

$$\vec{\mathcal{L}}(q) = \bigcup_{a:\delta(q,a)\neq\perp} a\vec{\mathcal{L}}(\delta(q,a)) \cup \begin{cases} \emptyset & q \notin F \\ \{\epsilon\} & q \in F \end{cases}$$

$$\vec{\mathcal{L}}(q) = \vec{\mathcal{L}}(p) \Leftrightarrow$$

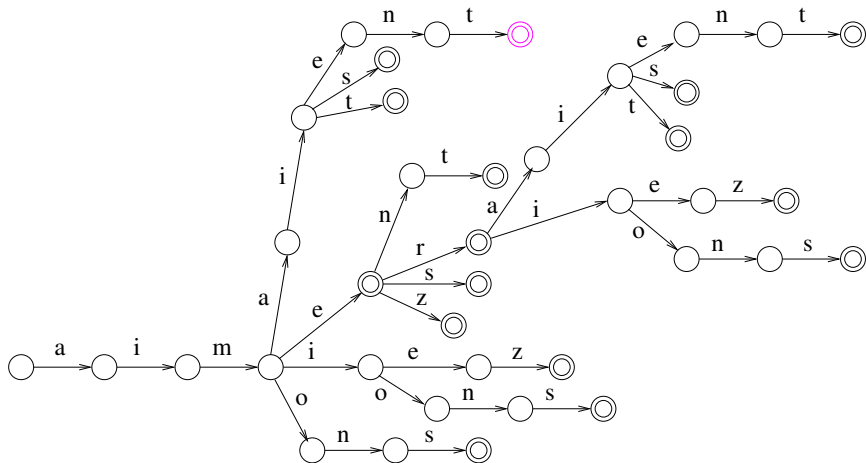
- $(q \in F \wedge p \in F) \vee (q \notin F \wedge p \notin F)$
- $\{a \in \Sigma : \delta(q, a) \in Q\} = \{b \in \Sigma : \delta(p, b) \in Q\}$
- $\forall a \in \Sigma : \delta(q, a) \in Q \quad \vec{\mathcal{L}}(\delta(q, a)) = \vec{\mathcal{L}}(\delta(p, a))$

$$\vec{\mathcal{L}}(q) = \bigcup_{a:\delta(q,a)\neq\perp} a\vec{\mathcal{L}}(\delta(q,a)) \cup \begin{cases} \emptyset & q \notin F \\ \{\epsilon\} & q \in F \end{cases}$$

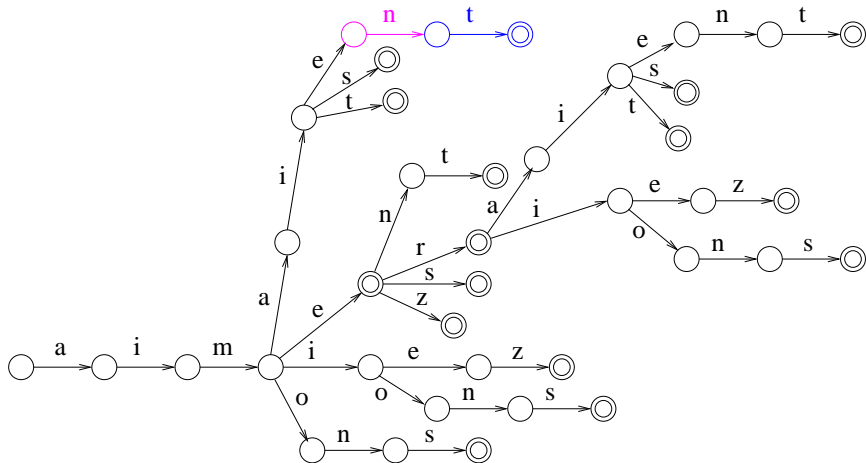
$$\vec{\mathcal{L}}(q) = \vec{\mathcal{L}}(p) \Leftrightarrow$$

- $(q \in F \wedge p \in F) \vee (q \notin F \wedge p \notin F)$
- $\{a \in \Sigma : \delta(q, a) \in Q\} = \{b \in \Sigma : \delta(p, b) \in Q\}$
- $\forall a \in \Sigma : \delta(q, a) \in Q \iff \delta(p, a) \in Q$

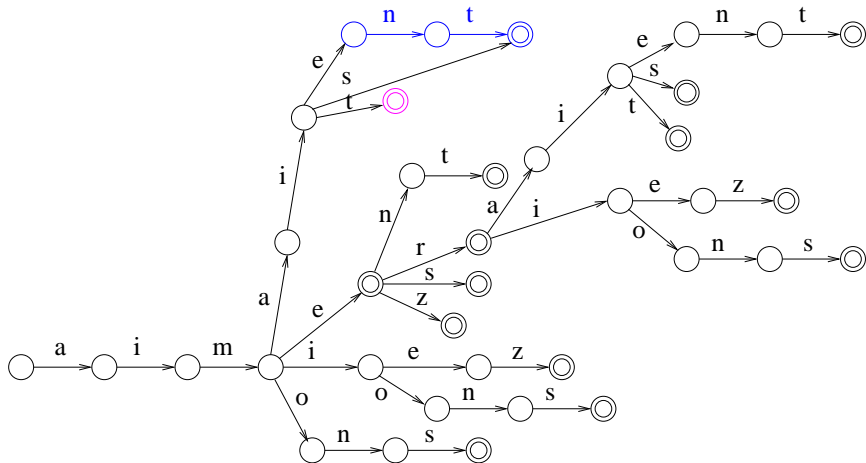
Co to jest minimalizacja drzewa?



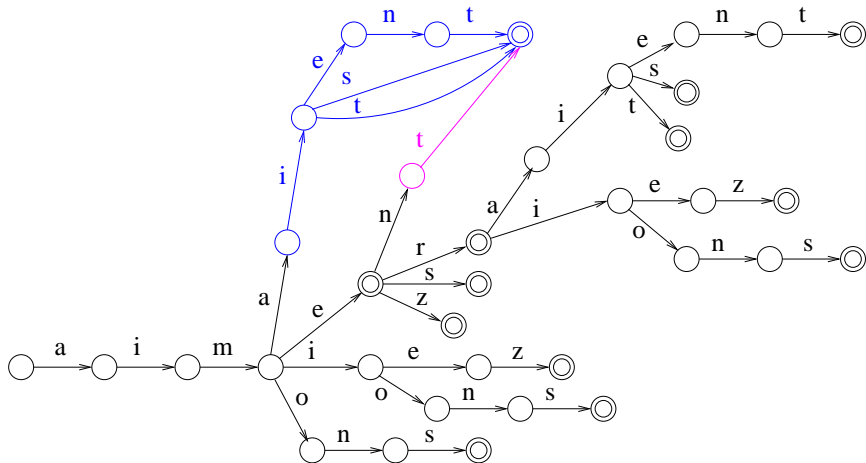
Co to jest minimalizacja drzewa?



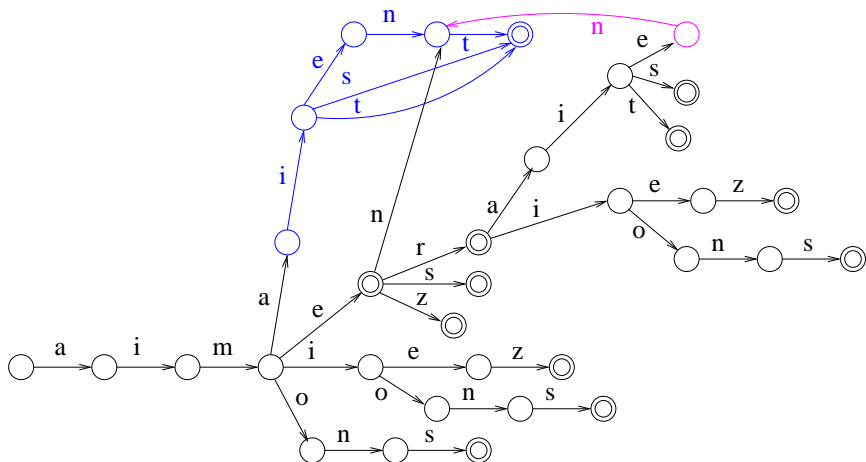
Co to jest minimalizacja drzewa?



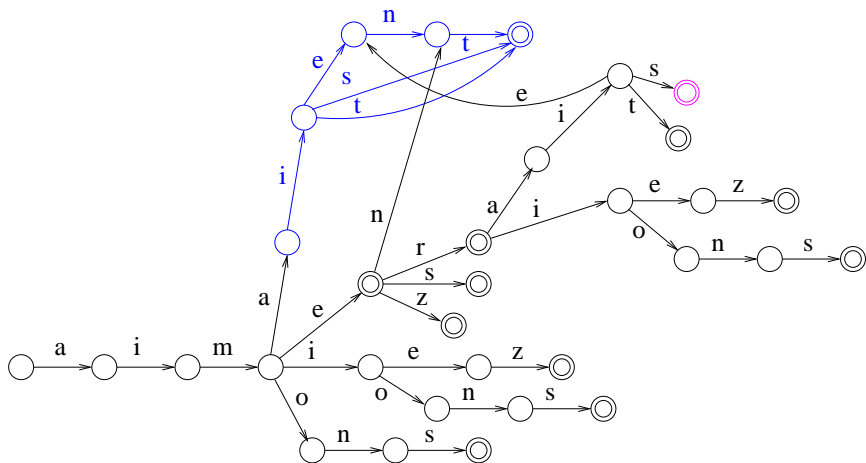
Co to jest minimalizacja drzewa?



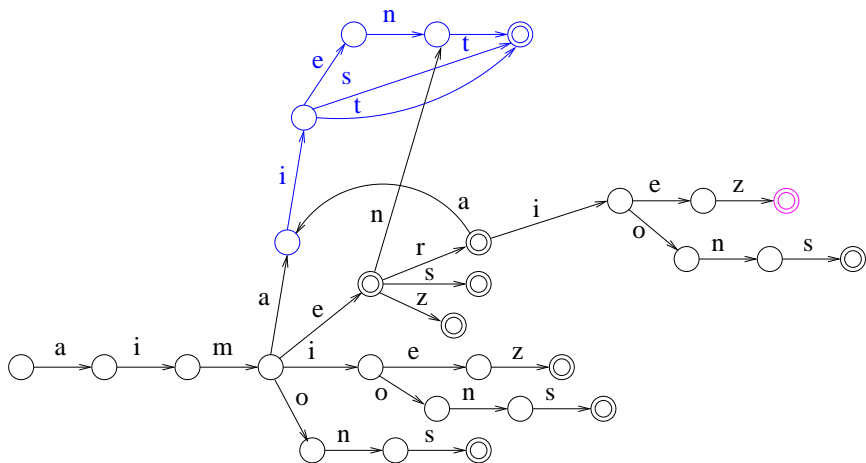
Co to jest minimalizacja drzewa?



Co to jest minimalizacja drzewa?



Co to jest minimalizacja drzewa?



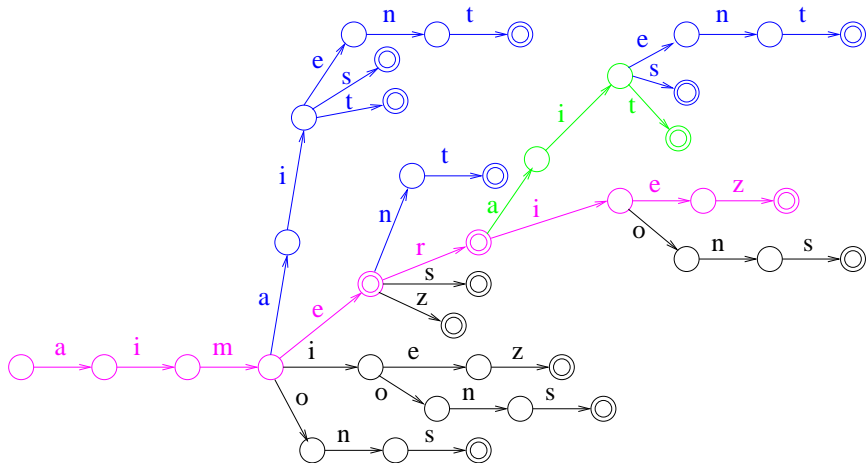
- Porównanie z użyciem definicji rekurencyjnej $\vec{\mathcal{L}}(q)$
- Stany odwiedzone wg. metody postorder, tak że dzieci zawsze mają niepowtarzalny prawostronny język
- Stany z niepowtarzalnymi $\vec{\mathcal{L}}(q)$ w tablicy rozproszonej
- Funkcja mieszająca wykorzystuje końcowość stanów i przejścia

Wynik: operacje na rejestrze są $\mathcal{O}(1)$

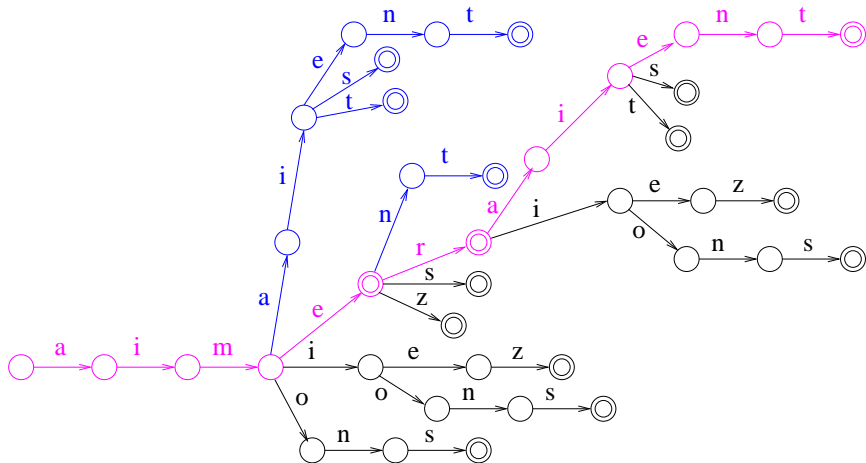
Są dwie możliwości:

- Dodajemy słowo do języka automatu i minimalizujemy całą zmienioną część automatu.
 - Brak ograniczeń na dane wejściowe.
 - Wielokrotne przetwarzania tych samych stanów.
- Dodajemy słowo do języka automatu i minimalizujemy tylko tę część, która już się nie zmienia.
 - Dane muszą być uporządkowane.
 - Duża szybkość bo stany przetwarzane jednokrotnie.

Synchronizacja – dane uporządkowane



Synchronizacja – dane uporządkowane



Przyrostowa konstr. z danych uporząd.

```
1:  function sorted_construction;
2:     $w' \leftarrow \epsilon$ ;
3:    while input not empty do
4:       $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:      while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:         $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:      end while;
8:      if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:      while  $i \leq |w|$  do
10:         $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:      end while;
12:       $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:    end while;
14:     $\text{repl\_or\_reg}(q_0, w')$ ;
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:    if  $v \neq \epsilon$  then
18:       $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
19:    end if;
20:    if  $\exists_{r \in R} r \equiv q$  then
21:      delete  $q$ ; return  $r$ ;
22:    else
23:       $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:    end if;
25:  end function;
```

Przyrostowa konstr. z danych uporząd.

```
1:  function sorted_construction;
2:     $w' \leftarrow \epsilon$ ;
3:    while input not empty do
4:       $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:      while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:         $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:      end while;
8:      if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:      while  $i \leq |w|$  do
10:         $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:      end while;
12:       $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:    end while;
14:    repl_or_reg( $q_0, w'$ );
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:    if  $v \neq \epsilon$  then
18:       $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
19:    end if;
20:    if  $\exists_{r \in R} r \equiv q$  then
21:      delete  $q$ ; return  $r$ ;
22:    else
23:       $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:    end if;
25:  end function;
```

Przyrostowa konstr. z danych uporząd.

```
1:  function sorted_construction;
2:       $w' \leftarrow \epsilon$ ;
3:      while input not empty do
4:           $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:          while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:               $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:          end while;
8:          if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:          while  $i \leq |w|$  do
10:              $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:          end while;
12:           $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:      end while;
14:      repl_or_reg( $q_0, w'$ );
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:      if  $v \neq \epsilon$  then
18:           $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\text{delta}(q, v_1), v_2\dots|v|)$ ;
19:      end if;
20:      if  $\exists_{r \in R} r \equiv q$  then
21:          delete  $q$ ; return  $r$ ;
22:      else
23:           $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:      end if;
25:  end function;
```

Przyrostowa konstr. z danych uporząd.

```
1:  function sorted_construction;
2:       $w' \leftarrow \epsilon$ ;
3:      while input not empty do
4:           $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:          while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:               $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:          end while;
8:          if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:          while  $i \leq |w|$  do
10:              $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:          end while;
12:           $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:      end while;
14:      repl_or_reg( $q_0, w'$ );
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:      if  $v \neq \epsilon$  then
18:           $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\text{delta}(q, v_1), v_2\dots|v|)$ ;
19:      end if;
20:      if  $\exists_{r \in R} r \equiv q$  then
21:          delete  $q$ ; return  $r$ ;
22:      else
23:           $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:      end if;
25:  end function;
```

Przyrostowa konstr. z danych uporząd.

```
1:  function sorted_construction;
2:       $w' \leftarrow \epsilon$ ;
3:      while input not empty do
4:           $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:          while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:               $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:          end while;
8:          if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:          while  $i \leq |w|$  do
10:              $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:          end while;
12:           $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:      end while;
14:      repl_or_reg( $q_0, w'$ );
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:      if  $v \neq \epsilon$  then
18:           $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2\dots|v|)$ ;
19:      end if;
20:      if  $\exists_{r \in R} r \equiv q$  then
21:          delete  $q$ ; return  $r$ ;
22:      else
23:           $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:      end if;
25:  end function;
```

Przyrostowa konstr. z danych uporząd.

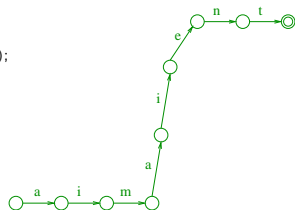
```
1:  function sorted_construction;
2:       $w' \leftarrow \epsilon$ ;
3:      while input not empty do
4:           $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:          while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:               $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:          end while;
8:          if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:          while  $i \leq |w|$  do
10:              $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:          end while;
12:           $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:      end while;
14:      repl_or_reg( $q_0, w'$ );
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:      if  $v \neq \epsilon$  then
18:           $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
19:      end if;
20:      if  $\exists_{r \in R} r \equiv q$  then
21:          delete  $q$ ; return  $r$ ;
22:      else
23:           $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:      end if;
25:  end function;
```


Przyrostowa konstr. z danych uporząd.

```
1:  function sorted_construction;
2:       $w' \leftarrow \epsilon$ ;
3:      while input not empty do
4:           $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:          while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:               $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:          end while;
8:          if  $i \leq |w'|$  then  $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ; end if;
9:          while  $i \leq |w|$  do
10:              $\delta(s, w_i) \leftarrow$  new state;  $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
11:          end while;
12:           $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
13:      end while;
14:       $\text{repl\_or\_reg}(q_0, w')$ ;
15:  end function;
16:  function repl_or_reg( $q, v$ );
17:      if  $v \neq \epsilon$  then
18:           $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
19:      end if;
20:      if  $\exists_{r \in R} r \equiv q$  then
21:          delete  $q$ ; return  $r$ ;
22:      else
23:           $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
24:      end if;
25:  end function;
```

Przyrostowa konstr. z danych uporządk.

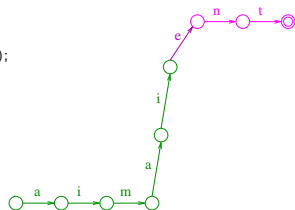
```
1:  function sorted_construction;
2:       $w' \leftarrow \epsilon$ ;
3:      while input not empty do
4:           $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:          while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:               $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:          end while;
8:          if  $i \leq |w'|$  then
9:               $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1} \dots |w'|)$ ;
10:          end if;
11:         while  $i \leq |w|$  do
12:              $\delta(s, w_i) \leftarrow$  new state;
13:              $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:         end while;
15:          $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:     end while;
17:     repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:     if  $v \neq \epsilon$  then
21:          $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2 \dots |v|)$ ;
22:     end if;
23:     if  $\exists_{r \in R} r \equiv q$  then
24:         delete  $q$ ; return  $r$ ;
25:     else
26:          $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:     end if;
28: end function;
```



aimaient

Przyrostowa konstr. z danych uporządk.

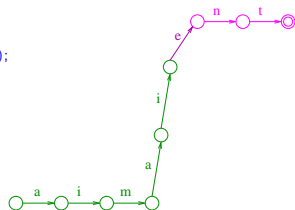
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

Przyrostowa konstr. z danych uporządk.

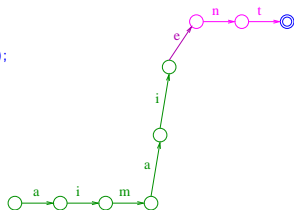
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

Przyrostowa konstr. z danych uporządk.

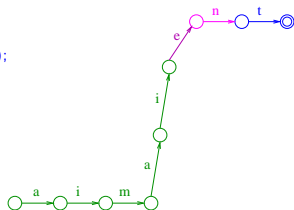
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

Przyrostowa konstr. z danych uporządk.

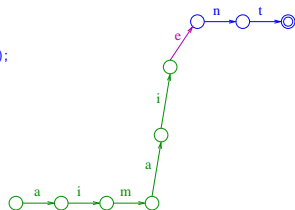
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

Przyrostowa konstr. z danych uporządk.

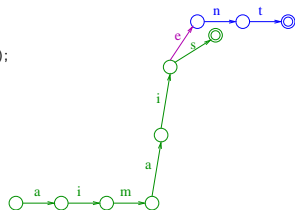
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

Przyrostowa konstr. z danych uporządk.

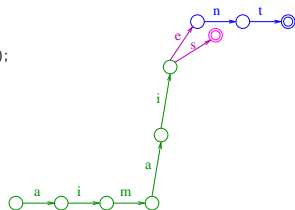
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimais

Przyrostowa konstr. z danych uporządk.

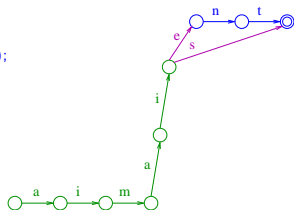
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1} \dots |w'|)$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2 \dots |v|)$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimait

Przyrostowa konstr. z danych uporządk.

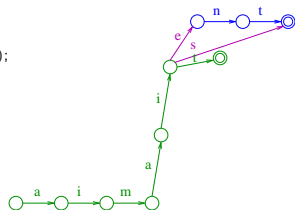
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1 \dots |w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimait

Przyrostowa konstr. z danych uporządk.

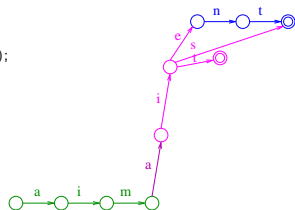
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1\dots|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2\dots|v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aimait

Przyrostowa konstr. z danych uporządk.

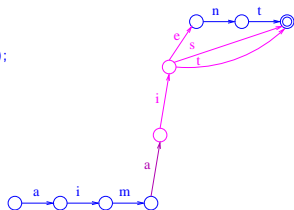
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1} \dots |w'|)$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17: function repl_or_reg( $q, v$ );
18:   if  $v \neq \epsilon$  then
19:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2 \dots |v|)$ ;
20:   end if;
21:   if  $\exists_{r \in R} r \equiv q$  then
22:     delete  $q$ ; return  $r$ ;
23:   else
24:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
25:   end if;
26: end function;
```



aime

Przyrostowa konstr. z danych uporządk.

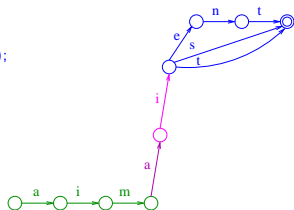
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1 \dots |w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

Przyrostowa konstr. z danych uporządk.

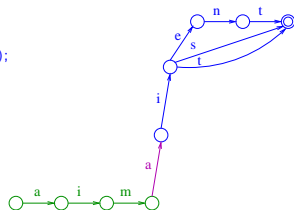
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1..|w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17: function repl_or_reg( $q, v$ );
18:   if  $v \neq \epsilon$  then
19:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2..|v|)$ ;
20:   end if;
21:   if  $\exists_{r \in R} r \equiv q$  then
22:     delete  $q$ ; return  $r$ ;
23:   else
24:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
25:   end if;
26: end function;
```



aime

Przyrostowa konstr. z danych uporządk.

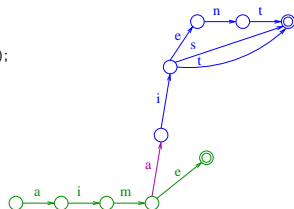
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1 \dots |w'|})$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_{2 \dots |v|})$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

Przyrostowa konstr. z danych uporządk.

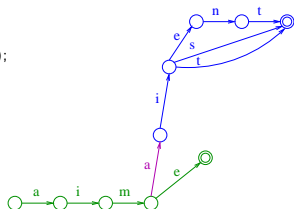
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1} \dots |w'|)$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2 \dots |v|)$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



aime

Przyrostowa konstr. z danych uporządk.

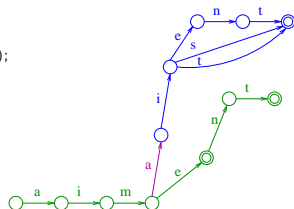
```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1} \dots |w'|)$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17: function repl_or_reg( $q, v$ );
18:   if  $v \neq \epsilon$  then
19:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2 \dots |v|)$ ;
20:   end if;
21:   if  $\exists_{r \in R} r \equiv q$  then
22:     delete  $q$ ; return  $r$ ;
23:   else
24:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
25:   end if;
26: end function;
```



aiment

Przyrostowa konstr. z danych uporządk.

```
1: function sorted_construction;
2:    $w' \leftarrow \epsilon$ ;
3:   while input not empty do
4:      $s \leftarrow q_0$ ;  $i \leftarrow 1$ ;  $w \leftarrow$  next word;
5:     while  $i \leq |w|$  and  $\delta(s, w_i) \neq \perp$  do
6:        $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
7:     end while;
8:     if  $i \leq |w'|$  then
9:        $\delta(s, w'_i) \leftarrow \text{repl\_or\_reg}(\delta(s, w'_i), w'_{i+1} \dots |w'|)$ ;
10:    end if;
11:    while  $i \leq |w|$  do
12:       $\delta(s, w_i) \leftarrow$  new state;
13:       $s \leftarrow \delta(s, w_i)$ ;  $i \leftarrow i + 1$ ;
14:    end while;
15:     $F \leftarrow F \cup \{s\}$ ;  $w' \leftarrow w$ ;
16:  end while;
17:  repl_or_reg( $q_0, w'$ );
18: end function;
19: function repl_or_reg( $q, v$ );
20:   if  $v \neq \epsilon$  then
21:      $\delta(q, v_1) \leftarrow \text{repl\_or\_reg}(\delta(q, v_1), v_2 \dots |v|)$ ;
22:   end if;
23:   if  $\exists_{r \in R} r \equiv q$  then
24:     delete  $q$ ; return  $r$ ;
25:   else
26:      $R \leftarrow R \cup \{q\}$ ; return  $q$ ;
27:   end if;
28: end function;
```



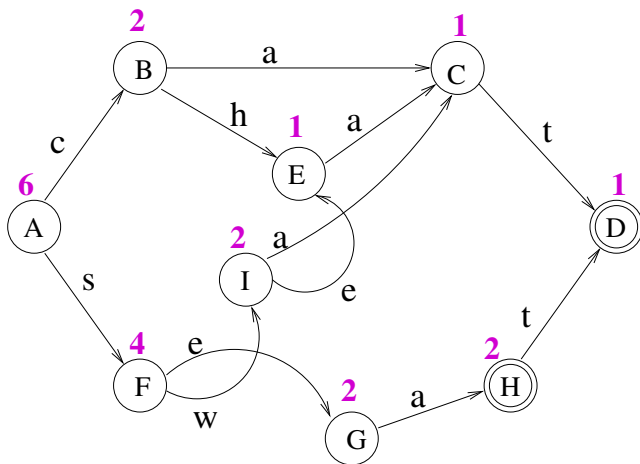
aiment

Nie wszystkie dane słownikowe – informacje związane ze słowami – da się upakować w automacie. Np. znaczenie słowa nie jest związane z jego końcówką, jest też różne dla różnych słów. W takim przypadku informacje przechowujemy poza automatem, a słowa numerujemy. Numer słowa jest indeksem określającym dodatkową informację. Numerowanie słów to realizacja minimalnej doskonałej funkcji mieszającej.

Doskonała funkcja mieszająca – funkcja mieszająca pozwalająca obliczyć indeks klucza zawsze za pierwszym razem (wykluczająca kolizje).

Minimalna doskonała funkcja mieszająca to doskonała funkcja mieszająca ze zbioru n kluczy na spójny przedział n liczb całkowitych.

Doskonała f. mieszająca – pierwsza metoda

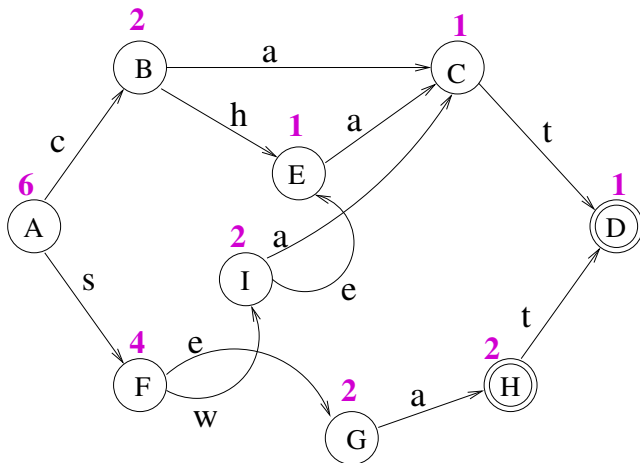


Doskonała f. mieszająca – obliczanie liczby słów w prawostronnym języku stanu

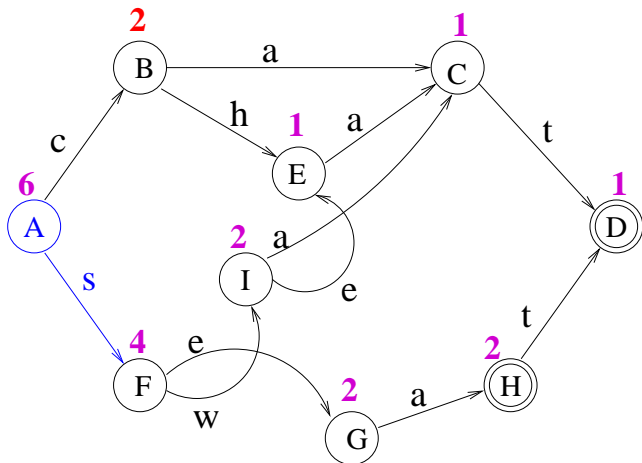
$$\vec{\mathcal{L}}(q) = \bigcup_{a:\delta(q,a)\neq\perp} a\vec{\mathcal{L}}(\delta(q,a)) \cup \begin{cases} \emptyset & q \notin F \\ \{\epsilon\} & q \in F \end{cases}$$

$$|\vec{\mathcal{L}}(q)| = \sum_{a:\delta(q,a)\neq\perp} |\vec{\mathcal{L}}(\delta(q,a))| + \begin{cases} 0 & q \notin F \\ 1 & q \in F \end{cases}$$

Doskonała f. mieszająca – pierwsza metoda

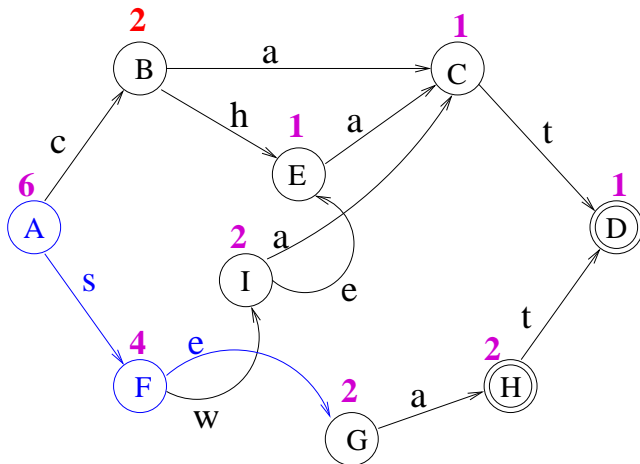


Doskonała f. mieszająca – pierwsza metoda



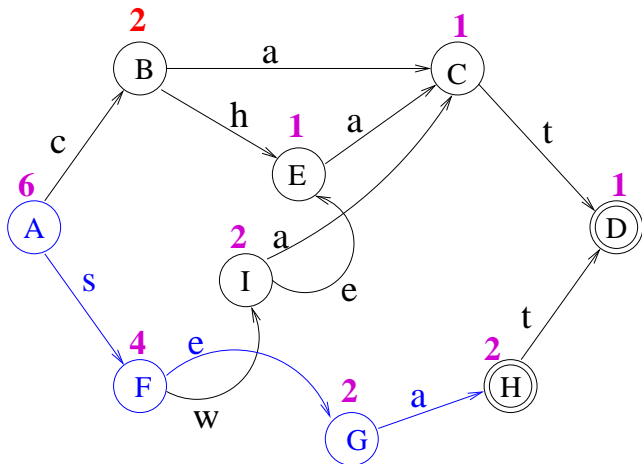
sea = 2

Doskonała f. mieszająca – pierwsza metoda



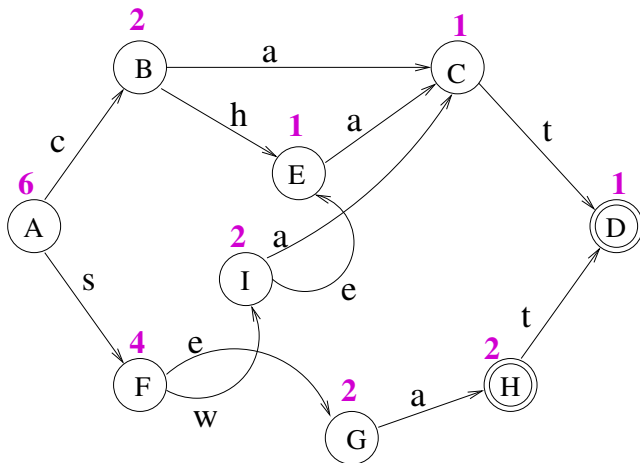
$$\text{sea} = 2 + 0$$

Doskonała f. mieszająca – pierwsza metoda



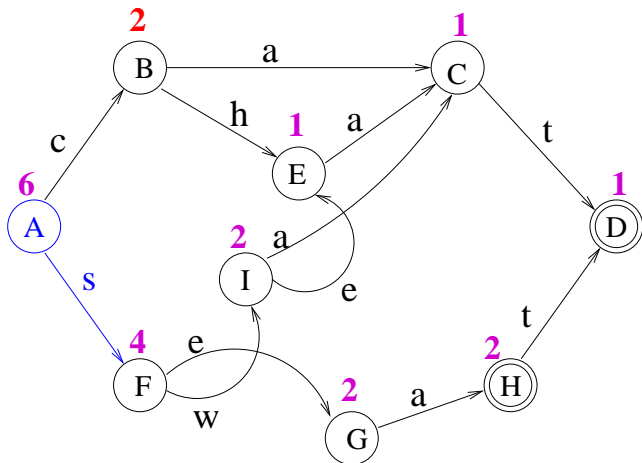
$$\text{sea} = 2 + 0 + 0 = 2$$

Doskonała f. mieszająca – pierwsza metoda



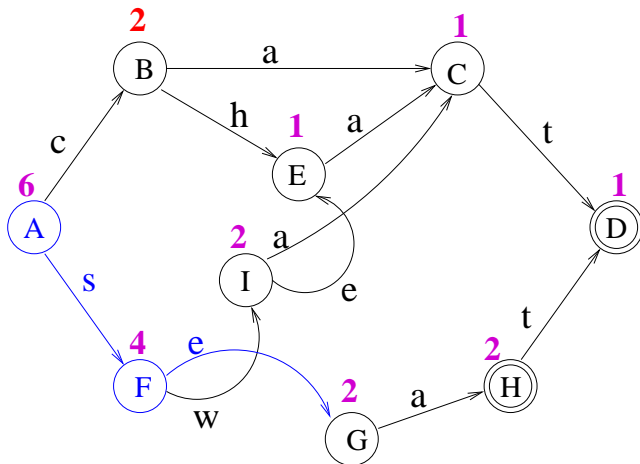
seat =

Doskonała f. mieszająca – pierwsza metoda



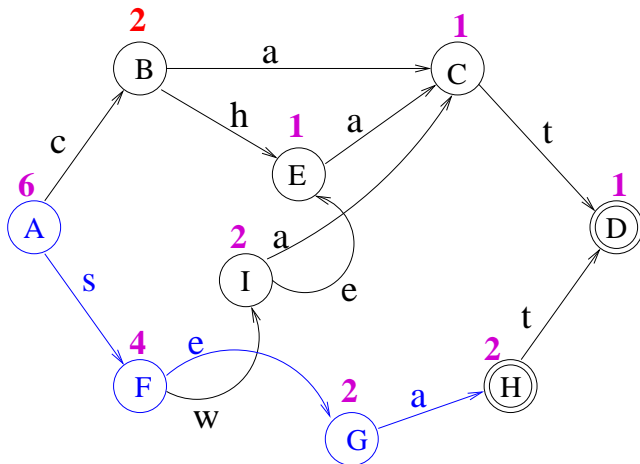
seat = 2

Doskonała f. mieszająca – pierwsza metoda



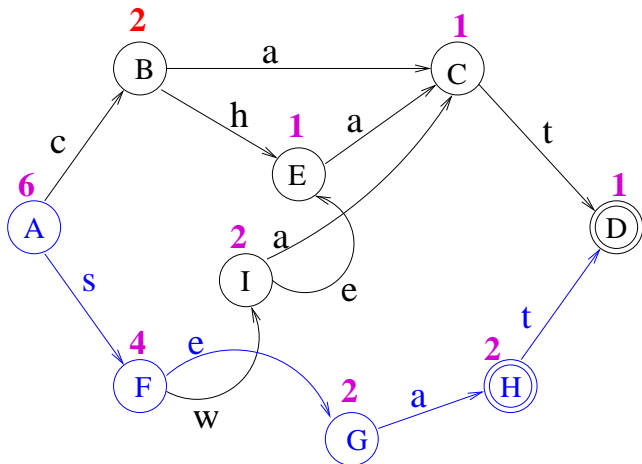
$$\text{seat} = 2 + 0$$

Doskonała f. mieszająca – pierwsza metoda



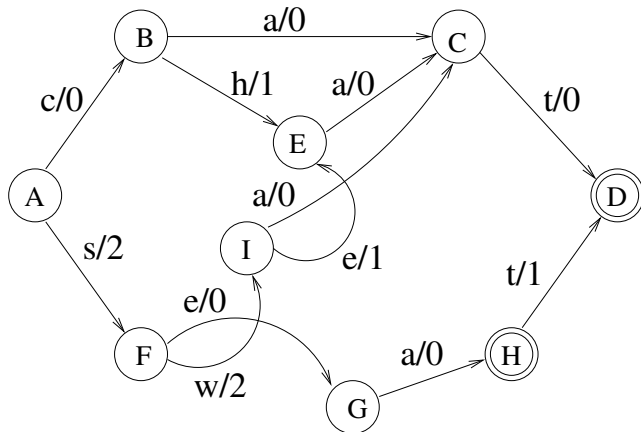
$$\text{seat} = 2 + 0 + 0$$

Doskonała f. mieszająca – pierwsza metoda

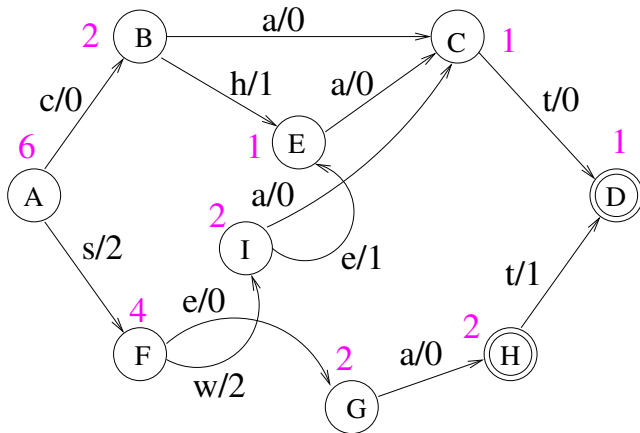


$$\text{seat} = 2 + 0 + 0 + 1 = 3$$

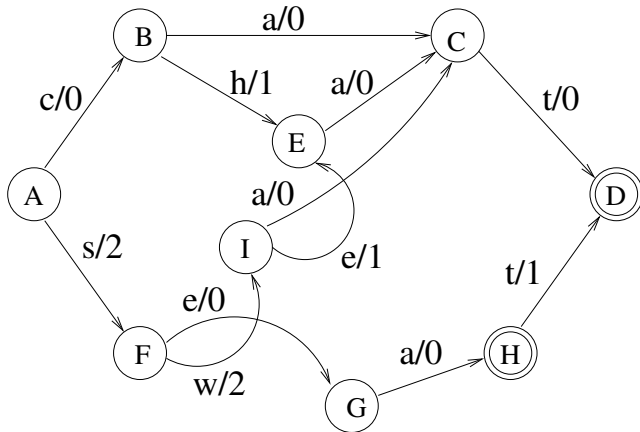
Doskonała f. mieszająca – druga metoda



Doskonała f. mieszająca – druga metoda

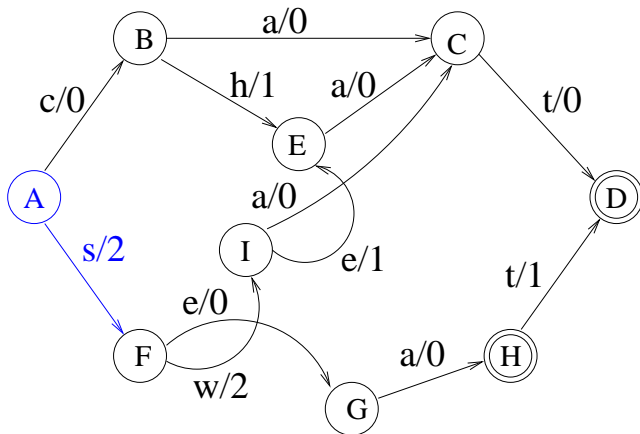


Doskonała f. mieszająca – druga metoda



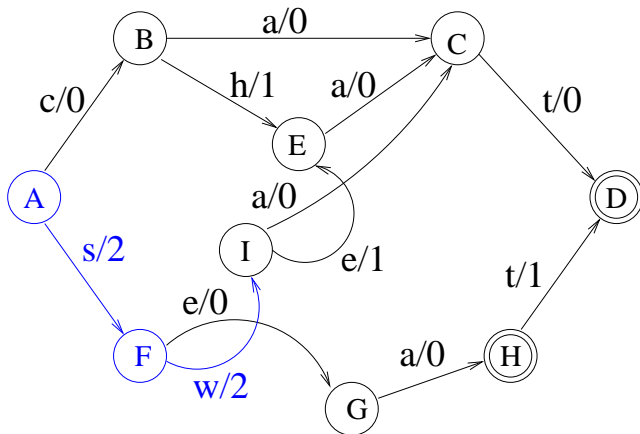
swat =

Doskonała f. mieszająca – druga metoda



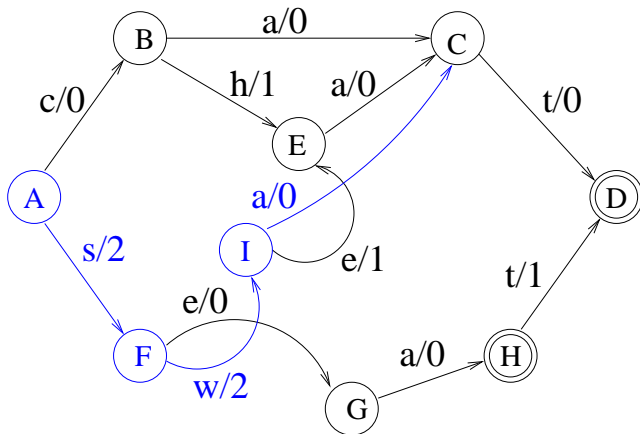
swat = 2

Doskonała f. mieszająca – druga metoda



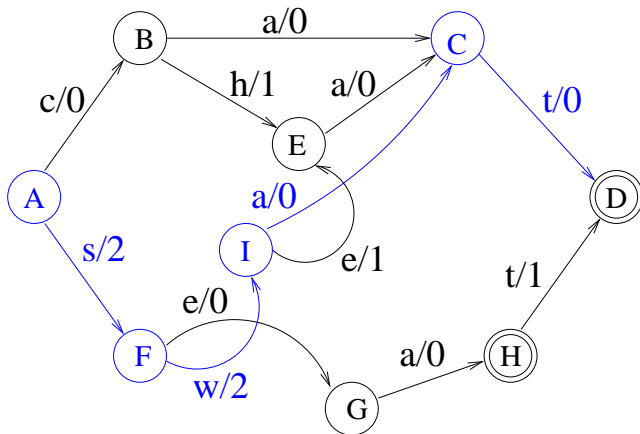
$$\text{swat} = 2 + 2$$

Doskonała f. mieszająca – druga metoda



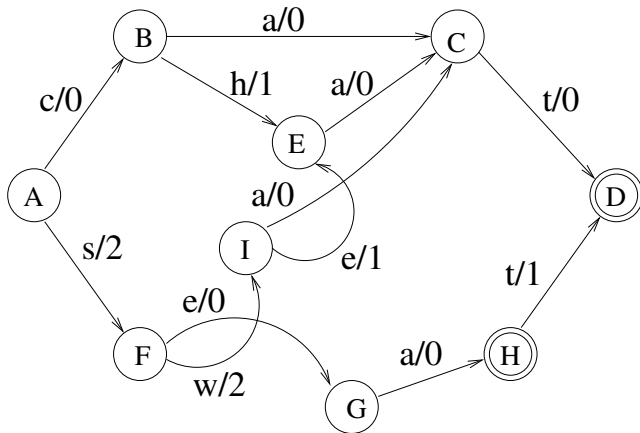
$$\text{swat} = 2 + 2 + 0$$

Doskonała f. mieszająca – druga metoda



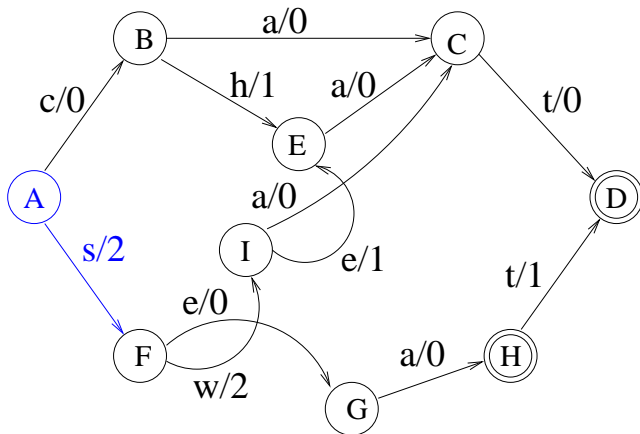
$$\text{swat} = 2 + 2 + 0 + 0 = 4$$

Doskonała f. mieszająca – druga metoda



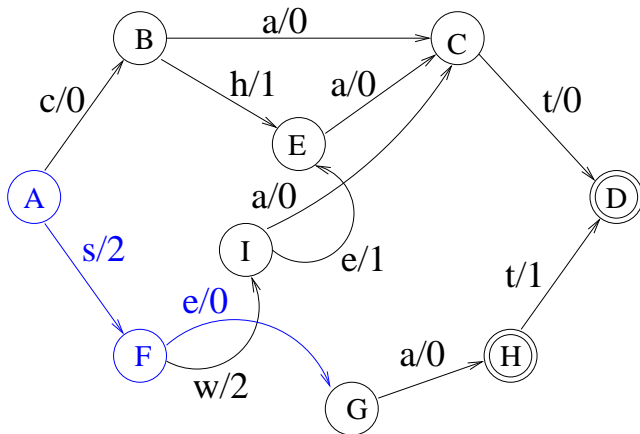
seat =

Doskonała f. mieszająca – druga metoda



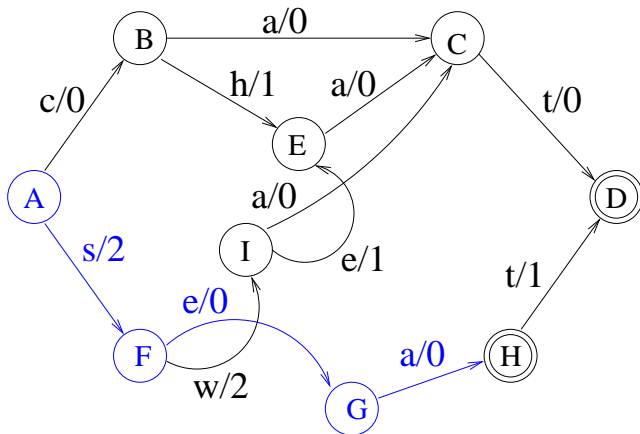
seat = 2

Doskonała f. mieszająca – druga metoda



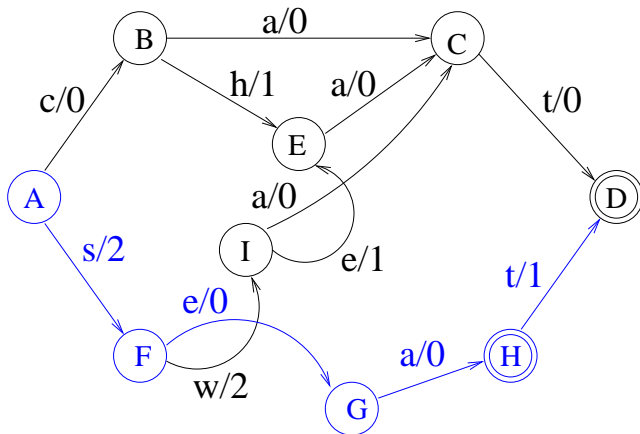
$$\text{seat} = 2 + 0$$

Doskonała f. mieszająca – druga metoda



$$\text{seat} = 2 + 0 + 0$$

Doskonała f. mieszająca – druga metoda



$$\text{seat} = 2 + 0 + 0 + 1 = 3$$

Literatura nt. przyrostowej budowy słowników i doskonałej funkcji mieszającej

- 1 Jan Daciuk, Stoyan Mihov, Bruce Watson, and Richard Watson, *Incremental Construction of Minimal Acyclic Finite State Automata*, Computational Linguistics, 26(1), pp. 3-16, March 2000. Dostępne przez: <http://acl.ldc.upenn.edu/J/J00/J00-1002.pdf>
- 2 Claudio Lucchiesi, Tomasz Kowaltowski, *Applications of Finite Automata Representing Large Vocabularies*, Software Practice and Experience, Jan. 1993, 23(1), pp. 15–30.
- 3 Jan Daciuk, Dawid Weiss, *Smaller Representation of Finite-State Automata*, Theoretical Computer Science, vol. 450, pp. 10-21, 7 Sept. 2012.