Algoritmusok felülnézetből

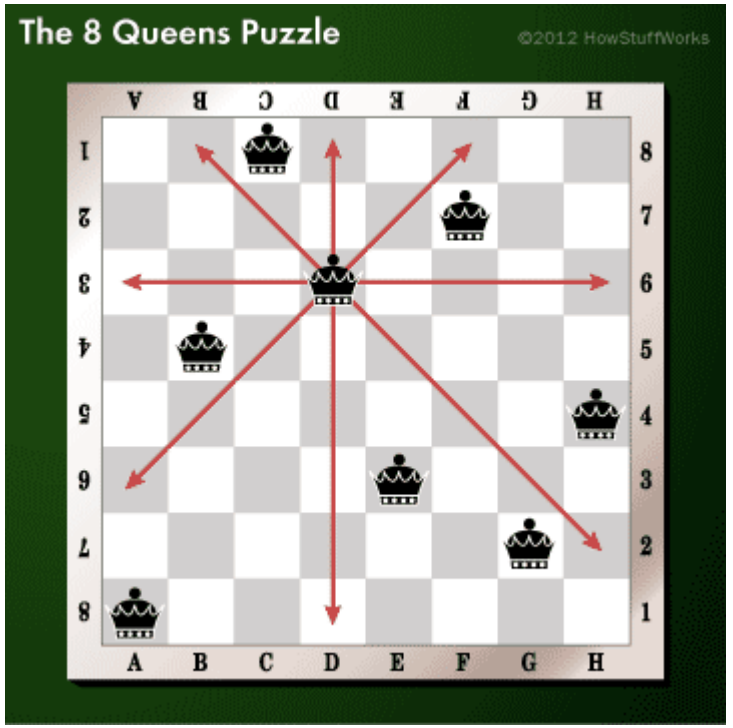
2. ELŐADÁS

Sapientia-EMTE

2015-16

Nyolc-királynő probléma

- 1848: SAKK-feladvány, Max Bezzel
- 1850: Illustrirte zeitung, Franz Nauck
  - Gauss: 76, 72
  - Nauck: 62, **92**
- 1972: Dijkstra
  - BACKTRACKING

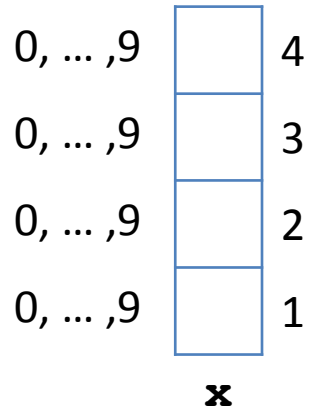


Kerékpár-zár probléma

- *4-pozíciós kerékpár-zár:*
  - *Mindegyik pozícióba valamelyik számjegy választható ki:* 0, 1, …, 9. *A feladatunk az, hogy generáljuk az összes lehetséges 4 számjegyű kódvektort:* (0,0,0,0), (0,0,0,1), (0,0,0,2), …, (9,9,9,9).
  - Érdekel az összes (v_1,v_2,v_3,v_4) alakú vektor, ahol $v_i \in \{0, 1, \ldots, 9\}$, i=1..4.

Kerékpár-zár probléma

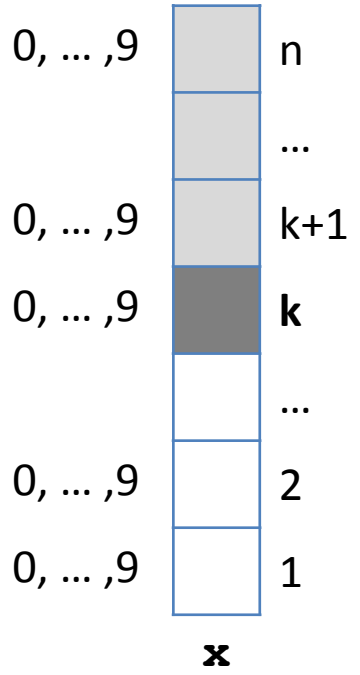
```
minden x[1] = 0,9 végezd
   minden x[2] = 0,9 végezd
      minden x[3] = 0,9 végezd
         minden x[4] = 0,9 végezd
            ki: x[1..4]  //10^4-szer
         vége minden
      vége minden
   vége minden
vége minden
```



Általános kerékpár-zár probléma

- *Generáljuk az összes n számjegyű kódvektort*

```
BTd(x[],n,k)
   minden x[k] = 0,9 végezd
      ha k < n akkor
         BTd(x,n,k+1)
      különben
         kiír(x,n)
      vége ha
   vége minden
vége BTd
```

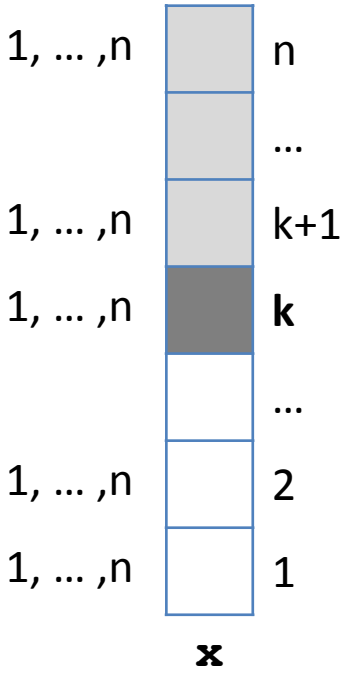


- **BTd(x,n,k)** : generálja, az x[k..n] tömbszakaszon, az összes (v_k, v_{k+1}, …, v_n) kódszakaszt.
- **BTd(x,n,1)** : generálja az összes n-hosszú kódvektort.

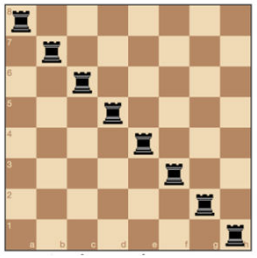
Descartes szorzat elemei

- *Generáljuk az {1,…,n} halmaz n-szeres descartes szorzatának elemeit*

```
BTd(x[],n,k)
   minden x[k] = 1,n végezd
      ha k < n akkor
         BTd(x,n,k+1)
      különben
         kiír(x,n)
      vége ha
   vége minden
vége BTd
```



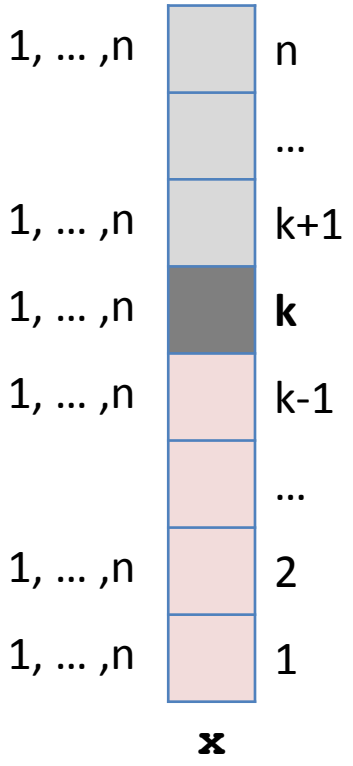
- **BTd(x,n,k)** : generálja, az x[k..n] tömbszakaszon, az összes (v_k, v_{k+1}, …, v_n) kódszakaszt.
- **BTd(x,n,1)** : generálja az összes n-hosszú kódvektort.



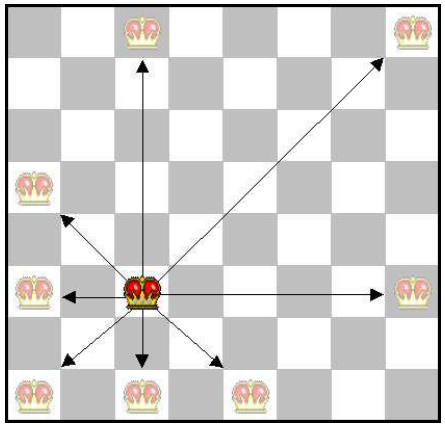
Permutációk

- *Generáljuk az {1,…,n} halmaz összes permutációit*

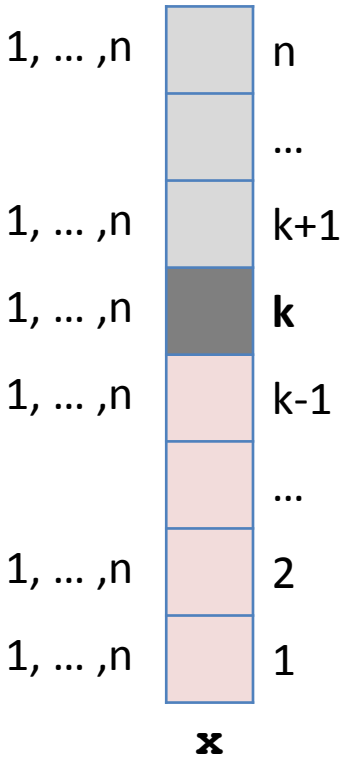
```
BTp(x[],n,k)
   minden x[k] = 1,n végezd
      ha ígéretes(x,k) akkor
         ha k < n akkor
            BTp(x,n,k+1)
         különben
            kiír(x,n)
         vége ha
      vége ha
   vége minden
vége BTp
```




```
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
```



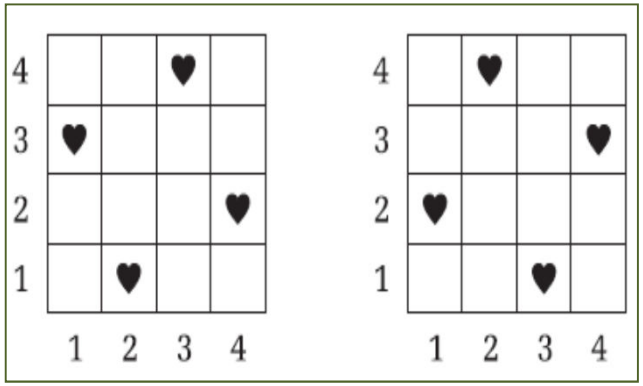
N-királynő probléma

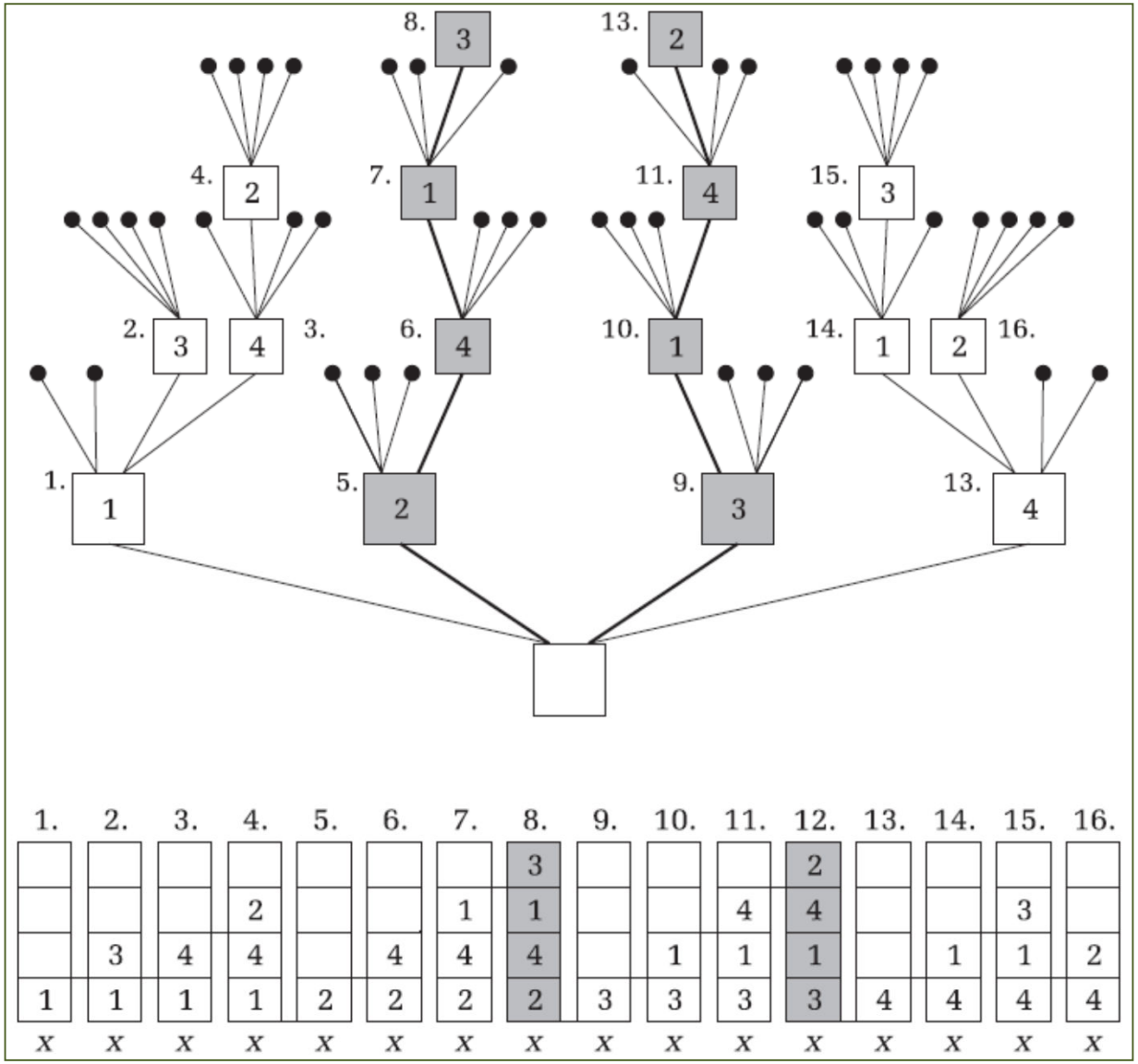



```
BTkirálynő(x[],n,k)
   minden x[k] = 1,n végezd
      ha ígéretes_királynő(x,k) akkor
         ha k < n akkor
            BTkirálynő(x,n,k+1)
         különben
            kiír(x,n)
         vége ha
      vége ha
   vége minden
vége BTkirálynő
```

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

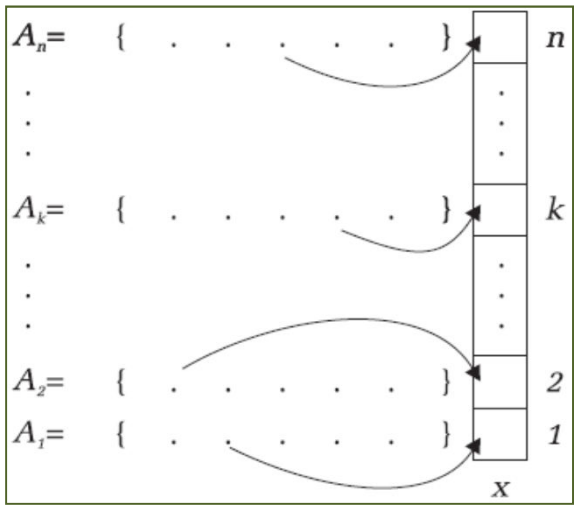
4-királynő probléma

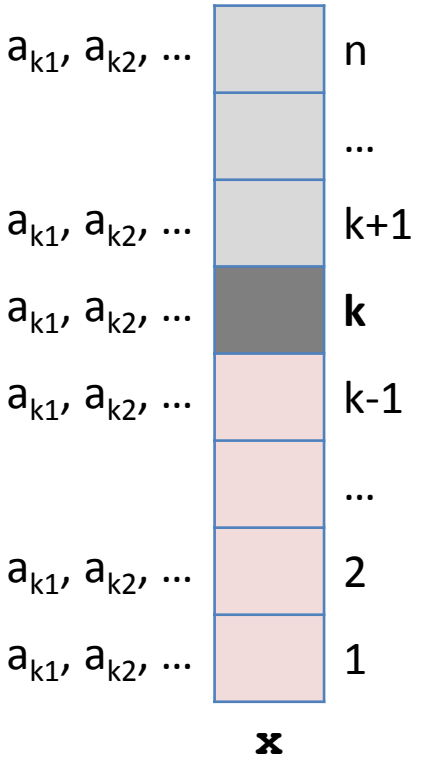




Általános BACKTRACKING modell

```
BT(x[],n,k)
   minden x[k] = ak1, ak2, … végezd
      ha ígéretes(x,n,k) akkor
         ha megoldás(x,n,k) akkor
            kiír(x,n,k)
         különben
            BT(x,n,k+1)
         vége ha
      vége ha
   vége minden
vége BT
```



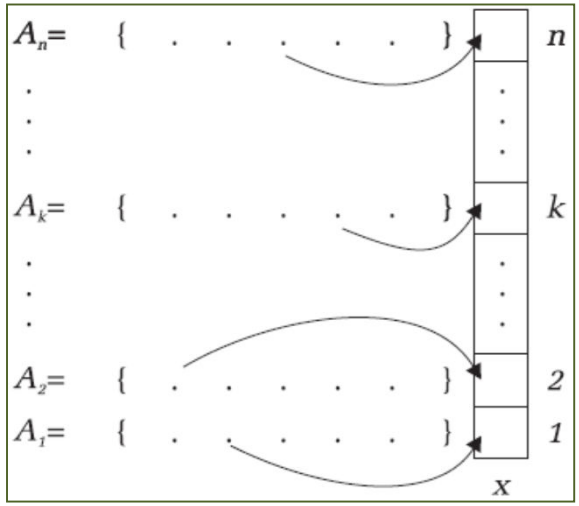


BACKTRACKING feladatok/stratégia

- SAJÁTOSSÁG: a feladat *összes* megoldásában érdekeltek vagyunk
- Optimalizálási feladatok: generáljuk az összes potenciális megoldást, és optimumot keressük ezek között (ágyúval a veréb után)
  - a `kiír` eljárást lecseréljük egy min/max keresőre
  - az optimális megoldást utólag írjuk ki
- Gyakran „nyers erő” megközelítésnek számít

Recept BACKTRACKING feladatokhoz

1. Hogyan kódolhatók a feladat megoldásai vektorokként?
2. Igyekszünk felállítani az ábrán bemutatott modellt.



3. Megírjuk az `ígéretes` függvényt, amely az algoritmus kulcselemének tekintendő!
4. Megírjuk a `megoldás` függvény!
5. Megírjuk a `kiír` eljárást!

Lásd a JEGYZETet

2.1. Halmazműveletek: Generáljuk az $\{1, 2, \ldots, n\}$ halmaz
1. p-szeres Descartes-szorzatának elemeit;
2. összes permutációit;
3. p-edrendű variációit;
4. p-edrendű kombinációit;
5. összes részhalmazát;
6. összes partícióját.

2.5. Szuperprímek: Generáljuk az összes n számjegyű szuperprímet. Egy természetes számot szuperprímnek nevezünk, ha prím és a számjegyei jobbról balra sorrendben történő egyenkénti levágásával nyert számok (tekintsük ezeket a szám prefixeinek) is mind prímek. Például 239 szuper prím, mert 239, 23 és 2 mind prímek.

2.7. Békák: Legyen egy $s[1 \ldots 2n+1]$ karakterlánc, amelyben az első n elem 0, az $(n+1)$-edik szóköz, a többi pedig 1-es. Generáljuk az összes lehetőséget, ahogyan a nullások (fehér békák) helyet cserélhetnek az egyesekkel (fekete békák), a következő feltételek mellett:
- a fehér békák csak balról jobbra, a feketék pedig csak jobbról balra szökdöshetnek.
- egy béka akkor haladhat előre, ha előtte szóköz van, vagy ha az előtte lévő béka előtt szóköz van, ez utóbbi esetben átugorhatja az előtte lévő békát.

2.9. 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!