Politechnika Wrocławska

Struktury danych i złożoność obliczeniowa Wykłady 6., 7.

Prof. dr hab. inż. Jan Magott

Problemy „łatwe” i „trudne”

Problemy „łatwe” to problemy rozwiązywalne w czasie wielomianowym.

Problemy „trudne” to takie, których prawie na pewno nie można tak rozwiązać.

W teorii złożoności obliczeniowej fundamentalnym pojęciem jest problem decyzyjny.

Problem decyzyjny

Dane:

….

Pytanie:

Pytanie, na które odpowiedź brzmi „tak” lub „nie”.

Problem decyzyjny π składa się ze zbioru parametrów, które nie muszą mieć nadanych wartości oraz pytania, na które odpowiedź brzmi „tak” lub „nie”.

Po ustaleniu wartości wszystkich parametrów problemu π otrzymujemy **instancję** (**konkretny problem**), którą oznaczamy symbolem I.

Zbiór instancji problemu π oznaczać będziemy symbolem D_π.

Problem optymalizacyjny

Dane wejściowe:

X – dziedzina funkcji,

R – zbiór liczb rzeczywistych,

$f: X \to R$ - funkcja celu.

Zadanie:

Znaleźć $x^* \epsilon X$ takie, że $f(x^*) = \min_{x \epsilon X} f(x)$

lub $f(x^*) = \max_{x \epsilon X} f(x)$

Problem decyzyjny związany z problemem optymalizacyjnym

Problem optymalizacyjny

Dane:

…

Zadanie:

Znaleźć $x^* \in X$ takie, że $f(x^*) = min_{x \in X} f(x)$ ($f(x^*) = max_{x \in X} f(x)$)

Problem decyzyjny związany z problemem optymalizacyjnym

Dane:

j.w., $r \in R$

Pytanie:

Czy istnieje $x^{**} \in X$ takie, że $f(x^{**}) \leq r$ ($f(x^{**}) \geq r$) ?

Złożoność problemu decyzyjnego a złożoność problemu optymalizacyjnego

Z rozwiązania problemu optymalizacyjnego można wyznaczyć rozwiązanie problemu decyzyjnego.

Problem optymalizacyjny

Dane: …

Zadanie: Znaleźć $x^* \in X$ takie, że $f(x^*) = min_{x \in X} f(x)$

Problem decyzyjny związany z problemem optymalizacyjnym

Dane: j.w., $r \in R$

Pytanie: Czy istnieje $x^{**} \in X$ takie, że $f(x^{**}) \leq r$

Rozwiązaniem problemu decyzyjnego jest „tak” $\Leftrightarrow$ $f(x^*) \leq r$

W praktyce ważna jest optymalizacja.

Teoria złożoności obliczeniowej jest zbudowana na pojęciu problemu decyzyjnego.

Twierdzenie

Z każdym problemem optymalizacyjnym można związać odpowiadający mu problem decyzyjny.

Złożoność problemu decyzyjnego a złożoność problemu optymalizacyjnego

Jeśli problem optymalizacyjny jest „łatwy”, to problem decyzyjny też jest „łatwy” .

Jeśli problem decyzyjny jest „trudny” (NP-zupełny),
to odpowiadający mu problem optymalizacyjny jest „trudny”.

Zatem:

W celu wykazania „łatwości” problemu decyzyjnego wystarczy wykazać łatwość odpowiadającego mu problemu optymalizacyjnego.

Dla wykazania „trudności” problemu optymalizacyjnego wystarczy wykazać trudność związanego z nim problemu decyzyjnego.

Intuicyjne pojmowanie klas P i NP

Klasa P zawiera te wszystkie problemy decyzyjne, dla których znaleziono wielomianowe czasowo algorytmy ich rozwiązania.

Klasa NP zawiera te wszystkie problemy decyzyjne, dla których znaleziono niedeterministyczne algorytmy wielomianowe ich rozwiązania (w wielomianowym czasie można odgadnąć rozwiązanie i sprawdzić czy to rozwiązanie daje odpowiedź "tak").

Problem decyzyjny π_1 jest nazywany ***NP-zupełnym***, jeśli:

1. $\pi_1 \epsilon NP$,
2. Dla każdego innego problemu decyzyjnego $\pi_2 \epsilon NP$ jest $\pi_2 \propto \pi_1$.

Zatem, jeśli istniałby algorytm wielomianowy do rozwiązywania jakiegokolwiek problemu NP-zupełnego, to każdy problem z klasy NP (w tym również problemy NP-zupełne) mógłby być rozwiązany za pomocą algorytmu wielomianowego.

Z bezskuteczności poszukiwań algorytmu wielomianowego dla któregokolwiek problemu NP-zupełnego wynika, że **prawie na pewno wszystkie problemy NP-zupełne można rozwiązać tylko przy użyciu algorytmów ponadwielomianowych**.

Decyzyjny problem szeregowania zadań niezależnych na dwu maszynach (procesorach)

Dane:

- $T = \{t_1, \ldots, t_i, \ldots, t_k\}$ - zbiór czasów wykonania k zadań niezależnych (bez ograniczeń kolejności wykonywania) gdzie $t_i \in N_+$,
- $B \in N_+$,
- $\sum_{i=1}^{k} t_i = 2B$.

Pytanie:

Czy można te zadania tak rozłożyć na dwie maszyny, że sumaryczny czas wykonania będzie równy B ?

Optymalizacyjny problem szeregowania zadań niezależnych na dwu maszynach (procesorach)

Dane:

- $T = \{t_1, \ldots, t_i, \ldots, t_k\}$ - zbiór czasów wykonania k zadań niezależnych (bez ograniczeń kolejności wykonywania) gdzie $t_i \in N_+$,
- $B \in N_+$,
- $\sum_{i=1}^{k} t_i = 2B$.

Zadanie:

Wyznaczyć minimalny czas wykonania.

Problem podziału zbioru

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+$,
- $\sum_{i=1}^{k} x_i = 2B$.

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że

$$\sum_{x_i \in X_1} x_i = B \ ?$$

Z ***Problemem podziału zbioru*** można skojarzyć następujący problem optymalizacyjny

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+$,
- $\sum_{i=1}^{k} x_i = 2B$.

Pytanie:

Wyznaczyć podzbiór $X_1 \subset X$ taki, że

$$min_{X_1 \in 2^X} |(\sum_{x_i \in X_1} x_i) - B|$$

gdzie 2^X jest zbiorem potęgowym (zbiorem wszystkich podzbiorów) zbioru X.

Przykłady problemów decyzyjnych

- Czy liczba naturalna n jest liczbą pierwszą?
- Problem podziału zbioru
- Problem spełnialności wyrażeń logicznych

Problem podziału zbioru jest przykładem problemu decyzyjnego

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+$,
- $\sum_{i=1}^{k} x_i = 2B$.

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że

$$\sum_{x_i \in X_1} x_i = B \ ?$$

Problem spełnialności wyrażeń logicznych

(ang. Satisfability problem)

Dane:

Funkcja boolowska n zmiennych logicznych

$$f(x_1, x_2, \dots, x_n): \{0,1\}^n \rightarrow \{0,1\}$$

Pytanie:

Czy istnieje przypisanie wartości 0 i 1 („false” i „true”) do zmiennych $x_1, x_2, \dots, x_n$ takie, że

$$f(x_1, x_2, \dots, x_n) = 1 \quad ?$$

Przykłady problemów optymalizacyjnych

Problem programowania liniowego
Problem plecakowy
Problem komiwojażera

Problem programowania liniowego

Dane:

Wektory $c_{1\times n}$, $b_{m\times 1}$, $x_{n\times 1}$, macierz $A_{m\times n}$

o składowych rzeczywistych

Funkcja celu $f(x) = c \cdot x$

Ograniczenia $A \cdot x \leq b$

Zadanie:

Wyznaczyć $x^* \in X$ takie, że $f(x^*) = max_{x\in X} f(x)$

Przykład problemu programowania liniowego

Wektory $c_{1\times n}$, $b_{m\times 1}$, $x_{n\times 1}$, macierz $A_{m\times n}$
o składowych rzeczywistych

Funkcja celu $f(x) = c \cdot x$

c_j - zysk z wyprodukowania jednej jednostki j-tego produktu,

x_j - liczba jednostek j-tego produktu,

$c_j \cdot x_j$ - zysk z wyprodukowania x_j jednostek j-tego produktu,

$c \cdot x$ – zysk całkowity,

Przykład problemu programowania liniowego

Wektory $c_{1\times n}$, $b_{m\times 1}$, $x_{n\times 1}$, macierz $A_{m\times n}$
o składowych rzeczywistych

Ograniczenia $A \cdot x \leq b$

b_i - liczba jednostek i-tego materiału do produkcji,

a_{ij} - liczba jednostek i-tego materiału potrzebna do wyprodukowania jednostki j-tego produktu

$a_{ij} \cdot x_j$ - liczba jednostek i-tego materiału potrzebna do wyprodukowania x_j jednostek j-tego produktu,

$a_{i.} \cdot x$ - liczba jednostek i-tego materiału potrzebna do wyprodukowania wolumenu x poszczególnych produktów,

$a_{i.} \cdot x \leq b_i$ - ograniczenie zużycia i-tego materiału

Problem programowania liniowego

Dane:

Wektory $c_{1\times n}$, $b_{m\times 1}$, $x_{n\times 1}$, macierz $A_{m\times n}$

o składowych rzeczywistych

Funkcja celu $f(x) = c \cdot x$

Ograniczenia $A \cdot x \leq b$

Zadanie:

Wyznaczyć $x^* \in X$ takie, że $f(x^*) = max_{x\in X} f(x)$

Algorytmy rozwiązujące problem programowania liniowego

- Algorytm simplex (niewielomianowy)
- Algorytm Khachiana (wielomianowy) 1979
- Algorytm Karmarkara (wielomianowy) 1984

W praktyce, często dla tych samych danych, Algorytm Khachiana wykonywany jest dłużej niż algorytm simplex.

W praktyce, często algorytmy inspirowane algorytmem Karmarkara mają podobną złożoność jak algorytm simplex.

Dyskretny problem plecakowy -wersja decyzyjna

Dane:

Skończony zbiór elementów $A = \{a_1, a_2, \dots, a_n\}$.

Rozmiar $s(a_i) > 0$ i wartość $w(a_i) > 0$ elementu a_i.

Pojemność plecaka $b > 0$ i stała $y > 0$.

Zadanie:

Czy istnieje podzbiór $A' \subset A$ taki, że:

$$\sum_{a_i \in A'} s(a_i) \leq b$$

$$\sum_{a_i \in A'} w(a_i) \geq y \quad ?$$

Dyskretny problem plecakowy - wersja optymalizacyjna

Dane:

Skończony zbiór elementów $A = \{a_1, a_2, \dots, a_n\}$.

Rozmiar $s(a_i) > 0$ i wartość $\mathrm{w}(a_i) > 0$ elementu a_i .

Pojemność plecaka $b > 0$.

Zadanie:

Czy istnieje podzbiór $A' \subset A$ taki, że:

$$\sum_{a_i \in A'} s(a_i) \leq b$$

$$max_{A' \subset A} \sum_{a_i \in A'} w(a_i)$$

Problem komiwojażera

Dane:

Zbiór miast $C = \{c_1, c_2, \ldots, c_m\}$, odległości $d_{ij} \epsilon N_+$ z miasta c_i do c_j i ograniczenie $b \epsilon N$.

(Nie ma wymogu $d_{ij} = d_{ji}$)

Pytanie:

Znaleźć kolejność odwiedzania wszystkich miast (permutację) $< c_{i[1]}, c_{i[2]}, \ldots, c_{i[m]} >$, gdzie $c_{i[j]}$ jest miastem odwiedzanym jako j-te, taką, że:

$$\sum_{j=1}^{m-1} d_{i[j],i[j+1]} + d_{i[m],i[1]} \leq b \, ?$$

Notacja o

Funkcja $f(n)$ ***jest*** $o(g(n))$, jeśli

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Uporządkowanie przykładowych ciągów

$$1, logn, \dots, \sqrt{n}, n, n\log n, n\sqrt{n}, n^2, n^3, \dots, 2^n, n!, n^n$$

L, P - ciąg, ciąg na prawo od niego

$$L = o(P)$$

Funkcją czasowej złożoności obliczeniowej (***złożonością obliczeniową***) algorytmu A rozwiązującego problem π jest funkcja przyporządkowująca każdej wartości rozmiaru $N(I)$ instancji $I \epsilon D_\pi$ maksymalną liczbę elementarnych kroków (lub jednostek czasu) komputera potrzebną do rozwiązania instancji o tym rozmiarze za pomocą algorytmu A .

czasowa

Algorytmem wielomianowym nazywamy algorytm, którego złożoność obliczeniowa jest $O(p(k))$, gdzie p jest pewnym wielomianem, k - rozmiarem rozwiązywanej instancji.

Każdy algorytm, którego złożoność nie może być tak ograniczona – nazywany jest **ponadwielomianowym.**

W praktyce korzystnie jest wyznaczyć rozmiar problemu za pomocą **jednego**, **dwu**, rzadko kiedy większej liczby **parametrów** określających liczbę elementów zbioru istotnego dla danego problemu.

Złożoność czasowa a złożoność pamięciowa algorytmów.

Częsty konflikt między kierunkami minimalizacji obu powyższych złożoności.

Modele obliczeń:

- Maszyna Turinga,
- Maszyna RAM,
- Programy komputerowe.

Porównanie modeli obliczeń.

Pojęcia teorii złożoności obliczeniowej są formułowane za pomocą maszyn Turinga:

- deterministycznej,
- niedeterministycznej,

które są modelami formalnymi obliczeń.

Alan Turing (1912-1954)

Brytyjski matematyk, filozof i kryptolog.

Twórca modelu formalnego komputera zwanego maszyną Turinga.

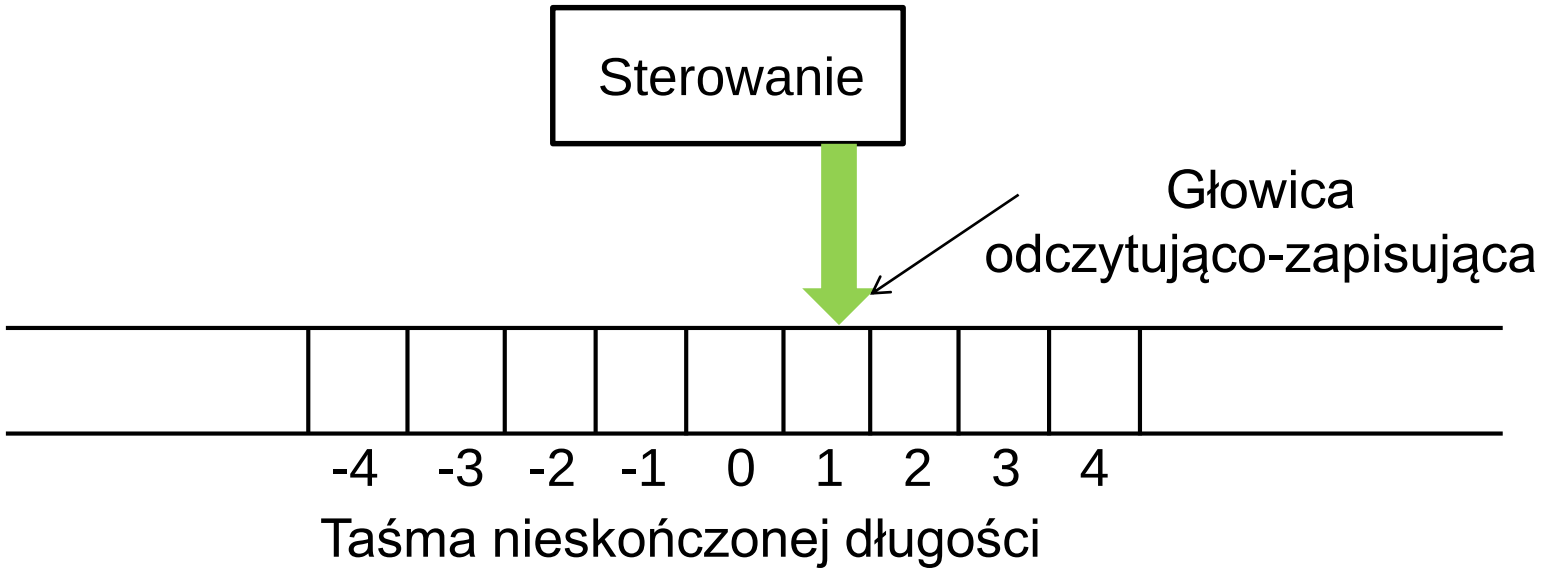
„Ojciec informatyki”

Twórca testu sprawdzającego czy maszyna jest inteligentna.

Test Turinga

Arbiter rozmawia w języku naturalnym z bytem, o którym nie wie czy jest człowiekiem czy maszyną. Jeśli o bycie, który jest maszyną nie potrafi powiedzieć czy jest człowiekiem czy maszyną, to maszyna przeszła test.

Deterministyczna jednotaśmowa maszyna Turinga (DMT)



Program dla DMT składa się z:

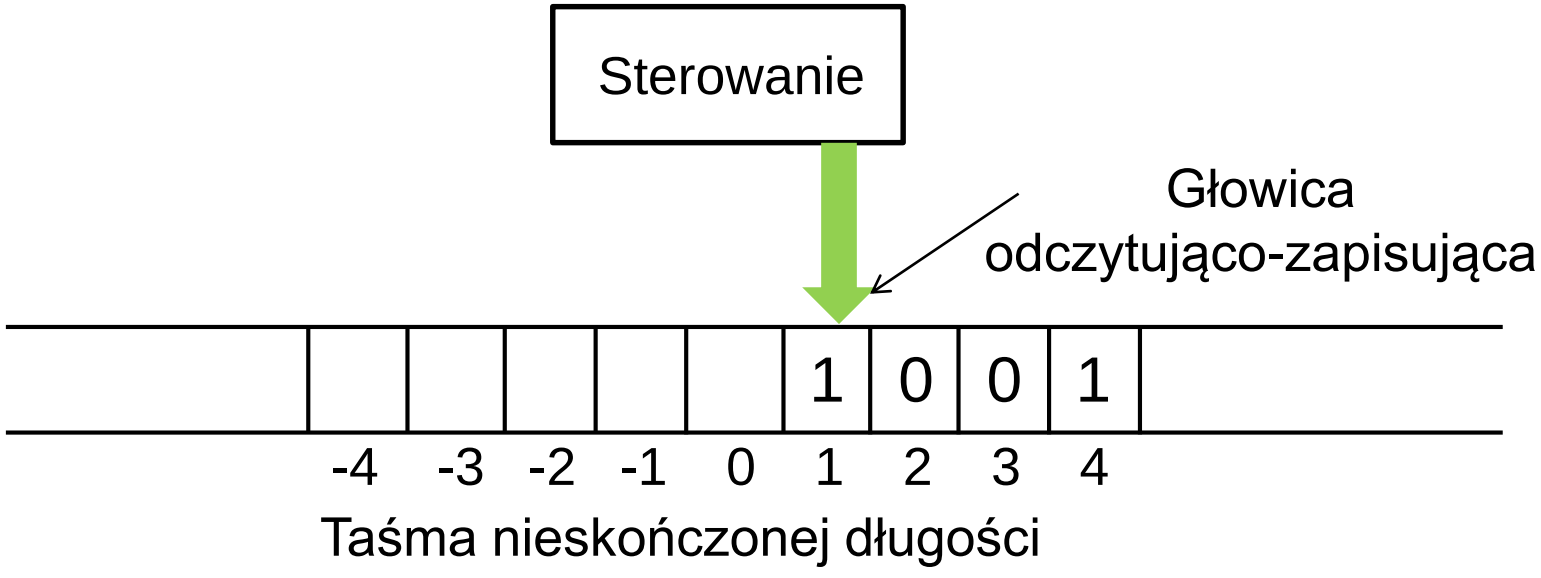
Skończonego zbioru symboli taśmy $\sum$ zawierającego separator (symbol pusty) $\sqcup$,

Skończonego zbioru stanów $Q = \{q_0, \dots, q_m\}$ zawierającego stan początkowy q_0, dwa wyróżnione stany końcowe q_Y (odpowiedź „tak”) i q_N (odpowiedź „nie”),

Funkcji przejść $\delta: (Q \backslash \{q_Y, q_N\}) \times \sum \rightarrow Q \times \sum \times \{-1, 0, +1\}$.

Dane programu na DMT

Pierwsze słowo x (skończony ciąg symboli) ze zbioru $\sum \setminus \{\sqcup\}$ zapisane po jednym symbolu w kolejnych komórkach taśmy o numerach od 1 do $|x|$. Kolejne słowa są oddzielone separatorami $\sqcup$.

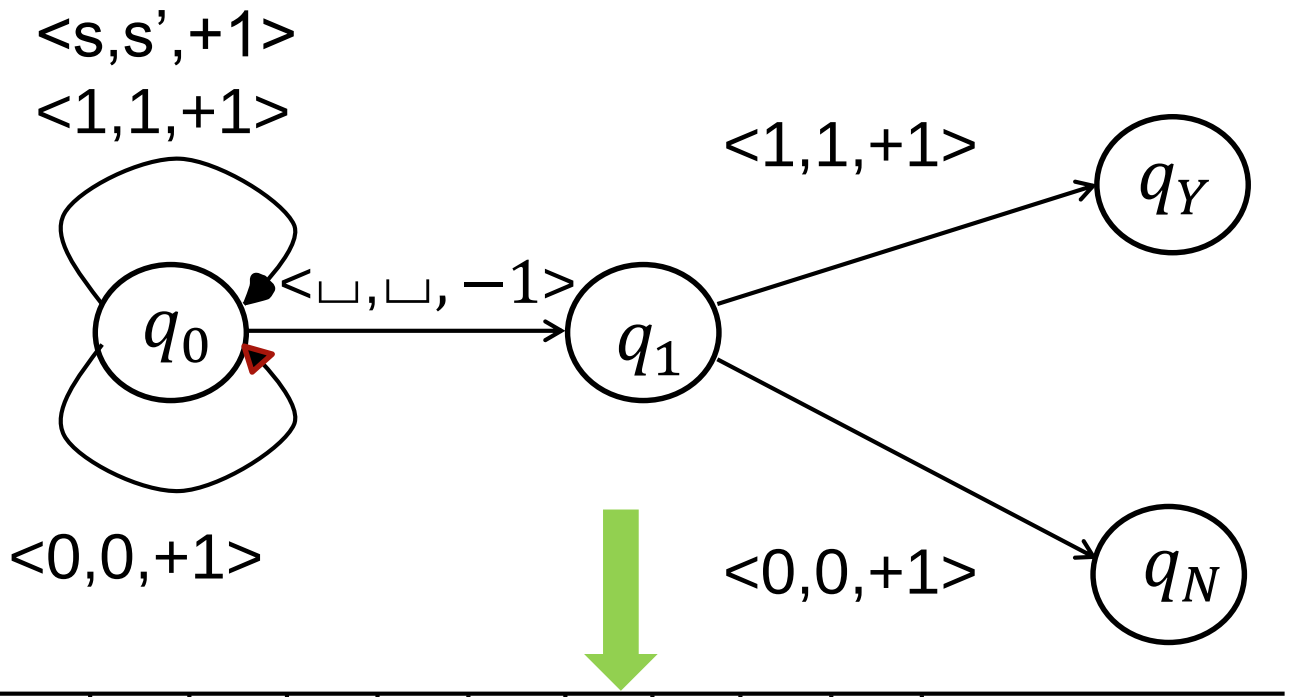


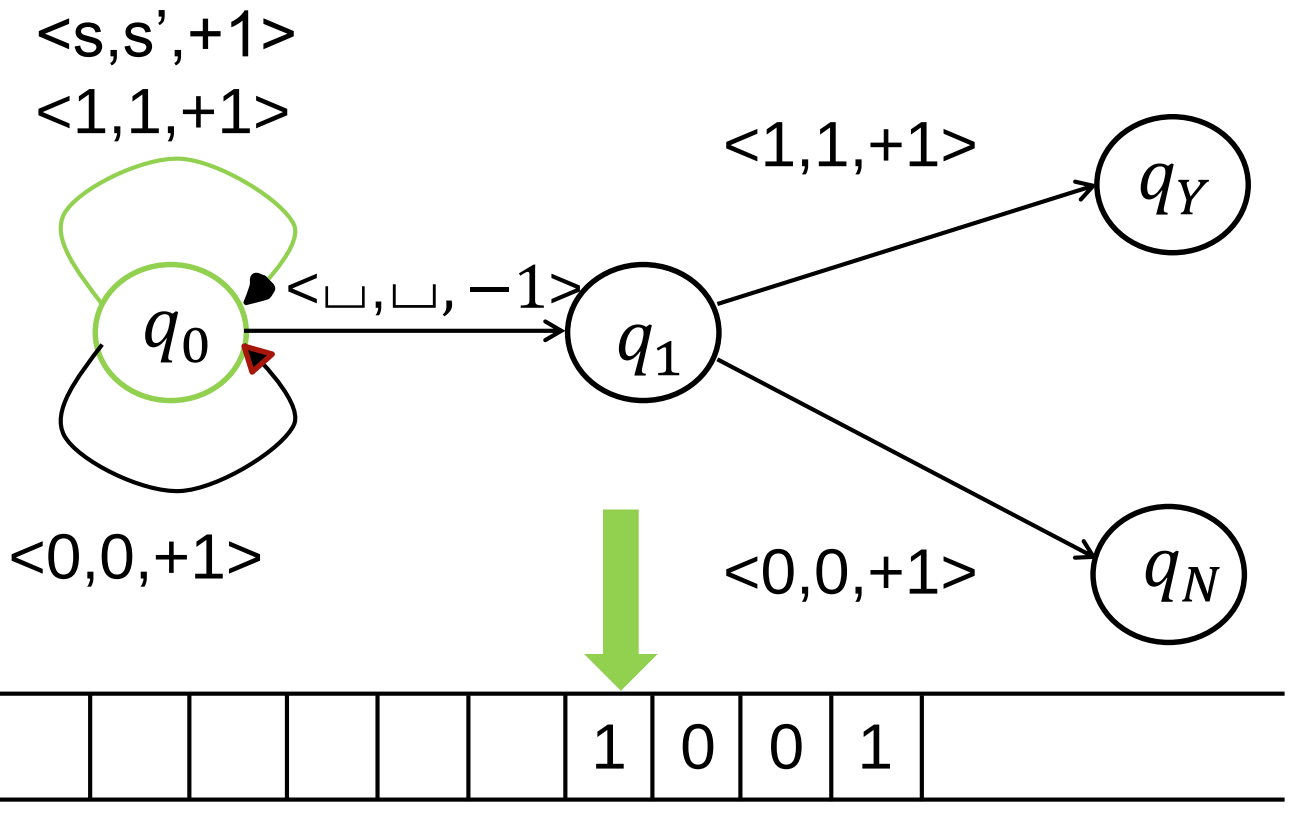
Wykonanie programu DMT

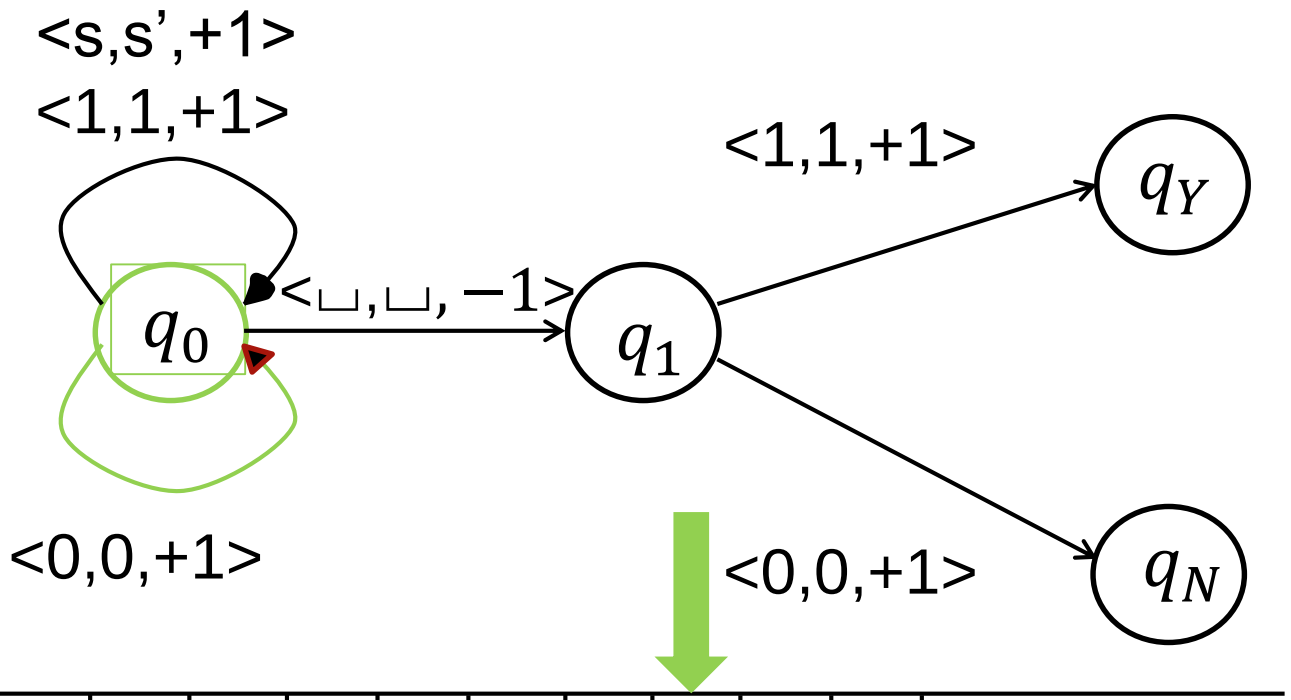
1. Maszyna znajduje się w stanie początkowym q_0, natomiast głowica odczytuje symbol s z komórki o numerze 1.
2. Jeśli DMT znajduje się w stanie $q \epsilon Q \backslash \{q_Y, q_N\}$, a w komórce, nad która jest głowica jest symbol $s \epsilon \sum$, to maszyna wykonuje czynności określone funkcją przejścia $\delta(q, s) = (q', s', \Delta)$:
   1. Głowica w miejsce symbolu s wpisuje s',
   2. Głowica przesuwa się o jedną komórkę: w lewo jeśli $\Delta = -1$, w prawo jeśli $\Delta = +1$, nie przesuwa się jeśli $\Delta = 0$,
   3. Stan DTM zmienia się na q'.
3. Wykonywanie trwa do czasu gdy DTM znajdzie się w stanie końcowym.

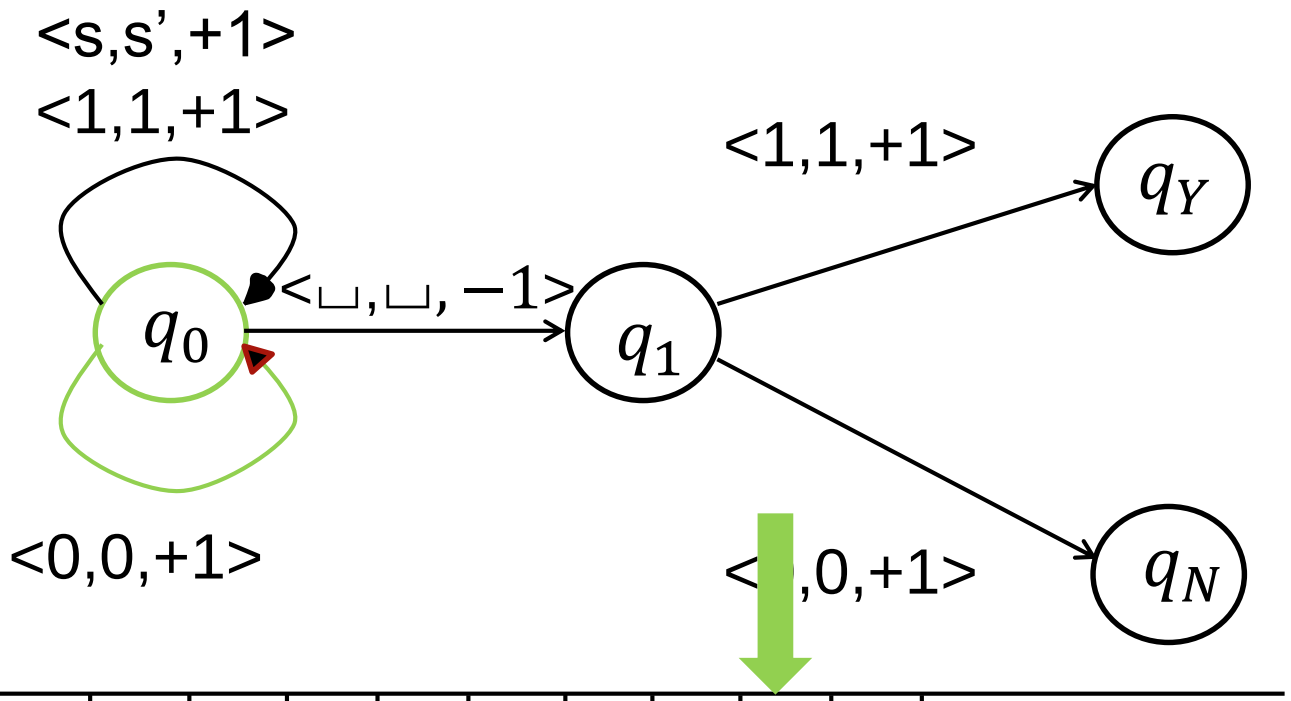
DTM rozwiązuje problem decyzyjny π przy kodowaniu e, jeśli zatrzymuje się dla wszystkich ciągów danych wejściowych i kończy obliczenia w stanie q_Y dla wszystkich danych wejściowych $x(I)$ instancji I takich, że $I \epsilon Y_\pi$ i tylko dla nich.

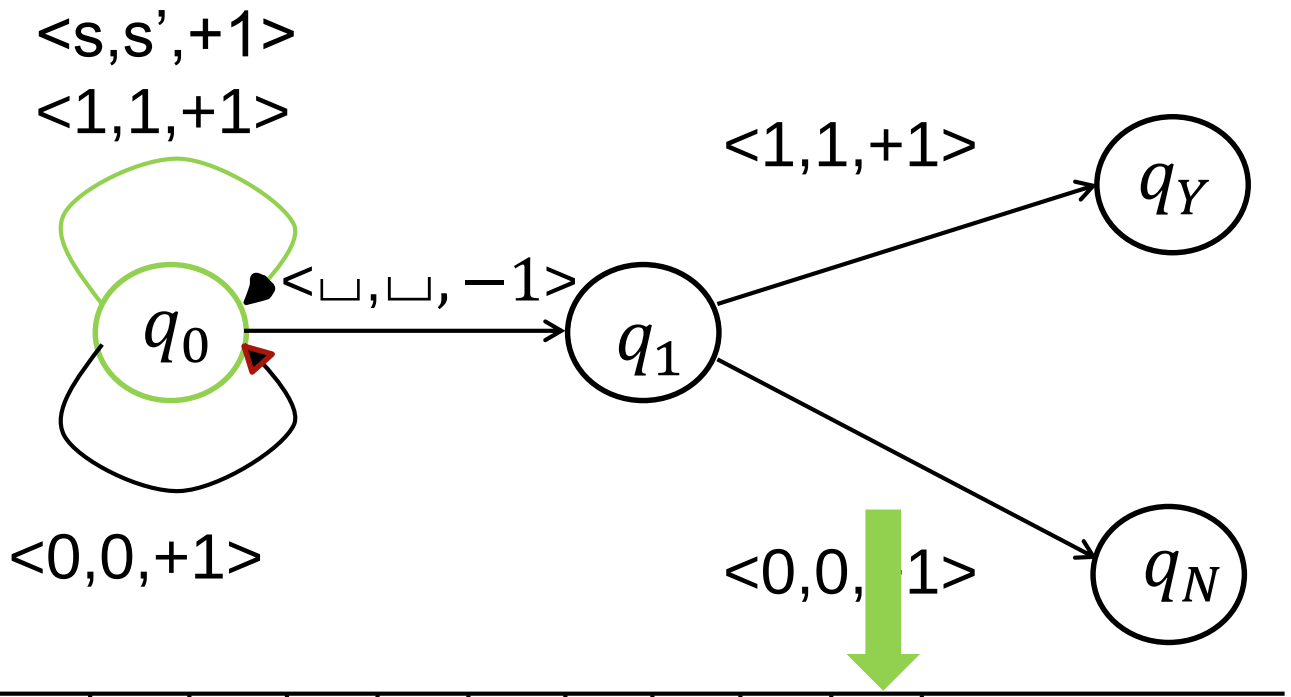
Y_π - zbiór wszystkich instancji problemu decyzyjnego π, dla których odpowiedzią jest „tak”.

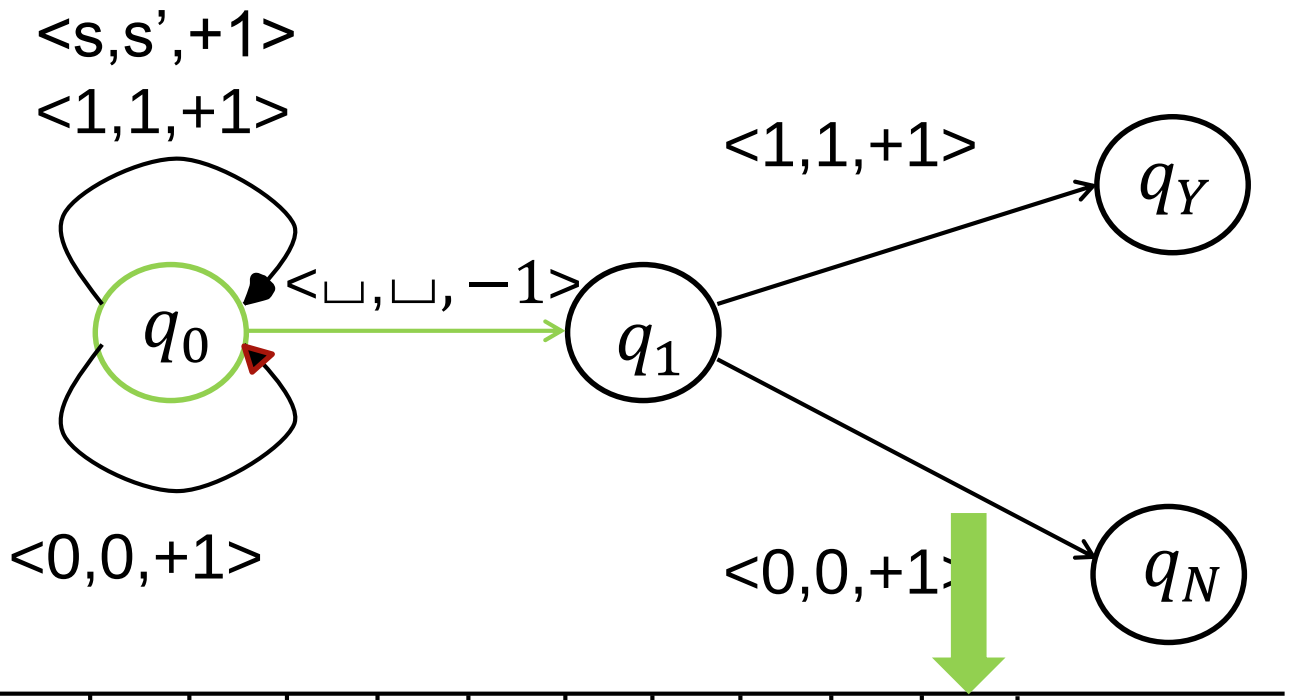

<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
<␣,␣, −1>
q_0
q_1
<0,0,+1>
<0,0,+1>
q_N
1
0
0
1

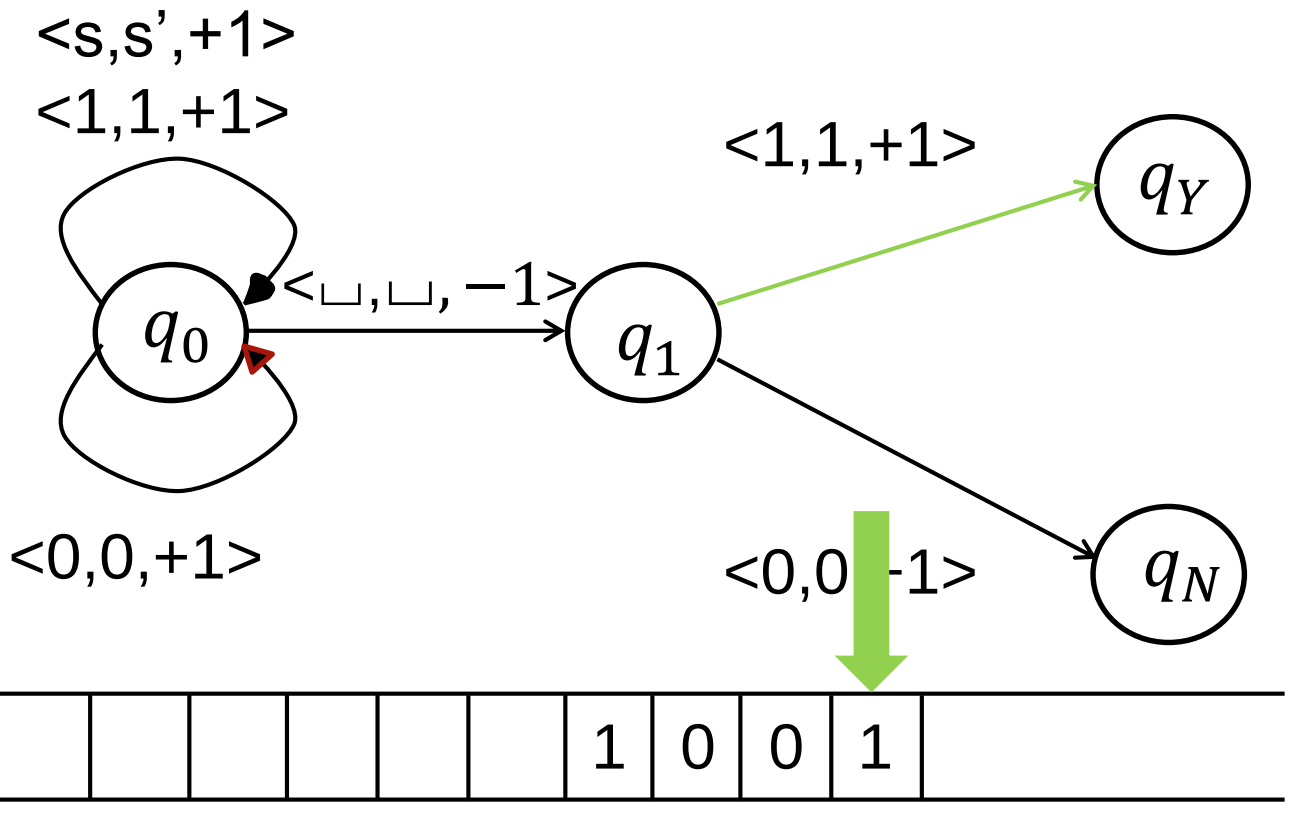

<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
<␣,␣, −1>
q_0
q_1
<0,0,+1>
<0,0,+1>
q_N
1
0
0
1

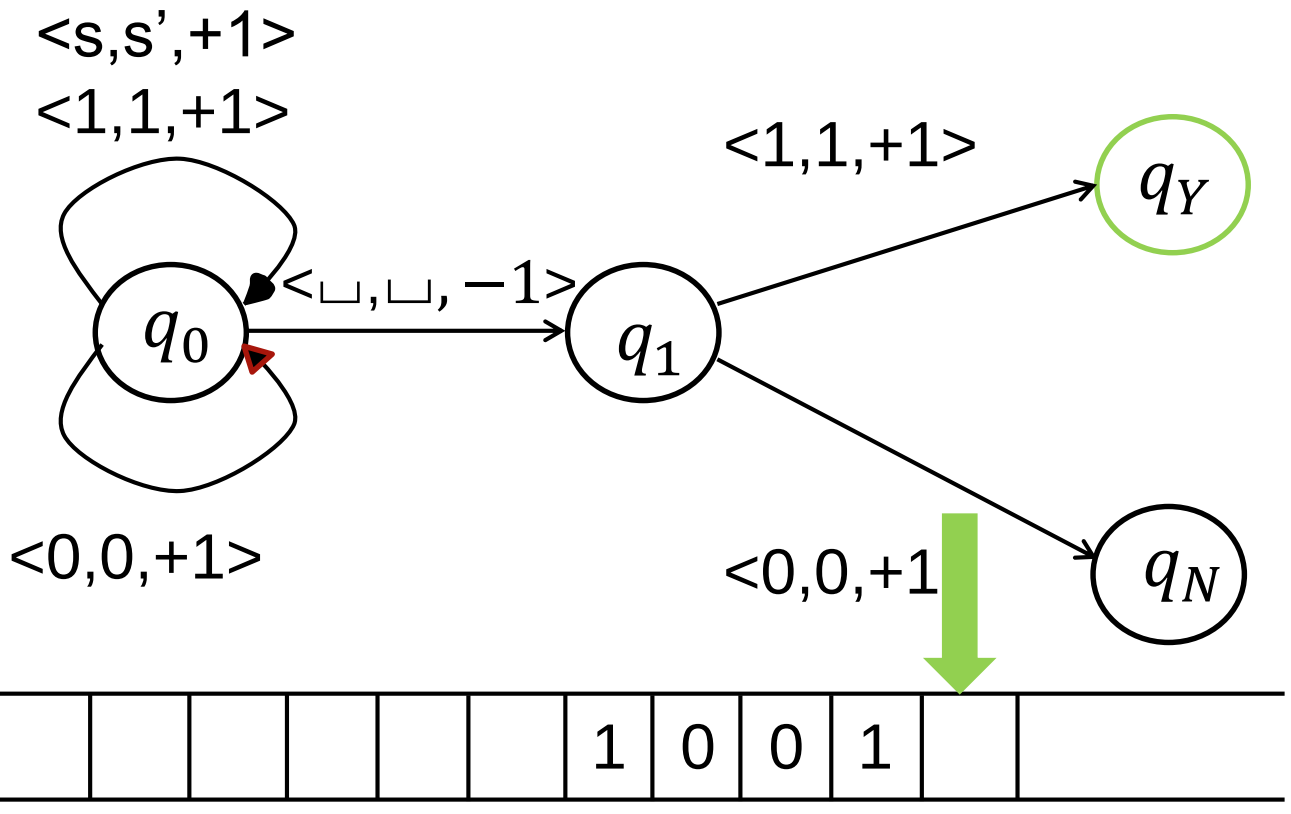

<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
q_0
<␣,␣, −1>
q_1
<0,0,+1>
<0,0,+1>
q_N
1
0
0
1


<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
q_0
<␣,␣, −1>
q_1
<0,0,+1>
<,0,+1>
q_N
1
0
0
1

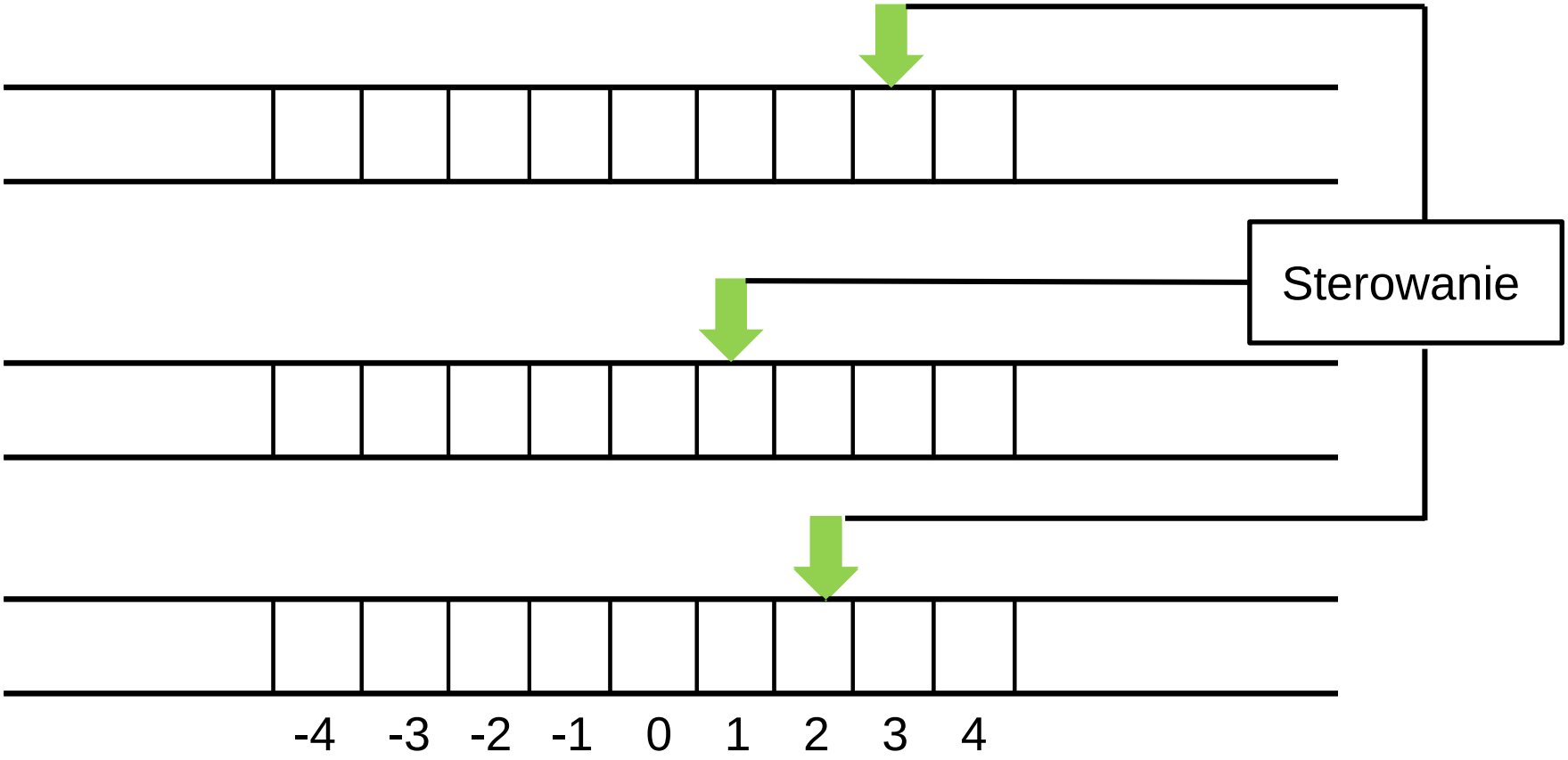

<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
<␣,␣, −1>
q_0
q_1
<0,0,+1>
<0,0,
1>
q_N
1
0
0
1


<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
<␣,␣, −1>
q_0
q_1
<0,0,+1>
<0,0,+1>
q_N
1
0
0
1


<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
<␣,␣, −1>
q_0
q_1
<0,0,+1>
<0,0 -1>
q_N
1
0
0
1


<s,s',+1>
<1,1,+1>
<1,1,+1>
q_Y
<␣,␣,−1>
q_0
q_1
<0,0,+1>
<0,0,+1
q_N
1
0
0
1

Deterministyczna k-taśmowa DMT



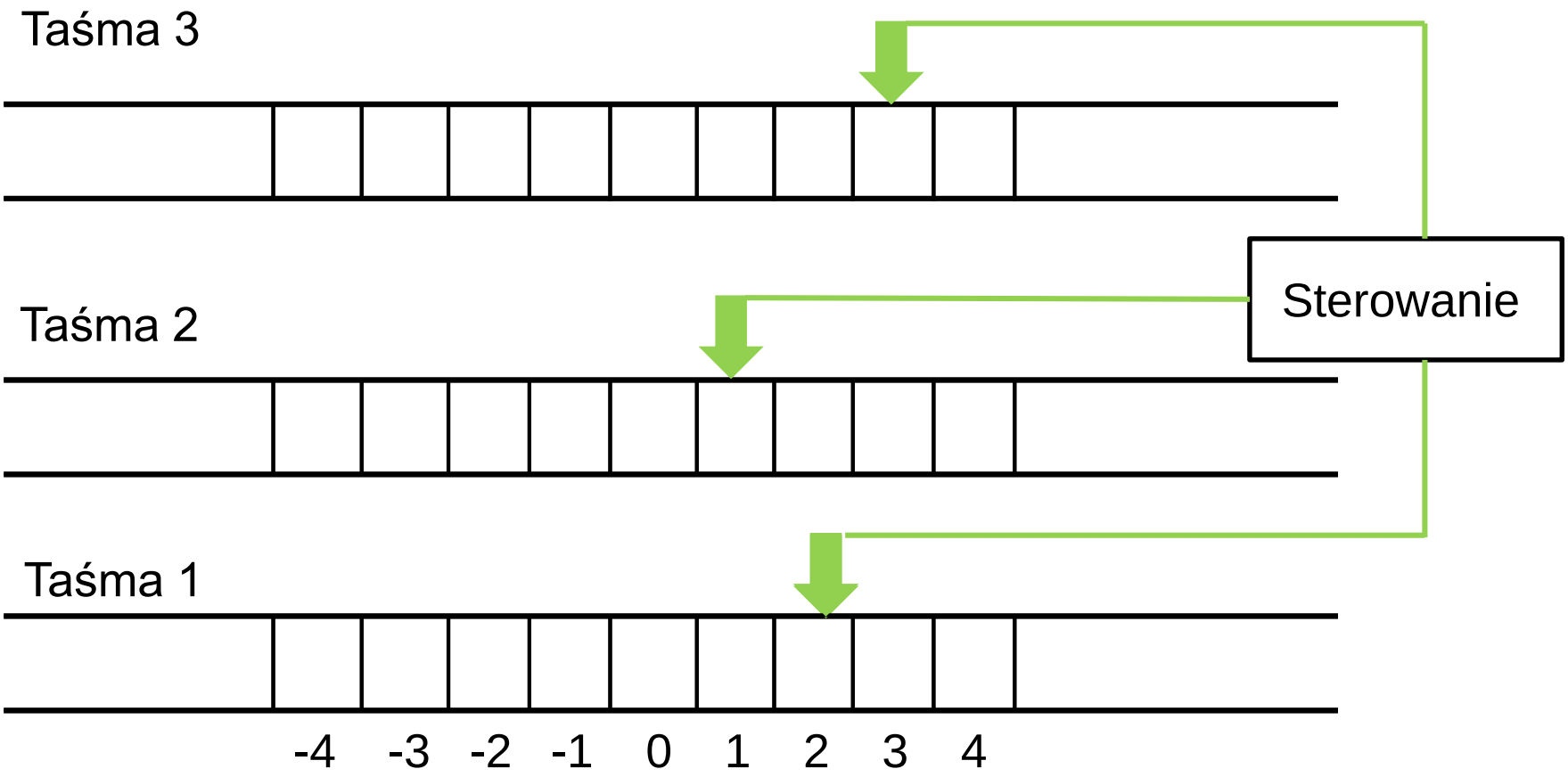
Program dla k- taśmowej DMT składa się ze:

Skończonego zbioru symboli $\sum$ zawierającego separator (symbol pusty) $\sqcup$ zapisywanych/odczytywanych na/z k taśm,

Skończonego zbioru stanów $Q = \{q_0, \ldots, q_m\}$ zawierającego stan początkowy q_0, dwa wyróżnione stany końcowe q_Y (odpowiedź „tak”) i q_N (odpowiedź „nie”),

Funkcji przejść $\delta: (Q \backslash \{q_Y, q_N\}) \times \Sigma^k \to Q \times \Sigma^k \times \{-1,0,1\}^k$.

Deterministyczna trójtaśmowa DMT



Cel:

Dodawanie dwu liczb dwójkowych tej samej długości, znajdujących się na Taśmach 1 i 2, których najstarsze pozycje znajdują się w komórkach o numerze 1.

Wynik tworzony jest na Taśmie 3. Cyfra najmłodszej pozycji powinna być umieszczona tak jak cyfry najmłodszych pozycji danych wejściowych. Głowica tej taśmy powinna wskazywać najstarszą cyfrę wyniku.

Plan:

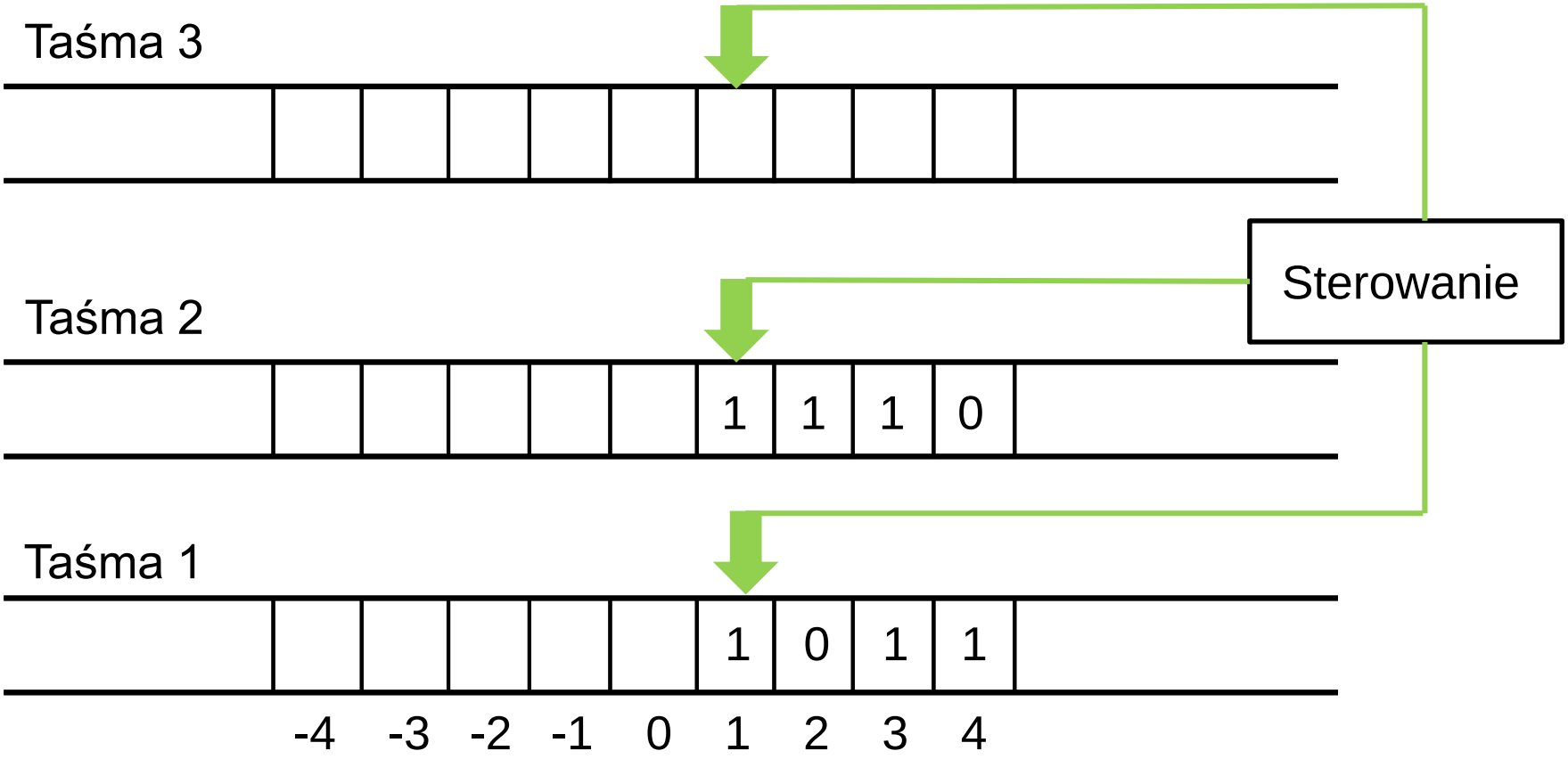
Wymagania:

Zewnętrzne objawy zachowania trójtaśmowej DMT.

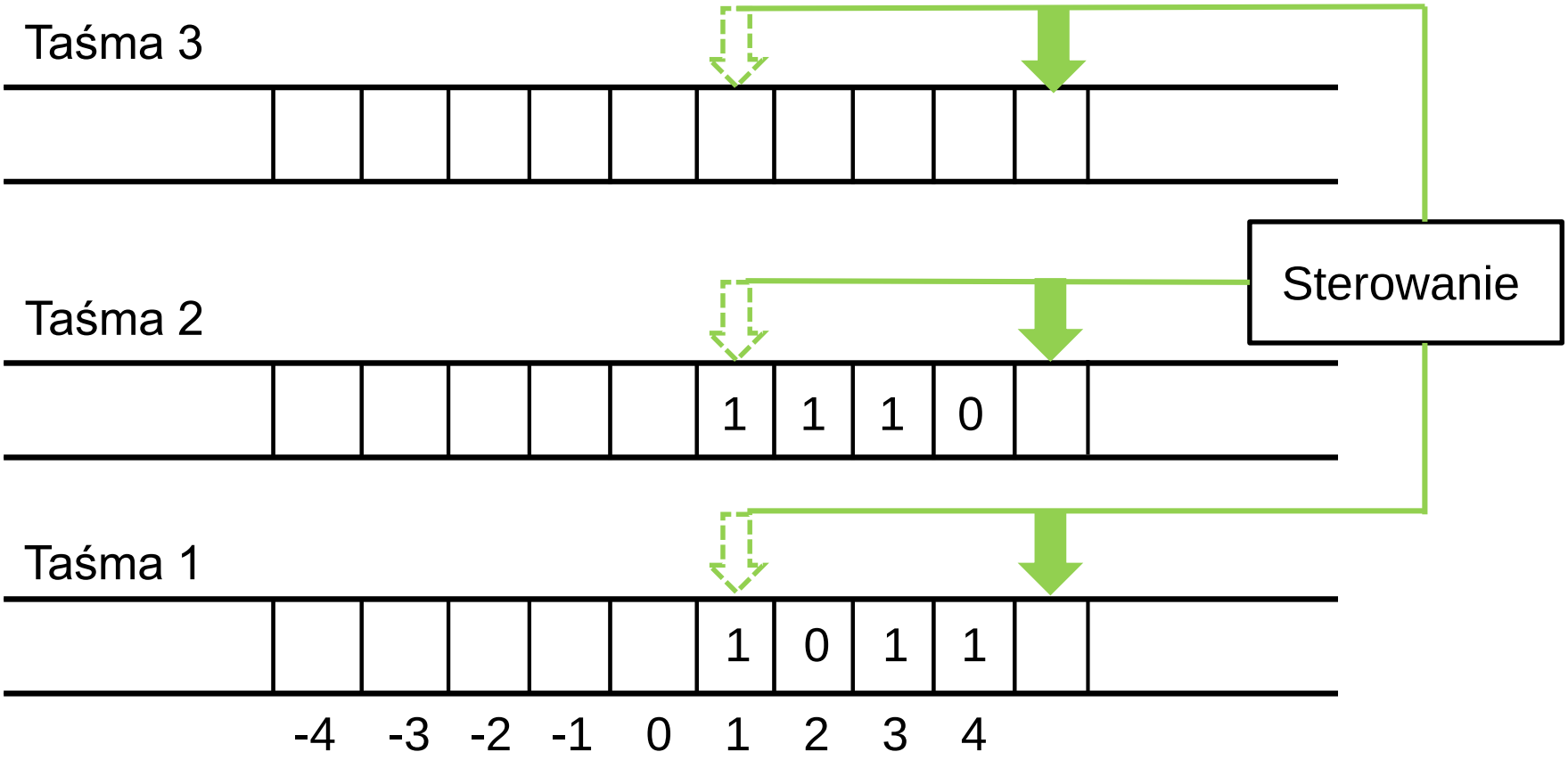
Specyfikacja:

Diagram przejść maszyny.

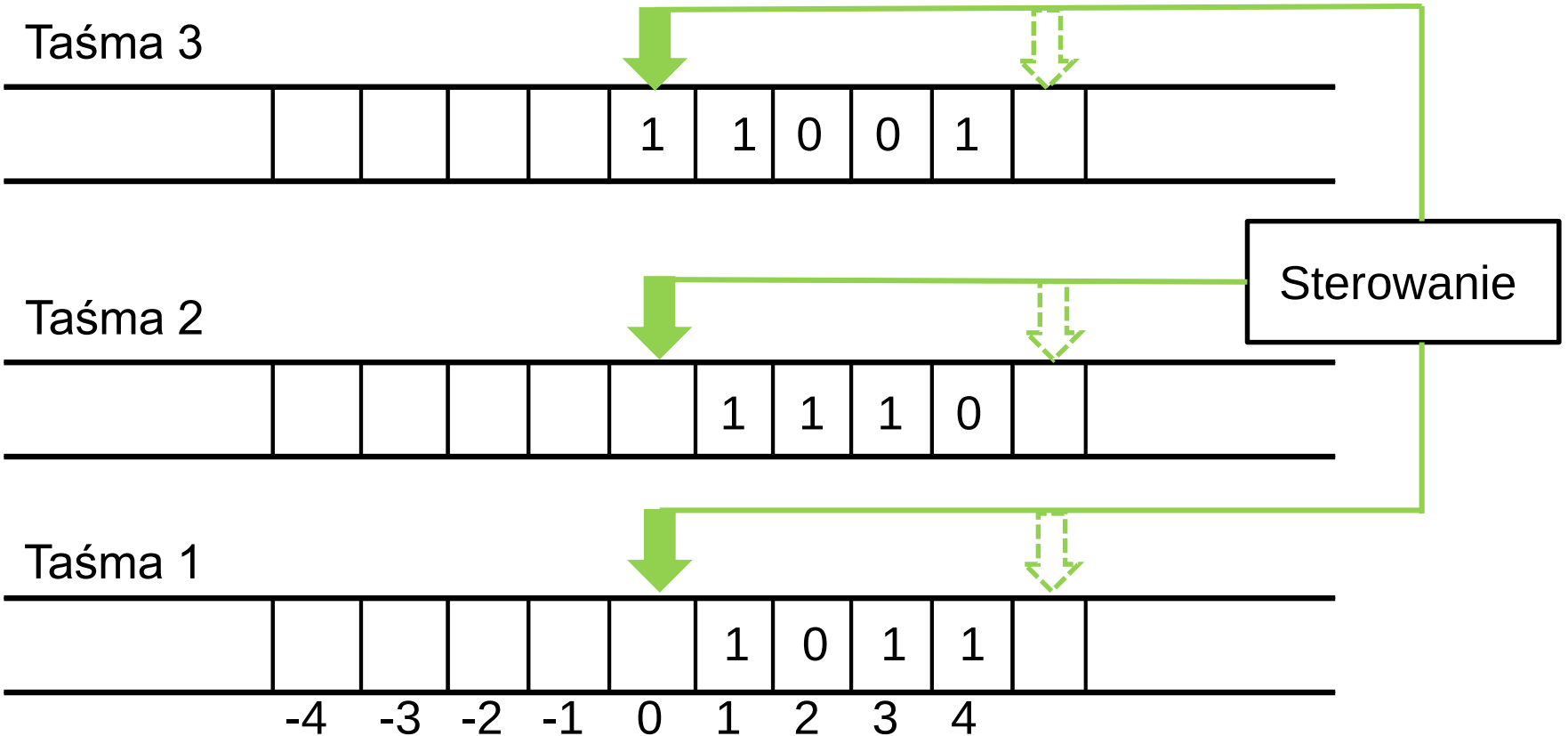
Start: Maszyna znajduje się w stanie początkowym q_0, natomiast głowice odczytują symbole z trzech taśm z komórek o numerze 1.



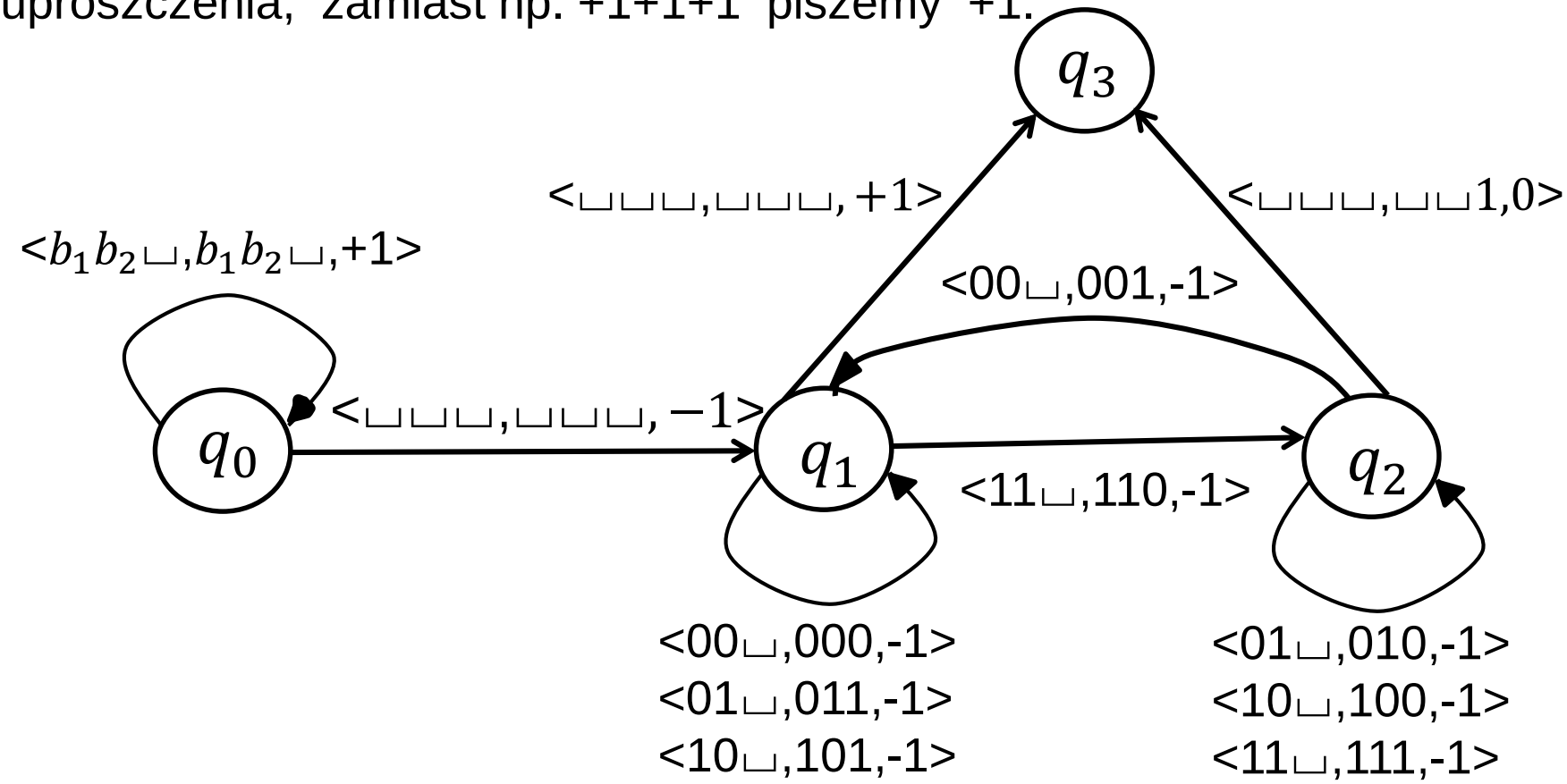
Głowice trzech taśm przesuwają się w prawo do osiągnięcia komórek ze znakami pustymi znajdujących się na prawo od liczb.



Głowice trzech taśm przesuwają się w lewo do osiągnięcia komórek ze znakami pustymi znajdujących się na lewo od liczb na Taśmach 1 i 2. Kolejne cyfry sumy zapisywane są na Taśmie 3.



b_i - cyfra dwójkowa i–tej taśmy.
W tym przykładzie, trzy głowice poruszają się synchronicznie. Stąd, dla uproszczenia, zamiast np. +1+1+1 piszemy +1.



Model *RAM* (ang. random access machine)

- Jeden procesor,
- Operacje elementarne (zapisania, dodawania, odejmowania, porównania dwu liczb, itp.) wymagają jednego kroku czasowego,
- Taśma wejściowa z głowicą odczytującą,
- Taśma wyjściowa z głowicą zapisującą,
- Pamięć danych jest zbiorem rejestrów z wyróżnionym rejestrem – akumulatorem, w którym wykonywane są obliczenia,
- Dostęp do pamięci wymaga jednego kroku,
- Możliwość adresowania pośredniego,
- Program, który nie jest przechowywany w pamięci jest sekwencją rozkazów (nie podlega automodyfikacji),
- Licznik rozkazów.

Model *RAM* c.d.

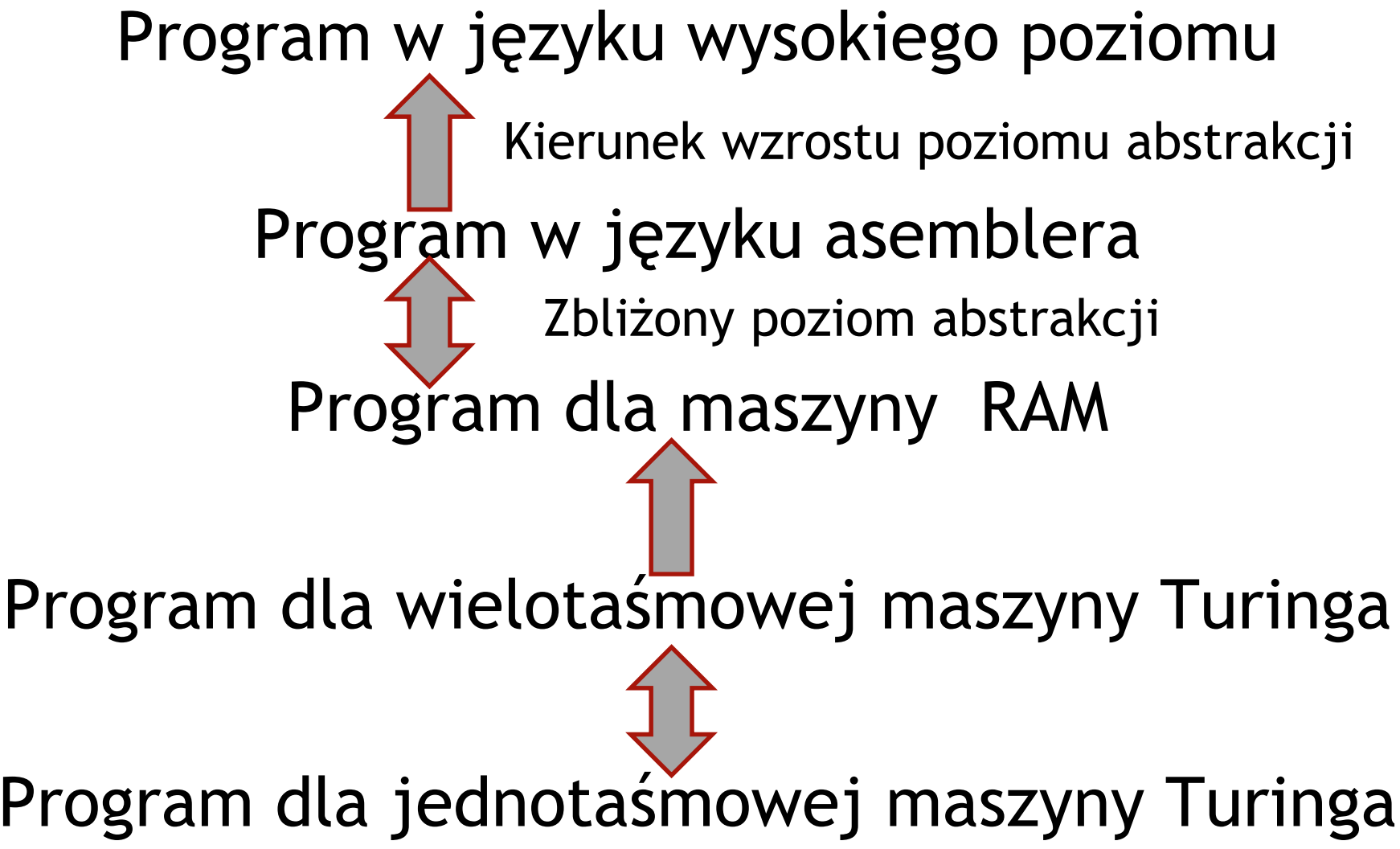
- Każda komórka taśm: wejściowej i wyjściowej oraz każdy rejestr może zawierać dowolną liczbę całkowitą.
- Po odczycie z komórki taśmy wejściowej (zapisie w komórce taśmy wyjściowej), głowica taśmy jest przesuwana o jedną pozycję w prawo. Treści wpisanej na taśmie wyjściowej nie można zmienić.
- Rozkazy (zbiór rozkazów nie jest precyzyjnie zdefiniowany, ale nie może zawierać instrukcji niespotykanych w rzeczywistych komputerach):
  - Arytmetyczne (+, -, x, /),
  - Wejścia-wyjścia,
  - Rozgałęzienia przepływu sterowania.

Racjonalny” model komputera

„Racjonalny” model komputera to taki, dla którego istnieje wielomian ograniczający od góry zakres pracy wykonywanej w jednostce czasu.

Model mogący wykonywać dowolnie dużą liczbę operacji równolegle nie jest „racjonalnym”.

Modele algorytmów



Kryteria kosztów operacji elementarnych

(zapisania, dodawania, odejmowania, porównania dwu liczb, itp.)

Logarytmiczne kryterium kosztów

Czas wykonania elementarnej operacji zależy liniowo od długości łańcucha danych kodujących liczby, a zatem od logarytmów liczb.

Analiza teoretyczna z użyciem DMT prowadzona jest przy tym kryterium.

Jednorodne kryterium kosztów

Czas wykonania elementarnej operacji jest jednostkowy.

Analiza praktyczna często oparta jest na tym kryterium.

Twierdzenie

Modele procesu obliczeń:

- ***Jednotaśmowa maszyna Turinga,***
- ***Wielotaśmowa maszyna Turinga,***
- ***Maszyna RAM***

są równoważne w tym sensie, że jeśli dany problem jest rozwiązywany przez jeden model w czasie ograniczonym od góry przez wielomian zależny od rozmiarów problemu, to przy założeniu logarytmicznego kryterium kosztów jest on również rozwiązywany przez każdy inny model w czasie ograniczonym od góry przez wielomian zależny od jego rozmiarów.

Jeśli czas działania programu (algorytmu) P na DMT rozwiązującego problem decyzyjny π jest ograniczony od góry wielomianem zależnym od długości języka L (danych wejściowych), tzn. czas działania $t \leqslant p(|L|)$ dla każdego L i pewnego wielomianu p, to algorytm P jest ***algorytmem wielomianowym***.

Jeżeli algorytm nie jest algorytmem wielomianowym, to jest nazywamy ***algorytmem ponadwielomianowym***.