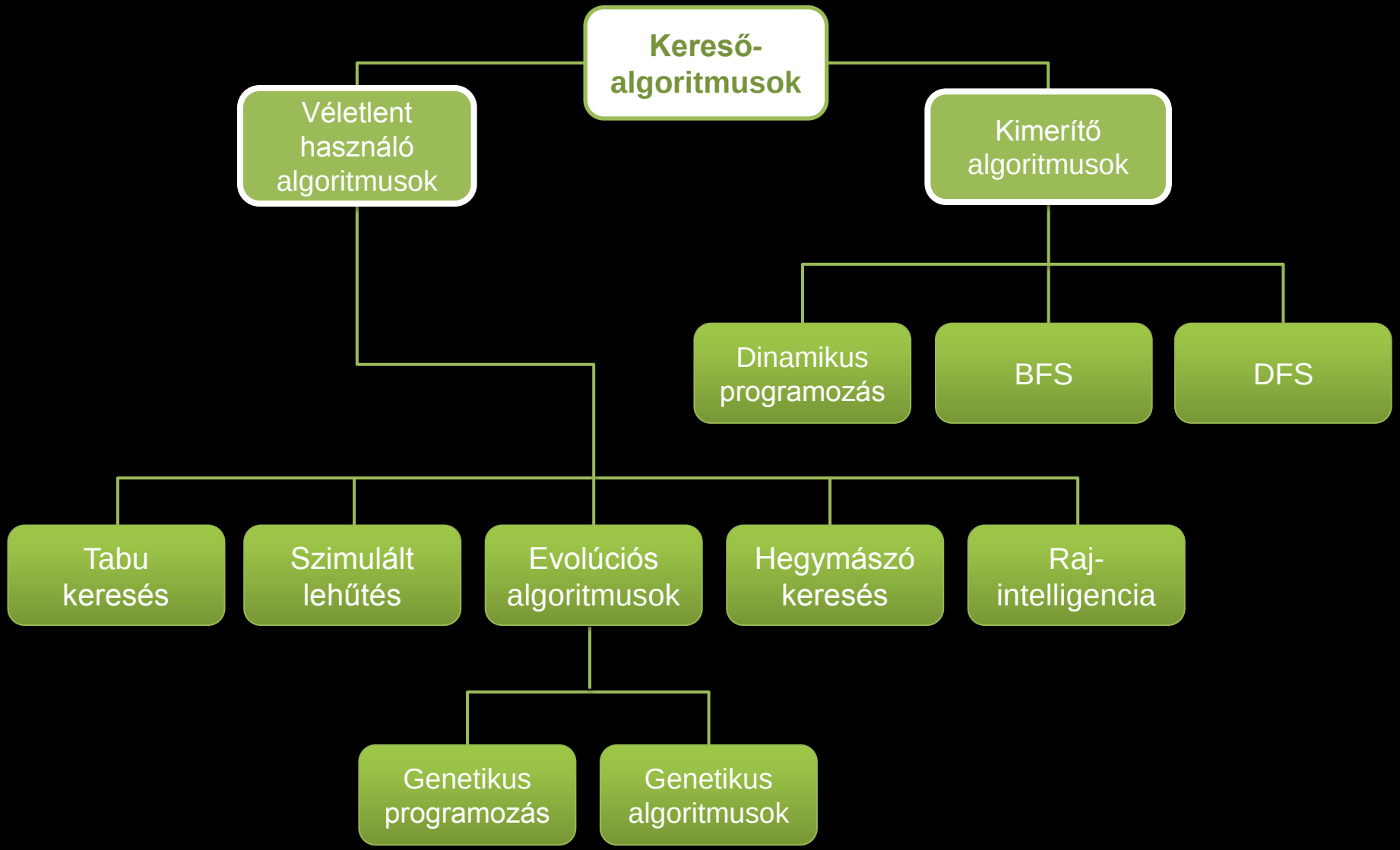
Genetikus algoritmusok

Zsolnai Károly - BME CS

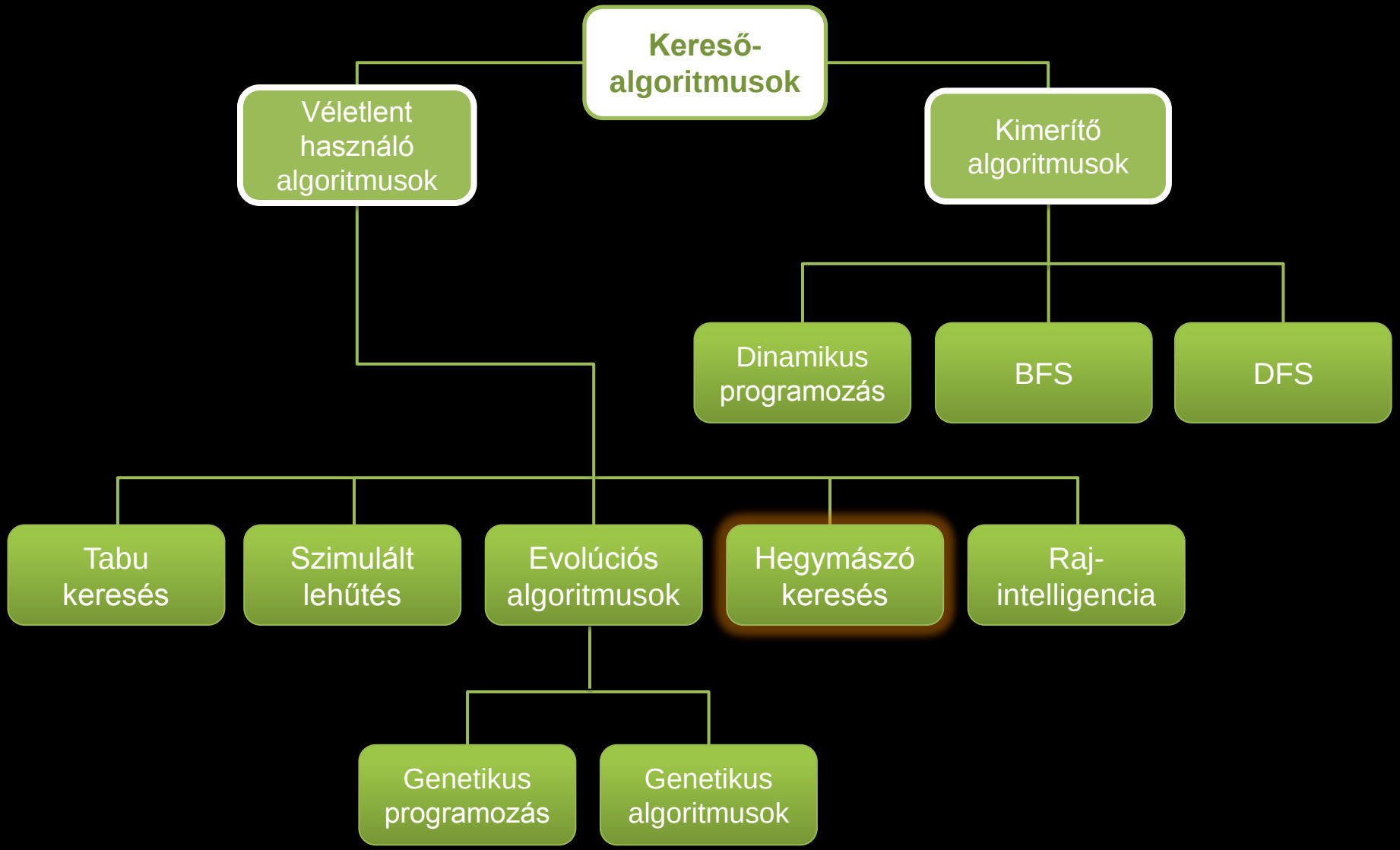
zsolnai@cs.bme.hu

Keresőalgoritmusok osztályai

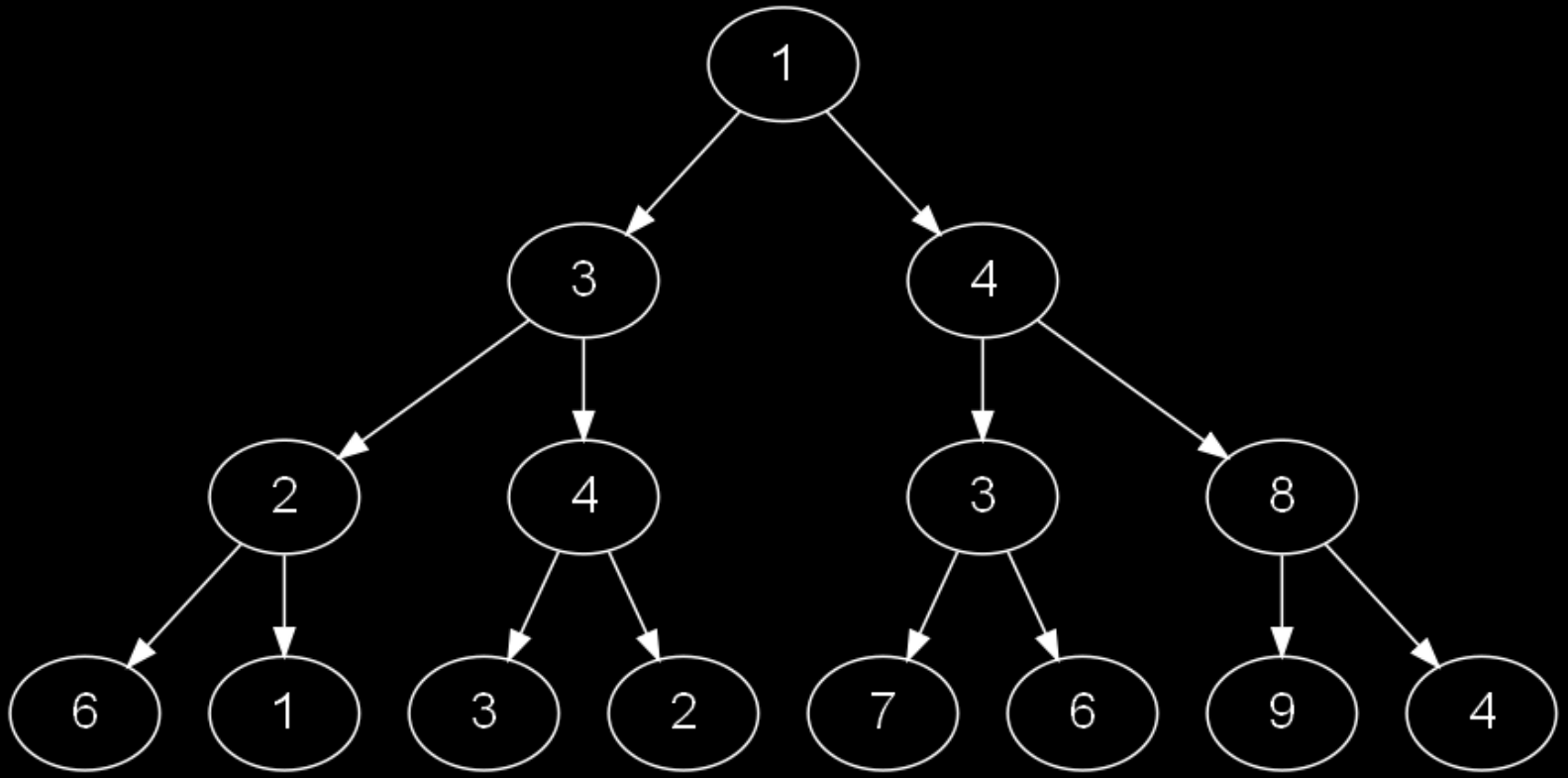


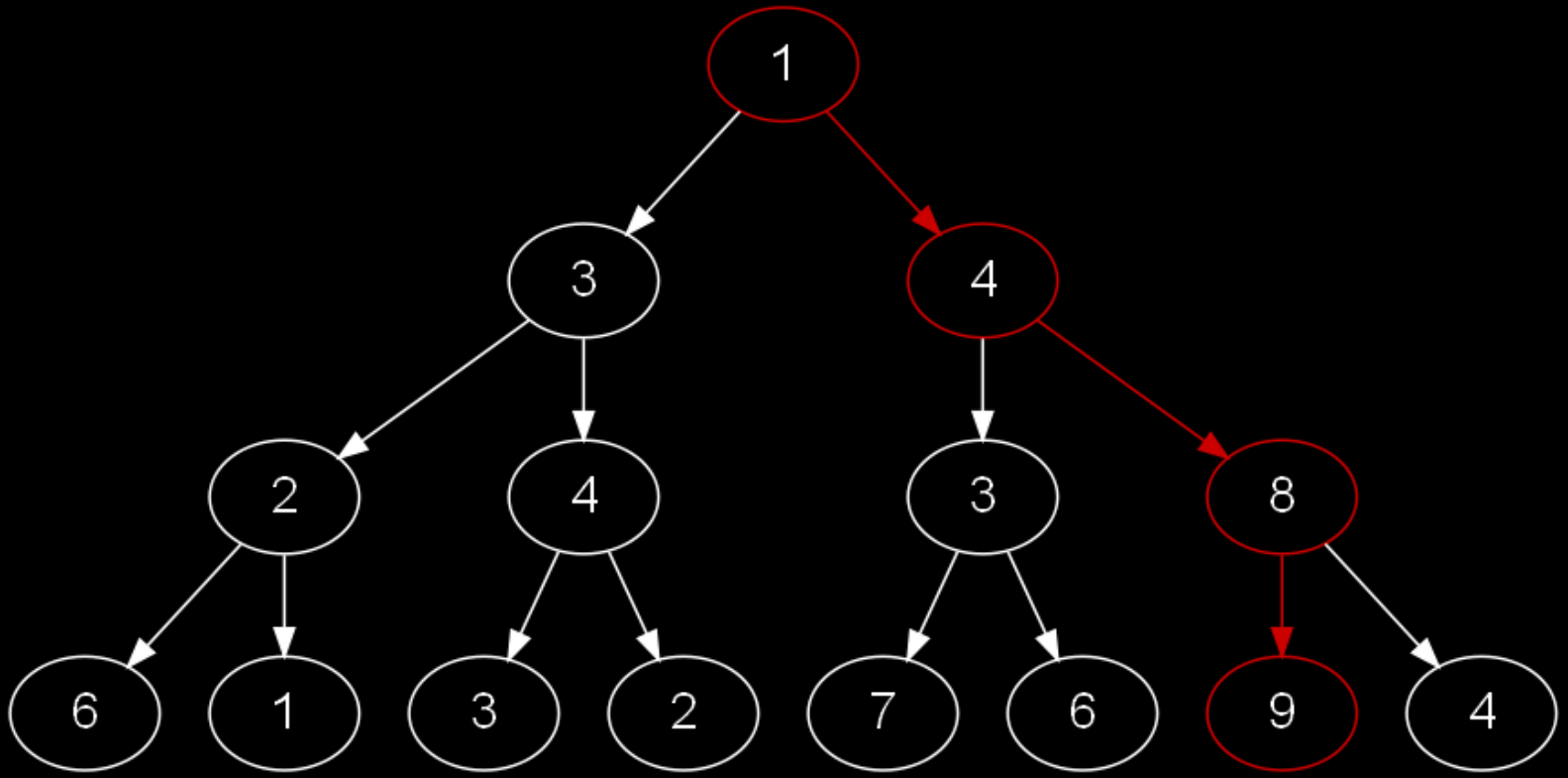
(a keresőalgoritmusok egy részhalmazát mutattuk be)

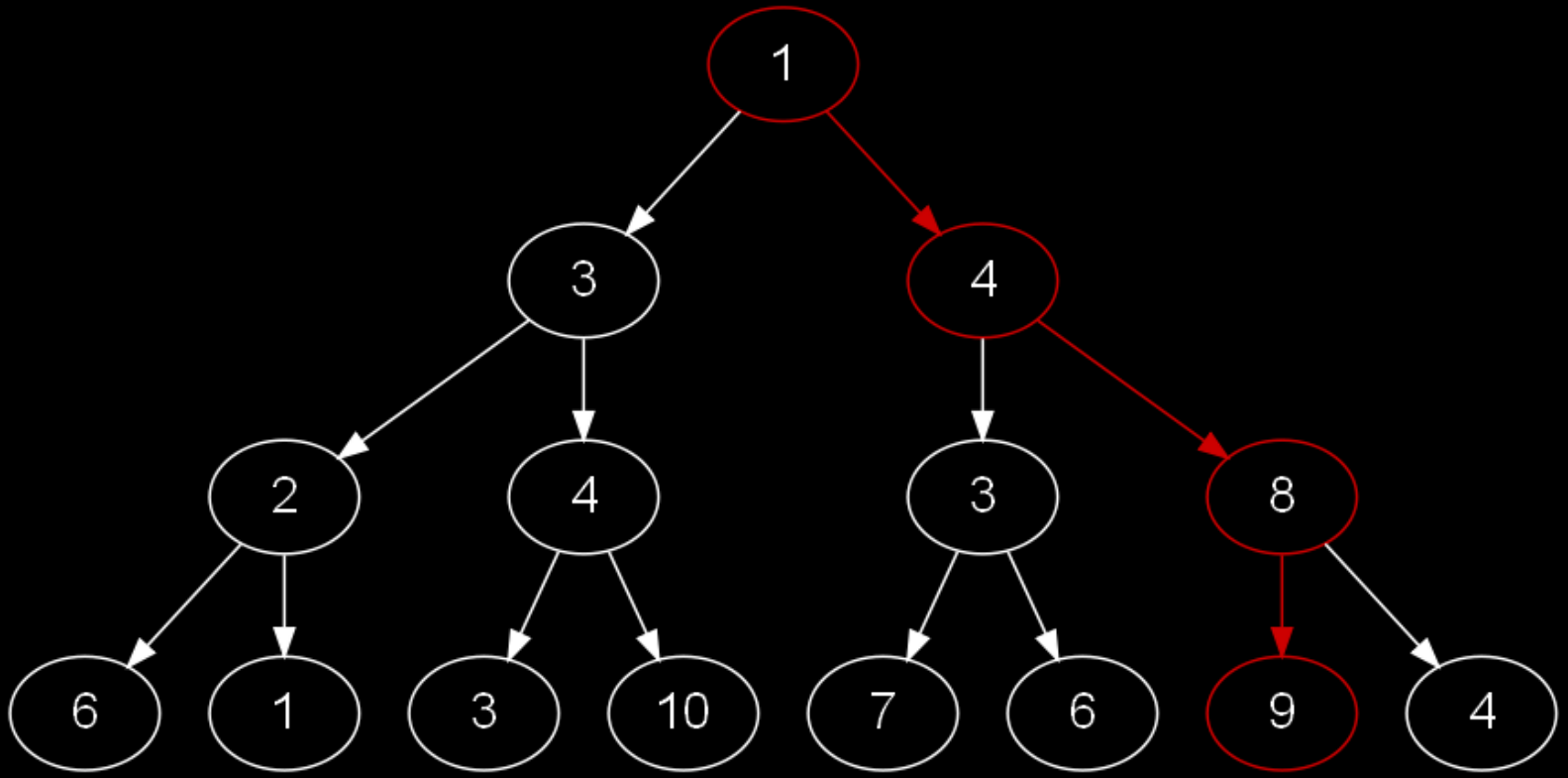
Keresőalgoritmusok osztályai

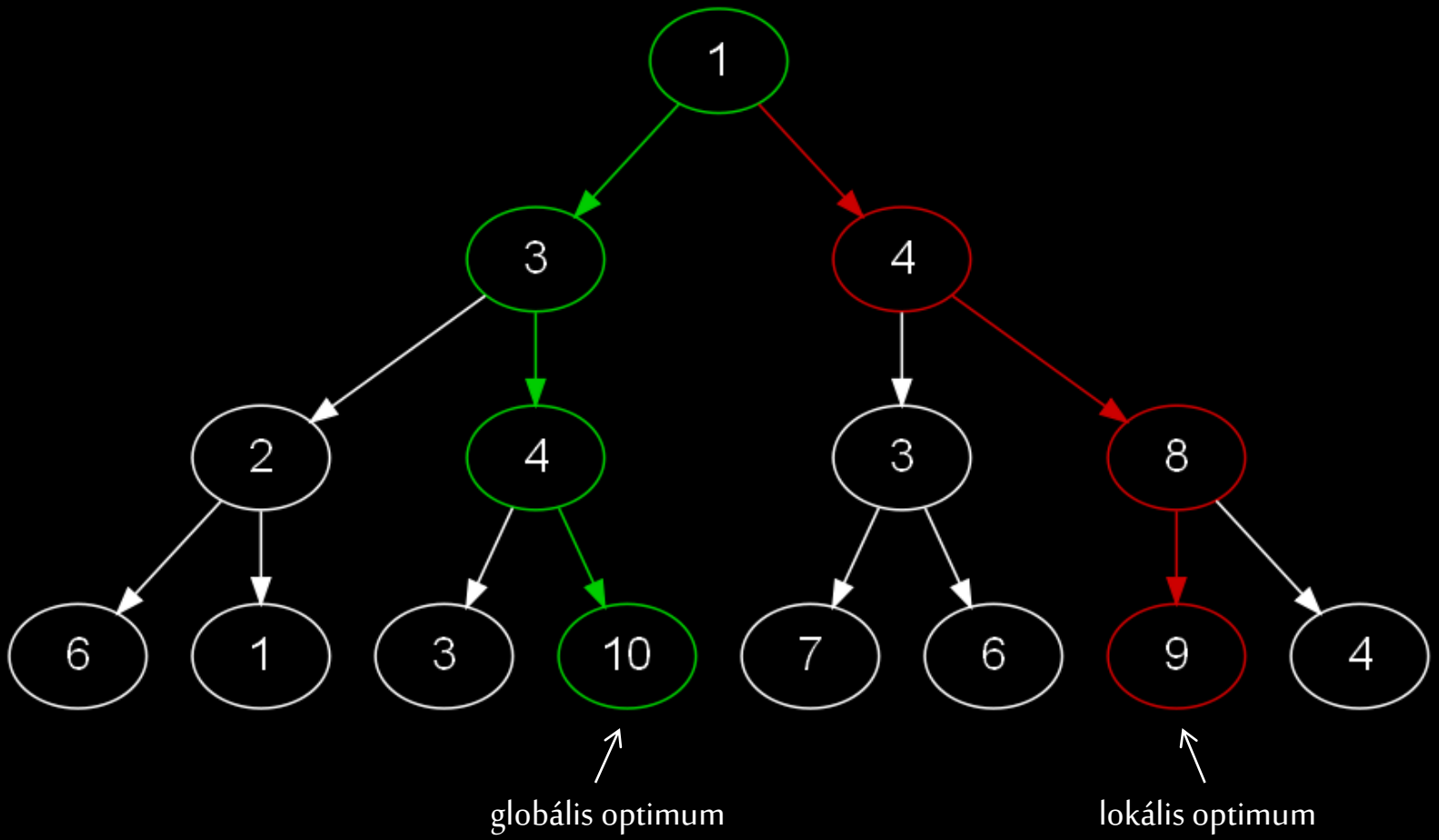


(a keresőalgoritmusok egy részhalmazát mutattuk be)

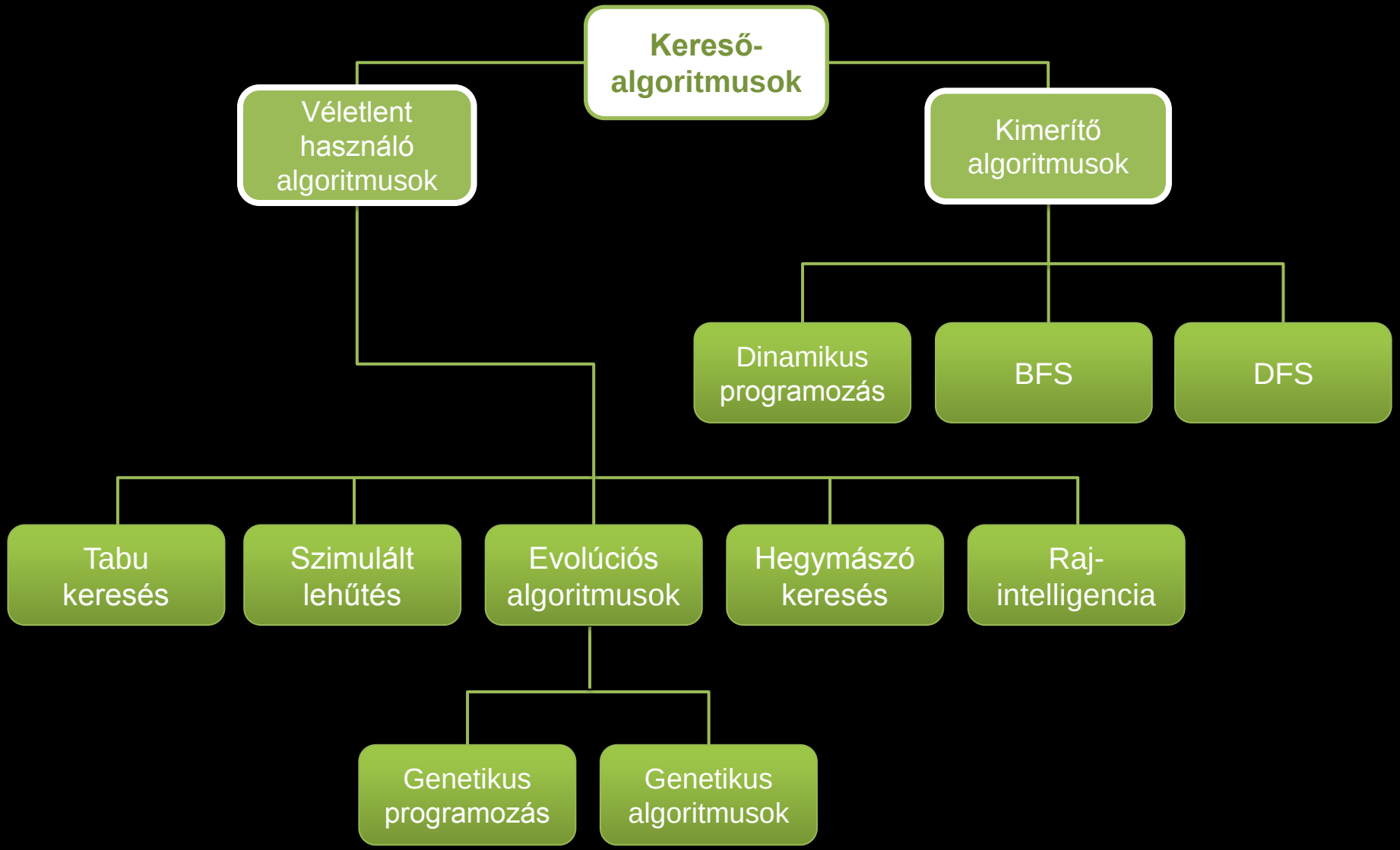

1
3
4
2
4
3
8
6
1
3
2
7
6
9
4


1
3
4
2
4
3
8
6
1
3
2
7
6
9
4


1
3
4
2
4
3
8
6
1
3
10
7
6
9
4

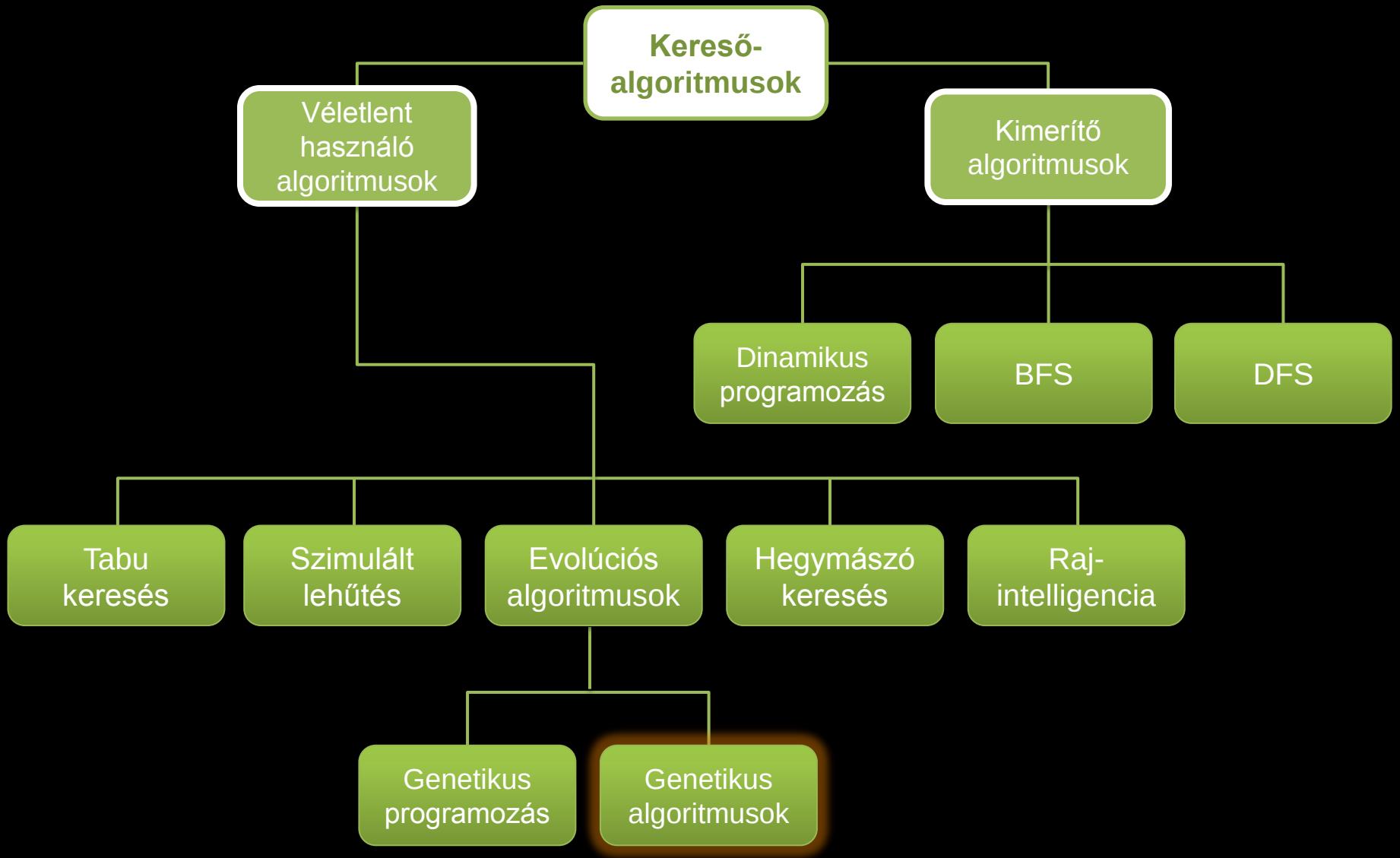

1
3
4
2
4
3
8
6
1
3
10
7
6
9
4
globális optimum
lokális optimum

Keresőalgoritmusok osztályai



(a keresőalgoritmusok egy részhalmazát mutattuk be)

Keresőalgoritmusok osztályai



(a keresőalgoritmusok egy részhalmazát mutattuk be)

A genetikus algoritmus a lehetséges megoldások egy populációját hozza létre, amelyekhez lépésenként új egyedeket ad, illetve a már meglévő egyedekre szelekciós, rekombinációs és mutációs operátorokat alkalmaz.

KERESÉSI ÉS OPTIMALIZÁCIÓS PROBLÉMÁK, AHOL:

- Sokdimenziós keresési terek jellemzők és a fontosabb változók közötti összefüggés ismeretlen
- A tradicionális megoldások elfogadhatatlan futási időt adnak (NP, NPC)
- A keresési tér nehezen, vagy nem is szűkíthető
- Egy megoldás jósága gyorsan ellenőrizhető, de egy jó megoldás előállítása nehézkes

GÉN

Definíció a DNS alapvető szerveződési egysége, valamilyen tulajdonságot kódol

Matematikai reprezentáció $x \in \mathbb{R}, \mathbb{B}, \mathbb{Z}, \ldots$

ALLÉL

Definíció egy-egy géntípus lehetséges megjelenési formáinak halmaza

Matematikai reprezentáció $\text{az } x \text{ által felvehető értékek halmaza}$

KROMOSZÓMA

Definíció a gének egy teljesnek tekinthető gyűjteménye

Matematikai reprezentáció $\vec{c} = (x_1, x_2, \ldots, x_n),\ x_1, \ldots, x_n \in \mathbb{R}, \mathbb{B}, \mathbb{Z}, \ldots$

$$\vec{c} = (\uparrow, \rightarrow, \downarrow, \leftarrow)$$

GENOTÍPUS

Definíció a kromoszóma néhány kiválasztott génje

Matematikai reprezentáció $\vec{g} = (\underbrace{x_1, x_2, \ldots, x_i}_{i}),\ x_1, x_2, \ldots, x_i \in \vec{c}, i < n$

FENOTÍPUS

Definíció a genotípus általánosítása

Matematikai reprezentáció $f : \vec{g} \rightarrow \text{tulajdonság}$

$$f(0, 1, 1, 0, 1, 1) = \text{”barna”}$$

$$f(\uparrow, \rightarrow) = \text{”jobbra fordul”}$$

POPULÁCIÓ

Definíció egy adott pillanatban létező potenciális megoldások gyűjteménye

Matematikai reprezentáció $\vec{p} = \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n\}$

EVOLÚCIÓS OPERÁTOROK

Definíció a természetben megfigyelhető szelekció, rekombináció és mutáció műveleteinek megfelelői

FITNESS

Definíció egy adott kromoszómával definiált egyedek túlélési esélye

Matematikai reprezentáció $f : \vec{c} \rightarrow \mathbb{R}$

EVOLÚCIÓ

Definíció a fenti operátorok iteratív alkalmazása a megoldáshalmaz javítása érdekében

Matematikai reprezentáció $f(\vec{c}_1) > f(\vec{c}_2) \Rightarrow P_s(\vec{c}_1) > P_s(\vec{c}_2)$

BINÁRIS KÓDOLÁS

$$\vec{c}_1 = [1010110100101010011]$$
$$\vec{c}_2 = [1001111000110011 0]$$

BINÁRIS KÓDOLÁS

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

0-1 Hátizsák probléma

Adott súlyú és értékű elemeket kell véges kapacitású hátizsákban elhelyezni, cél az összérték maximalizálása.

Kódolás $$\vec{x}_i = \begin{cases} 1, & \text{ha } benne\ van, \\ 0, & \text{ha } nincs\ benne. \end{cases}$$

BINÁRIS KÓDOLÁS

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

0-1 Hátizsák probléma

Adott súlyú és értékű elemeket kell véges kapacitású hátizsákban elhelyezni, cél az összérték maximalizálása.

Kódolás

$$\vec{x}_i = \begin{cases} 1, & \text{ha } benne\ van, \\ 0, & \text{ha } nincs\ benne. \end{cases}$$

Fitness függvény

$$f(\vec{c}) = \sum_{i=0}^{n} x_i\, v_i$$

Feltétel

$$\sum_{i=0}^{n} x_i\, w_i \leq w_l$$

PERMUTÁCIÓS KÓDOLÁS

$$\vec{c}_1 = [4\ \ 8\ \ 5\ \ 1\ \ 3\ \ 6\ \ 9\ \ 2\ \ 7]$$
$$\vec{c}_2 = [2\ \ 6\ \ 3\ \ 9\ \ 5\ \ 4\ \ 1\ \ 7\ \ 8]$$

PERMUTÁCIÓS KÓDOLÁS

$$\vec{c}_1 = [4\ 8\ 5\ 1\ 3\ 6\ 9\ 2\ 7]$$
$$\vec{c}_2 = [2\ 6\ 3\ 9\ 5\ 4\ 1\ 7\ 8]$$

Utazóügynök probléma

Egy irányított gráfban keresünk minimális összsúlyú Hamilton-kört.

Kódolás $x_i = i.\ \text{meglátogatott város indexe}$

PERMUTÁCIÓS KÓDOLÁS

$$\vec{c}_1 = [4\ 8\ 5\ 1\ 3\ 6\ 9\ 2\ 7]$$
$$\vec{c}_2 = [2\ 6\ 3\ 9\ 5\ 4\ 1\ 7\ 8]$$

Utazóügynök probléma

Egy irányított gráfban keresünk minimális összsúlyú Hamilton-kört.

Kódolás $x_i = i.\ \text{meglátogatott város indexe}$

Fitness függvény $f(x_i) = \frac{1}{\sum_{i=0}^{n-1} w(v_i, v_{i+1})}$

Feltétel $w(n) = w(0)$

VALÓS ÉRTÉK KÓDOLÁS

$$\vec{c}_1 = [13.5 \;\; 27.9 \;\; 1.22 \;\; 4.3 \;\; 1.9]$$
$$\vec{c}_2 = [E \; G \; F \; C \; A \; O \; L \; T \; D \; B]$$
$$\vec{c}_3 = [60° \uparrow \; -120° \uparrow \; 45° \uparrow]$$

VALÓS ÉRTÉK KÓDOLÁS

$$\vec{c}_1 = [13.5 \;\; 27.9 \;\; 1.22 \;\; 4.3 \;\; 1.9]$$
$$\vec{c}_2 = [E \; G \; F \; C \; A \; O \; L \; T \; D \; B]$$
$$\vec{c}_3 = [60° \uparrow \; -120° \uparrow \; 45° \uparrow]$$

Labirintus

Egy útvesztő valamely kezdőpontjától kell minél messzebb eljutni.

Kódolás $\vec{c}_3$-ban x_i az i. cselekvés típusa

VALÓS ÉRTÉK KÓDOLÁS

$$\vec{c}_1 = [13.5 \;\; 27.9 \;\; 1.22 \;\; 4.3 \;\; 1.9]$$
$$\vec{c}_2 = [E \; G \; F \; C \; A \; O \; L \; T \; D \; B]$$
$$\vec{c}_3 = [60° \uparrow \; -120° \uparrow \; 45° \uparrow]$$

Labirintus

Egy útvesztő valamely kezdőpontjától kell minél messzebb eljutni.

Kódolás $\vec{c}_3$-ban x_i az i. cselekvés típusa

Fitness függvény $f(x_i) = \sqrt{(s_0 - t_0)^2 + (s_1 - t_1)^2}$

Azaz a standard euklideszi távolság a starttól, vagy tetszőleges egyéb norma

KERESZTEZÉS EGY PONTON

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110110]$$

$$\vec{c}_3 = [0000000000000000]$$

KERESZTEZÉS EGY PONTON

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

$$\vec{c}_3 = [0000000000000000]$$

$$k \in (0, n-1)$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i < k \\ \vec{c}_2(i), & \text{ha } i \geq k \end{cases}$$

KERESZTEZÉS EGY PONTON

$$\vec{c}_1 = [1010110|1001010011]$$
$$\vec{c}_2 = [1001111|0001100110]$$

$$\vec{c}_3 = [0000000|0000000000]$$

$$k \in (0, n-1)$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i < k \\ \vec{c}_2(i), & \text{ha } i \geq k \end{cases}$$

KERESZTEZÉS EGY PONTON

$$\vec{c}_1 = [1010110|1001010011]$$
$$\vec{c}_2 = [1001111|0001100110]$$

$$\vec{c}_3 = [1010110|0001100110]$$

$$k \in (0, n-1)$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i < k \\ \vec{c}_2(i), & \text{ha } i \geq k \end{cases}$$

KERESZTEZÉS EGY PONTON

$$\vec{c}_1 = [1010110|1001010011]$$
$$\vec{c}_2 = [1001111|0001100110]$$

$$\vec{c}_3 = [1010110|0001100110]$$

$$k \in (0, n-1)$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i < k \\ \vec{c}_2(i), & \text{ha } i \geq k \end{cases}$$

$$\vec{c}_4(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i \geq k \\ \vec{c}_2(i), & \text{ha } i < k \end{cases}$$

KERESZTEZÉS EGY PONTON

$$\vec{c}_1 = [1010110|1001010011]$$
$$\vec{c}_2 = [1001111|0001100110]$$

$$\vec{c}_3 = [1010110|0001100110]$$
$$\vec{c}_4 = [1001111|1001010011]$$

$$k \in (0, n-1)$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i < k \\ \vec{c}_2(i), & \text{ha } i \geq k \end{cases}$$

$$\vec{c}_4(i) = \begin{cases} \vec{c}_1(i), & \text{ha } i \geq k \\ \vec{c}_2(i), & \text{ha } i < k \end{cases}$$

KERESZTEZÉS EGY PONTON

KERESZTEZÉS KÉT PONTON

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110110]$$

$$\vec{c}_3 = [0000000000000000]$$

KERESZTEZÉS KÉT PONTON

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

$$\vec{c}_3 = [0000000000000000]$$

$$k_1, k_2 \in (0, n-1),\ k_1 < k_2$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } (k_1 < i \vee k_2 \geq i) \\ \vec{c}_2(i), & \text{ha } (k_1 \geq i \wedge k_2 < i) \end{cases}$$

KERESZTEZÉS KÉT PONTON

$$\vec{c}_1 = [10101|101001010|011]$$
$$\vec{c}_2 = [10011|110001100|110]$$

$$\vec{c}_3 = [00000|000000000|000]$$

$$k_1, k_2 \in (0, n-1),\ k_1 < k_2$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } (k_1 < i \vee k_2 \geq i) \\ \vec{c}_2(i), & \text{ha } (k_1 \geq i \wedge k_2 < i) \end{cases}$$

KERESZTEZÉS KÉT PONTON

$$\vec{c}_1 = [10101|1010010 10|011]$$
$$\vec{c}_2 = [10011|110001100|110]$$

$$\vec{c}_3 = [10011|1010010 10|110]$$

$$k_1, k_2 \in (0, n-1),\ k_1 < k_2$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } (k_1 < i \vee k_2 \geq i) \\ \vec{c}_2(i), & \text{ha } (k_1 \geq i \wedge k_2 < i) \end{cases}$$

VÉLETLEN KERESZTEZÉS

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110110]$$

$$\vec{c}_3 = [0000000000000000]$$

VÉLETLEN KERESZTEZÉS

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

$$\vec{c}_3 = [0000000000000000]$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } p_i = 0 \\ \vec{c}_2(i), & \text{ha } p_i = 1 \end{cases}$$

VÉLETLEN KERESZTEZÉS

$$\vec{c}_1 = [1010110100101010011]$$
$$\vec{c}_2 = [1001111000110011 0]$$

$$\vec{c}_3 = [1011111100111 0010]$$

$$\vec{c}_3(i) = \begin{cases} \vec{c}_1(i), & \text{ha } p_i = 0 \\ \vec{c}_2(i), & \text{ha } p_i = 1 \end{cases}$$

ARITMETIKAI KERESZTEZÉS

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

$$\vec{c}_3 = [0000000000000000]$$

ARITMETIKAI KERESZTEZÉS

$$\vec{c}_1 = [1010110100101011]$$
$$\vec{c}_2 = [1001111000110011]$$

$$\vec{c}_3 = [0000000000000000]$$

$$\vec{c}_3 = \vec{c}_1 \oplus \vec{c}_2$$

ARITMETIKAI KERESZTEZÉS

$$\vec{c}_1 = [1010110100101 0011]$$
$$\vec{c}_2 = [1001111000110 0110]$$

$$\vec{c}_3 = [0011001100011 0101]$$

$$\vec{c}_3 = \vec{c}_1 \oplus \vec{c}_2$$

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101010011]$$

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101010011]$$

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \neg(\vec{c}(i)), & \text{ha } p_i < \varepsilon \end{cases}$$

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101001011]$$

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \neg(\vec{c}(i)), & \text{ha } p_i < \varepsilon \end{cases}$$

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101010011]$$
$$\downarrow$$
$$\vec{c}' = [1011010110100011 11]$$

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \neg(\vec{c}(i)), & \text{ha } p_i < \varepsilon \end{cases}$$

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101011]$$
$$\downarrow$$
$$\vec{c}' = [1011010110100011]$$

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \neg(\vec{c}(i)), & \text{ha } p_i < \varepsilon \end{cases}$$

Általánosítható-e a modell valós értékek kódolására is?

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101010011]$$
$$\downarrow$$
$$\vec{c}' = [1011010110100011 1]$$

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \neg(\vec{c}(i)), & \text{ha } p_i < \varepsilon \end{cases}$$

Általánosítható-e a modell valós értékek kódolására is?

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \vec{c}(i) \pm \delta, & \text{ha } p_i < \varepsilon \end{cases}$$

BINÁRIS MUTÁCIÓ

$$\vec{c} = [1010110100101010011]$$
$$\downarrow$$
$$\vec{c}' = [1011010110100011 1]$$

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \neg(\vec{c}(i)), & \text{ha } p_i < \varepsilon \end{cases}$$

Általánosítható-e a modell valós értékek kódolására is?

$$\vec{c}'(i) = \begin{cases} \vec{c}(i), & \text{ha } p_i \geq \varepsilon \\ \vec{c}(i) \pm \delta, & \text{ha } p_i < \varepsilon \end{cases} \rightarrow \quad \delta = 1 \quad \checkmark$$

FITNESS ARÁNYOS SZELEKCIÓ

Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

FITNESS ARÁNYOS SZELEKCIÓ

Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

```
1: p⃗ = {c⃗_1, c⃗_2, ..., c⃗_n}
2: r ← Σ_{i=0}^{n} f(c⃗_i)
3: loop
4:     legyen k ∈ [0, r)
5:     for all c⃗_i in p⃗ do
6:         if Σ_{j=0}^{i} f(c⃗_i) < k then
7:             continue      // nem ő nyert
8:         else
9:             válaszd ki a c⃗_i kromoszómát
10:            break         // ő nyert
11:        end if
12:    end for
13: end loop
```

FITNESS ARÁNYOS SZELEKCIÓ

Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

1: $\vec{p} = \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n\}$
2: $r \leftarrow \sum_{i=0}^{n} f(\vec{c}_i)$
3: **loop**
4: legyen $k \in [0, r)$
5: **for all** $\vec{c}_i$ in $\vec{p}$ **do**
6: **if** $\sum_{j=0}^{i} f(\vec{c}_i) < k$ **then**
7: **continue** // nem ő nyert
8: **else**
9: válaszd ki a $\vec{c}_i$ kromoszómát
10: **break** // ő nyert
11: **end if**
12: **end for**
13: **end loop**

FITNESS ARÁNYOS SZELEKCIÓ

Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

1: $\vec{p} = \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n\}$
2: $r \leftarrow \sum_{i=0}^{n} f(\vec{c}_i)$
3: **loop**
4: legyen $k \in [0, r)$
5: **for all** $\vec{c}_i$ in $\vec{p}$ **do**
6: **if** $\sum_{j=0}^{i} f(\vec{c}_i) < k$ **then**
7: **continue** // nem ő nyert
8: **else**
9: válaszd ki a $\vec{c}_i$ kromoszómát
10: **break** // ő nyert
11: **end if**
12: **end for**
13: **end loop**

FITNESS ARÁNYOS SZELEKCIÓ

Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

```
1:  p⃗ = {c⃗_1, c⃗_2, ..., c⃗_n}
2:  r ← Σ_{i=0}^{n} f(c⃗_i)
3:  loop
4:      legyen k ∈ [0, r)
5:      for all c⃗_i in p⃗ do
6:          if Σ_{j=0}^{i} f(c⃗_i) < k then
7:              continue      // nem ő nyert
8:          else
9:              válaszd ki a c⃗_i kromoszómát
10:             break         // ő nyert
11:         end if
12:     end for
13: end loop
```

FITNESS ARÁNYOS SZELEKCIÓ

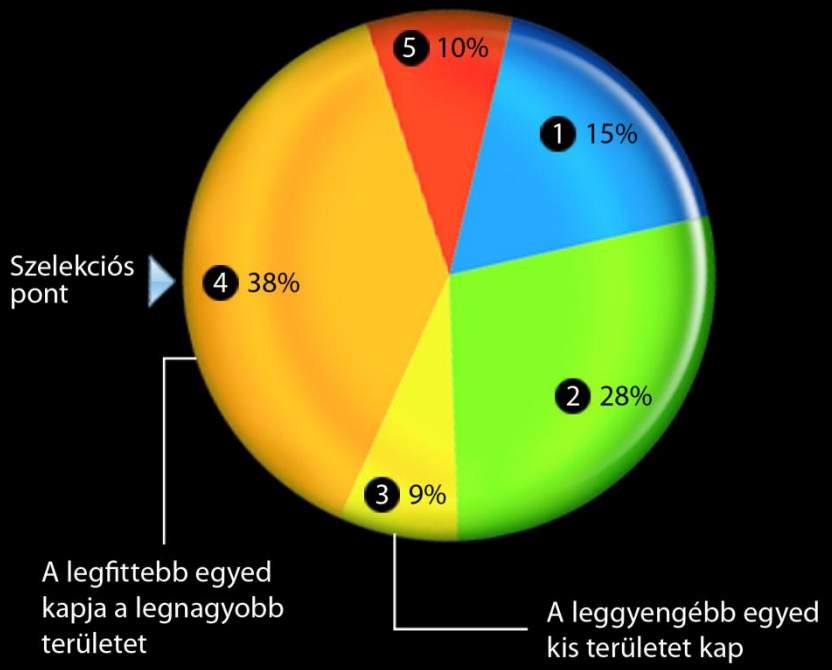
Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

```
1:  p⃗ = {c⃗_1, c⃗_2, ..., c⃗_n}
2:  r ← Σ_{i=0}^{n} f(c⃗_i)
3:  loop
4:      legyen k ∈ [0, r)
5:      for all c⃗_i in p⃗ do
6:          if Σ_{j=0}^{i} f(c⃗_i) < k then
7:              continue      // nem ő nyert
8:          else
9:              válaszd ki a c⃗_i kromoszómát
10:             break         // ő nyert
11:         end if
12:     end for
13: end loop
```

FITNESS ARÁNYOS SZELEKCIÓ

Alapelv $\forall i: \; P_s(\vec{c}_i) \propto f(\vec{c}_i)$

1: $\vec{p} = \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n\}$
2: $r \leftarrow \sum_{i=0}^{n} f(\vec{c}_i)$
3: **loop**
4: legyen $k \in [0, r)$
5: **for all** $\vec{c}_i$ in $\vec{p}$ **do**
6: **if** $\sum_{j=0}^{i} f(\vec{c}_i) < k$ **then**
7: **continue** // nem ő nyert
8: **else**
9: válaszd ki a $\vec{c}_i$ kromoszómát $\rightarrow$
10: **break** // ő nyert
11: **end if**
12: **end for**
13: **end loop**



ELITISTA SZELEKCIÓ

Alapelv Csak a k darab legjobb kromoszómát örökítjük tovább

$$f(\vec{c}_1) = 13$$
$$f(\vec{c}_2) = 11$$
$$f(\vec{c}_3) = 27$$
$$f(\vec{c}_4) = 22$$
$$f(\vec{c}_5) = 31$$
$$f(\vec{c}_6) = 5$$

ELITISTA SZELEKCIÓ

Alapelv Csak a k darab legjobb kromoszómát örökítjük tovább

$$
\begin{array}{ll}
f(\vec{c}_1) = 13 & \\
f(\vec{c}_2) = 11 & \\
f(\vec{c}_3) = 27 & \xrightarrow{\text{sort}} \\
f(\vec{c}_4) = 22 & \\
f(\vec{c}_5) = 31 & \\
f(\vec{c}_6) = 5 &
\end{array}
\begin{array}{l}
f(\vec{c}_5) = 31 \\
f(\vec{c}_3) = 27 \\
f(\vec{c}_4) = 22 \\
f(\vec{c}_1) = 13 \\
f(\vec{c}_2) = 11 \\
f(\vec{c}_6) = 5
\end{array}
$$

ELITISTA SZELEKCIÓ

Alapelv Csak a k darab legjobb kromoszómát örökítjük tovább

$$
\begin{array}{ll}
f(\vec{c}_1) = 13 & f(\vec{c}_5) = 31 \\
f(\vec{c}_2) = 11 & f(\vec{c}_3) = 27 \\
f(\vec{c}_3) = 27 & f(\vec{c}_4) = 22 \\
f(\vec{c}_4) = 22 & f(\vec{c}_1) = 13 \\
f(\vec{c}_5) = 31 & f(\vec{c}_2) = 11 \\
f(\vec{c}_6) = 5 & f(\vec{c}_6) = 5
\end{array}
\quad \xrightarrow{\text{sort}} \quad
$$

ALGORITMUS

1: **init** $\vec{p} = \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n\}$
2: $\forall i: \; f_i \leftarrow f(\vec{c}_i)$
3: **loop**
4: $\quad \vec{p}' \subseteq \vec{p}$
5: $\quad \vec{c}'_1 \leftarrow \text{keresztezés}(\vec{c}_i, \vec{c}_j), \;\; \vec{c}_i, \vec{c}_j \in \vec{p}'$
6: $\quad \vec{c}'_2 \leftarrow \text{mutáció}(\vec{c}_i), \;\; \vec{c}_i \in \vec{p}'$
7: $\quad \forall i: \; f_i \leftarrow f(\vec{c}_i)$
8: $\quad \vec{p} := \vec{p}' \cup \vec{c}'_1 \cup \vec{c}'_2$
9: **end loop**

ALGORITMUS

1: **init** $\vec{p} = \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n\}$
2: $\forall i: \ f_i \leftarrow f(\vec{c}_i)$
3: **loop**
4: $\quad \vec{p}' \subseteq \vec{p}$
5: $\quad \vec{c}'_1 \leftarrow \text{keresztezés}(\vec{c}_i, \vec{c}_j), \ \ \vec{c}_i, \vec{c}_j \in \vec{p}'$
6: $\quad \vec{c}'_2 \leftarrow \text{mutáció}(\vec{c}_i), \ \ \vec{c}_i \in \vec{p}'$
7: $\quad \forall i: \ f_i \leftarrow f(\vec{c}_i)$
8: $\quad \vec{p} := \vec{p}' \cup \vec{c}'_1 \cup \vec{c}'_2$
9: **end loop**

Kilépési feltételek

- Generáció limit
- Max fitness limit
- Átlag fitness limit
- Konvergencia beállása

1	0	1	1	1

1	0	0	0	1

0	0	1	0	1

1	1	0	1	0

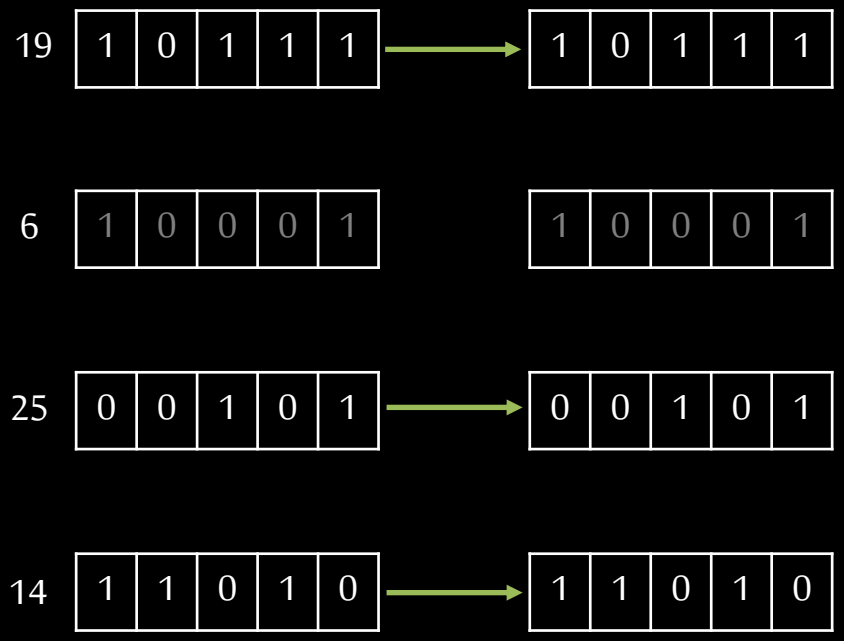
Kiinduló populáció

19	1	0	1	1	1
6	1	0	0	0	1
25	0	0	1	0	1
14	1	1	0	1	0

Kiinduló populáció

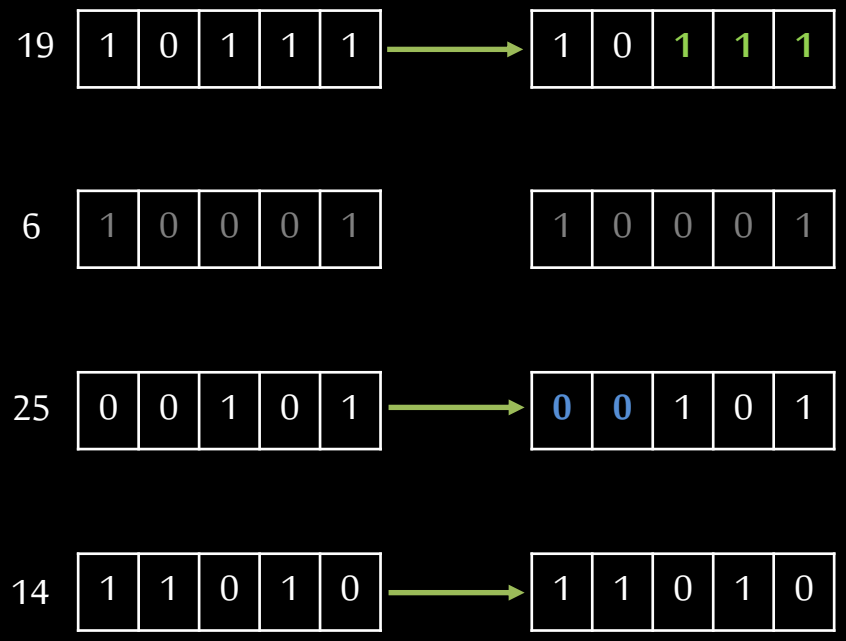
19	**1**	**0**	**1**	**1**	**1**
6	1	0	0	0	1
25	**0**	**0**	**1**	**0**	**1**
14	**1**	**1**	**0**	**1**	**0**

Kiinduló populáció



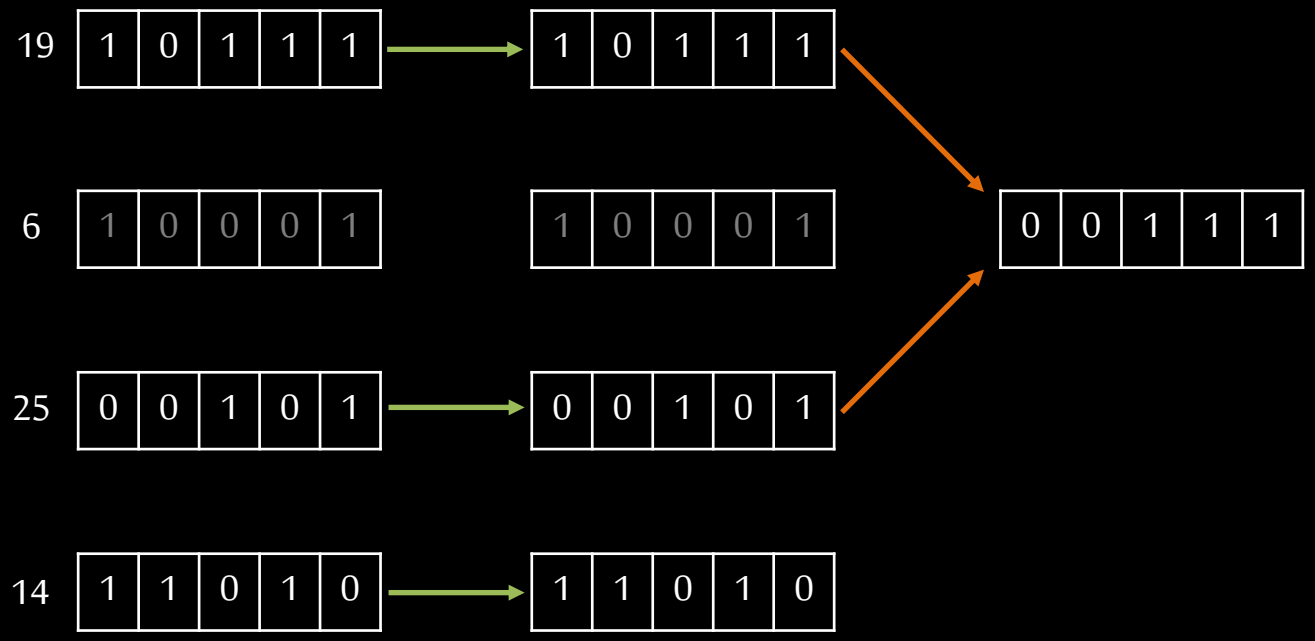
Kiinduló populáció

Szelekció



Kiinduló populáció

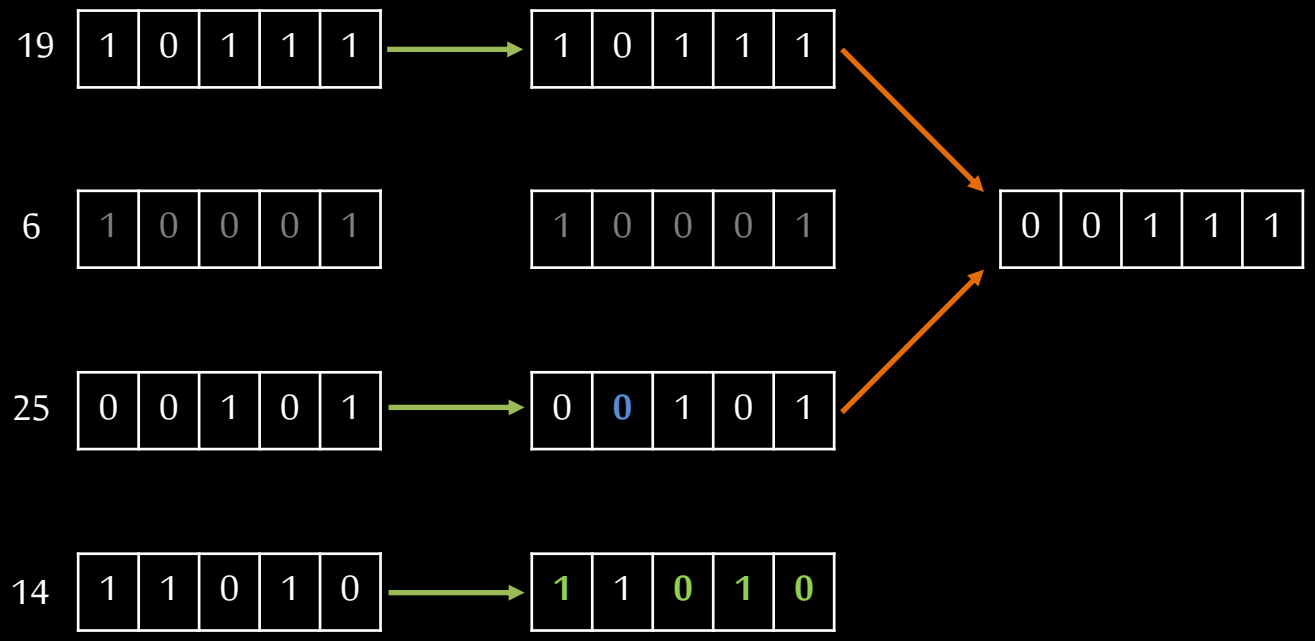
Szelekció



Kiinduló populáció

Szelekció

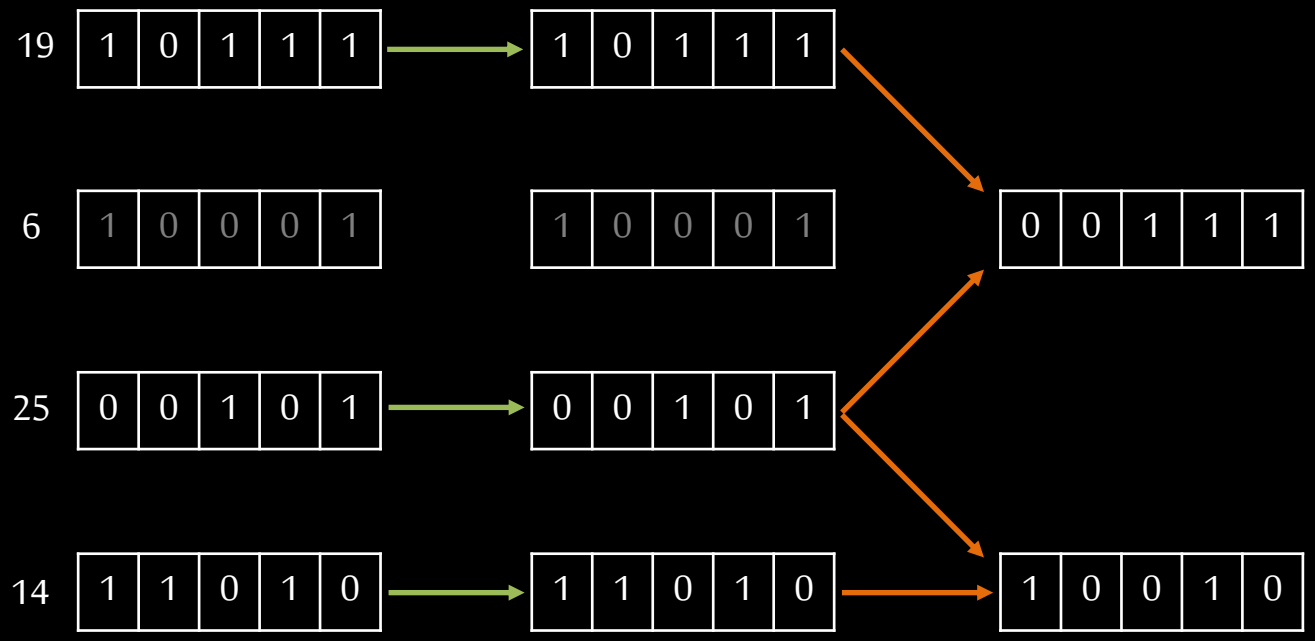
Keresztezés



Kiinduló populáció

Szelekció

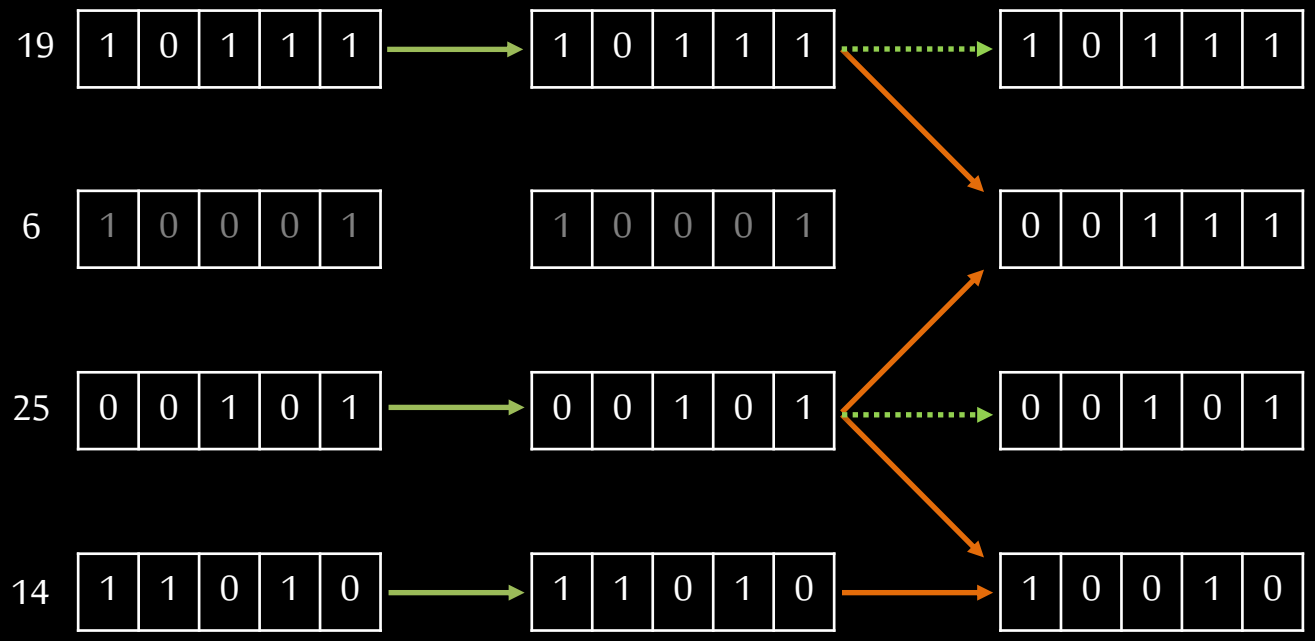
Keresztezés



Kiinduló populáció

Szelekció

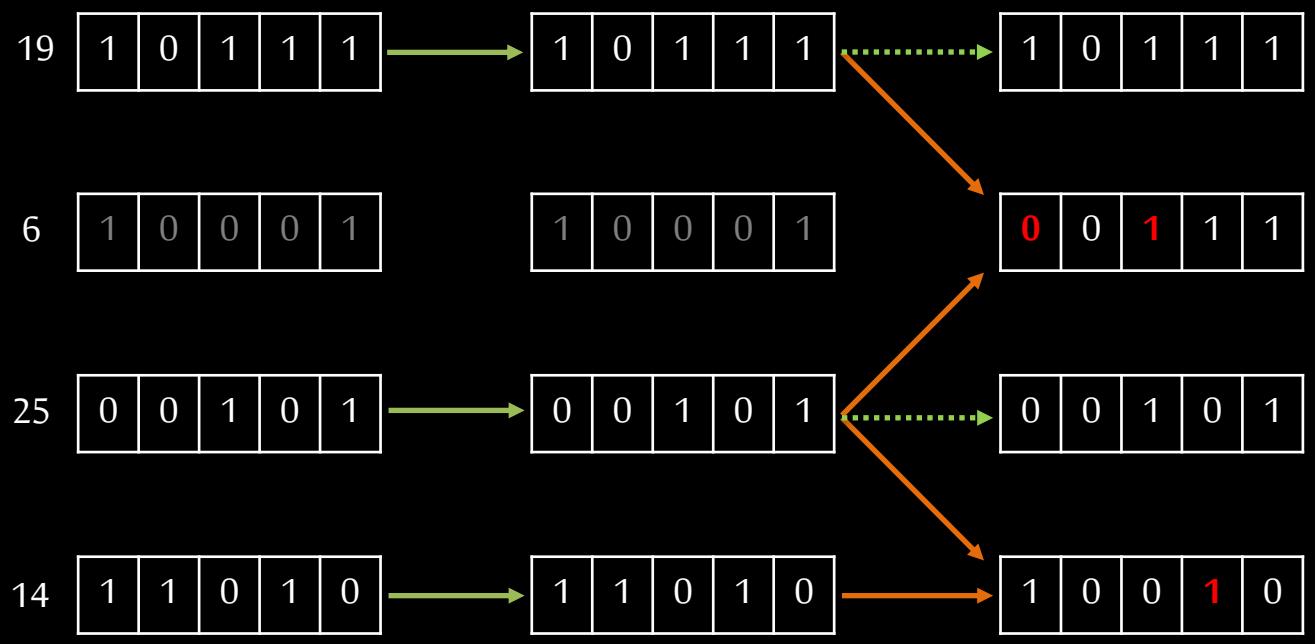
Keresztezés



Kiinduló populáció

Szelekció

Keresztezés

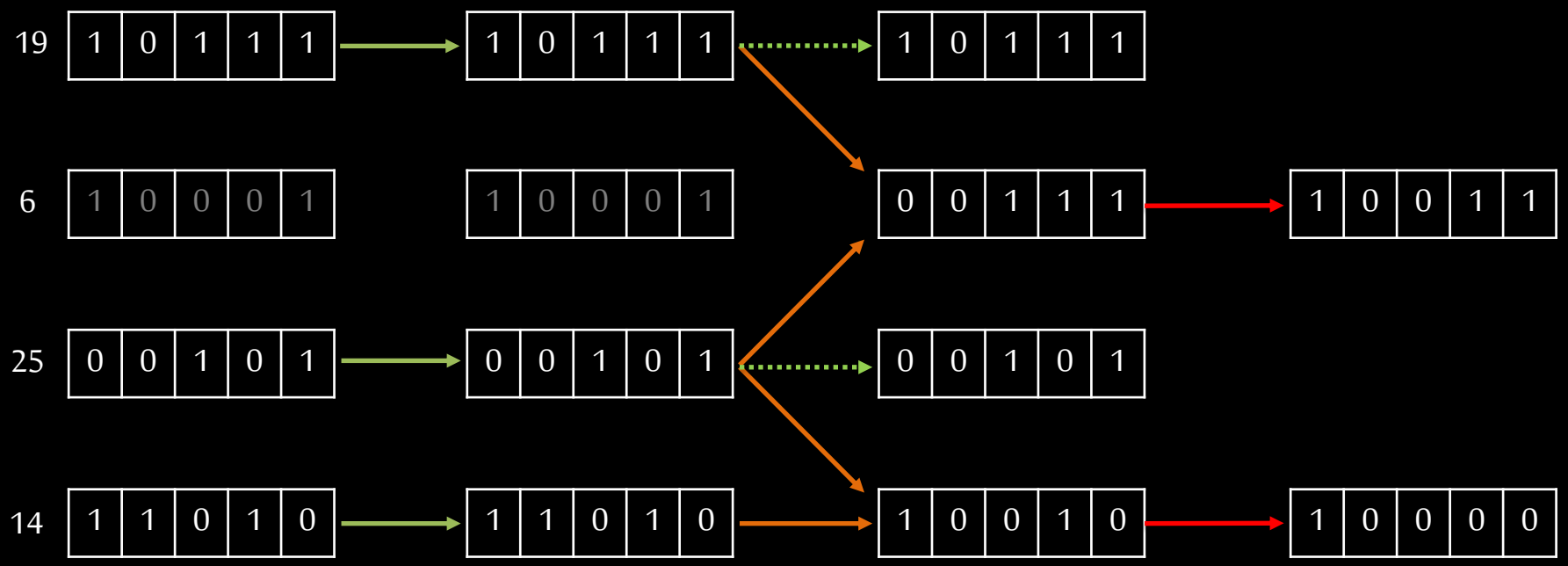


Kiinduló populáció

Szelekció

Keresztezés

Mutáció

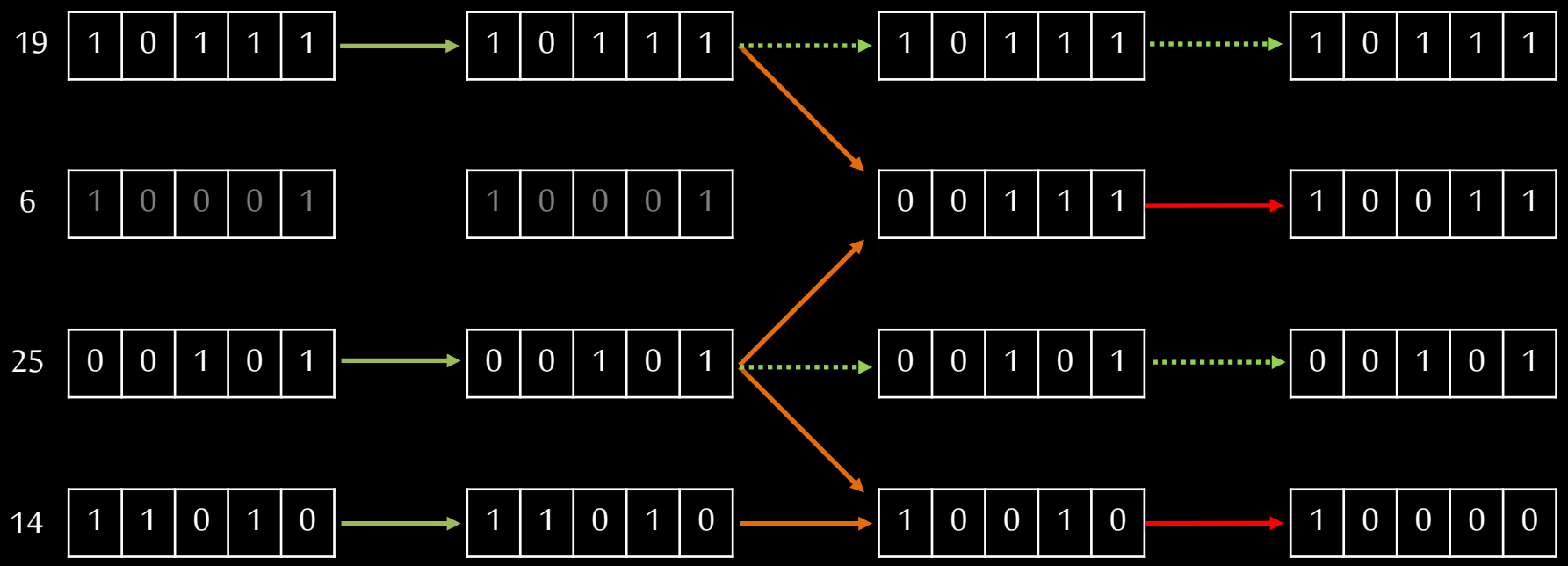


Kiinduló populáció

Szelekció

Keresztezés

Mutáció

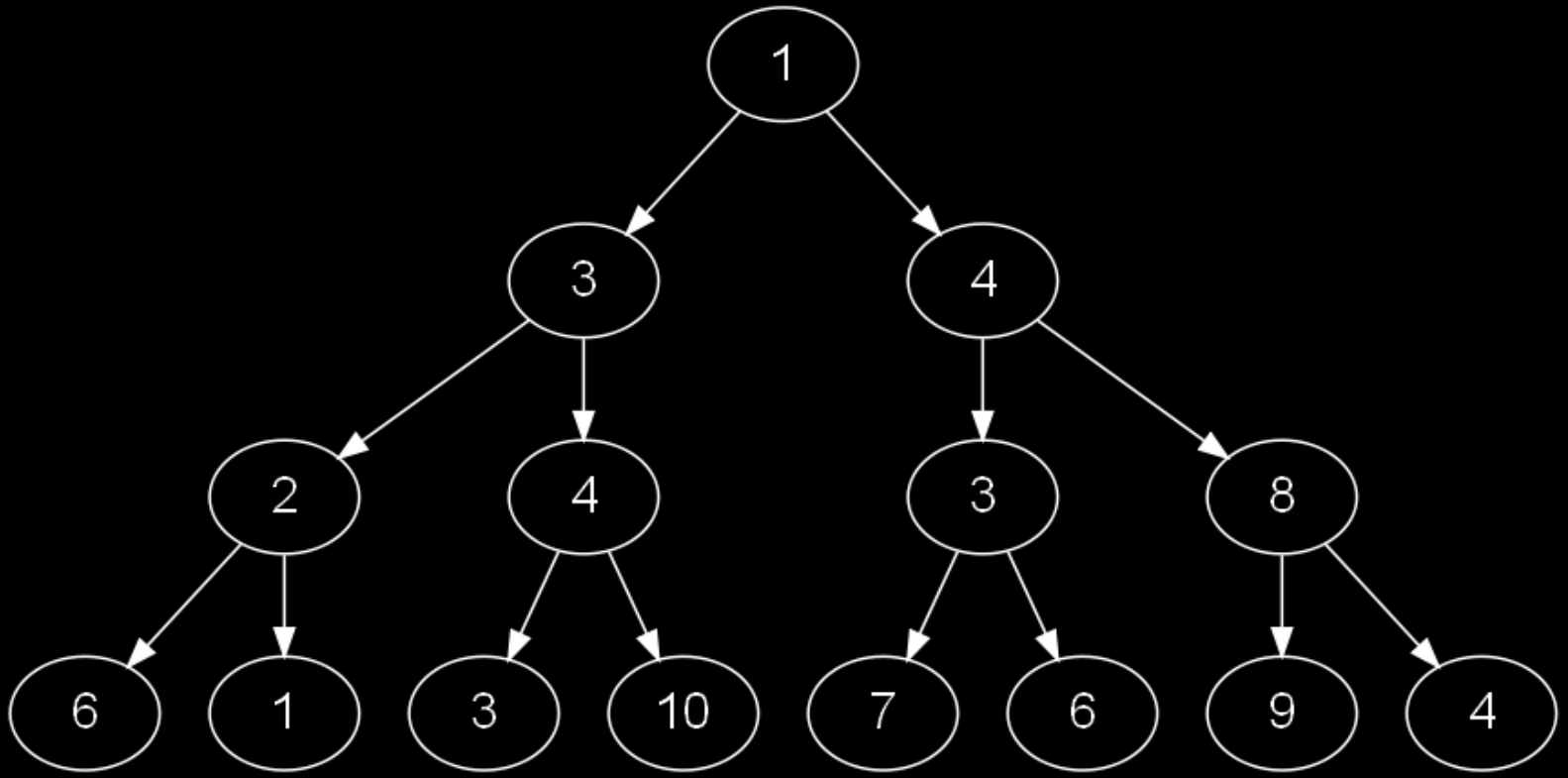


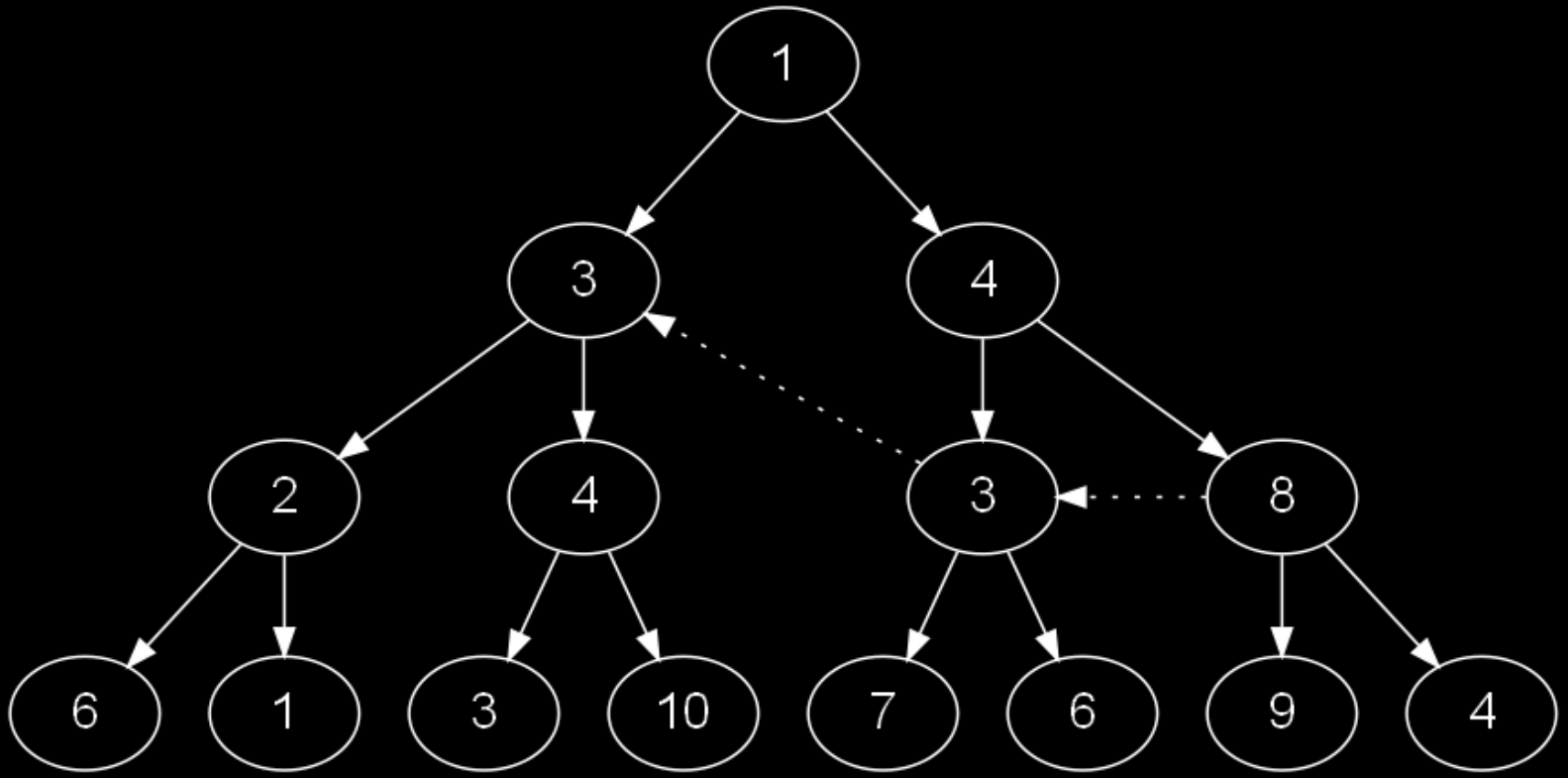
Kiinduló populáció

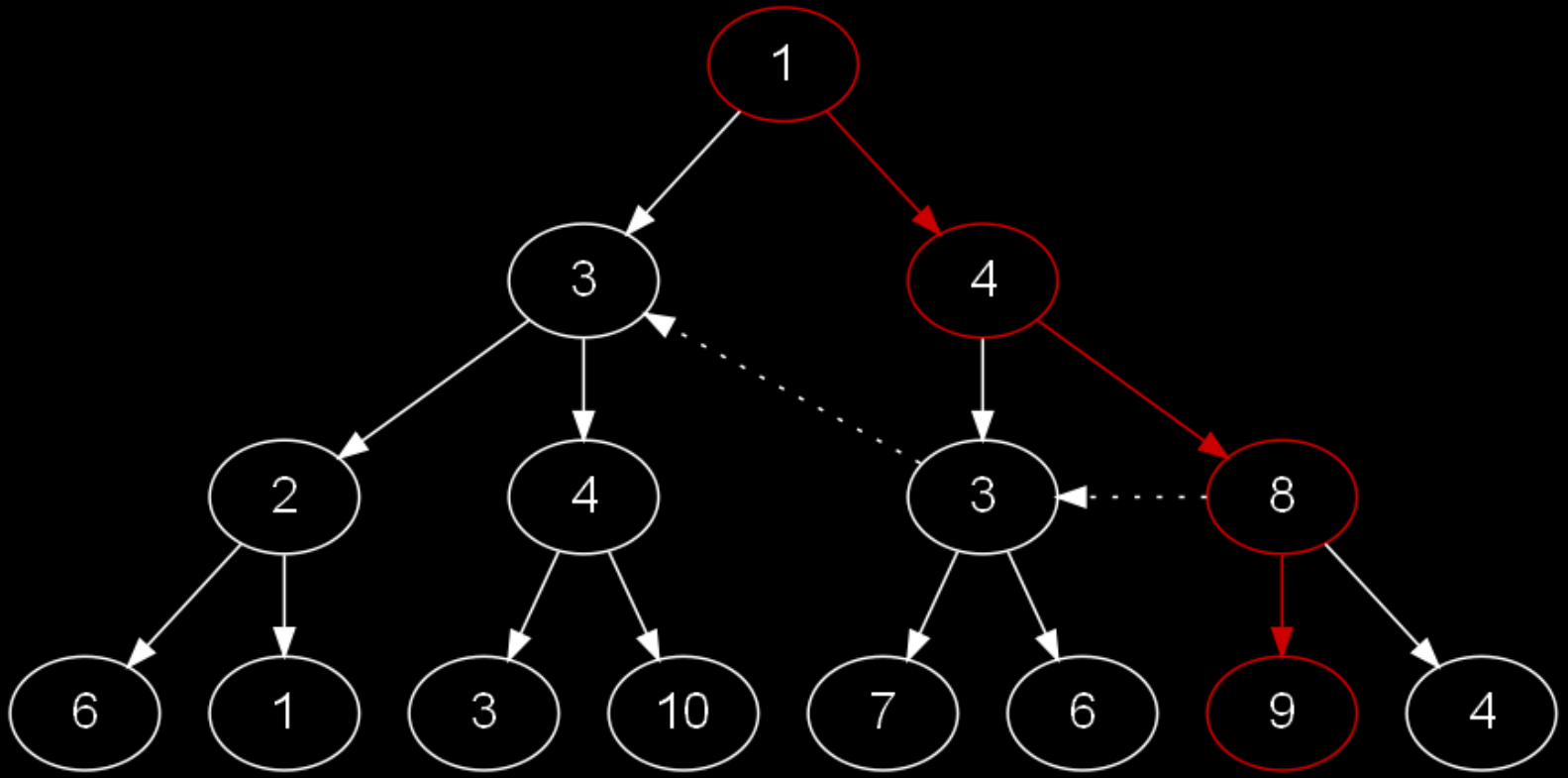
Szelekció

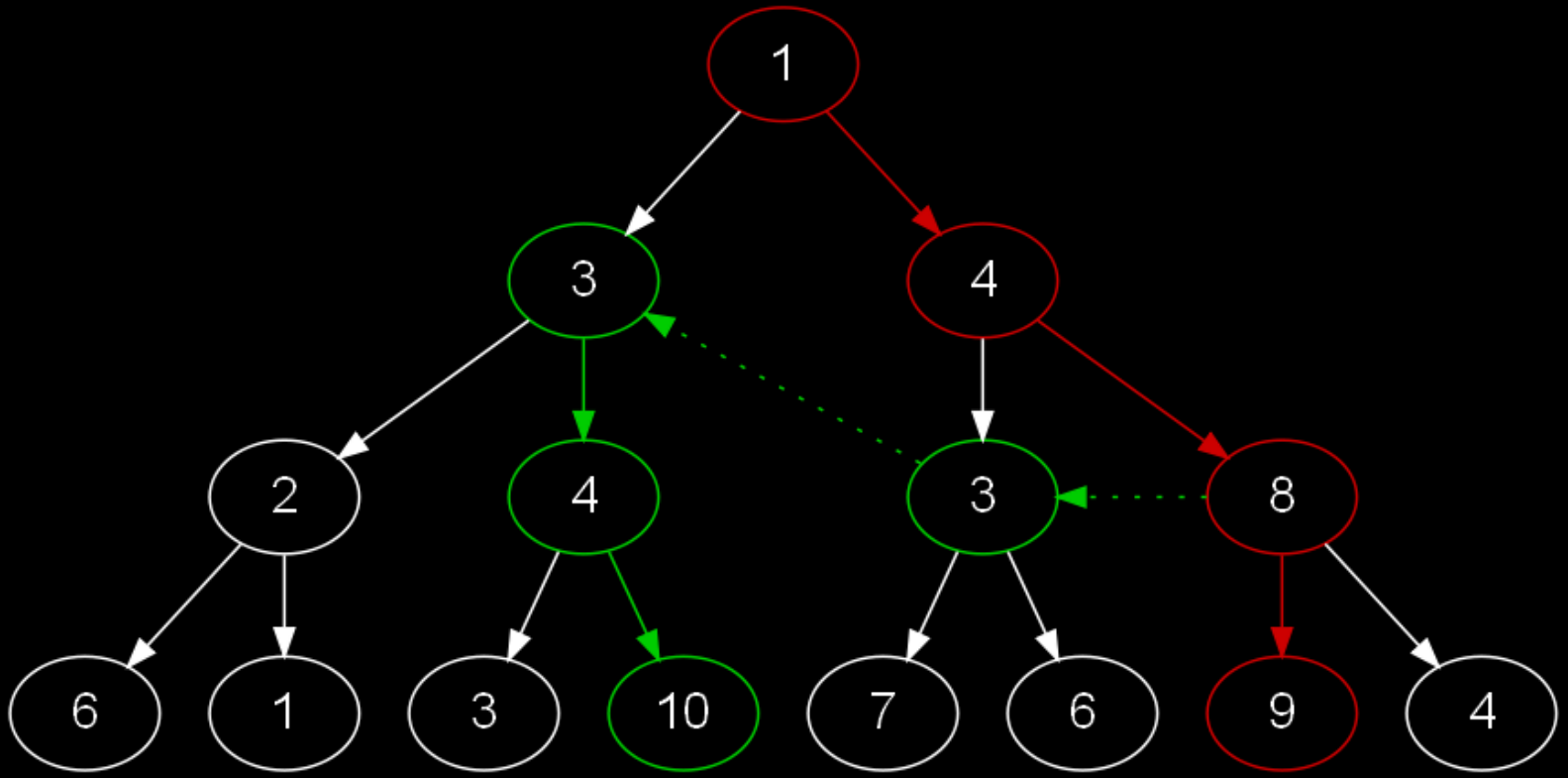
Keresztezés

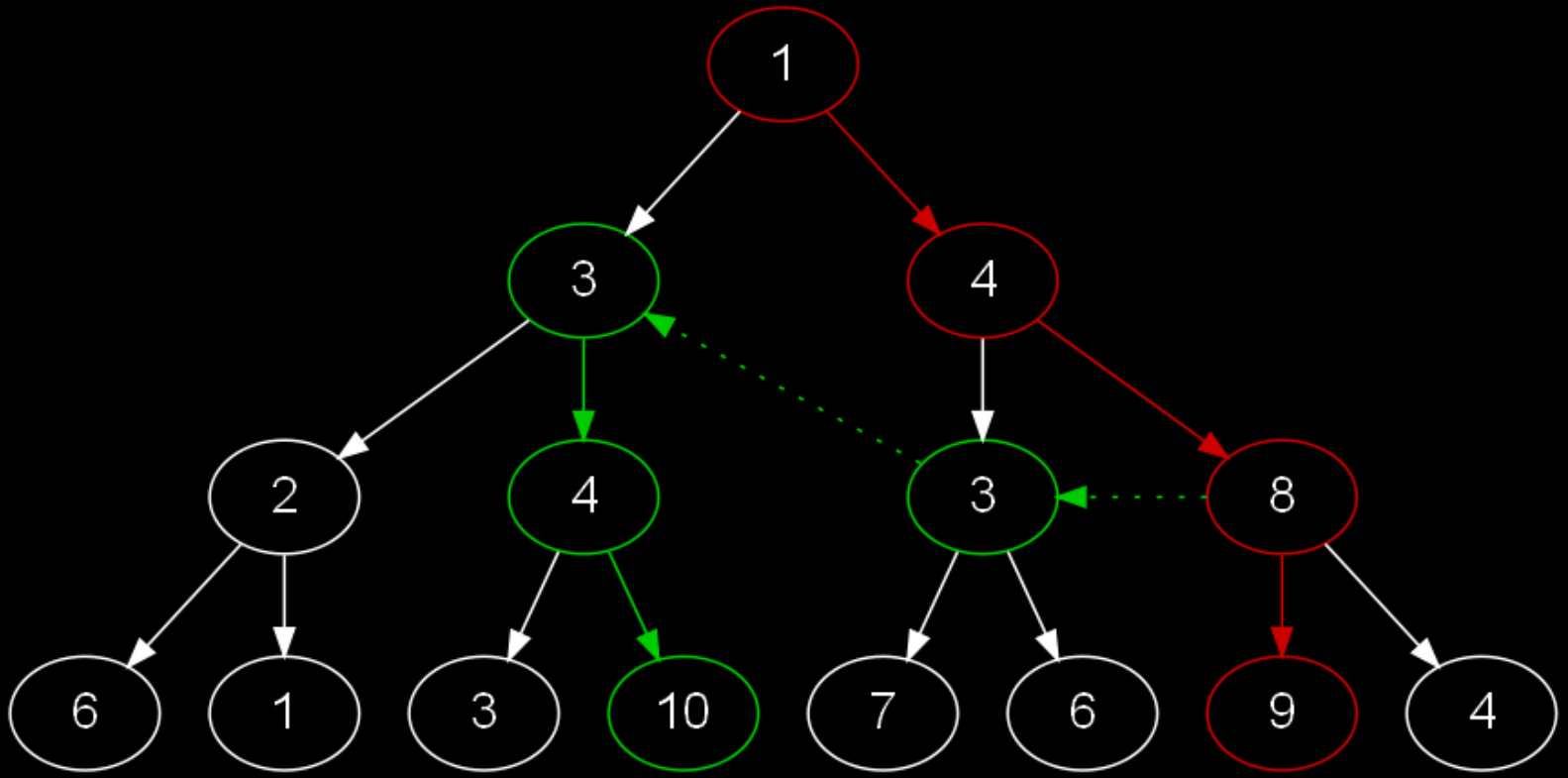
Mutáció


1
3
4
2
4
3
8
6
1
3
10
7
6
9
4


1
3
4
2
4
3
8
6
1
3
10
7
6
9
4


1
3
4
2
4
3
8
6
1
3
10
7
6
9
4


1
3
4
2
4
3
8
6
1
3
10
7
6
9
4



Érdemes úgy elképzelni, mintha a felrajzoltak helyett egy teljes gráfunk lenne, ahol a nem látszó éleket a mutáció valószínűségének megfelelő eséllyel igénybe vehetjük (szaggatott vonal).

Hátizsák probléma implementáció: http://cg.iit.bme.hu/~zsolnai/gfx/genetic/

```
#0      best fitness: 26        avg fitness: 13.250000

fitness:  26    01111111
fitness:  23    11101111
fitness:  18    10101011
fitness:  18    01010111
fitness:  17    10101101
fitness:  16    00010111
fitness:  14    11100101
fitness:  13    10011100
fitness:  13    01100101
fitness:  13    00011010
fitness:  13    10101010
fitness:  13    10101010
fitness:  12    11010001
fitness:  11    11000101
fitness:  14    11101100        [mutation]
fitness:  13    10110001        [mutation]
fitness:  11    00100101        [mutation]
fitness:  13    10111000        [mutation]
fitness:  21    10101111        [crossover]
fitness:  16    00011110        [crossover]
```

„Mona Lisa” festés poligonokból implementáció: http://cg.iit.bme.hu/~zsolnai/

„Mona Lisa” festés poligonokból implementáció: http://cg.iit.bme.hu/~zsolnai/

„Mona Lisa” festés poligonokból implementáció: http://cg.iit.bme.hu/~zsolnai/

Telepítési, fordítási útmutató a README-ben.

„Mona Lisa” festés poligonokból implementáció: http://cg.iit.bme.hu/~zsolnai/

Telepítési, fordítási útmutató a README-ben.

Gabriele Greco SDL implementációja:

Forrás: http://www.ggsoft.org/archives/genetic.zip

Bináris: http://www.ggsoft.org/archives/genetic_test.zip (SDL.dll kellhet hozzá)

Heap szervezés

A legegyszerűbb módszer, nem rendel az adatokhoz külön struktúrát, így a keresés lineáris futásidejű, azaz n darab rekord esetén $O(n)$.

Heap szervezés

A legegyszerűbb módszer, nem rendel az adatokhoz külön struktúrát, így a keresés lineáris futásidejű, azaz n darab rekord esetén $O(n)$.

Indexelt szervezés

Sűrű index – minden egyes adatrekordra mutatót állítunk
Ritka index – az adatrekordok egy csoportjára állítunk mutatókat

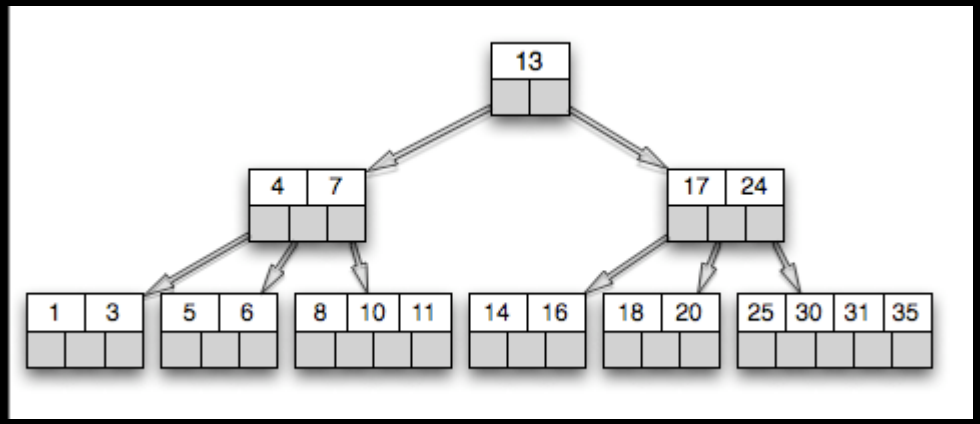
Ha az indexállomány a kulcs szerint rendezett, lehet bináris keresést használni, n darab rekord esetén $O(log_2 n)$.

Többszintű ritka indexek, B*-fa

Az indexállományban való keresést tovább gyorsítjuk azzal, hogy az indexhez egy további ritka indexet készítünk. Így egy fát kapunk, amiben a keresés $O(log_k n)$ nagyságrendű, amennyiben a fa elágazási tényezője k.

Többszintű ritka indexek, B*-fa

Az indexállományban való keresést tovább gyorsítjuk azzal, hogy az indexhez egy további ritka indexet készítünk. Így egy fát kapunk, amiben a keresés $O(log_k n)$ nagyságrendű, amennyiben a fa elágazási tényezője k.



Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

$t_i^{(q)}$ - az i. select futási ideje

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

$t_i^{(q)}$ - az i. select futási ideje

μ_i - hányszor kérték az adott lekérdezést

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

$t_i^{(q)}$ - az i. select futási ideje

μ_i - hányszor kérték az adott lekérdezést

N_q - ennyi lekérdezésre teszteltünk

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

$$f(\vec{c}) = \frac{1}{\sum_{i=1}^{N_q} \mu_i \, t_i^{(q)}}$$

$t_i^{(q)}$ - az $i.$ select futási ideje

μ_i - hányszor kérték az adott lekérdezést

N_q - ennyi lekérdezésre teszteltünk

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

$$f(\vec{c}) = \frac{1}{\sum_{i=1}^{N_q} \mu_i \, t_i^{(q)}}$$

$t_i^{(q)}$ - az i. select futási ideje

μ_i - hányszor kérték az adott lekérdezést

N_q - ennyi lekérdezésre teszteltünk

$$f(\vec{c}) = \frac{1}{\sum_{i=1}^{N_I} \eta_i \, t_i^{(I)}}$$

$t_i^{(I)}$ - az i. beszúrás futási ideje

η_i - hányszor kérték az adott beszúrást

N_I - ennyi beszúrásra teszteltünk

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

$$f(\vec{c}) = \frac{1}{\sum_{i=1}^{N_q} \mu_i \, t_i^{(q)}} + \frac{1}{\sum_{i=1}^{N_I} \eta_i \, t_i^{(I)}}$$

- $t_i^{(q)}$ - az $i.$ select futási ideje
- μ_i - hányszor kérték az adott lekérdezést
- N_q - ennyi lekérdezésre teszteltünk
- $t_i^{(I)}$ - az $i.$ beszúrás futási ideje
- η_i - hányszor kérték az adott beszúrást
- N_I - ennyi beszúrásra teszteltünk

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

lekérdezés és beszúrás fontosságának súlyozása

$$f(\vec{c}) = \alpha \frac{1}{\sum_{i=1}^{N_q} \mu_i \, t_i^{(q)}} + \beta \frac{1}{\sum_{i=1}^{N_I} \eta_i \, t_i^{(I)}}$$

- $t_i^{(q)}$ - az $i.$ select futási ideje
- μ_i - hányszor kérték az adott lekérdezést
- N_q - ennyi lekérdezésre teszteltünk
- $t_i^{(I)}$ - az $i.$ beszúrás futási ideje
- η_i - hányszor kérték az adott beszúrást
- N_I - ennyi beszúrásra teszteltünk

Adatbázis indexelési feladat

Feladat

Egy olyan indexre van szükségünk, ami az adatbázisból való olvasás és írás sebességét optimalizálja.

Kódolás

A kromoszóma legyen egy bináris vektor, ahol a vektor elemei a rekordok, az értékek pedig 0/1 az alapján, hogy teszünk-e rá mutatót, vagy sem.

$$\vec{c}_i = [1010110100101011]$$

Fitness függvény

lekérdezés és beszúrás fontosságának súlyozása

$$f(\vec{c}) = \alpha \frac{1}{\sum_{i=1}^{N_q} \mu_i \, t_i^{(q)}} + \beta \frac{1}{\sum_{i=1}^{N_I} \eta_i \, t_i^{(I)}}$$

- $t_i^{(q)}$ - az $i.$ select futási ideje
- μ_i - hányszor kérték az adott lekérdezést
- N_q - ennyi lekérdezésre teszteltünk
- $t_i^{(I)}$ - az $i.$ beszúrás futási ideje
- η_i - hányszor kérték az adott beszúrást
- N_I - ennyi beszúrásra teszteltünk

További alkalmazások az adatbázisokhoz

Fizikai adatszervezés optimalizálása

http://130.203.133.150/viewdoc/summary;jsessionid=A5CAB6783D8946BEEF75E034FD2CE254?doi=10.1.1.206.1862

Elosztott adatbázisokra bejövő queryk sorrendezése, szétosztása

http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=476497&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D476497

Adatbázis kiszolgáló fenntartási költségeinek minimalizálása

http://smartech.gatech.edu/handle/1853/6618

Természetes illesztések futásidő-optimalizálása

http://connection.ebscohost.com/c/articles/4477437/access-path-optimization-relational-joins

Elosztott adatbázisok vertikális partícionálása ideális módon

http://ceur-ws.org/Vol-547/91.pdf

A GENETIKUS ALGORITMUSOKRÓL TEHÁT AZ ALÁBBIAKAT ÁLLÍTHATJUK

- Nagy részük probléma- és reprezentációfüggetlen
- Az átlagos fitness érték az optimumhoz konvergálnak
- A konvergálás kapcsán már útközben látható a pillanatnyi eredmény
- Tetszőlegesen nagy dimenziójú problémákra alkalmazhatók
- Mutációval kiszabadulhatnak a lokális optimumhelyek rabságából
- Más gyors, de nem optimális keresési algoritmusokkal jól keverhetők
- Kiválóan párhuzamosíthatók, alacsony a kommunikációs bonyolultságuk
- Tárigényük $p \cdot O(n)$

Ökölszabály: akkor használandó, ha egy problémát elég egyszer megoldani!

Referenciák

Gajdos Sándor – Adatbázisok
Mûegyetemi kiadó, 2001.

Marcin Korytkowski, Marcin Gabryel, Robert Nowicki and Rafal Scherer -
Genetic Algorithm for Database Indexing
http://link.springer.com/chapter/10.1007%2F978-3-540-24844-6_179