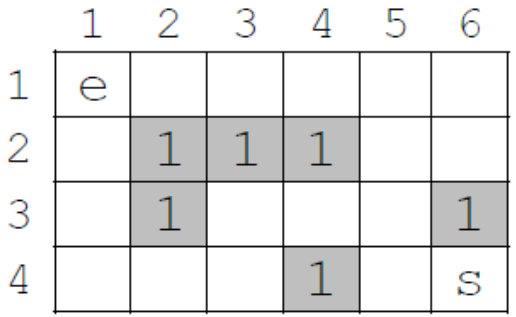
- út[]: a kurrens utat tárolja;
- útmin[]: a legrövidebb utat tárolja;
- kmin: a legrövidebb út hosszát tárolja;
- x, y: a kurrens pozíció;
- k: hányadik állomás a kurrens pozíció az aktuális úton;

```
backtracking_egér(a[][],ut[],utmin[],kmin,xs,ys,x,y,k)
    út[k].x = x
    út[k].y = y
    ha x == xs és y == ys akkor    // a sajthoz érkeztünk
        ha k < kmin akkor          // rövidebb utat találtunk
            útmin[1..k] = út[1..k]
            kmin = k
        vége ha
```

BackTracking-egér (2)

```
    különben
        a[x][y] = 1 // hurkok elkerülése végett
        ha a[x-1][y] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x-1,y,k+1)
        vége ha
        ha a[x][y+1] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x,y+1,k+1)
        vége ha
        ha a[x+1][y] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x+1,y,k+1)
        vége ha
        ha a[x][y-1] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x,y-1,k+1)
        vége ha
        a[x][y] = 0 // visszaállítjuk a szabad utat
    vége ha
vége backtracking_egér
```

Oszd-meg-és-uralkodj-egér (1)



- a függvény a legrövidebb út hosszát téríti vissza;
- a ∞ hossz jelentése: nincs út;
- x, y: a kurrens pozíció;
- a h1, h2, h3, h4 lokális változók a kurrens feladat fiúrészfeladatainak optimumértékeittárolják, időszakosan;
- a minimum függvény visszatéríti a paraméterértékek minimumát;
- a függvény meghívása: oszd_meg_és_uralkodj_egér(a,xs,ys,xe,ye).

```
oszd_meg_és_uralkodj_egér(a[][],xs,ys,x,y)
    ha x == xs és y == ys akkor // a sajthoz érkeztünk
        return 0
    vége ha
```

```
    h1 = ∞
    h2 = ∞
    h3 = ∞
    h4 = ∞
    a[x][y] = 1 // hurkok elkerülése végett
    ha a[x-1][y] == 0 akkor
        h1 = oszd_meg_és_uralkodj_egér(a,xs,ys,x-1,y)
    vége ha
    ha a[x][y+1] == 0 akkor
        h2 = oszd_meg_és_uralkodj_egér(a,xs,ys,x,y+1)
    vége ha
    ha a[x+1][y] == 0 akkor
        h3 = oszd_meg_és_uralkodj_egér(a,xs,ys,x+1,y)
    vége ha
    ha a[x][y-1] == 0 akkor
        h4 = oszd_meg_és_uralkodj_egér(a,xs,ys,x,y-1)
    vége ha
    a[x][y] = 0 // visszaállítjuk a szabad utat
    return minimum(h1,h2,h3,h4) +1
vége oszd_meg_és_uralkodj_egér
```

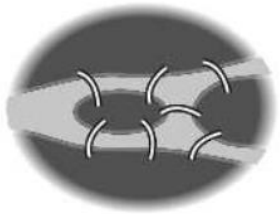
Sík-backtrack

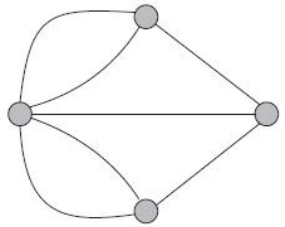
- **Labirintus:** Adott egy labirintus az a[1..n][1..m] bináris tömbben (1-essel kódoljuk a falat, 0-val a folyosót). Adott még egy személy pozíciója a labirintusban (az x koordináta a sorindex, az y az oszlopindex). Generáljuk az összes hurokmentes utat, amelyen a személy kijuthat a labirintusból.
- **Fénykép**: Adott egy fekete-fehér kép, amelyet az a[1..n][1..m] bináris mátrixban tároltunk (a 0 a fehér tartományokat, az 1 a fekete foltokat /tárgyakat/ ábrázolja). Állapítsuk meg a képen található tárgyak számát!

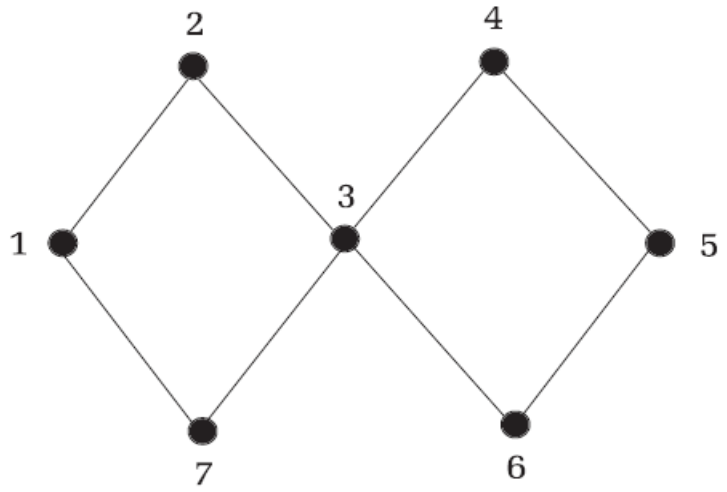
Euler gráfok











12.2. ábra. *A königsbergi hidak modellezése gráfokkal*

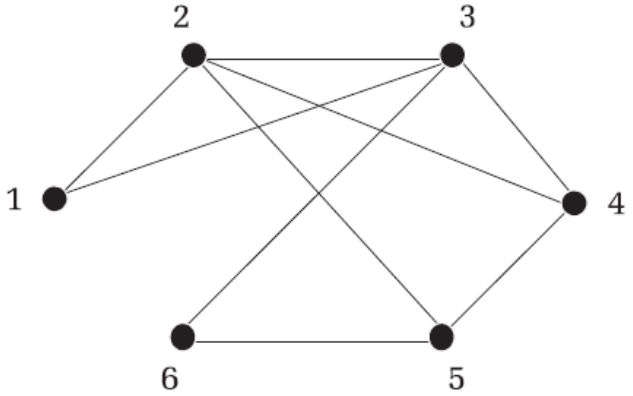
A *Hamilton-út* a gráf minden pontját tartalmazza (egyszer és csakis egyszer). A *Hamilton-kör* a gráf minden pontját pontosan egyszer tartalmazza. Ha egy gráf tartalmaz Hamilton-kört, akkor *Hamilton-gráf*nak nevezzük.

Egy gráf akkor és csak akkor Euler-gráf, ha összefüggő és bármely csúcsának a foka páros.

1. Ellenőrizd, hogy egy gráf euler-e!
2. Találj egy zárt euler vonalat!
3. Generáld az összes zárt euler vonalat!

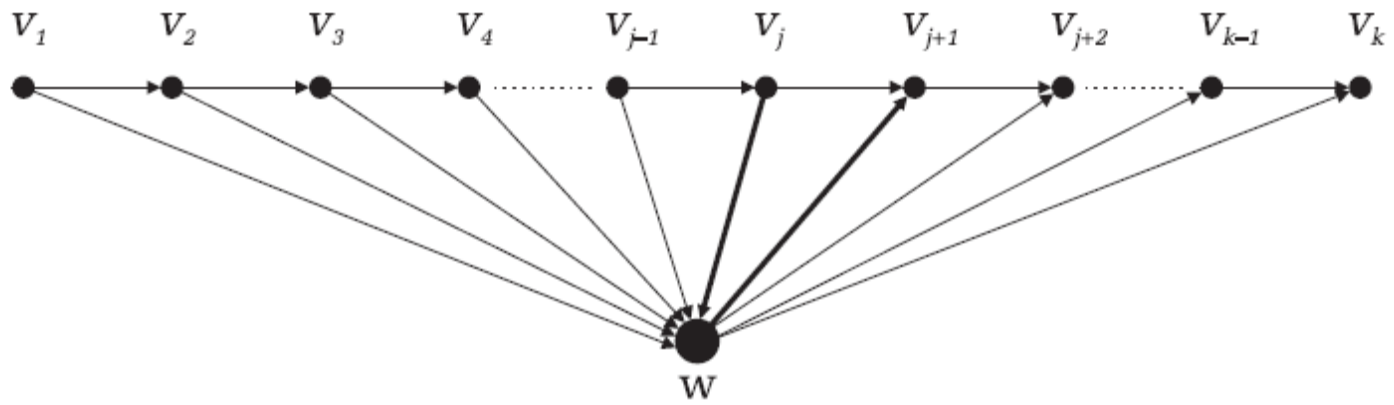


Hamilton gráfok



A *nyílt Euler-vonal* a gráf minden élét tartalmazza (egyszer és csakis egyszer). A *zárt Euler-vonal* kezdőpontja megegyezik a végpontjával és a gráf minden élét pontosan egyszer tartalmazza. Ha egy gráf tartalmaz zárt Euler-vonalat, akkor *Euler-gráf*nak nevezzük (1.6–1.9. ábra).

(**Rédei László**, 1934). Minden turné gráfban (olyan irányított gráf, amelyben bármely pontpár között vagy oda, vagy vissza él van) van irányított Hamilton-út.

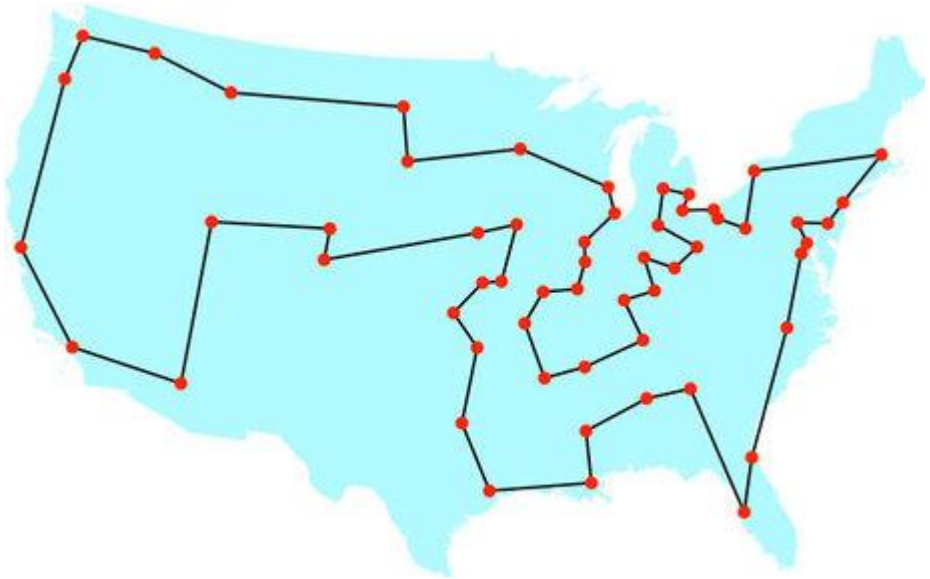


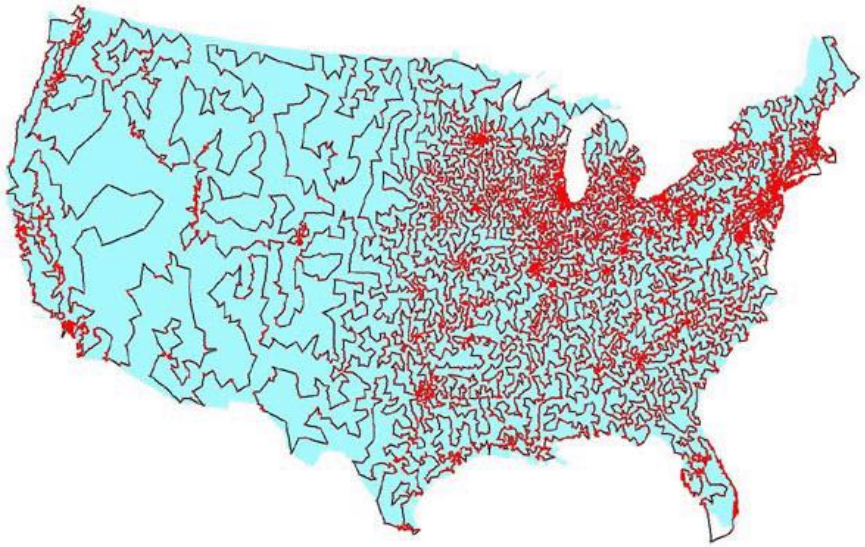
Utazó ügynök probléma (TSP)

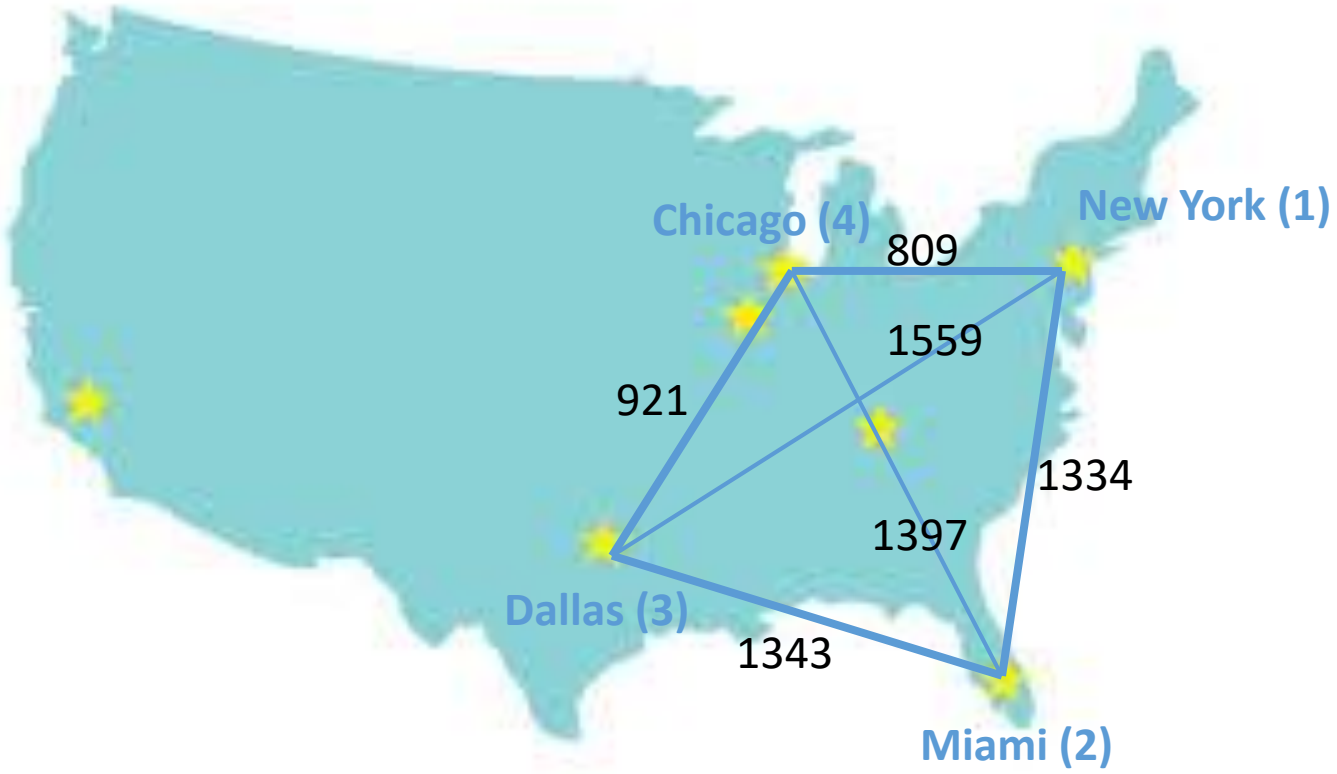


A feladat

- Adva van *n* város, illetve az útiköltség bármely két város között, keressük a legolcsóbb utat egy adott városból indulva, amely minden várost pontosan egyszer érint, majd a kiindulási városba ér vissza.

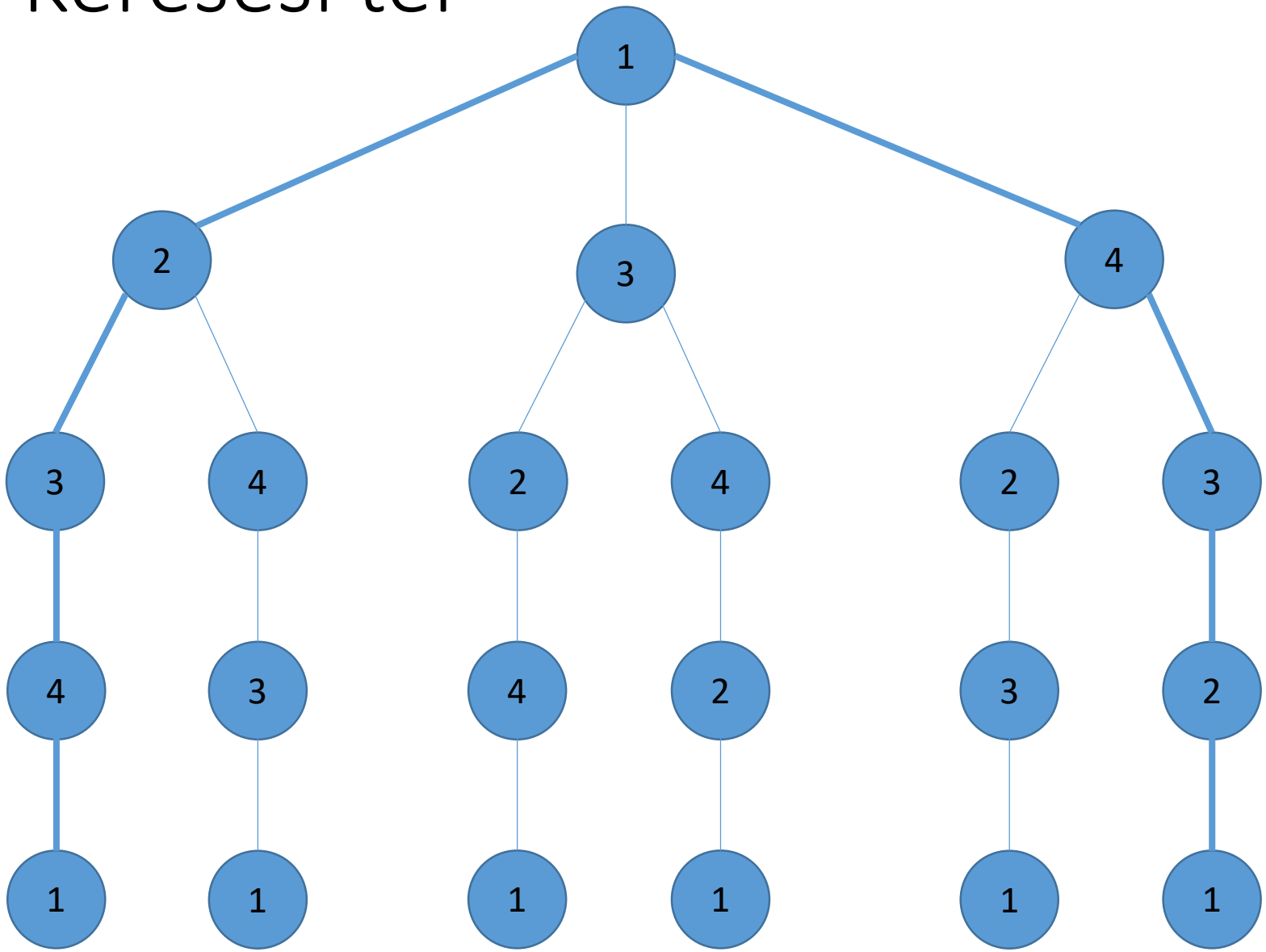






	New York	Miami	Dallas	Chicago
1. New York	—	1334	1559	809
2. Miami	1334	—	1343	1397
3. Dallas	1559	1343	—	921
4. Chicago	809	1397	921	—

Keresési tér



Számítási nehézség

- A legkézenfekvőbb megoldás az összes permutáció végignézése
  - ((n-1)!)/2 út közül kell választanunk egyet, a legrövidebbet
- Dinamikus programozás ($O(n^2 2^n)$)
  - 1962: Bellman / Held–Karp
- **Teljes megoldású algoritmus**
  - 1954: 49 amerikai város
  - 1977: 120 város Németország környékéről
  - 1987: 2392 pont
  - 2004: Svédország 24 978 városa
  - 2006: 85 900 (CONCORDE algoritmus)
    - branch-and-bound módszerrel

?

1,**2**,**3**,**4**,**1**
1,**2**,**4**,**3**,**1**
1,**3**,**2**,**4**,**1**
1,**3**,**4**,**2**,**1**
1,**4**,**2**,**3**,**1**
1,**4**,**3**,**2**,**1**

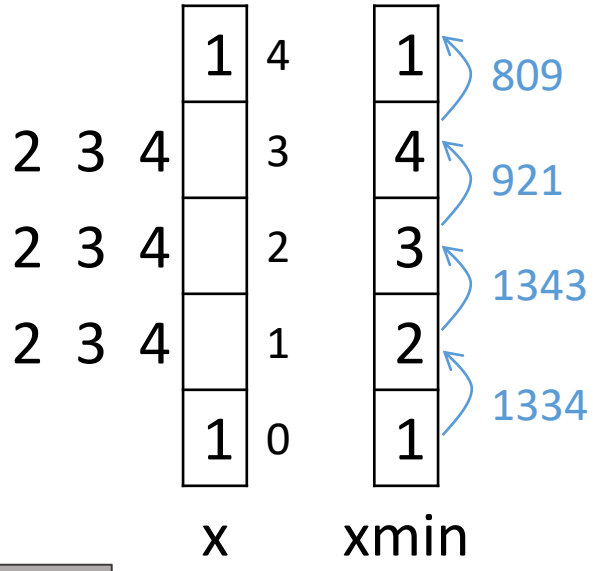


Generáljuk a permutációkat (Backtracking)

```
BTp(x[],n,k)
   minden x[k] = 2,n végezd
      ha ígéretes(x,k) akkor
         ha k < n-1 akkor
            BTp(x,n,k+1)
         különben
            frissít_min_ut(xmin,x,n,d)
         vége ha
      vége ha
   vége minden
vége BTp
```

```
ígéretes(x[],k)
   minden i = 1,k-1 végezd
      ha x[i] == x[k] akkor
         return HAMIS
      vége ha
   vége minden
   return IGAZ
vége ígéretes
```

Minimum keresés a generált utak között



d	1	2	3	4
1	0	1334	1559	809
2	1334	0	1343	1397
3	1559	1343	0	921
4	809	1397	921	0