Szeregowanie zadań

Wykład nr 3

dr Hanna Furmańczyk

04.03.2020

Algorytm McNaughtona

1. Wylicz optymalną długość
   $C^*_{max} = \max\{\sum_{j=1,\ldots,n} p_j/m, \max_{j=1,\ldots,n} p_j\}$,
2. Szereguj kolejno zadania na maszynie, po osiągnięciu C^*_{max} przerwij zadanie i (jeśli się nie zakończyło) kontynuuj je na następnym procesorze począwszy od chwili 0.

Algorytm McNaughtona

1. Wylicz optymalną długość $C^*_{max} = \max\{\sum_{j=1,\ldots,n} p_j/m, \max_{j=1,\ldots,n} p_j\}$,
2. Szereguj kolejno zadania na maszynie, po osiągnięciu C^*_{max} przerwij zadanie i (jeśli się nie zakończyło) kontynuuj je na następnym procesorze począwszy od chwili 0.

Złożoność obliczeniowa:

Algorytm McNaughtona

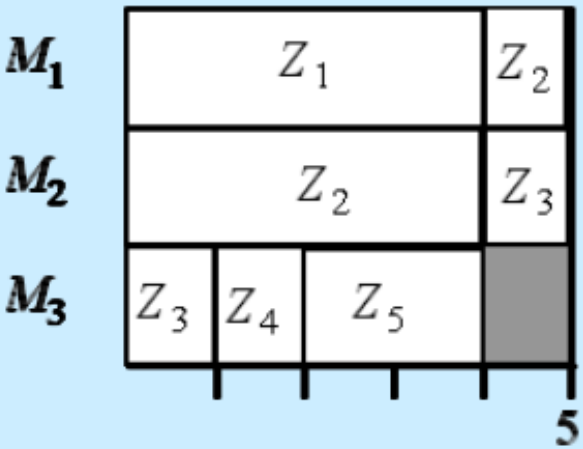
1. Wylicz optymalną długość $C^*_{max} = \max\{\sum_{j=1,\ldots,n} p_j/m, \max_{j=1,\ldots,n} p_j\}$,
2. Szereguj kolejno zadania na maszynie, po osiągnięciu C^*_{max} przerwij zadanie i (jeśli się nie zakończyło) kontynuuj je na następnym procesorze począwszy od chwili 0.

Złożoność obliczeniowa: $O(n)$.

Przykład

Przykład. m=3, n=5, $p_1,\ldots,p_5$=4,5,2,1,2.

$\Sigma_{i=1,\ldots,5}\, p_i$=14, max p_i=5,
$C_{\max}$*=max{14/3,5}=5.



Twierdzenie

Problem $P2||C_{max}$ jest NP-trudny.

Twierdzenie

Problem $P2||C_{max}$ jest NP-trudny.

Dowód

Redukcja problemu *podziału* do $P2||C_{max}$.

Twierdzenie

Problem $P2||C_{max}$ jest NP-trudny.

Dowód

Redukcja problemu *podziału* do $P2||C_{max}$.
Problem podziału:
Dany jest ciąg liczb naturalnych $a_1, a_2, \ldots, a_n$ tż. $S = \sum_{i=1}^{n} a_i$ jest l. parzystą.
Pytanie: Czy istnieje jego podciąg o sumie $S/2$?

Twierdzenie

Problem $P2||C_{max}$ jest NP-trudny.

Dowód

Redukcja problemu *podziału* do $P2||C_{max}$.
Problem podziału:
Dany jest ciąg liczb naturalnych $a_1, a_2, \ldots, a_n$ tż. $S = \sum_{i=1}^{n} a_i$ jest l. parzystą.
Pytanie: Czy istnieje jego podciąg o sumie $S/2$?
Redukcja: bierzemy n zadań o $p_j = a_j$, dwie maszyny. Pytamy o istnienie uszeregowania z $C_{max} \leq S/2$.

Twierdzenie

Problem $P2||C_{max}$ jest NP-trudny.

Dowód

Redukcja problemu *podziału* do $P2||C_{max}$.
Problem podziału:
Dany jest ciąg liczb naturalnych $a_1, a_2, \ldots, a_n$ tż. $S = \sum_{i=1}^{n} a_i$ jest l. parzystą.
Pytanie: Czy istnieje jego podciąg o sumie $S/2$?
Redukcja: bierzemy n zadań o $p_j = a_j$, dwie maszyny. Pytamy o istnienie uszeregowania z $C_{max} \leq S/2$.

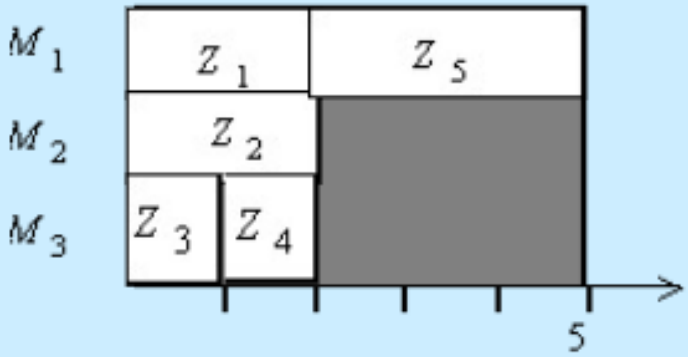
Dokładne algorytmy - programowanie dynamiczne, złożoność wykładnicza;
algorytmy przybliżone

Szeregowanie listowe (ang. *List Scheduling* LS); ogólnie

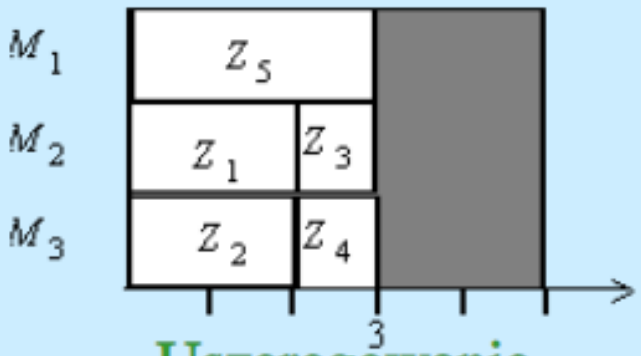
- Ustal kolejność zadań na liście.
- Za każdym razem, gdy zwalnia się jakaś maszyna/maszyny, wybieraj pierwsze (według „listy") wolne (w tym momencie)[zadania zależne] zadania i przypisuj je do zwalniających się procesorów.

Przykład

Przykład. $m=3$, $n=5$, $p_1,\ldots,p_5=2,2,1,1,3$.



Uszeregowanie listowe



Uszeregowanie optymalne

Szeregowanie listowe dla $P||C_{max}$

Z ustalonego ciągu zadań wybieraj pierwsze wolne (według „listy"), przypisując je zawsze do zwalniającego się procesora.

Szeregowanie listowe dla $P||C_{max}$

Z ustalonego ciągu zadań wybieraj pierwsze wolne (według „listy"), przypisując je zawsze do zwalniającego się procesora.

Algorytm LS jest 2-przybliżony $C_{max}(LS) \leq (2 - m^{-1})C^*_{max}$.

Szeregowanie LPT (ang. *Longest Processing Time*)

Szereguj listowo, przy czym zadania na liście są wstępnie posortowane według nierosnących czasów wykonania p_j.

Szeregowanie LPT (ang. *Longest Processing Time*)

Szereguj listowo, przy czym zadania na liście są wstępnie posortowane według nierosnących czasów wykonania p_j.

LPT jest 4/3-przybliżony: $C_{max}(LPT) \leq (4/3 - (3m)^{-1})C^*_{max}$.

Zadania zależne, podzielne

- w ogólności problem jest NP-trudny
- istnieje algorytm ($O(n^2)$) dla $P2|pmtn, prec|C_{max}$ i $P|pmtn, forest|C_{max}$

- problem NP-trudny
- znane przypadki wielomianowe:
  - $P|p_i = 1, in - forest|C_{max}$, $P|p_i = 1, out - forest|C_{max}$ (alg. Hu, $O(n)$)
  - $P2|p_i = 1, prec|C_{max}$ (alg. Coffmana-Grahama, $O(n^2)$)

- problem NP-trudny
- znane przypadki wielomianowe:
  - $P|p_i = 1, in - forest|C_{max}$, $P|p_i = 1, out - forest|C_{max}$ (alg. Hu, $O(n)$)
  - $P2|p_i = 1, prec|C_{max}$ (alg. Coffmana-Grahama, $O(n^2)$)

ALgorytm Hu - wstęp

- redukcja *out-forest* do *in-forest*: odwrócenie relacji *prec*, a po uzyskaniu harmonogramu – odwrócenie go,

- problem NP-trudny
- znane przypadki wielomianowe:
  - $P|p_i = 1, in - forest|C_{max}$, $P|p_i = 1, out - forest|C_{max}$ (alg. Hu, $O(n)$)
  - $P2|p_i = 1, prec|C_{max}$ (alg. Coffmana-Grahama, $O(n^2)$)

ALgorytm Hu - wstęp

- redukcja *out-forest* do *in-forest*: odwrócenie relacji *prec*, a po uzyskaniu harmonogramu – odwrócenie go,
- redukcja $in\text{-}forest \rightarrow in - tree$: dodanie „dodatkowego korzenia" dla wszystkich drzew, a po uzyskaniu harmonogramu usunięcie go

- problem NP-trudny
- znane przypadki wielomianowe:
  - $P|p_i = 1, in - forest|C_{max}$, $P|p_i = 1, out - forest|C_{max}$ (alg. Hu, $O(n)$)
  - $P2|p_i = 1, prec|C_{max}$ (alg. Coffmana-Grahama, $O(n^2)$)

ALgorytm Hu - wstęp

- redukcja *out-forest* do *in-forest*: odwrócenie relacji *prec*, a po uzyskaniu harmonogramu – odwrócenie go,
- redukcja $in\text{-}forest \rightarrow in - tree$: dodanie „dodatkowego korzenia" dla wszystkich drzew, a po uzyskaniu harmonogramu usunięcie go
- algorytm w skrócie: LS z *prec* + lista utworzona wg nierosnącej odległości od korzenia drzewa

Algorytm Hu $P|p_i = 1, in - tree|C_{max}$

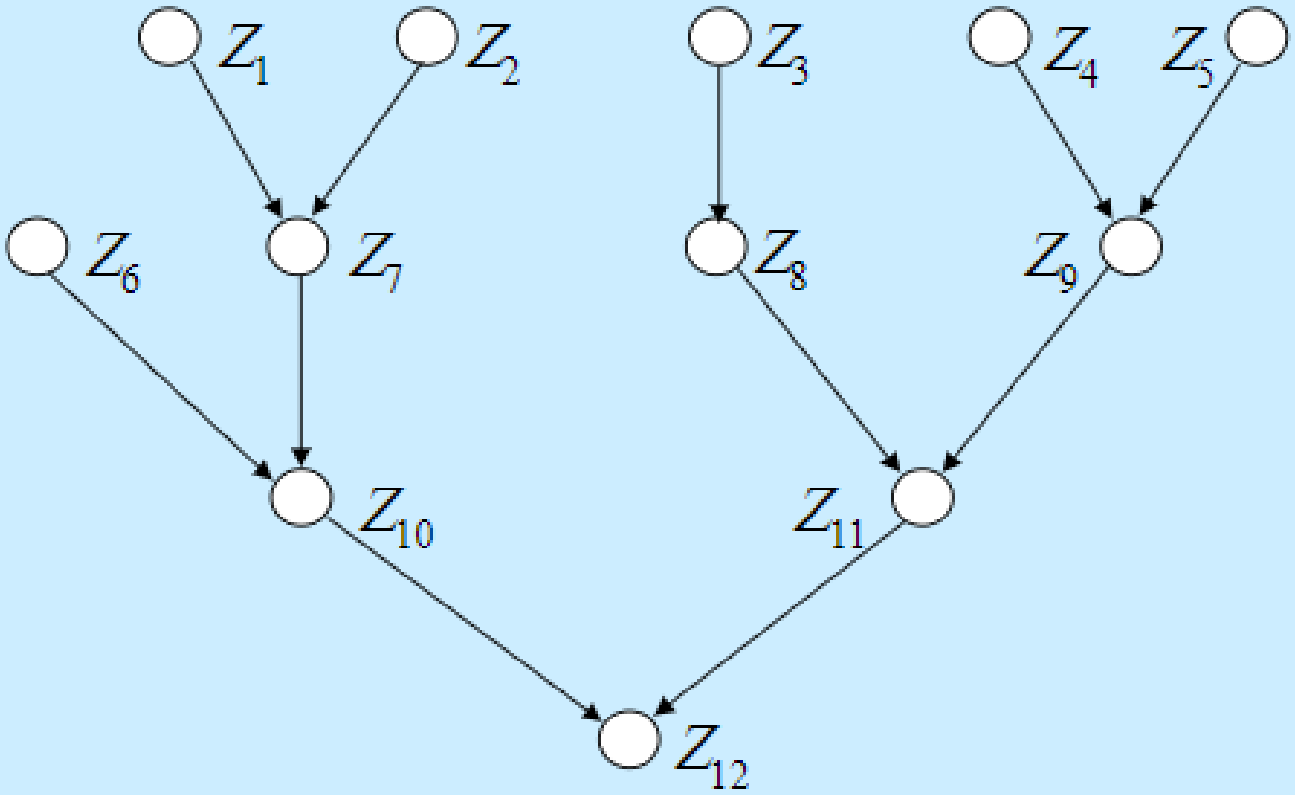
1. Ustal dla każdego zadania jego *poziom* – liczba węzłów na drodze do korzenia.
2. $t := 1$;
3. **repeat**
   - Wyznacz listę L_t zadań wolnych w chwili t;
   - Uporządkuj L_t według nierosnącego poziomu;
   - Przypisz m (lub mniej) zadań z początku L_t do maszyn;
   - Usuń przypisane zadania z grafu;
   - $t := t + 1$;

   until uszeregowano wszystkie zadania;

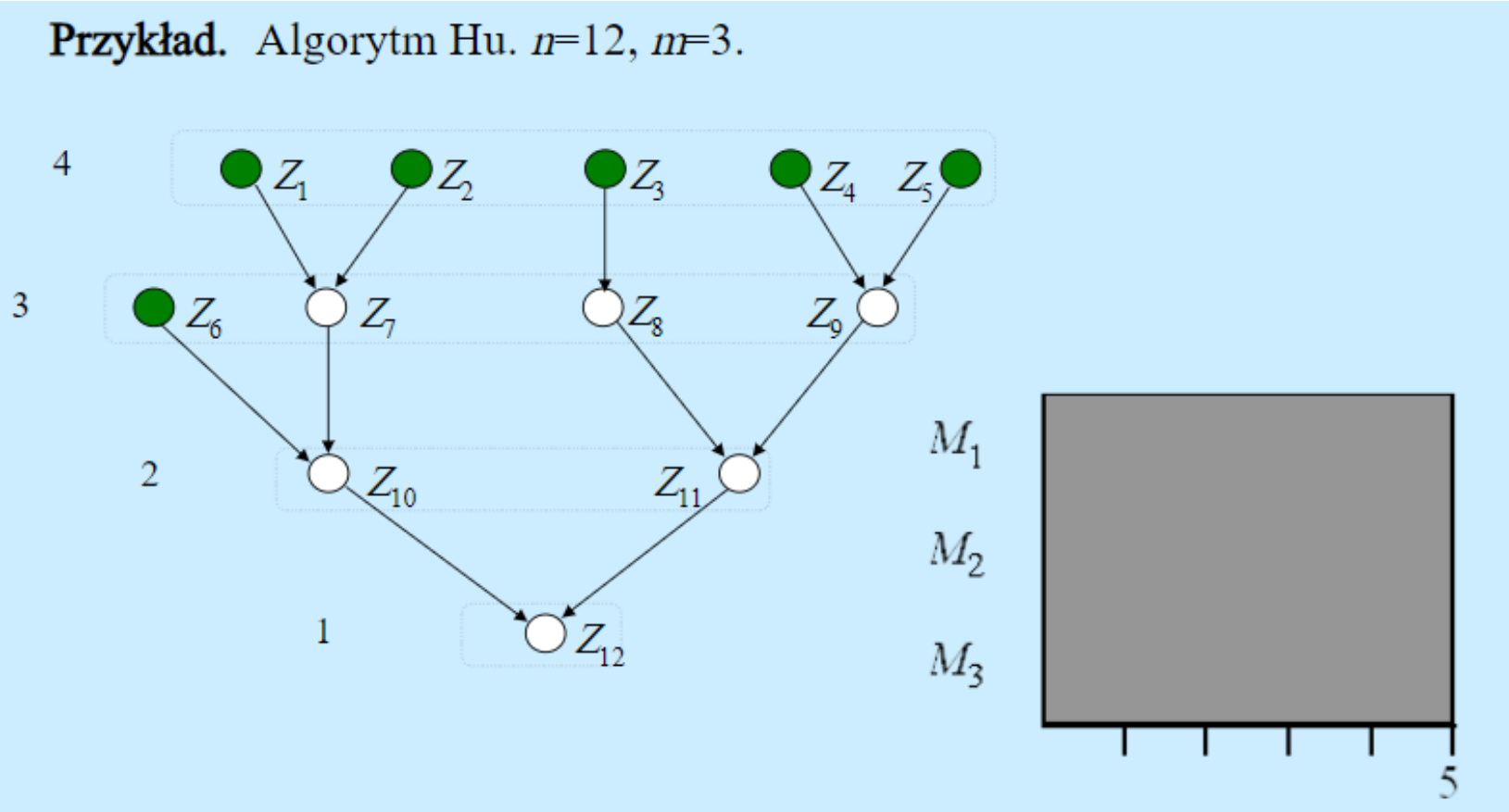
Algorytm Hu - przykład

Zadania niepodzielne

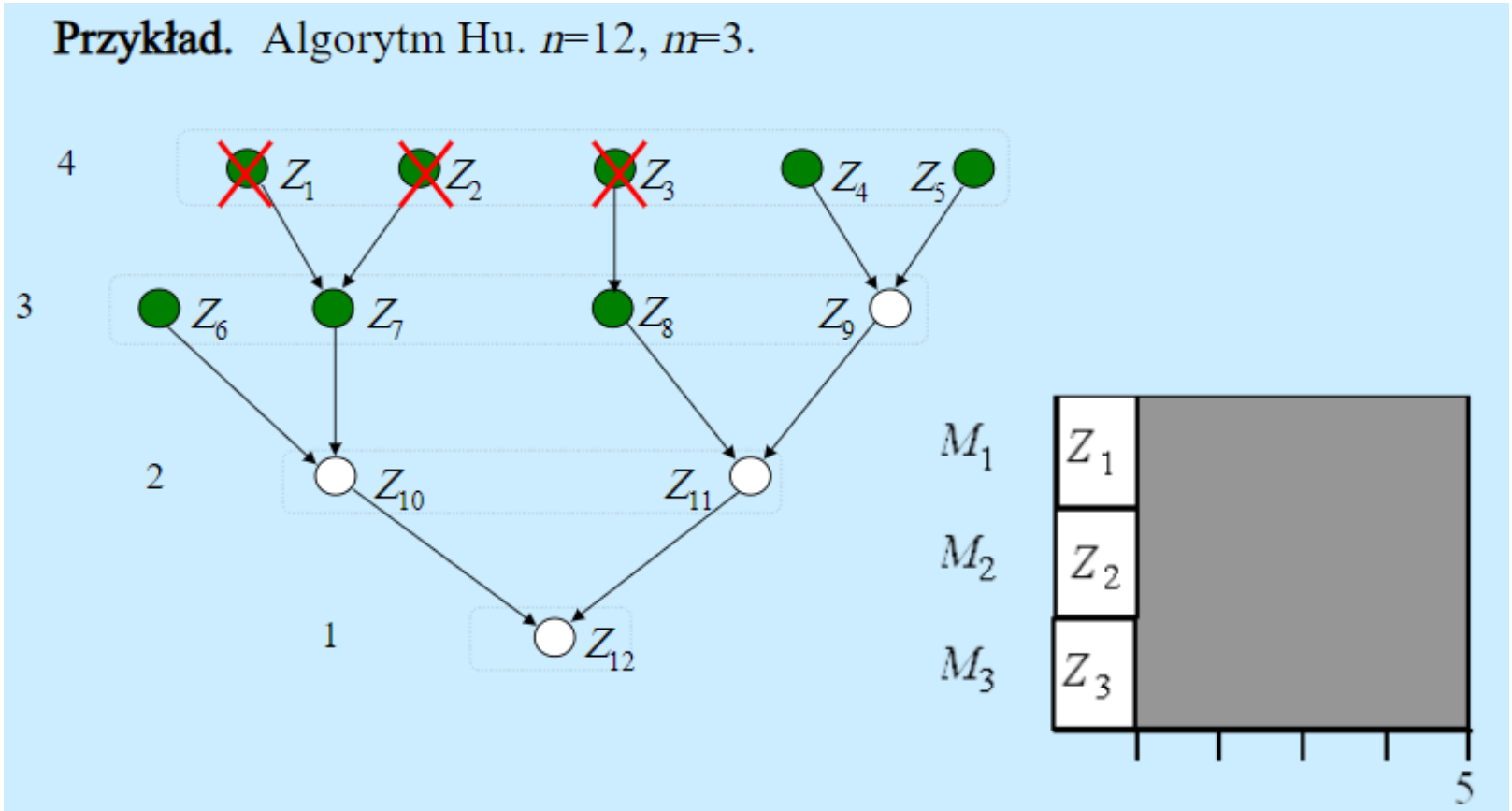
Przykład. Algorytm Hu. n=12, m=3.



Algorytm Hu - przykład

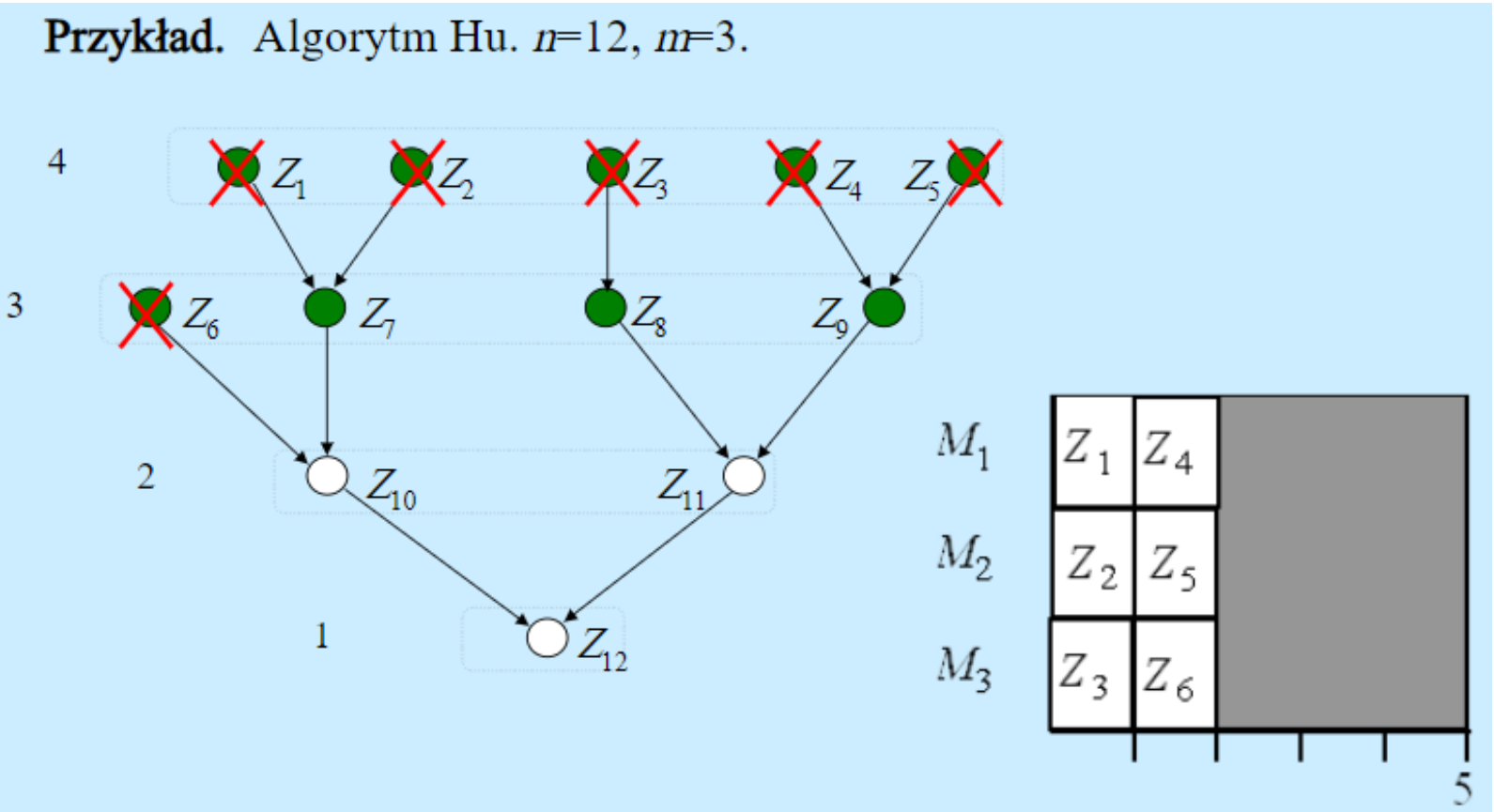


Algorytm Hu - przykład



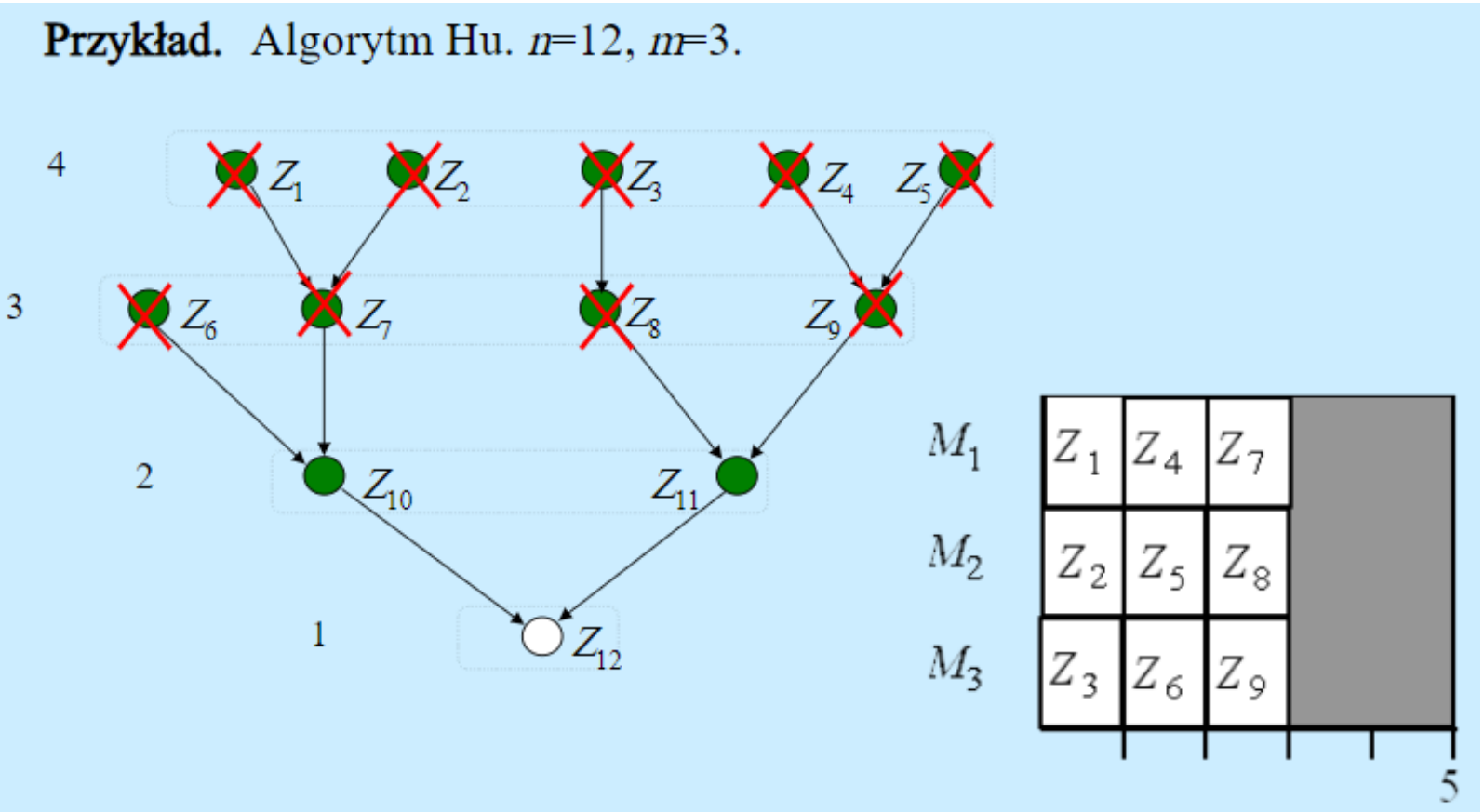
Algorytm Hu - przykład

Przykład. Algorytm Hu. n=12, m=3.



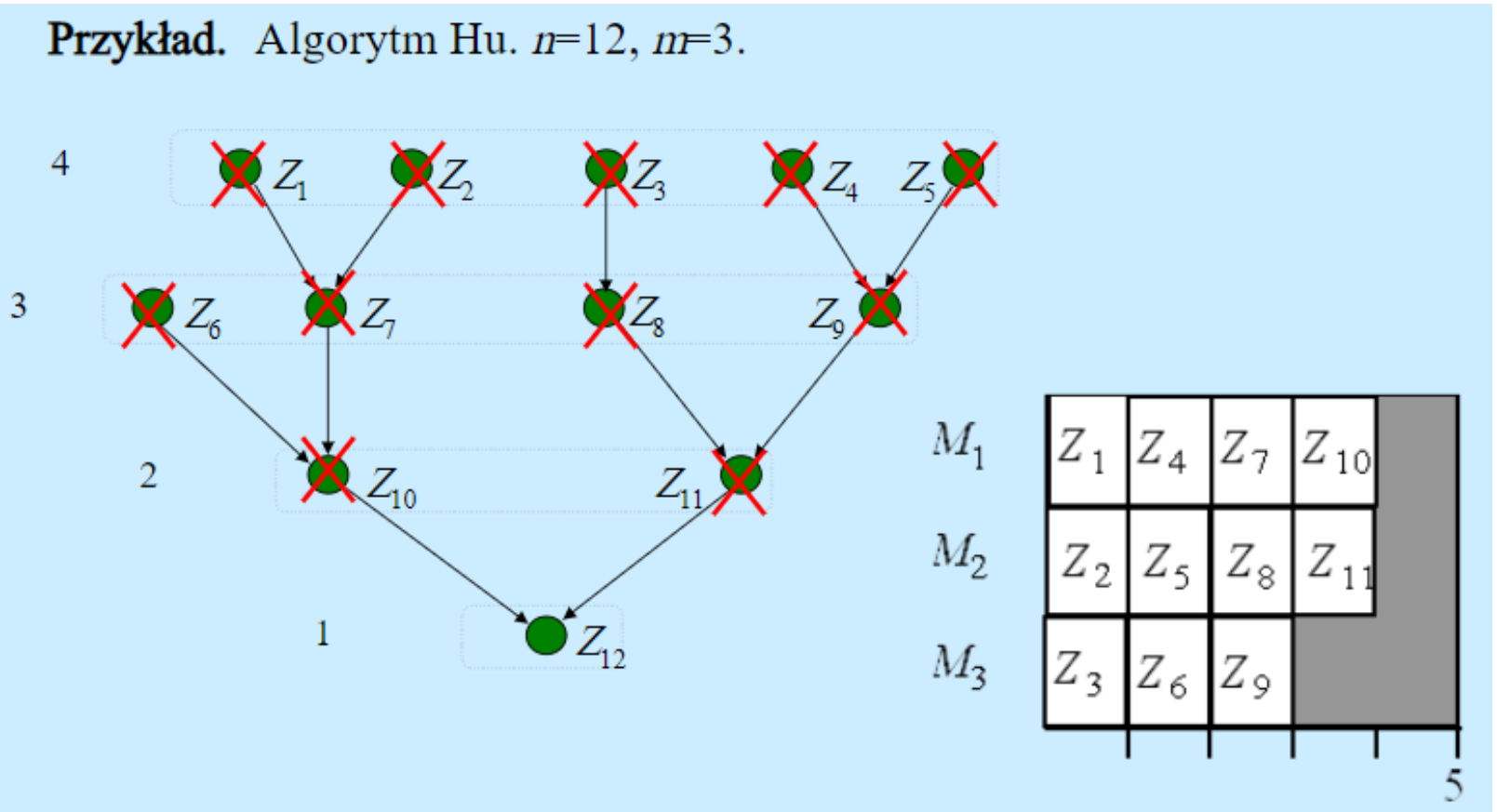
Algorytm Hu - przykład

Przykład. Algorytm Hu. n=12, m=3.



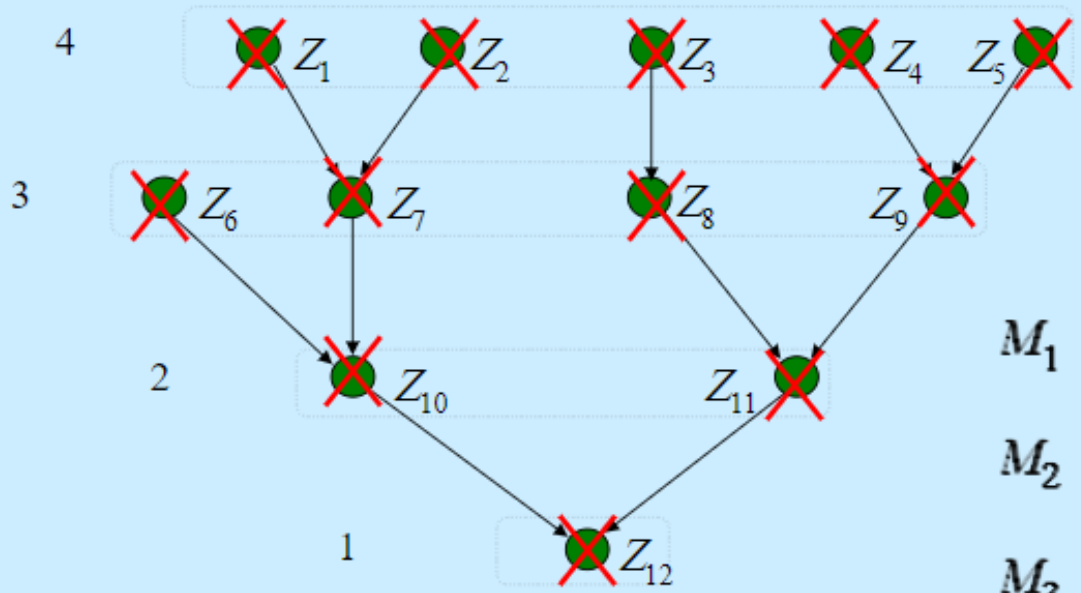
Algorytm Hu - przykład

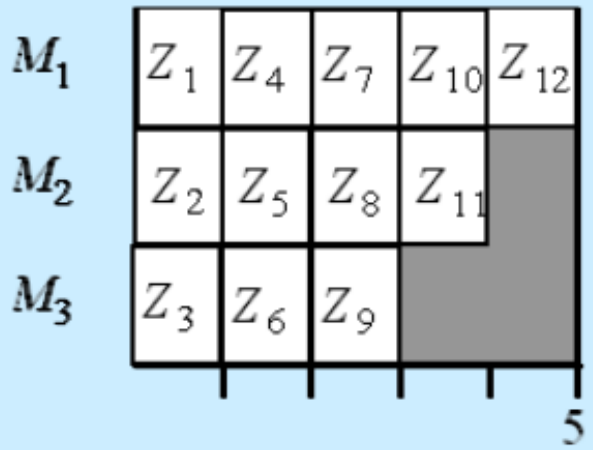
Przykład. Algorytm Hu. n=12, m=3.



Algorytm Hu - przykład

Przykład. Algorytm Hu. n=12, m=3.





Slajdy autorstwa P. Semprucha