Plan wykładu
Procesy
Wątki
Planowanie przydziału procesora
Algorytmy szeregowania
Planowanie w systemach wieloprocesorowych
Ocena algorytmów

Systemy operacyjne

Zarządzanie procesami

Arkadiusz Chrobot

Katedra Systemów Informatycznych, Politechnika Świętokrzyska w Kielcach

Kielce, 5 listopada 2024

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Plan wykładu

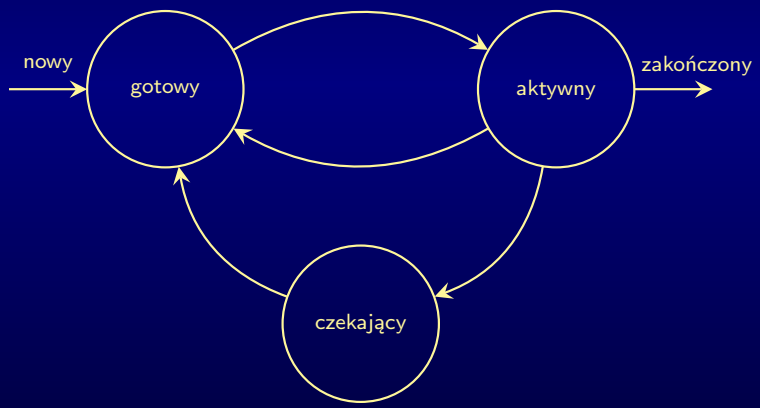
1. Procesy
   1. Proces sekwencyjny
   2. Cykl życia procesu
   3. Deskryptor procesu
   4. Procesy współbieżne
2. Wątki
3. Planowanie przydziału procesora
   1. Motywacja
   2. Kolejki
   3. Planiści
   4. Przełączanie kontekstu
4. Algorytmy szeregowania procesów
   1. Algorytm FCFS
   2. Algorytm SJF
   3. Algorytm SRT
   4. Algorytm priorytetowy
   5. Algorytm rotacyjny
   6. Wielopoziomowe kolejki
   7. Wielopoziomowe kolejki ze sprzężeniem zwrotnym
   8. Przykłady
5. Szeregowanie w systemach wieloprocesorowych
6. Ocena algorytmów

Proces sekwencyjny

Definicja

Program komputerowy, który został załadowany do pamięci komputera i który jest wykonywany nazywa się *procesem sekwencyjnym*. Wyraz *sekwencyjny* oznacza, że instrukcje programu są wykonywane przez proces w takiej kolejności, w jakiej zostały umieszczone w pliku z kodem wynikowym. Podczas wykonania proces może tworzyć inne procesy sekwencyjne, które mogą być wykonywane *współbieżnie*. Każdy proces posiada swój odrębny fragment pamięci operacyjnej i zestaw adresów przez który może się do niej odwoływać, czyli własną *przestrzeń adresową*. Wewnętrznie pamięć procesu jest podzielona na obszary kodu, danych, stosu, sterty i inne. W rzeczywistości procesy mogą współdzielić fragmenty swej pamięci ze względów oszczędności (obszar kodu, biblioteki współdzielone) lub z konieczności komunikacji (pamięć dzielona). Wymiennie proces nazywany jest *zadaniem*.

Cykl życia procesu



Stan procesu określa etap jego realizacji. Podczas przetwarzania procesu jego stan ulega zmianom. Graf obok ilustruje możliwości zmiany stanu procesu. Jest to diagram uogólniony dla większości systemów operacyjnych. Stan *gotowy* oznacza, że proces *może być* wykonany przez procesor. Stan *aktywny* oznacza, że proces jest w danej chwili wykonywany przez procesor. W tym stanie może znajdować się tylko jeden proces w komputerze jednoprocesorowym. Proces w stanie *czekający* oczekuje na zdarzenie (np.: zakończenie operacji wejścia-wyjścia) i nie może być w chwili bieżącej wykonywany przez procesor.

Deskryptor procesu

Deskryptor procesu zwany również *blokiem kontrolnym procesu* jest to zmienna (w języku Pascal byłby to rekord, w języku C struktura) przechowująca informacje na temat procesu. Jest niezbędna do zarządzania procesami i jest tworzona dla każdego z nich przez system operacyjny. Każdy deskryptor procesu jest więc strukturą należącą do systemu operacyjnego i jest umieszczany w jego przestrzeni adresowej. Blok kontrolny procesu może zawierać takie informacje, jak: stan procesu, stan licznika rozkazów i innych rejestrów procesora dla tego procesu, priorytet procesu, informacje o pamięci procesu, itd. Posiada on też pola wskaźnikowe, które pozwalają go łączyć, wraz z innymi blokami w większe struktury (listy, drzewa, itd.).

Współbieżność

Jeśli w pamięci systemu komputerowego rezyduje równocześnie kilka procesów, to mogą one być wykonywane *współbieżnie*. W przypadku komputerów jednoprocesorowych oznacza to, że procesor co pewien czas jest przydzielany na zmianę różnym procesom. Taką współbieżność nazywamy *pseudorównoległością*. W systemach wieloprocesorowych każdy z procesorów może w danej chwili wykonywać odrębny proces. Takie przetwarzanie nazywamy *przetwarzaniem równoległym*. Za stosowaniem współbieżności przemawia możliwość podziału zasobów fizycznych i logicznych między wielu użytkowników, przyspieszenie przetwarzania (w architekturach wieloprocesorowych), podział programu na niezależnie wykonywane jednostki i zwiększenie stopnia wykorzystania systemu komputerowego.

Tworzenie procesów

System operacyjny może utworzyć nowy proces na żądanie innego, istniejącego procesu. Proces tworzący nazywany jest *procesem macierzystym*. Tworzenie nowego procesu realizowane jest przez odpowiednie wywołanie systemowe. System operacyjny może przydzielić nowemu procesowi zasoby z puli zasobów wolnych lub część zasobów należących do procesu macierzystego. Ten drugi sposób zapobiega nadmiernemu rozmnażaniu procesów. Proces macierzysty może również przekazać procesowi potomnemu niezbędne do jego wykonania informacje. Po utworzeniu procesu potomnego proces rodzicielski może wykonywać się z nim współbieżnie, lub czekać na jego zakończenie. Powiązania „rodzinne" między procesami są odnotowywane w ich deskryptorach.

Przykład

W systemie Unix do tworzenia nowego procesu wykorzystywane jest wywołanie fork(), tworzy ono nowy proces, który współdzieli z procesem macierzystym obszar kodu, ale ma odrębny obszar danych. Wywołanie fork() zwraca wartość 0 dla procesu potomnego, a procesowi macierzystemu zwraca identyfikator potomka (PID). Jeśli programista chce aby nowy proces wykonywał inny kod niż jego proces macierzysty, to może zastąpić go poprzez użycie wywołania systemowego execve().

Kończenie procesu

Proces potomny może zakończyć się i przekazać swój kod zakończenia za pomocą odpowiedniego wywołania systemowego (w Uniksie exit()) procesowi macierzystemu. Proces macierzysty może również zakończyć działanie procesu potomnego posługując się jego identyfikatorem i odpowiednim wywołaniem systemowym (w Uniksie kill()). Za sytuację anormalną uznaje się zwykle zakończenie procesu macierzystego przed procesami potomnymi. Oznacza to, że żaden proces nie czeka na wyniki ich pracy (w Uniksie to oczekiwanie jest realizowane za pomocą wywołania systemowego wait()). Systemy operacyjne mogą w różny sposób obsługiwać tę sytuację. Część z nich *kończy kaskadowo* działanie osieroconych procesów potomnych, inne stosują swoisty mechanizm adopcji takich procesów. Do tej ostatniej grupy należy Unix. Procesy osierocone w tym systemie są przekazywane procesowi, który się nigdy nie kończy (init) i który co pewien czas wywołuje funkcję systemową wait().

Współpraca między procesami współbieżnymi

W zależności od intencji programisty i usług dostarczanych przez system operacyjny procesy współbieżne mogą ze sobą współpracować bądź też nie. Oto porównanie takich procesów:

Proces niezależny

- na jego stan nie wpływa żaden inny proces,
- jego działanie jest deterministyczne,
- jego działanie daje się powielać,
- jego działanie może być wstrzymywane i wznawiane bez żadnych skutków ubocznych.

Proces współpracujący

- jego stan jest dzielony z innymi procesami,
- jego wykonanie jest niedeterministyczne,
- wynik jego działania może zależeć od wykonania innych procesów.

Współpraca między procesami współbieżnymi

W zależności od intencji programisty i usług dostarczanych przez system operacyjny procesy współbieżne mogą ze sobą współpracować bądź też nie. Oto porównanie takich procesów:

Proces niezależny

- na jego stan nie wpływa żaden inny proces,
- jego działanie jest deterministyczne,
- jego działanie daje się powielać,
- jego działanie może być wstrzymywane i wznawiane bez żadnych skutków ubocznych.

Proces współpracujący

- jego stan jest dzielony z innymi procesami,
- jego wykonanie jest niedeterministyczne,
- wynik jego działania może zależeć od wykonania innych procesów.

Współpraca między procesami współbieżnymi

W zależności od intencji programisty i usług dostarczanych przez system operacyjny procesy współbieżne mogą ze sobą współpracować bądź też nie. Oto porównanie takich procesów:

Proces niezależny

- na jego stan nie wpływa żaden inny proces,
- jego działanie jest deterministyczne,
- jego działanie daje się powielać,
- jego działanie może być wstrzymywane i wznawiane bez żadnych skutków ubocznych.

Proces współpracujący

- jego stan jest dzielony z innymi procesami,
- jego wykonanie jest niedeterministyczne,
- wynik jego działania może zależeć od wykonania innych procesów.

Współpraca między procesami współbieżnymi

W zależności od intencji programisty i usług dostarczanych przez system operacyjny procesy współbieżne mogą ze sobą współpracować bądź też nie. Oto porównanie takich procesów:

Proces niezależny

- na jego stan nie wpływa żaden inny proces,
- jego działanie jest deterministyczne,
- jego działanie daje się powielać,
- jego działanie może być wstrzymywane i wznawiane bez żadnych skutków ubocznych.

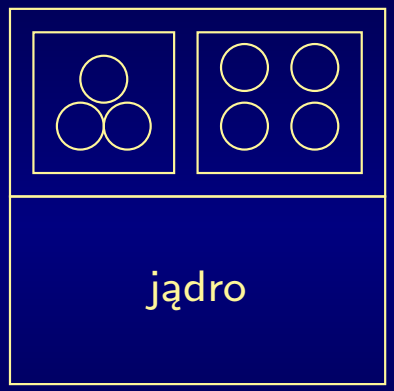
Proces współpracujący

- jego stan jest dzielony z innymi procesami,
- jego wykonanie jest niedeterministyczne,
- wynik jego działania może zależeć od wykonania innych procesów.

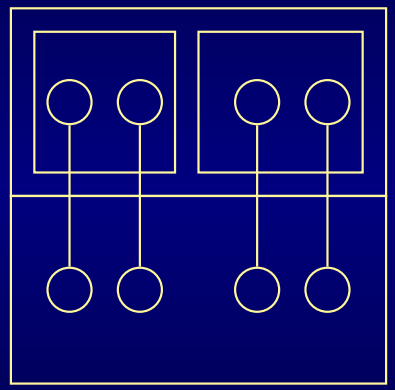
Wątki

Wątek (ang. thread) jest pojęciem pokrewnym pojęciu procesu. Podstawowa różnica polega na tym, że wątki nie mają osobnej przestrzeni adresowej, a współdzielą ten sam obszar pamięci. Pojedynczy proces może być wykonywany jako jeden wątek lub jako grupa wątków. Za stosowaniem wątków przemawia fakt, że przełączanie procesora między nimi jest na ogół mniej kosztowne niż przełączanie między procesami (trzeba zapamiętać mniej informacji). W niektórych systemach operacyjnych różnica w tych kosztach jest duża (Windows), w innych mała lub wręcz niewielka (Linux). Wątki mogą poprawiać interaktywność aplikacji (GUI) lub efektywność ich działania (demony). Mogą być one implementowane całkowicie w trybie użytkownika (biblioteka pth), mogą one być obsługiwane przez system operacyjny (Solaris, Windows, wywołanie clone() w Linuksie) lub można jednocześnie stosować oba podejścia (Java w różnych wersjach i dla różnych platform systemowych). Szeregowanie wątków może przebiegać na podstawie zawartości macierzy szeregowania lub podlegać takim samym mechanizmom jak szeregowanie zwykłych procesów. Wątki nazywane są często procesami lekkimi (ang. lightweight process), a tradycyjne procesy procesami ciężkimi (ang. heavy process). Ze względu na współdzielenie przestrzeni adresowej oprogramowanie wątków może być trudne.

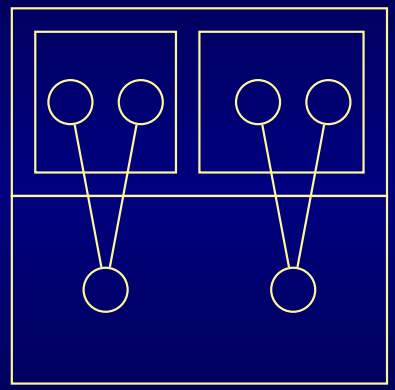
Wątki



(a) Implementacja w trybie użytkownika



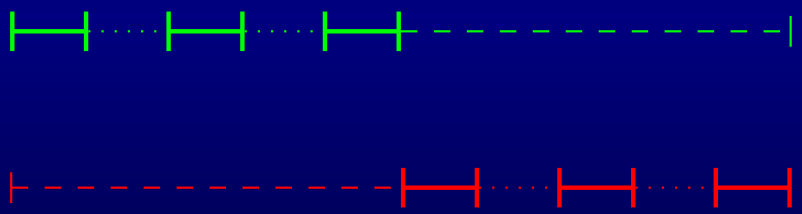
(b) Implementacja w trybie jądra



(c) Implementacja hybrydowa

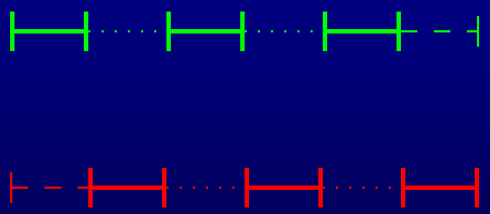
Rysunek: Implementacje wątków

Motywacja



Wykonanie każdego procesu jest podzielone na fazy. Rozróżniamy dwa rodzaje faz: fazę procesora, w której procesowi jest przydzielony procesor i fazę wejścia-wyjścia, w której na rzecz procesu wykonywana jest operacja wejścia-wyjścia lub inna czynność nieangażująca procesora. Jeśli dwa lub większa liczba procesów byłaby wykonywana szeregowo, tak jak obrazuje to ilustracja obok (linia pogrubiona - faza procesora, linia kropkowana - faza wejścia-wyjścia, linia kreskowana - czas przed lub po wykonaniu procesu), to procesy musiałby czekać długo na swoje wykonanie, a urządzenia wejścia-wyjścia i procesor byłyby naprzemiennie bezczynne. Takie procesy można jednak wykonywać współbieżnie, tzn. jeśli proces, który do tej pory korzystał z procesora wszedł w fazę korzystania z urządzeń peryferyjnych, to procesor może zostać przydzielony innemu procesowi, który jest gotów do wykonania. Takie postępowanie zmniejsza *czas przetwarzania i czas oczekiwania procesu*, jak również zwiększa i równoważy obciążenie procesora i urządzeń zewnętrznych, oraz podnosi *przepustowość* systemu (ilość pracy wykonaną w jednostce czasu).

Motywacja

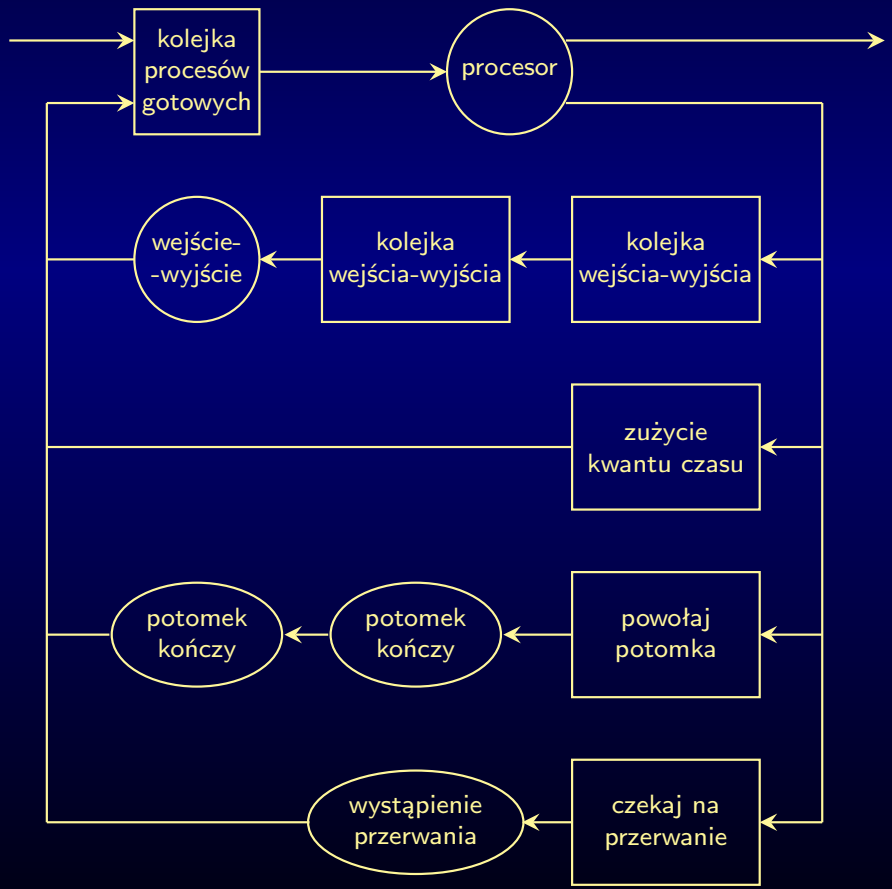


Wykonanie każdego procesu jest podzielone na fazy. Rozróżniamy dwa rodzaje faz: fazę procesora, w której procesowi jest przydzielony procesor i fazę wejścia-wyjścia, w której na rzecz procesu wykonywana jest operacja wejścia-wyjścia lub inna czynność nieangażująca procesora. Jeśli dwa lub większa liczba procesów byłaby wykonywana szeregowo, tak jak obrazuje to ilustracja obok (linia pogrubiona - faza procesora, linia kropkowana - faza wejścia-wyjścia, linia kreskowana - czas przed lub po wykonaniu procesu), to procesy musiałby czekać długo na swoje wykonanie, a urządzenia wejścia-wyjścia i procesor byłyby naprzemiennie bezczynne. Takie procesy można jednak wykonywać współbieżnie, tzn. jeśli proces, który do tej pory korzystał z procesora wszedł w fazę korzystania z urządzeń peryferyjnych, to procesor może zostać przydzielony innemu procesowi, który jest gotów do wykonania. Takie postępowanie zmniejsza *czas przetwarzania i czas oczekiwania procesu*, jak również zwiększa i równoważy obciążenie procesora i urządzeń zewnętrznych, oraz podnosi *przepustowość* systemu (ilość pracy wykonaną w jednostce czasu).

Kolejki

Podstawową strukturą danych wykorzystywaną w zarządzaniu procesami są opisane wcześniej deskryptory procesów. Bloki kontrolne wszystkich procesów, które są gotowe do wykonania powiązane są w *kolejkę procesów gotowych*. Wyraz kolejka w tym kontekście nie oznacza konkretnej struktury danych, a miejsce w którym procesy oczekują na przydział procesora. Taka kolejka może być stosem, kolejką FIFO lub innym rodzajem listy, a nawet innym rodzajem struktur danych (tablicą, drzewem). System operacyjny utrzymuje również kolejki oczekiwania na realizację operacji wejścia-wyjścia. Te kolejki tworzą deskryptory procesów będących w stanie oczekiwania. Często upraszczając nie mówimy, że deskryptor procesu jest w kolejce lecz, że proces znajduje się w kolejce.

Diagramy kolejek



Aby zwizualizować migrację procesów między kolejkami stosuje się często tzw. diagramy kolejek. Jeden z nich jest zaprezentowany obok.

Planiści

Decyzję o tym w jakiej kolejności i które procesy zostaną umieszczone w kolejkach podejmują mechanizmy systemu operacyjnego nazywane *mechanizmami szeregującymi* lub *planistami* (ang. scheduler). Najważniejsze mechanizmy szeregujące odpowiedzialne są za przydział procesora procesom gotowym do wykonania. We współczesnych, głównie interaktywnych systemach operacyjnych istnieje tylko jeden rodzaj takich planistów. Jest to *planista krótkoterminowy*. Taki planista wybiera z kolejki procesów gotowych proces, który jako następny otrzyma procesor. Wywoływany jest on bardzo często i musi krótko działać, aby nie tworzyć zbyt dużych narzutów czasowych. Drugi rodzaj planistów, który był właściwy głównie dla systemów wsadowych, to *planiści długoterminowi*. Taki planista był wywoływany wtedy, kiedy kończył swe działanie jakiś proces. Jego zadaniem było wybrać zadanie z puli zadań do wykonania, które trafiało do kolejki zadań gotowych do wykonania. Planista długoterminowy dbał o to by w tej kolejce znajdowały się zarówno procesy ograniczone przez wejście-wyjście jak i procesy ograniczone przez procesor. Dzięki temu praca tych jednostek była równoważona. Dbał on również o zachowanie stopnia *wieloprogramowości* (stałej w czasie liczby procesów w pamięci). W niektórych systemach operacyjnych stosowano również *planistów średnioterminowych*. Ich zadanie polegało na podejmowaniu decyzji, który z procesów ma być wycofany z pamięci operacyjnej do pamięci pomocniczej, aby umożliwić wykonanie innym procesom.

Przełączanie kontekstu

Jeśli proces traci procesor na rzecz innego procesu, lub w wyniku innego zdarzenia (np.: obsługa przerwania, operacja wejścia-wyjścia), to należy zapamiętać informacje niezbędne do kontynuowania jego wykonania w przyszłości. Te informacje nazywane są *kontekstem procesu*. Kontekst procesu, który utracił procesor zastępowany jest kontekstem procesu, który procesor uzyskał. Tę operację nazywa się *przełączaniem kontekstu* i dąży się do tego, aby czas poświęcony na nią był jak najmniejszy. Kontekst procesu tracącego procesor jest zapamiętywany w jego deskryptorze.

Koordynator

Przekazaniem sterowania do procesu, który wybrał planista krótkoterminowy zajmuje się koordynator (ang. dispatcher). Jego zadanie polega na przełączeniu kontekstu procesów, przełączeniu procesora w tryb użytkownika i wykonaniu skoku do adresu w programie użytkownika, pod którym znajduje się kolejny rozkaz do wykonania przez program.

Wstęp

Sytuacja, która została przedstawiona we wstępie do opisu zagadnienia planowania procesów jest idealną i w rzeczywistości występuje rzadko (jeśli w ogóle). We wszystkich procesach występują naprzemiennie fazy procesora i wejścia-wyjścia, ale procesy ograniczone przez procesor charakteryzują się małą liczbą faz procesora, które są za to długie, z kolei procesy ograniczone przez wejście-wyjście mają dużo bardzo krótkich faz procesora. Bazując na tych spostrzeżeniach można optymalizować różne wielkości związane z efektywnością pracy systemu komputerowego, np.:

1. wykorzystanie procesora,
2. przepustowość (liczbę wykonanych procesów w jednostce czasu),
3. czas cyklu przetwarzania (czas od przedłożenia procesu do wykonania, do jego zakończenia),
4. czas oczekiwania (czas od przedłożenia procesu do rozpoczęcia jego wykonania),
5. czas odpowiedzi (czas reakcji procesu na żądania użytkownika - ważny w systemach interaktywnych),

Zostaną zaprezentowane podstawowe strategie używane przez planistę krótkoterminowego, z których większość będzie oceniona pod względem średniego czasu oczekiwania procesów.

Wstęp

Sytuacja, która została przedstawiona we wstępie do opisu zagadnienia planowania procesów jest idealną i w rzeczywistości występuje rzadko (jeśli w ogóle). We wszystkich procesach występują naprzemiennie fazy procesora i wejścia-wyjścia, ale procesy ograniczone przez procesor charakteryzują się małą liczbą faz procesora, które są za to długie, z kolei procesy ograniczone przez wejście-wyjście mają dużo bardzo krótkich faz procesora. Bazując na tych spostrzeżeniach można optymalizować różne wielkości związane z efektywnością pracy systemu komputerowego, np.:

1. wykorzystanie procesora,
2. przepustowość (liczbę wykonanych procesów w jednostce czasu),
3. czas cyklu przetwarzania (czas od przedłożenia procesu do wykonania, do jego zakończenia),
4. czas oczekiwania (czas od przedłożenia procesu do rozpoczęcia jego wykonania),
5. czas odpowiedzi (czas reakcji procesu na żądania użytkownika - ważny w systemach interaktywnych),

Zostaną zaprezentowane podstawowe strategie używane przez planistę krótkoterminowego, z których większość będzie oceniona pod względem średniego czasu oczekiwania procesów.

Wstęp

Sytuacja, która została przedstawiona we wstępie do opisu zagadnienia planowania procesów jest idealną i w rzeczywistości występuje rzadko (jeśli w ogóle). We wszystkich procesach występują naprzemiennie fazy procesora i wejścia-wyjścia, ale procesy ograniczone przez procesor charakteryzują się małą liczbą faz procesora, które są za to długie, z kolei procesy ograniczone przez wejście-wyjście mają dużo bardzo krótkich faz procesora. Bazując na tych spostrzeżeniach można optymalizować różne wielkości związane z efektywnością pracy systemu komputerowego, np.:

1. wykorzystanie procesora,
2. przepustowość (liczbę wykonanych procesów w jednostce czasu),
3. czas cyklu przetwarzania (czas od przedłożenia procesu do wykonania, do jego zakończenia),
4. czas oczekiwania (czas od przedłożenia procesu do rozpoczęcia jego wykonania),
5. czas odpowiedzi (czas reakcji procesu na żądania użytkownika - ważny w systemach interaktywnych),

Zostaną zaprezentowane podstawowe strategie używane przez planistę krótkoterminowego, z których większość będzie oceniona pod względem średniego czasu oczekiwania procesów.

Wstęp

Sytuacja, która została przedstawiona we wstępie do opisu zagadnienia planowania procesów jest idealną i w rzeczywistości występuje rzadko (jeśli w ogóle). We wszystkich procesach występują naprzemiennie fazy procesora i wejścia-wyjścia, ale procesy ograniczone przez procesor charakteryzują się małą liczbą faz procesora, które są za to długie, z kolei procesy ograniczone przez wejście-wyjście mają dużo bardzo krótkich faz procesora. Bazując na tych spostrzeżeniach można optymalizować różne wielkości związane z efektywnością pracy systemu komputerowego, np.:

1. wykorzystanie procesora,
2. przepustowość (liczbę wykonanych procesów w jednostce czasu),
3. czas cyklu przetwarzania (czas od przedłożenia procesu do wykonania, do jego zakończenia),
4. czas oczekiwania (czas od przedłożenia procesu do rozpoczęcia jego wykonania),
5. czas odpowiedzi (czas reakcji procesu na żądania użytkownika - ważny w systemach interaktywnych),

Zostaną zaprezentowane podstawowe strategie używane przez planistę krótkoterminowego, z których większość będzie oceniona pod względem średniego czasu oczekiwania procesów.

Wstęp

Sytuacja, która została przedstawiona we wstępie do opisu zagadnienia planowania procesów jest idealną i w rzeczywistości występuje rzadko (jeśli w ogóle). We wszystkich procesach występują naprzemiennie fazy procesora i wejścia-wyjścia, ale procesy ograniczone przez procesor charakteryzują się małą liczbą faz procesora, które są za to długie, z kolei procesy ograniczone przez wejście-wyjście mają dużo bardzo krótkich faz procesora. Bazując na tych spostrzeżeniach można optymalizować różne wielkości związane z efektywnością pracy systemu komputerowego, np.:

1. wykorzystanie procesora,
2. przepustowość (liczbę wykonanych procesów w jednostce czasu),
3. czas cyklu przetwarzania (czas od przedłożenia procesu do wykonania, do jego zakończenia),
4. czas oczekiwania (czas od przedłożenia procesu do rozpoczęcia jego wykonania),
5. czas odpowiedzi (czas reakcji procesu na żądania użytkownika - ważny w systemach interaktywnych),

Zostaną zaprezentowane podstawowe strategie używane przez planistę krótkoterminowego, z których większość będzie oceniona pod względem średniego czasu oczekiwania procesów.

Wstęp

Sytuacja, która została przedstawiona we wstępie do opisu zagadnienia planowania procesów jest idealną i w rzeczywistości występuje rzadko (jeśli w ogóle). We wszystkich procesach występują naprzemiennie fazy procesora i wejścia-wyjścia, ale procesy ograniczone przez procesor charakteryzują się małą liczbą faz procesora, które są za to długie, z kolei procesy ograniczone przez wejście-wyjście mają dużo bardzo krótkich faz procesora. Bazując na tych spostrzeżeniach można optymalizować różne wielkości związane z efektywnością pracy systemu komputerowego, np.:

1. wykorzystanie procesora,
2. przepustowość (liczbę wykonanych procesów w jednostce czasu),
3. czas cyklu przetwarzania (czas od przedłożenia procesu do wykonania, do jego zakończenia),
4. czas oczekiwania (czas od przedłożenia procesu do rozpoczęcia jego wykonania),
5. czas odpowiedzi (czas reakcji procesu na żądania użytkownika - ważny w systemach interaktywnych),

Zostaną zaprezentowane podstawowe strategie używane przez planistę krótkoterminowego, z których większość będzie oceniona pod względem średniego czasu oczekiwania procesów.

Wywłaszczanie i jego brak

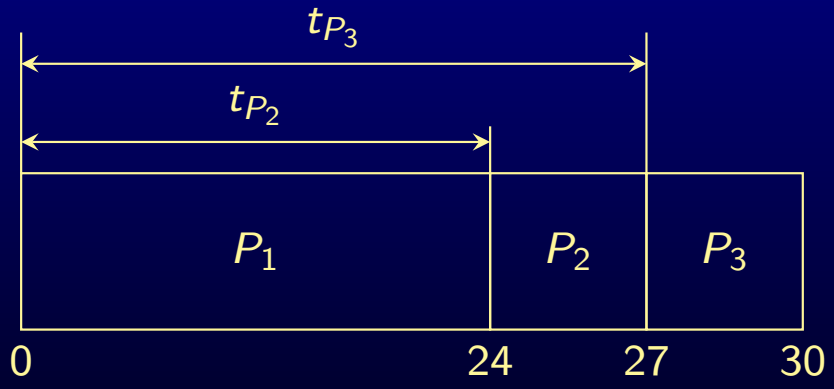
W zależności od tego, czy proces wyłącznie dobrowolnie oddaje procesor (np.: po zgłoszeniu zapotrzebowania na operację wejścia-wyjścia), czy też może mu on zostać odebrany (np.: w wyniku działania czasomierza), rozróżniamy dwa rodzaje strategii szeregowania procesów: szeregowanie *bez wywłaszczeń* i szeregowanie *z wywłaszczaniem* (ang. preemptive). Systemy operacyjne stosujące pierwszą strategię nazywamy systemami z *kooperacją*, a systemy stosujące drugą po prostu systemami *z wywłaszczaniem*. Większość współczesnych systemów należy do drugiej kategorii.

FCFS

Algorytm FCFS (ang. First-Come First-Served), nazywany także FIFO jest najprostszym algorytmem szeregowania. Procesor jest przyznawany procesom w takiej kolejności w jakiej są one umieszczone w kolejce procesów gotowych. Brak jest wywłaszczania. FCFS może prowadzić do *efektu konwoju*, tzn. oczekiwania procesów ograniczonych przez *wejście-wyjście* na zakończenie realizacji długich faz procesora procesu ograniczonego przez procesor.

Średni czas oczekiwania

Proces	Czas trwania fazy
P_1	24 ms
P_2	3 ms
P_3	3 ms



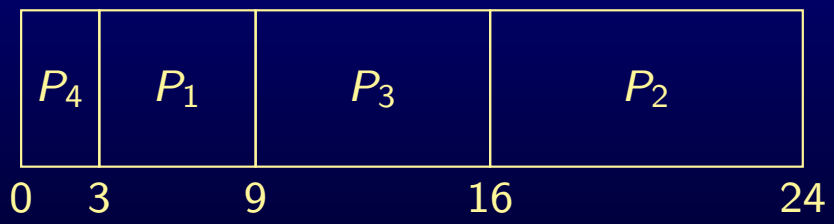
Średni czas oczekiwania można policzyć posługując się diagramem Gantta. Rysunek obok przedstawia taki diagram (w jego skład nie wchodzą linie wymiarowe). Obrazuje on kolejność wykonania procesów. Na jego dole znajdują się czasy oczekiwania kolejnych procesów na wykonanie. Są one sumą czasów wykonania ich poprzedników. Średni czas oczekiwania procesów jest w przypadku algorytmu FCFS średnią arytmetyczną czasów oczekiwania poszczególnych procesów w kolejce, czyli $\frac{1}{n} \cdot \sum_{k=1}^{n} t_k = (0 + 24 + 27)/3 = 17ms$

Algorytm SJF

Algorytm SJF (ang. Shortest Job First) - najpierw najkrótsze zadanie jest algorytmem optymalnym, jeśli chcemy uzyskań minimalny czas oczekiwania procesów. Procesor jest przydzielany procesom według długości trwania ich faz procesora (tzn. zaczynając od tego o najkrótszej fazie, a kończąc na tym o najdłuższej). Problemem jest jednak przewidywanie czasu trwania kolejnej fazy procesu. Stosuje się oszacowania statystyczne, bazujące na historii wykonania procesu. Te oszacowania opierają się na średniej wykładniczej. Jeśli zapamiętywane są tylko dwie ostatnie fazy procesora procesu, to ta średnia ma postać: $\tau_{n+1} = \alpha \cdot \tau_n + (1 - \alpha) \cdot \tau_{n-1}$, gdzie τ jest czasem wykonania fazy procesora, a α współczynnikiem z przedziału [0,1]. Jeśli jest zapamiętywanych więcej faz, to wzór na średnią wykładniczą staje się bardziej skomplikowany: $\tau_{n+1} = \alpha \cdot \tau_n + (1 - \alpha) \cdot \alpha \cdot \tau_{n-1} + \ldots + (1 - \alpha)^j \cdot \alpha \cdot \tau_{n-j} + \ldots + (1 - \alpha)^{n+1} \cdot \tau_0$ Za τ_0 przyjmuje się stałą lub wartość średnią dla wszystkich procesów w systemie. Opis odnosi się do algorytmu w wersji bez wywłaszczeń.

Średni czas oczekiwania

Proces	Czas trwania fazy
P_1	6 ms
P_2	8 ms
P_3	7 ms
P_4	3 ms



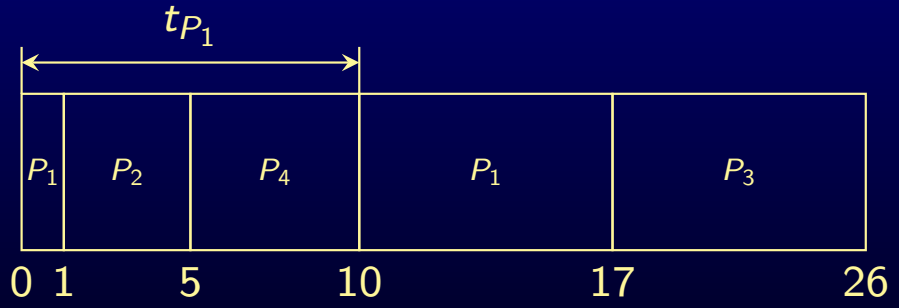
Średni czas oczekiwania procesów dla algorytmu SJF bez wywłaszczeń liczony jest w ten sam sposób jak dla algorytmu FCFS, czyli dla danych z rysunku będzie to: $(3 + 16 + 9 + 0)/4 = 7ms$.

Algorytm SRT

Algorytm SRT (ang. Shortest Remaining Time) - najpierw najkrótszy pozostały czas, jest odmianą algorytmu SJF z wywłaszczaniem. Procesor jest odbierany wykonywanemu procesowi wtedy, kiedy do kolejki procesów gotowych nadchodzi proces o czasie trwania fazy procesora krótszym, niż czas konieczny do zakończenia fazy procesora bieżącego zadania.

Średni czas oczekiwania

Proces	Czas trwania fazy	Czas nadejścia
P_1	8 ms	0 ms
P_2	4 ms	1 ms
P_3	9 ms	2 ms
P_4	5 ms	3 ms



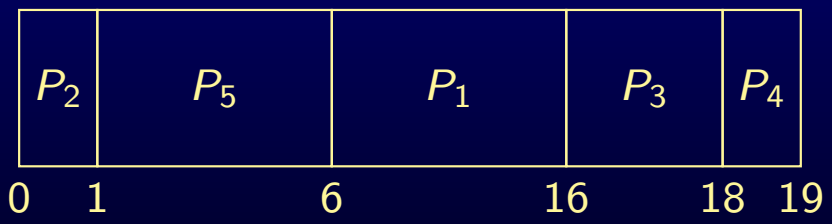
Licząc średni czas oczekiwania procesów dla algorytmu SRT należy uwzględnić czasy częściowego wykonania faz procesora wywłaszczanych procesów i odjąć je od czasów ich oczekiwania. Należy również uwzględnić i odjąć od czasu oczekiwania czas nadejścia do kolejki procesów gotowych, dla wszystkich procesów. Reasumując otrzymujemy: $((10-1)+(1-1)+(17-2)+(5-3))/4 = 26/4 = 6,5ms$.

Szeregowanie priorytetowe

Algorytm priorytetowy przydziela procesor procesom według przypisanego im *priorytetu*. Priorytet jest liczbą całkowitą, która określa „ważność" procesu. Zazwyczaj im niższa jest wartość tej liczby, tym proces ma wyższy priorytet. Priorytety mogą być przydzielane *zewnętrznie* jak i *wewnętrznie*. Szeregowanie priorytetowe może odbywać się z wywłaszczaniem, jak i bez. Algorytm SJF (w obu wersjach) jest formą algorytmu priorytetowego. Przy planowaniu priorytetowym może dojść do zjawiska, które nazywamy *głodzeniem procesu*. Zachodzi ono wtedy, gdy w systemie jest proces o bardzo niskim priorytecie. Jeśli inne procesy będą miały zawsze wyższe priorytety to rzeczony proces może nigdy nie otrzymać dostępu do procesora. Aby uniknąć tego zjawiska stosuje się okresowe *postarzanie procesów*, czyli zwiększanie ich priorytetów.

Średni czas oczekiwania

Proces	Czas trwania fazy	Priorytet
P_1	10 ms	3
P_2	1 ms	1
P_3	2 ms	3
P_4	1 ms	4
P_5	5 ms	2

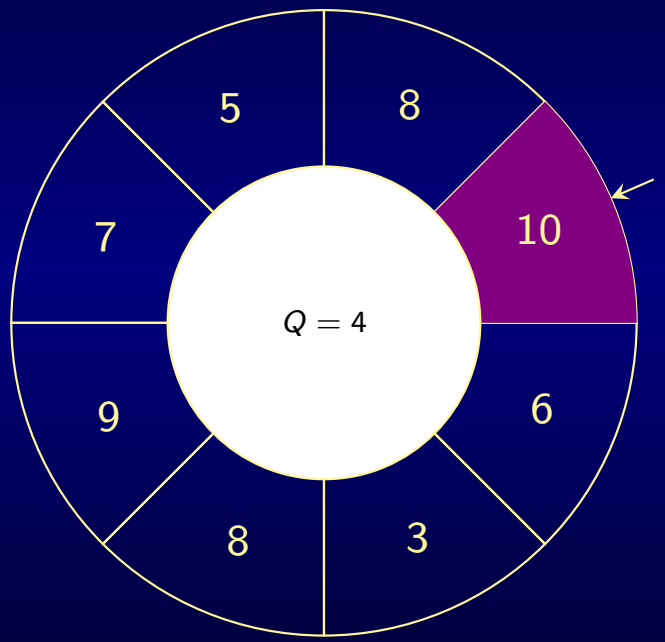


Średni czas oczekiwania będzie policzony dla algorytmu priorytetowego bez wywłaszczeń. Dla danych z rysunku obok będzie to $(1 + 6 + 16 + 18)/5 = 8,2ms$.

Szeregowanie rotacyjne

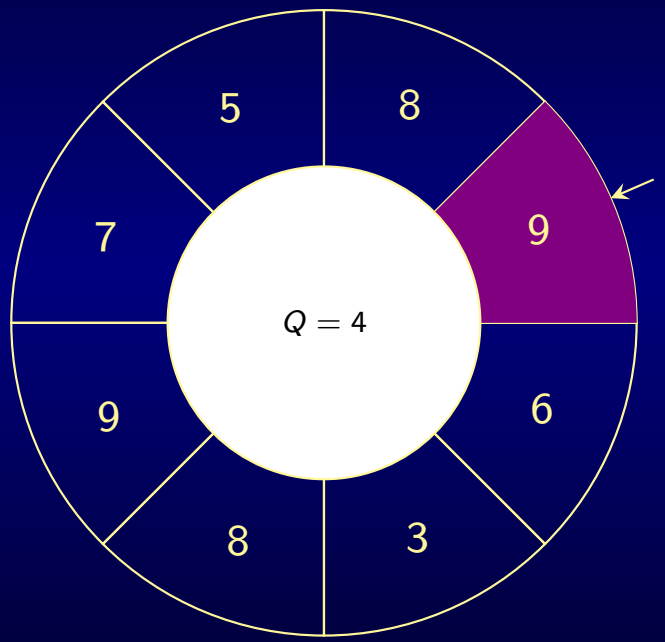
Algorytm rotacyjny (ang. *round robin*) zwany także karuzelowym jest algorytmem z wywłaszczeniami, stosowanym głównie w systemach interaktywnych. Każdemu procesowi system operacyjny przyznaje pewien *kwant* czasu na wykonanie fazy procesora. Jeśli proces wykona swoją fazę wcześniej, to dobrowolnie oddaje procesor innym procesom. Jeśli nie to procesor jest mu odbierany, a on wędruje na koniec kolejki procesów gotowych i musi czekać aż wszystkie pozostałe procesy wykorzystają swój kwant czasu. Kolejka procesów gotowych jest więc listą cykliczną. Dobierając kwant czasu należy uważać, aby nie był zbyt długi (wówczas otrzymamy algorytm FCFS) lub zbyt krótki (wówczas procesor więcej czasu będzie poświęcał na przełączanie kontekstu, niż na wykonywanie procesów).

Algorytm rotacyjny



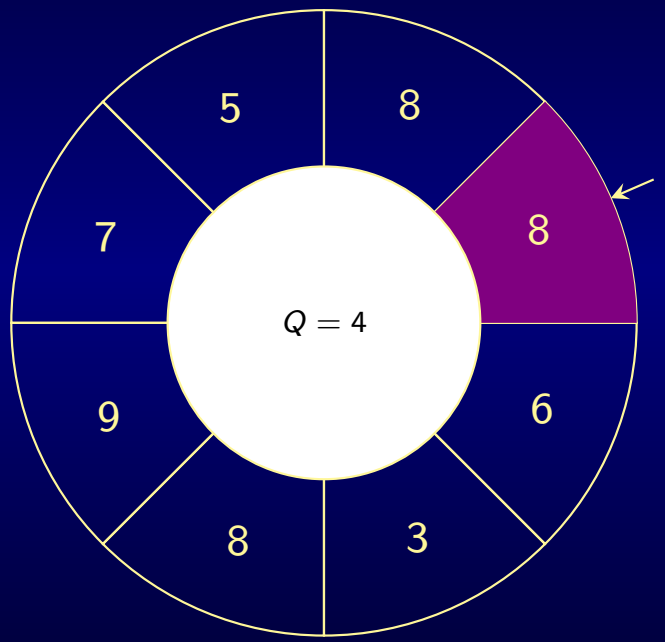
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



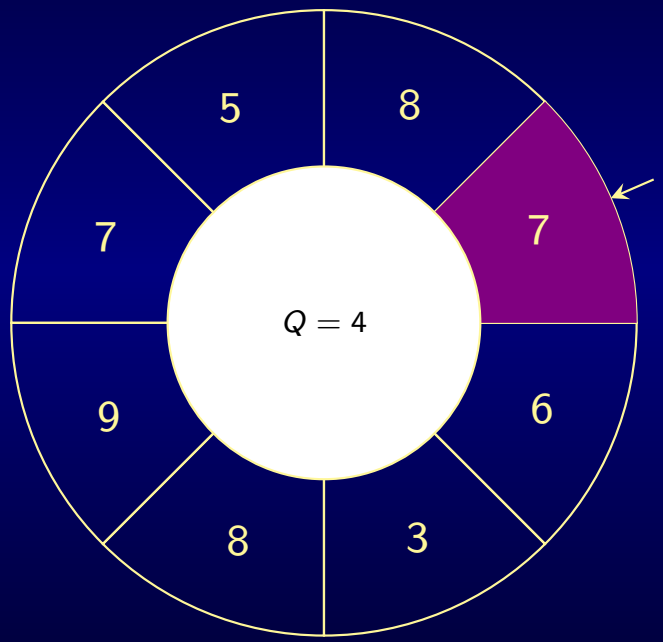
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



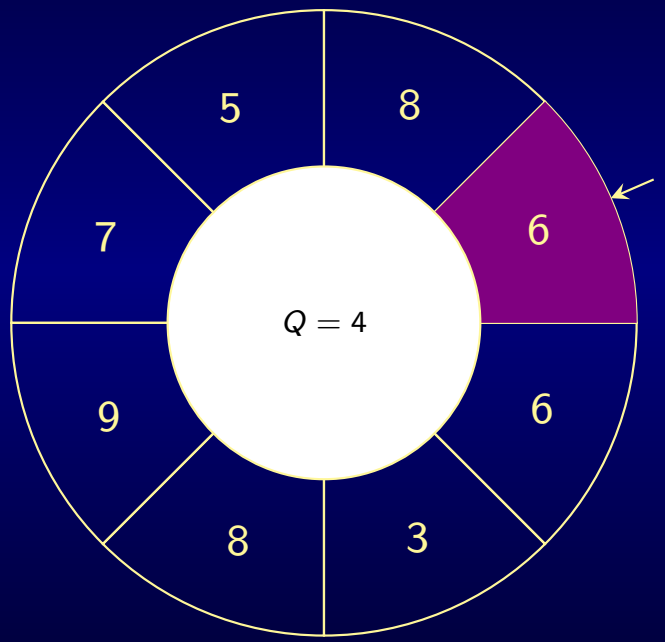
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



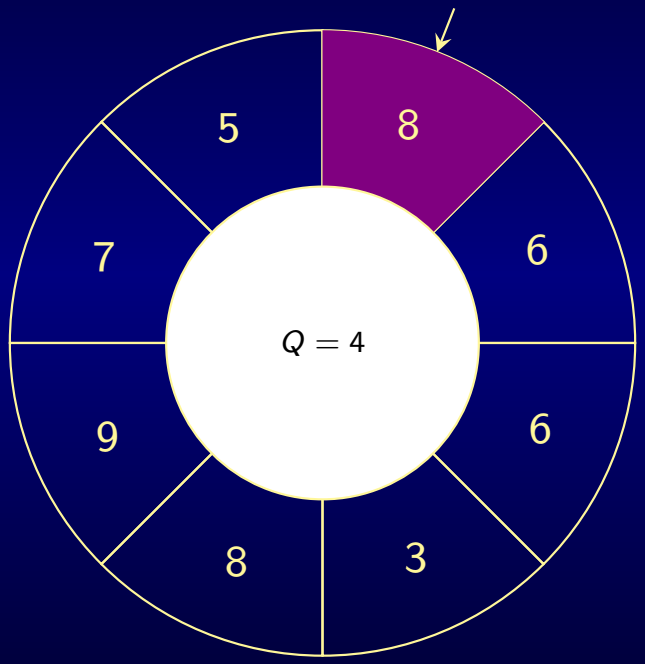
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



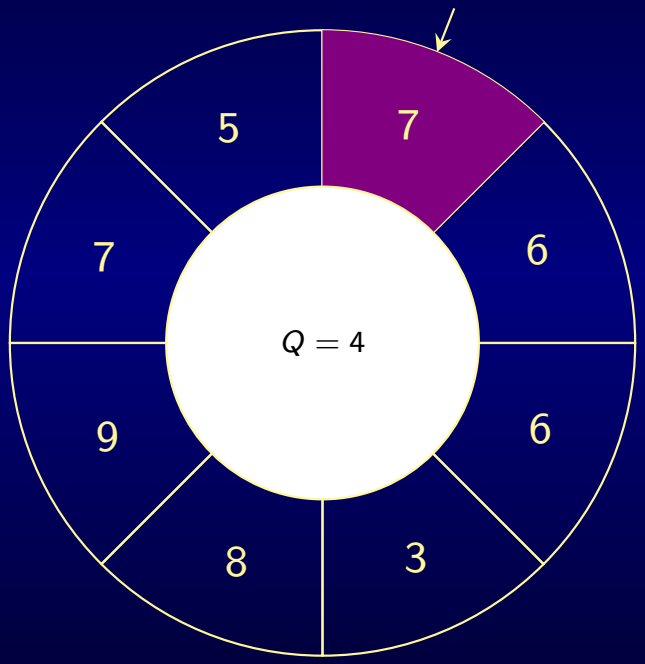
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



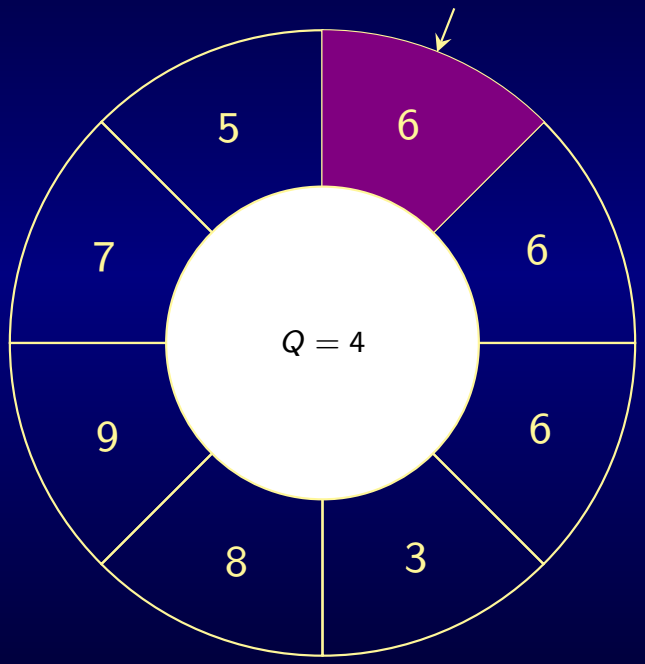
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



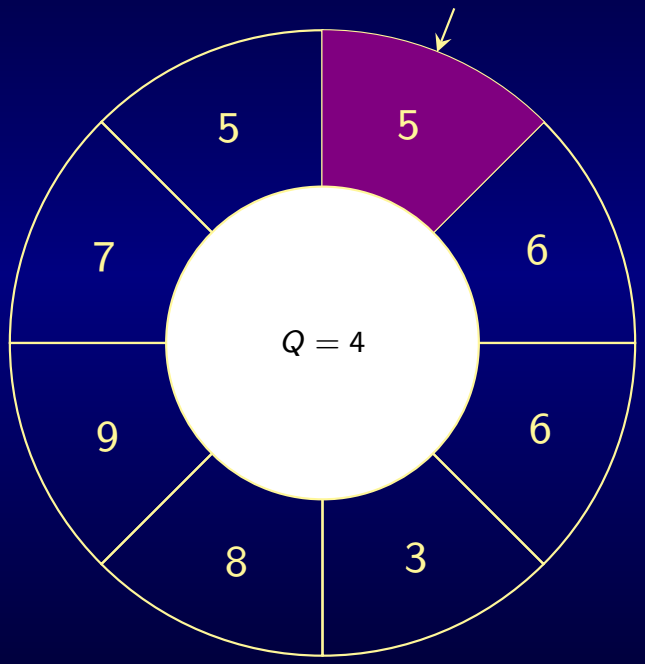
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



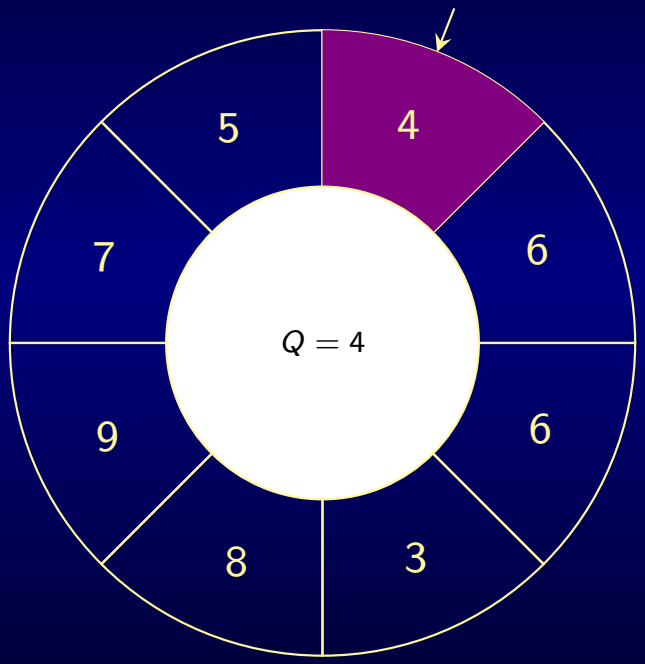
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



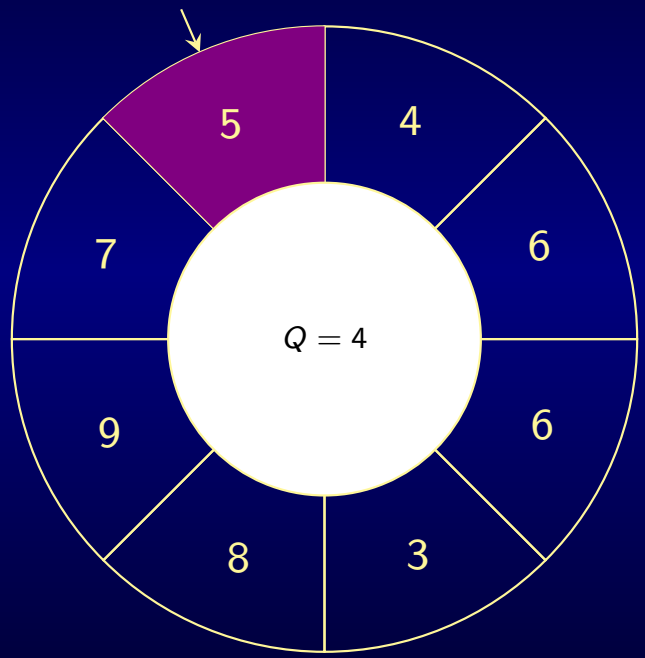
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



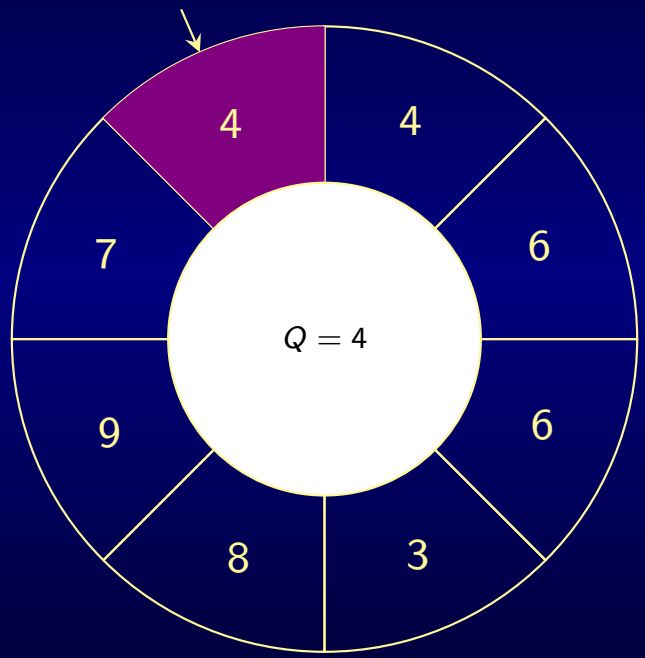
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



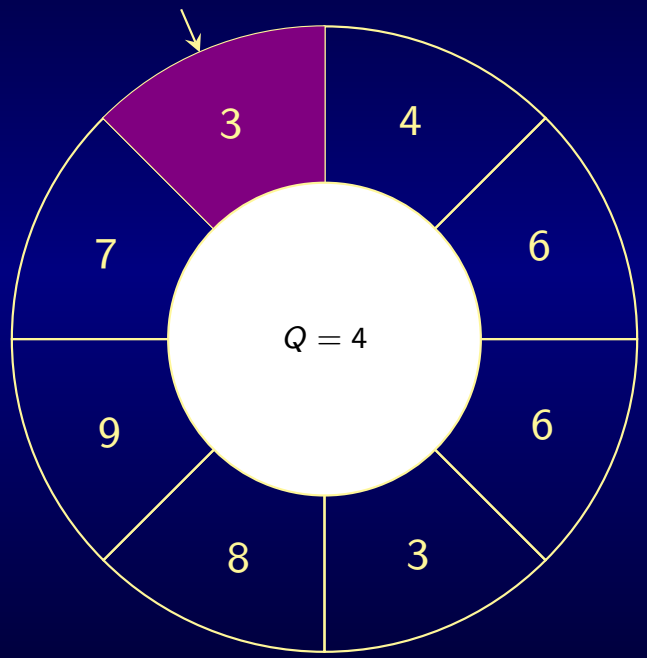
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



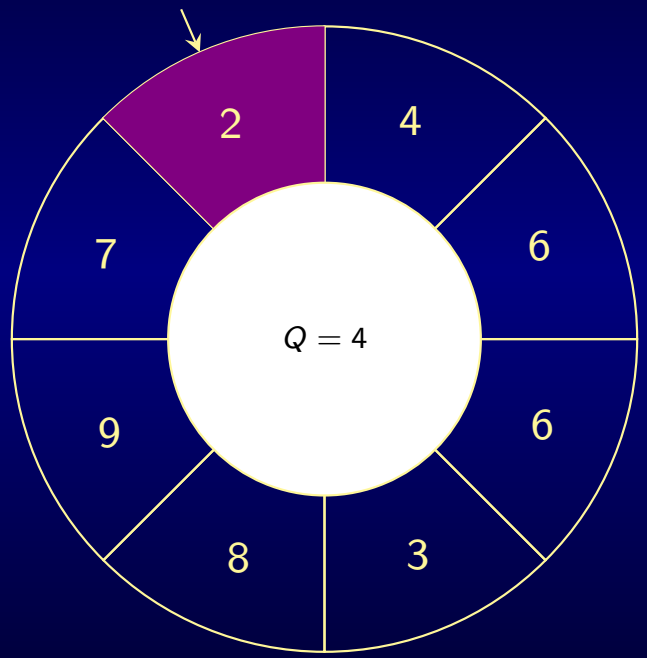
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



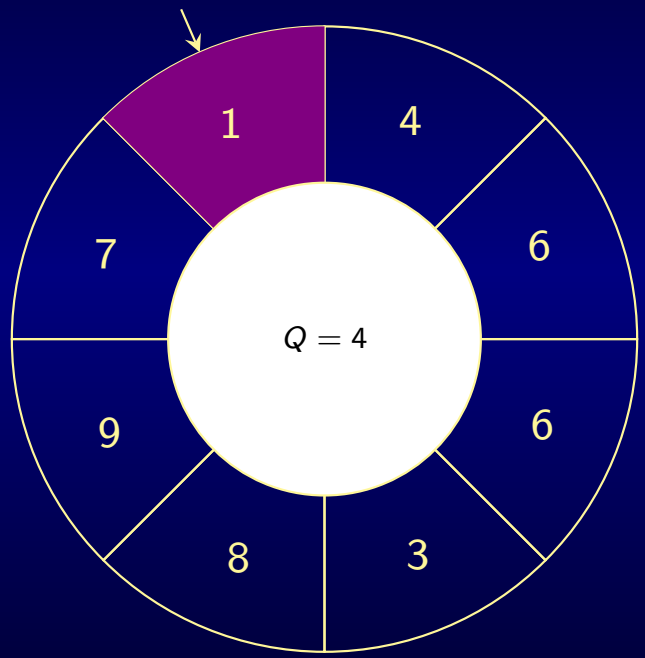
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



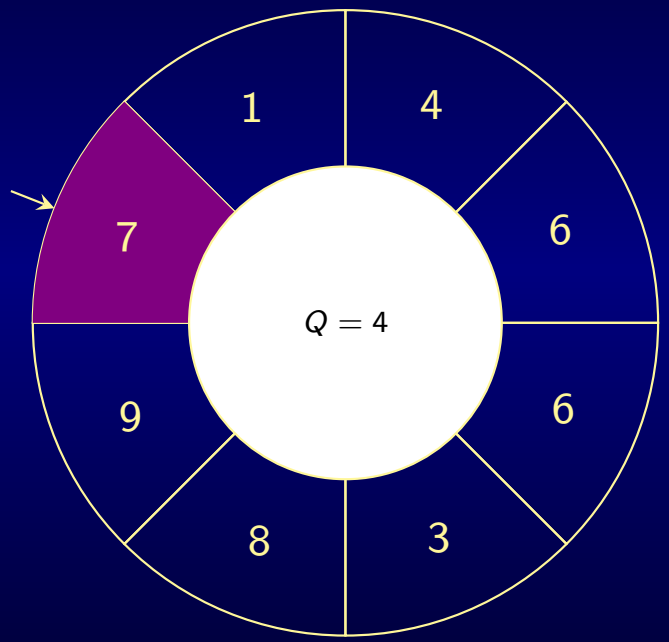
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



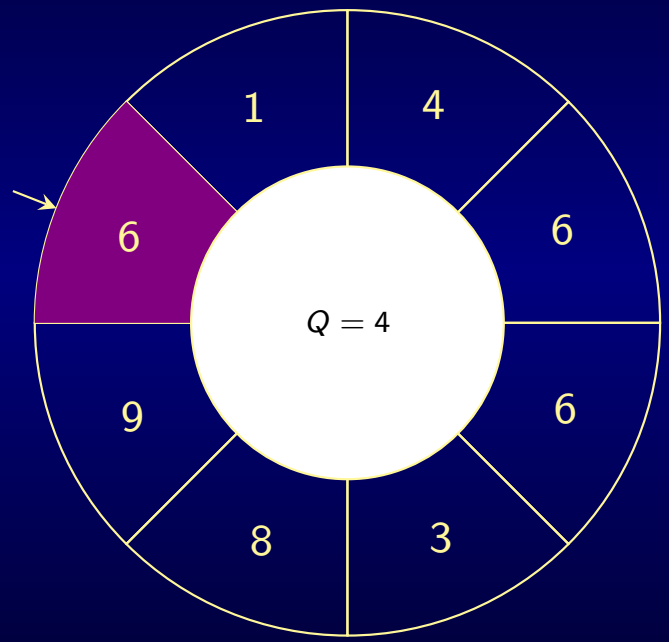
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



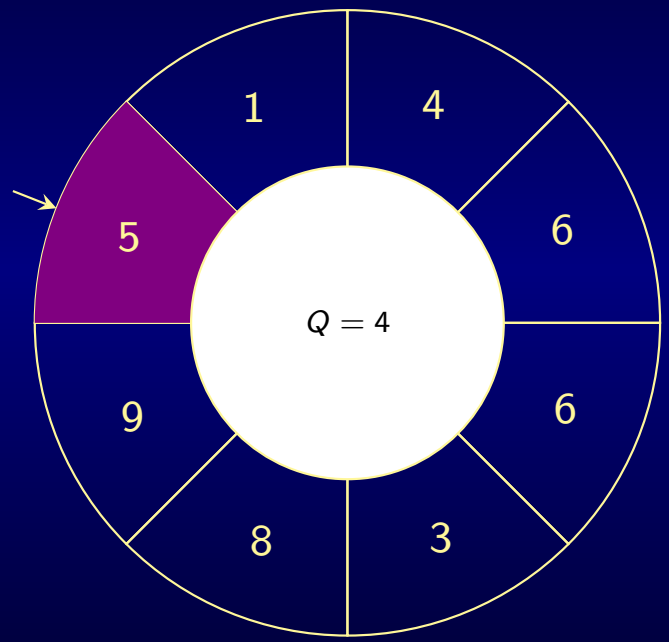
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



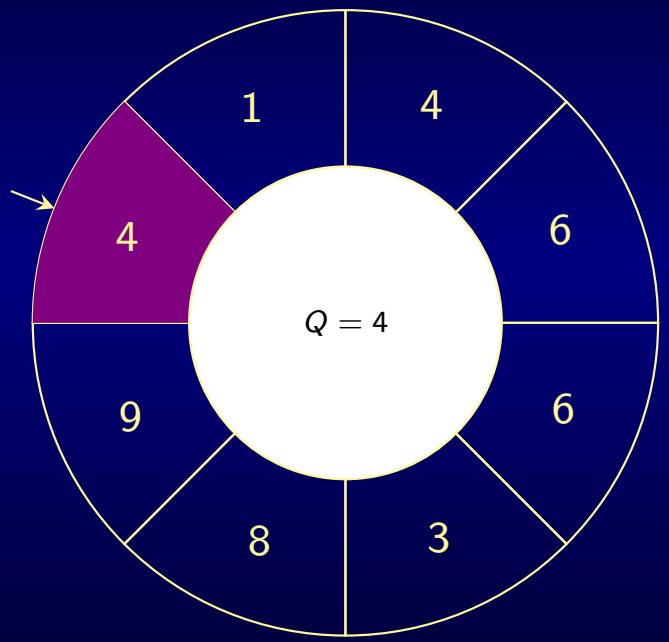
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



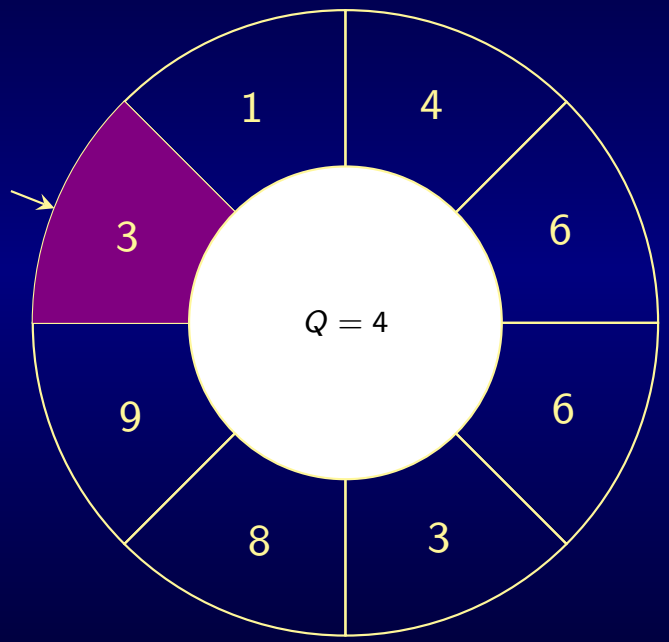
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



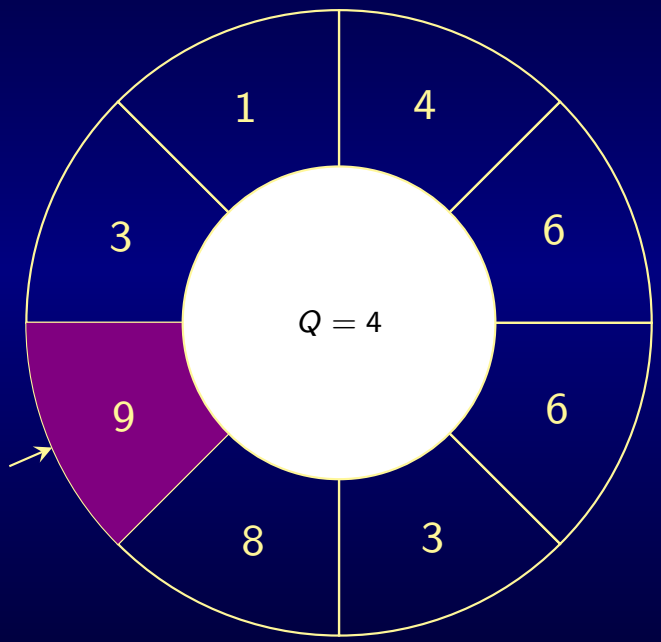
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



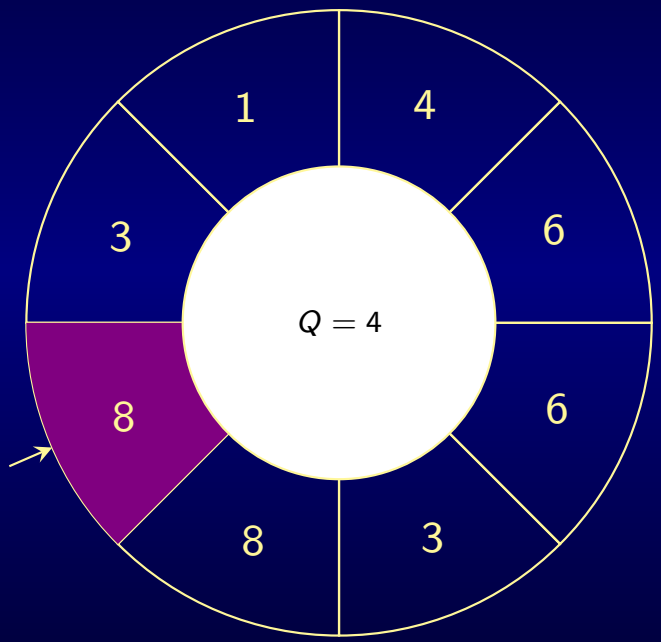
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



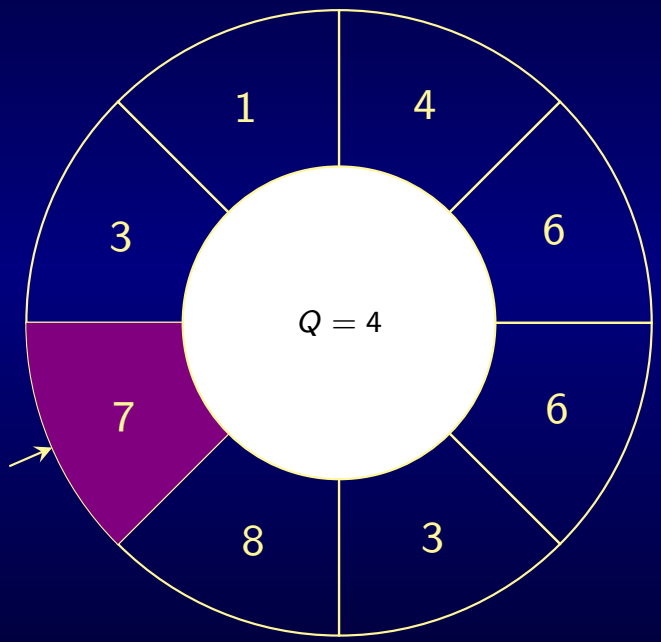
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



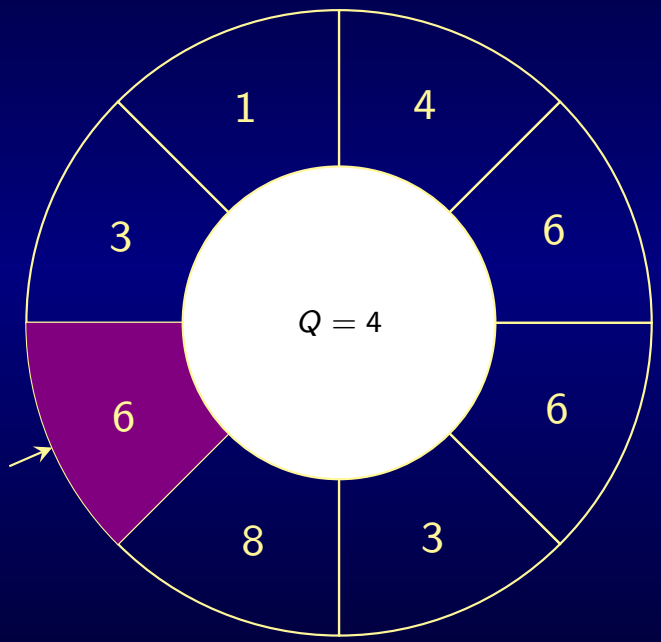
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



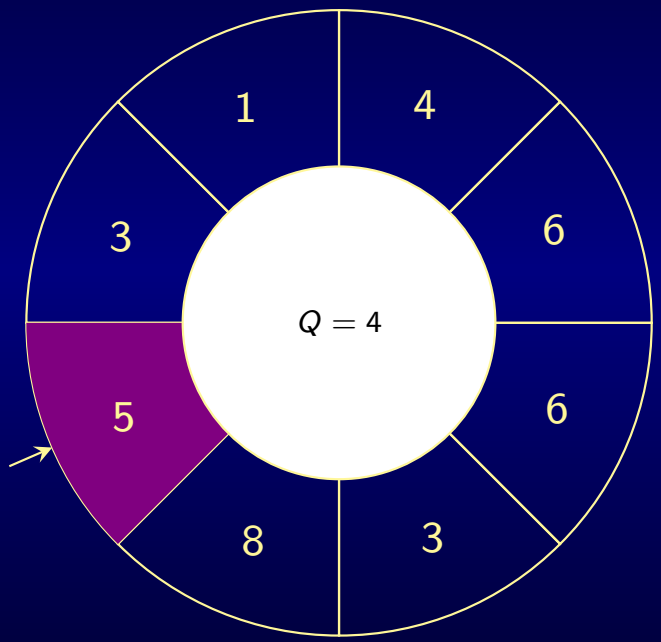
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



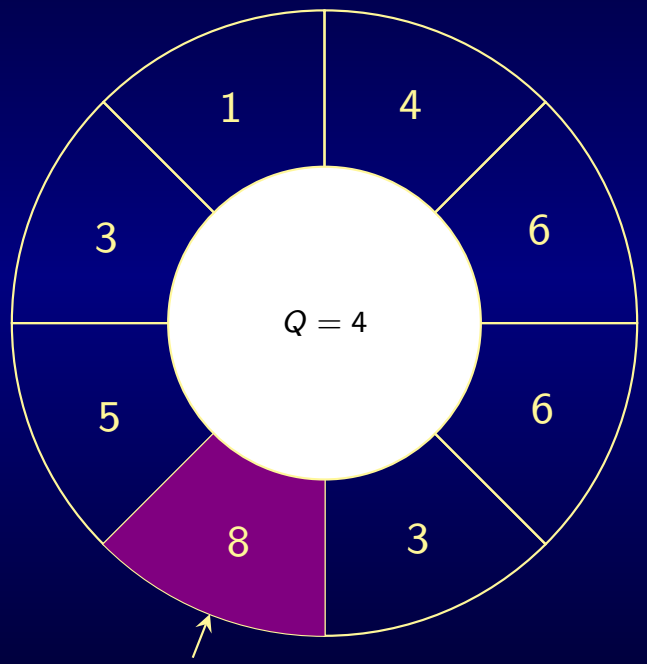
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



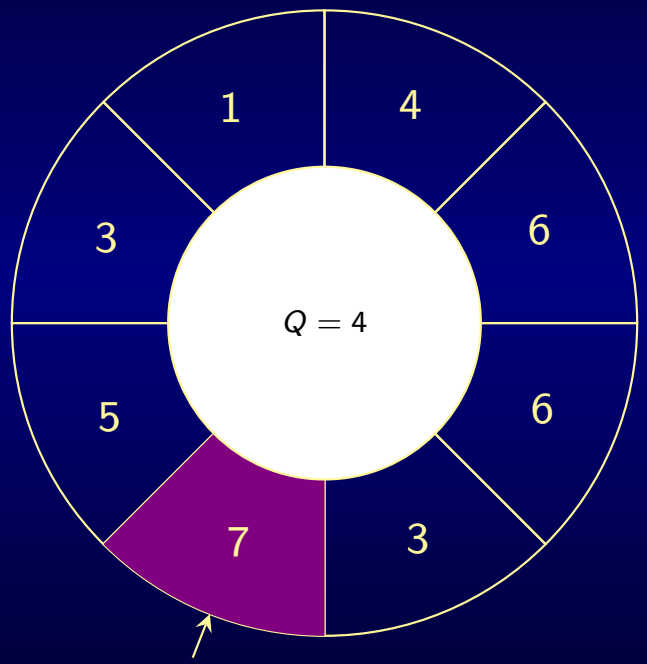
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



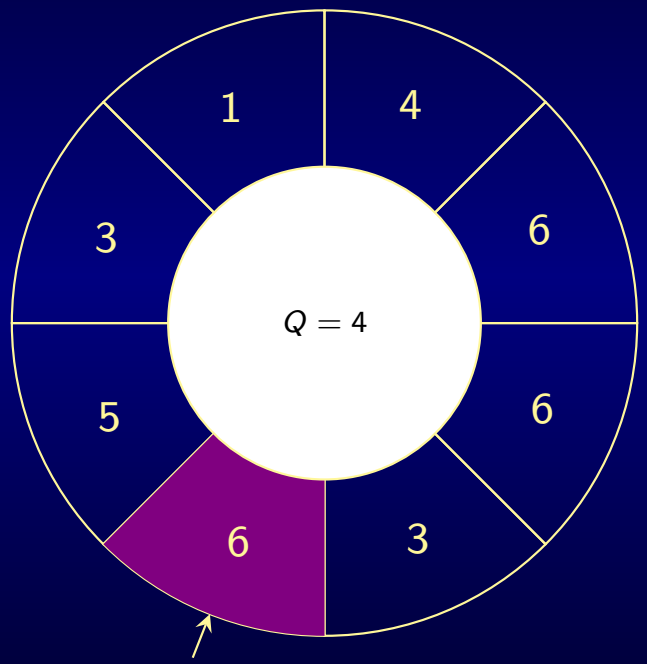
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



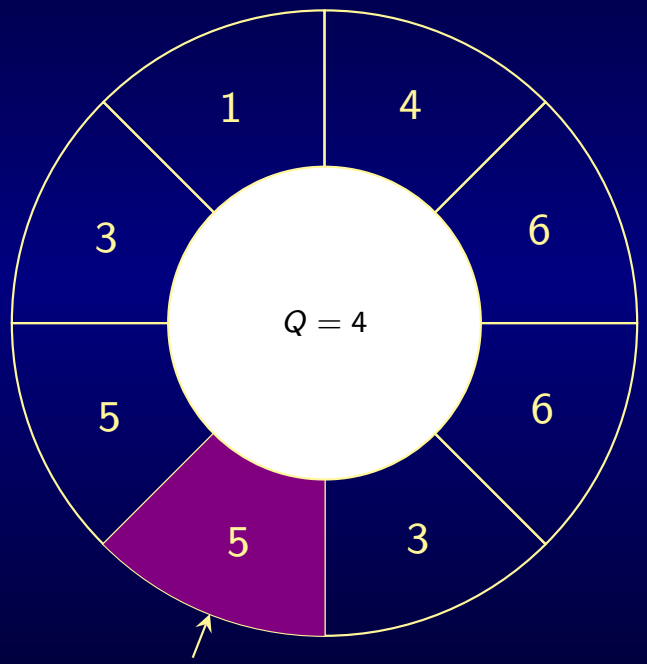
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



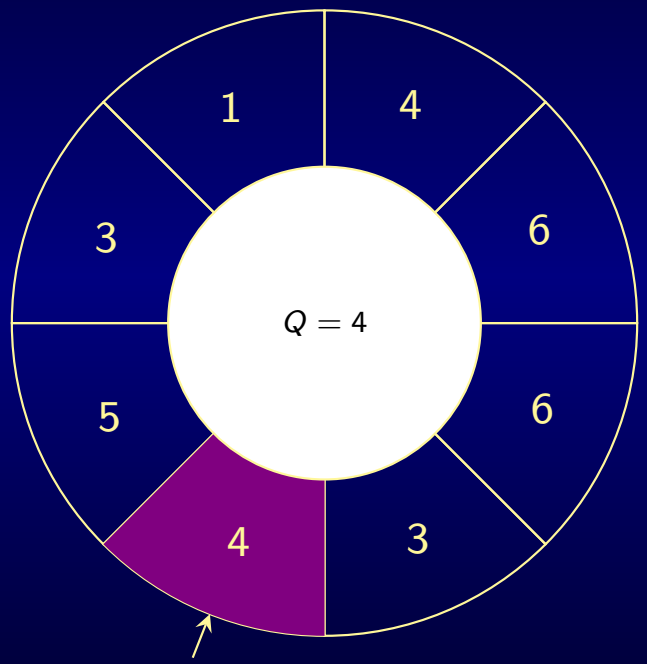
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



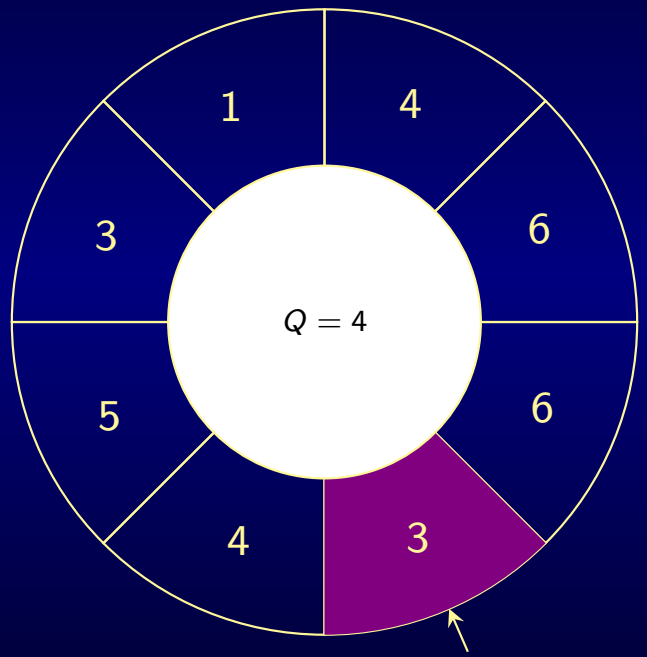
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



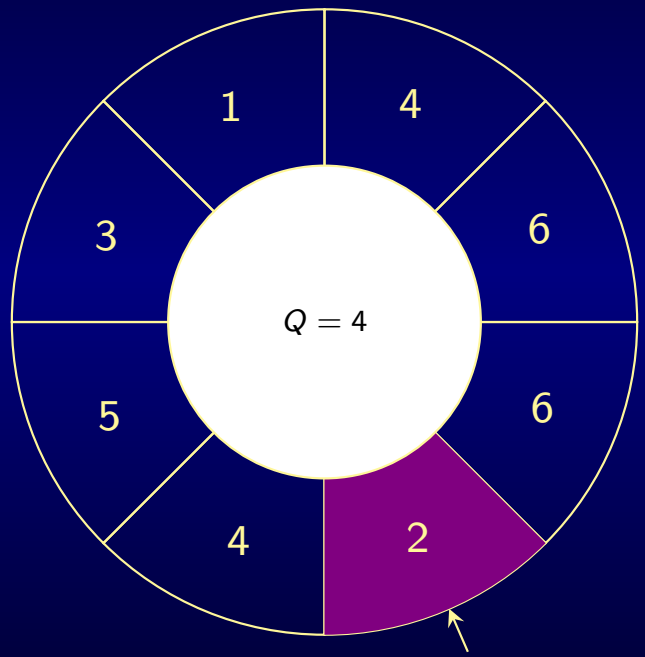
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



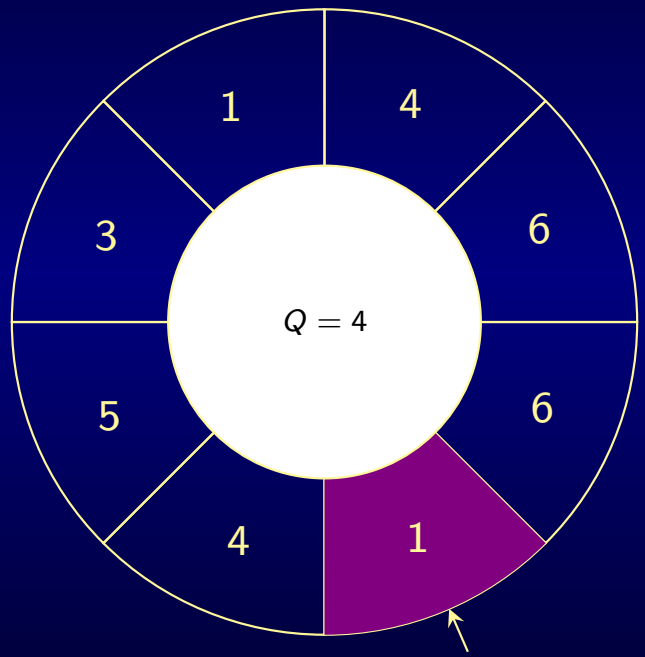
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



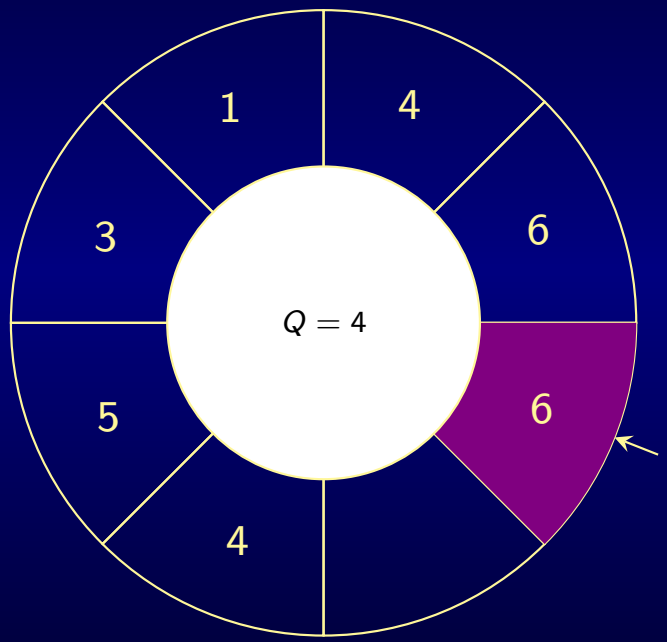
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



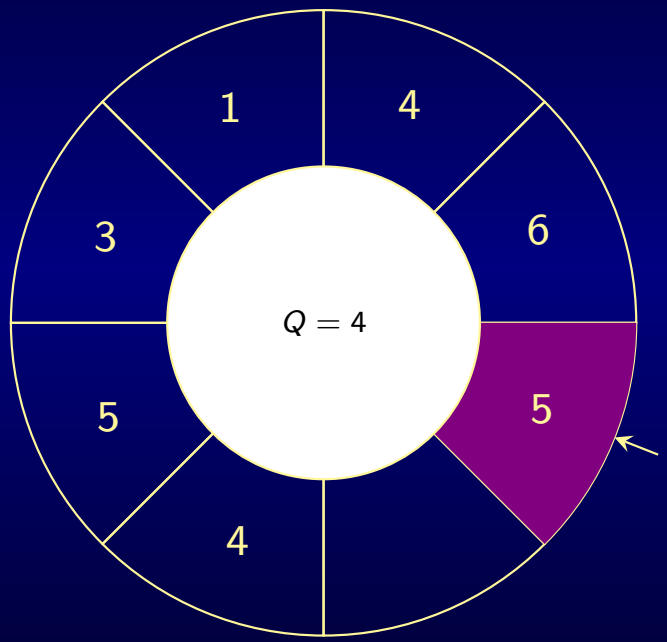
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



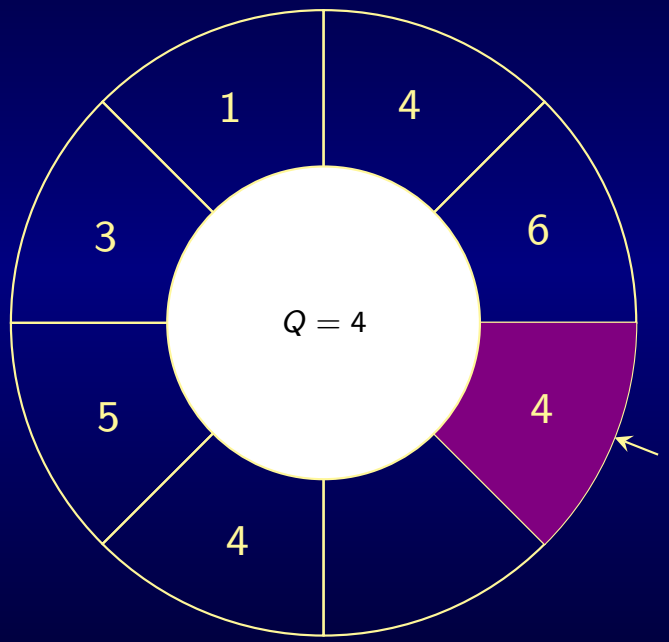
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



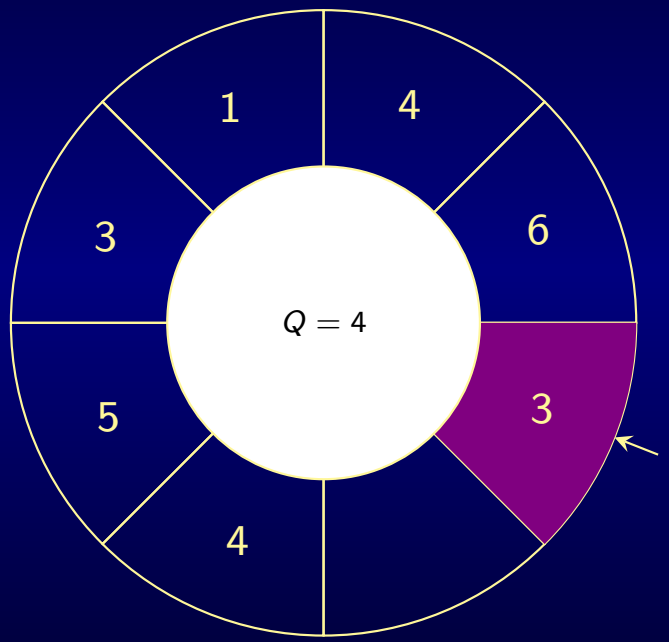
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



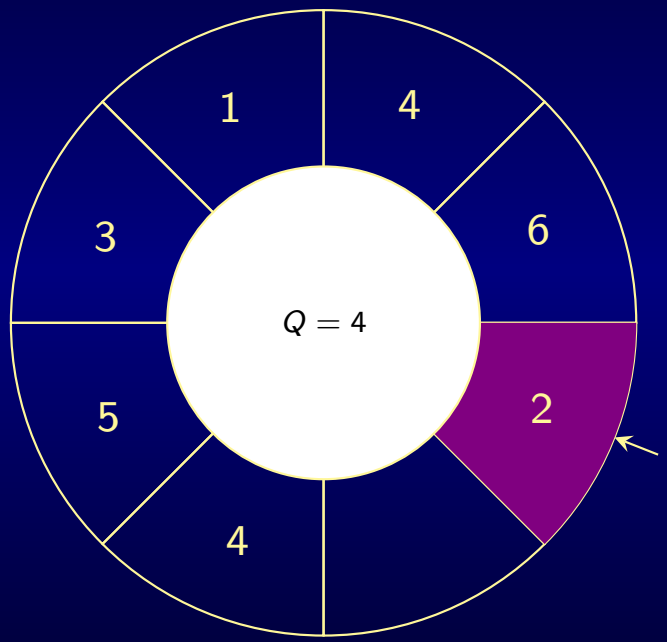
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



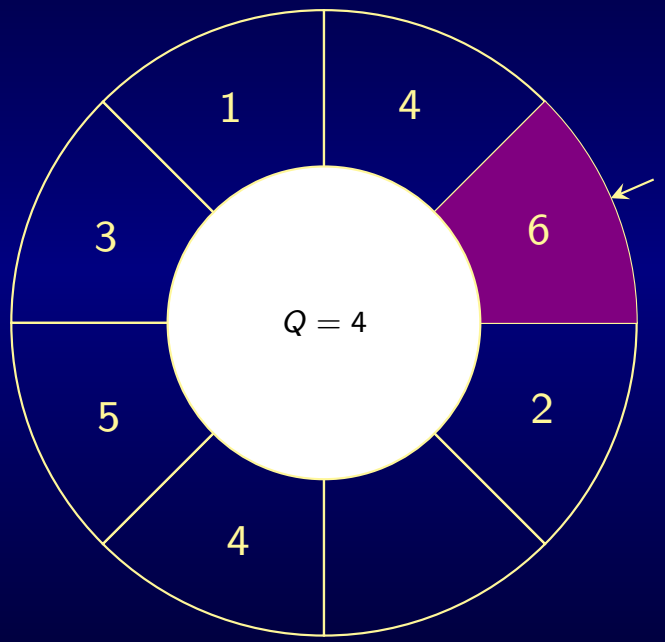
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



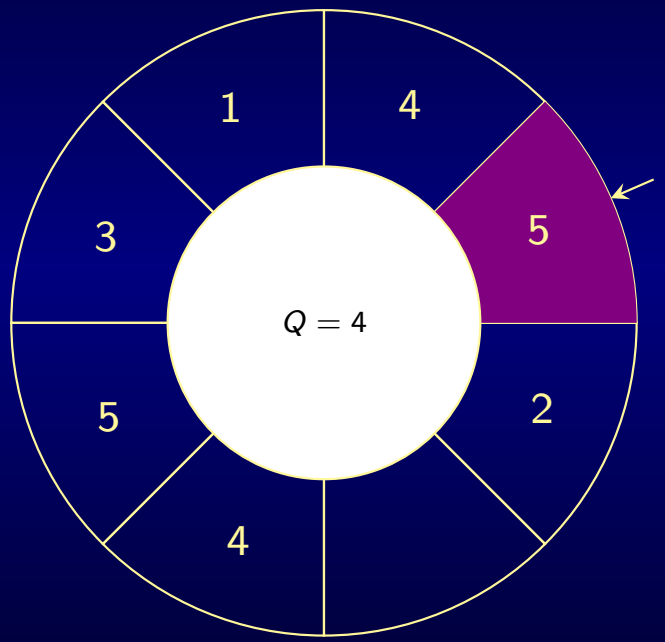
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



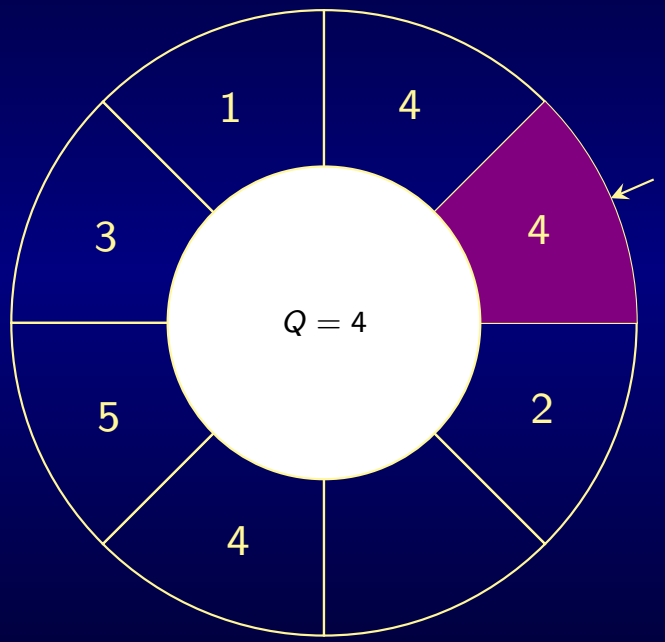
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



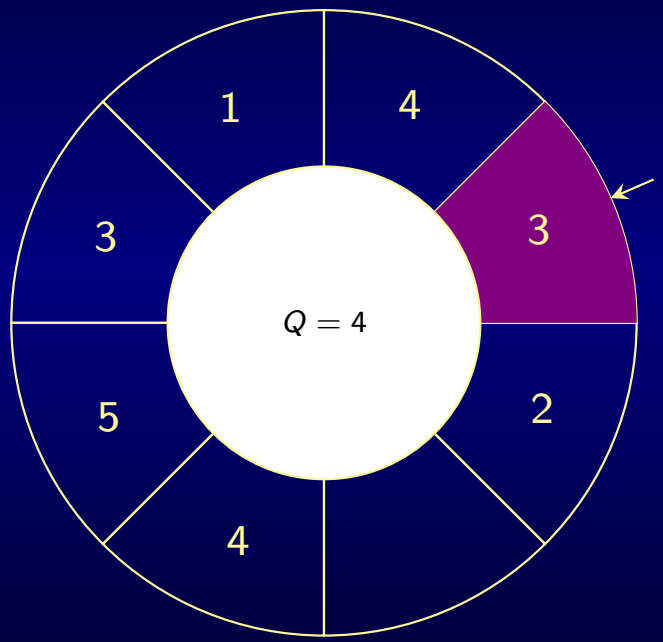
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



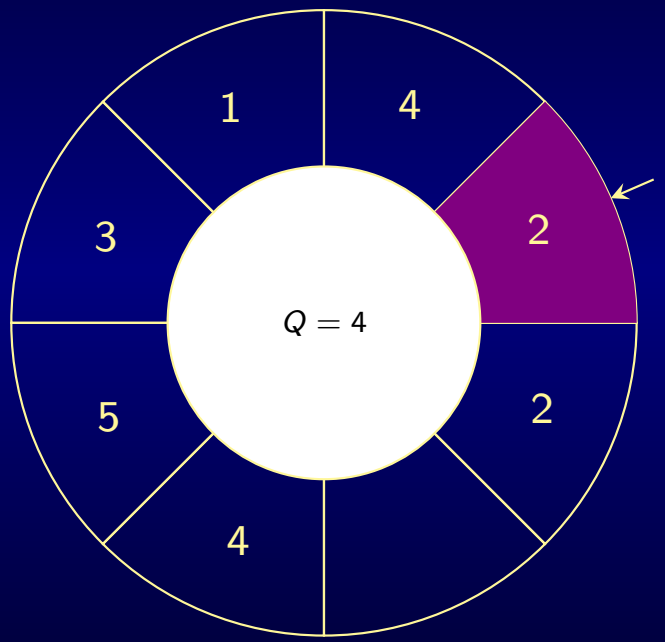
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



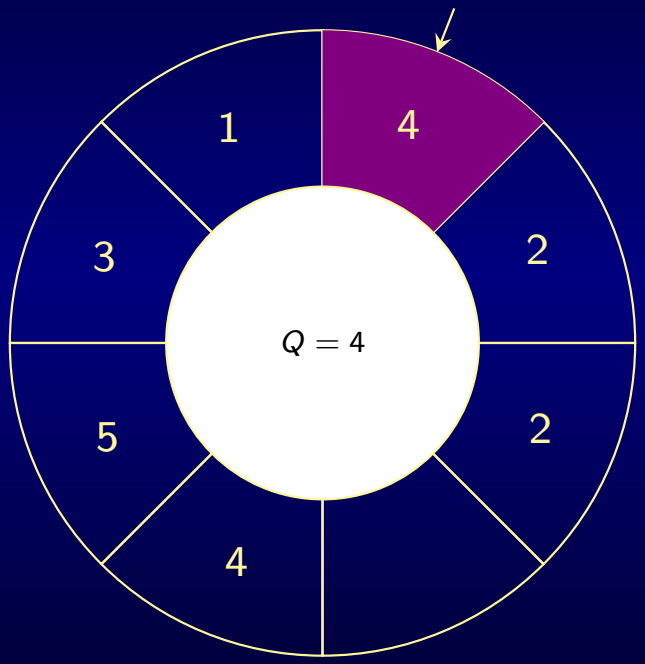
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



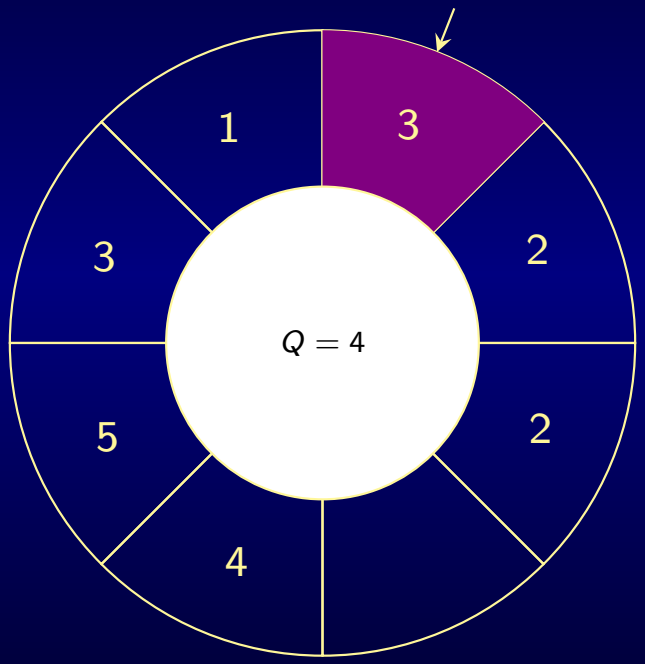
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



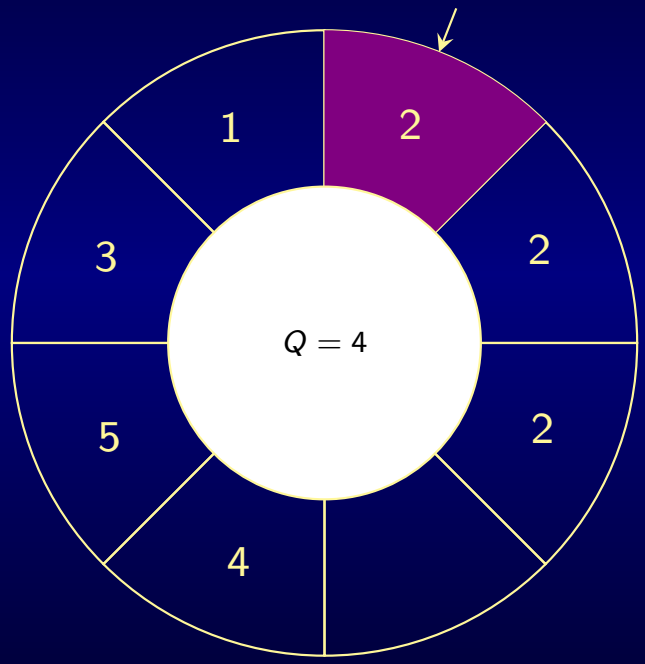
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



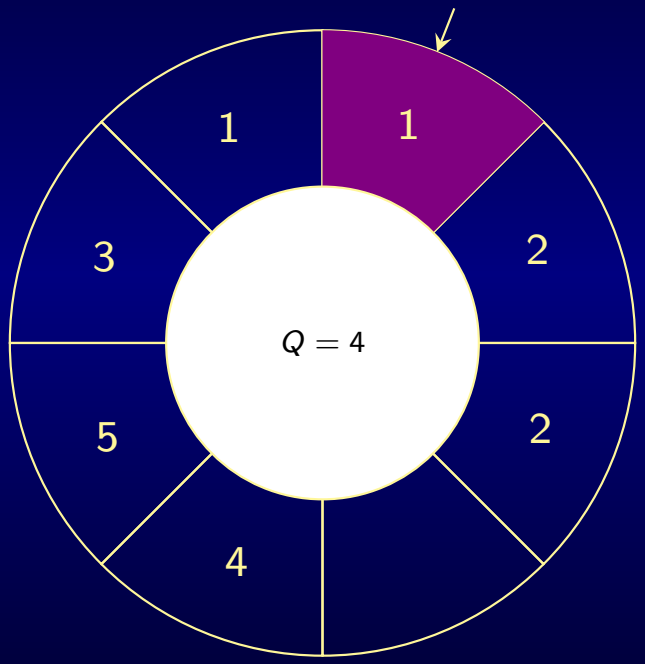
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



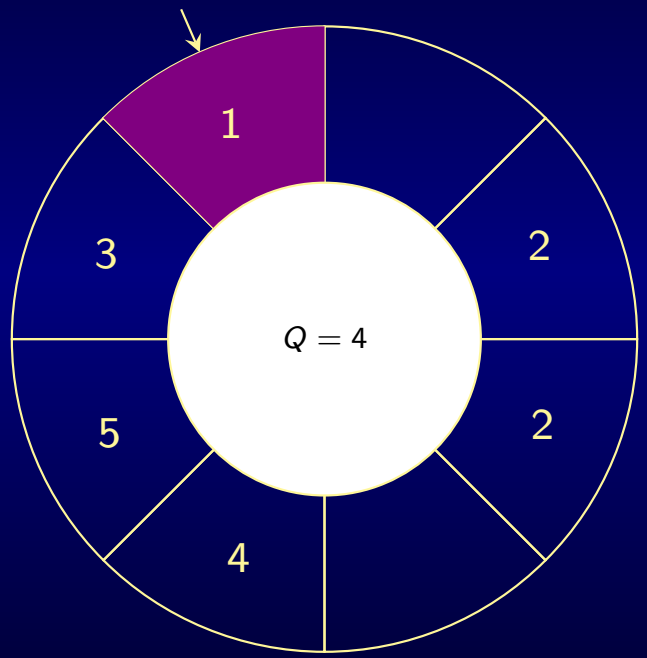
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



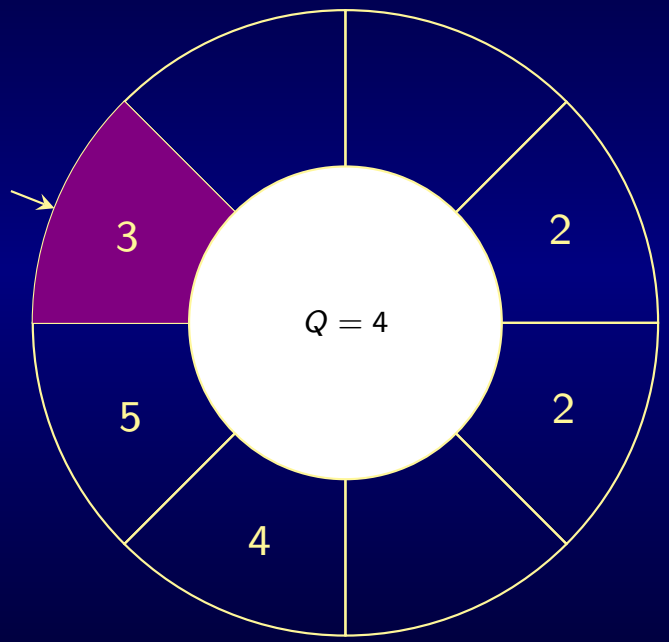
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



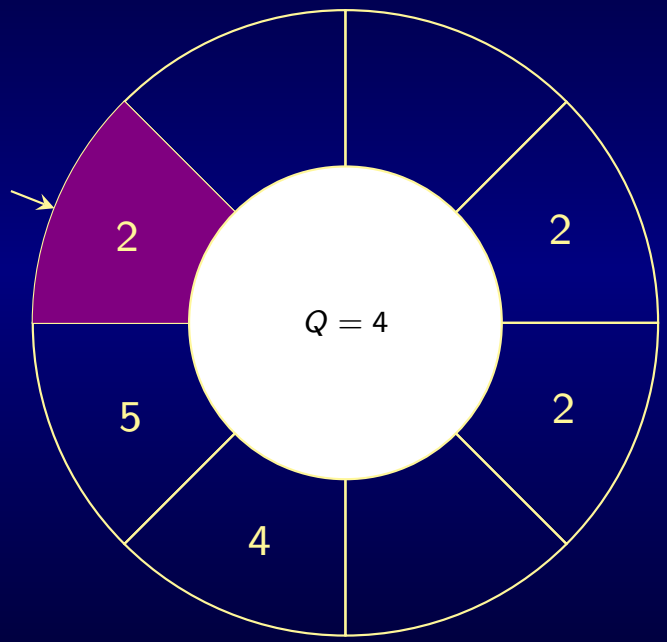
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



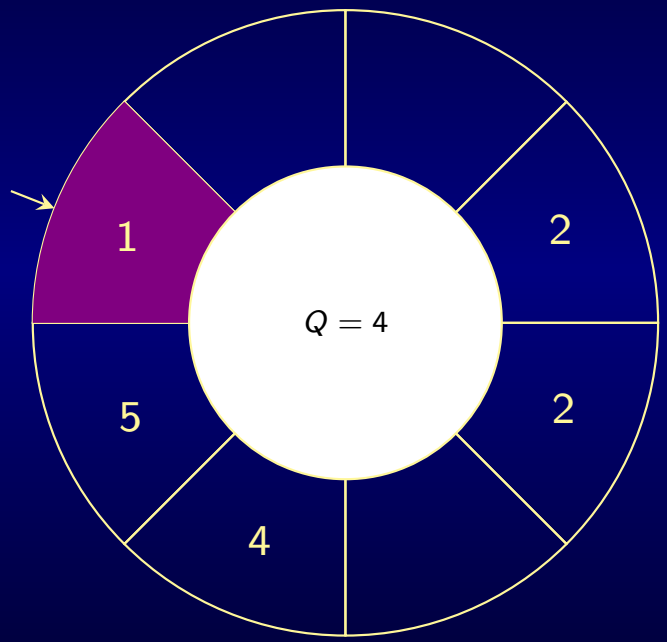
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



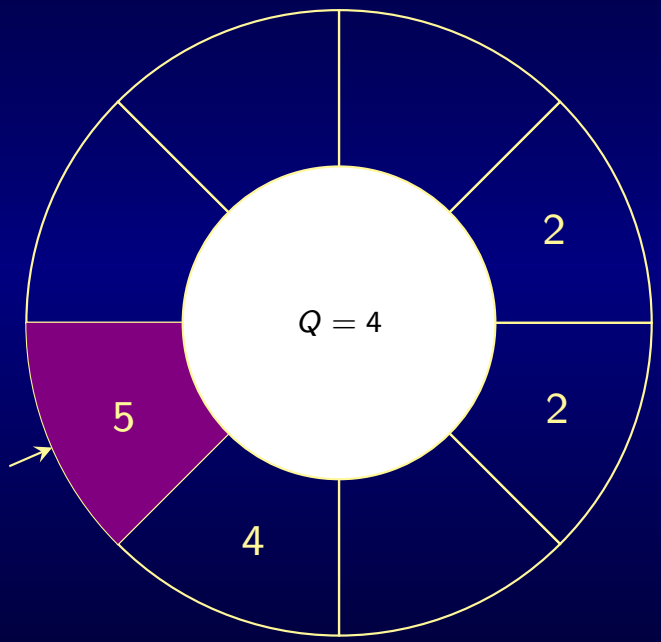
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



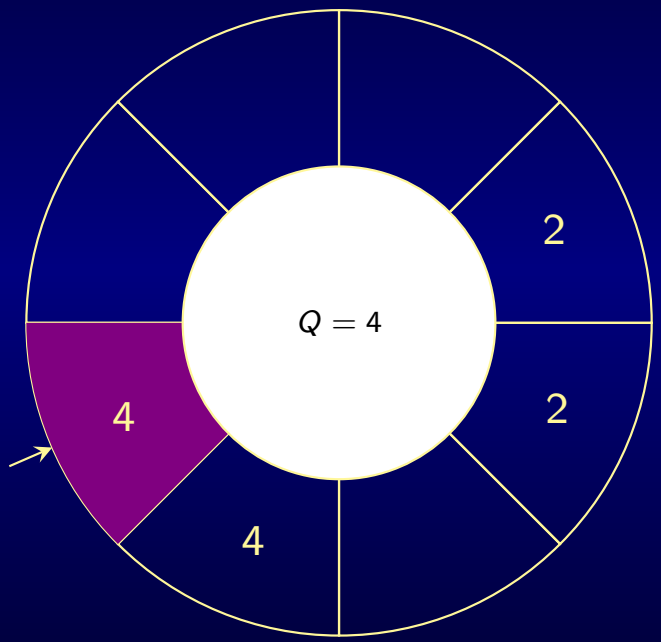
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



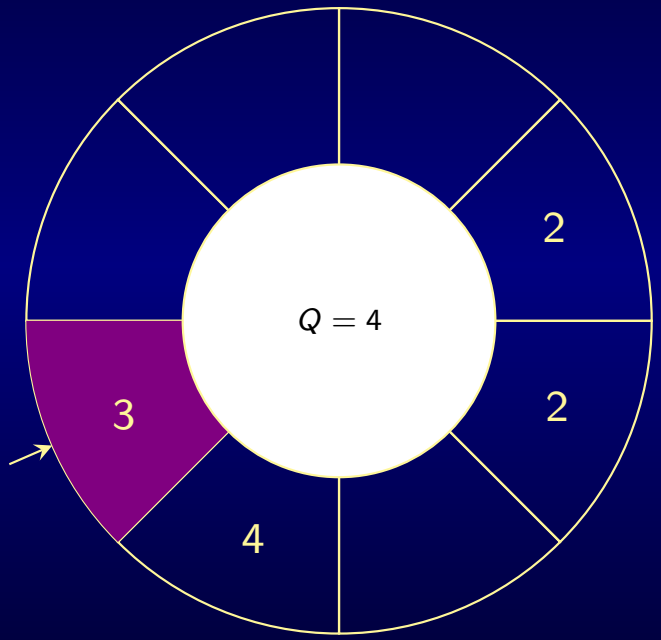
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



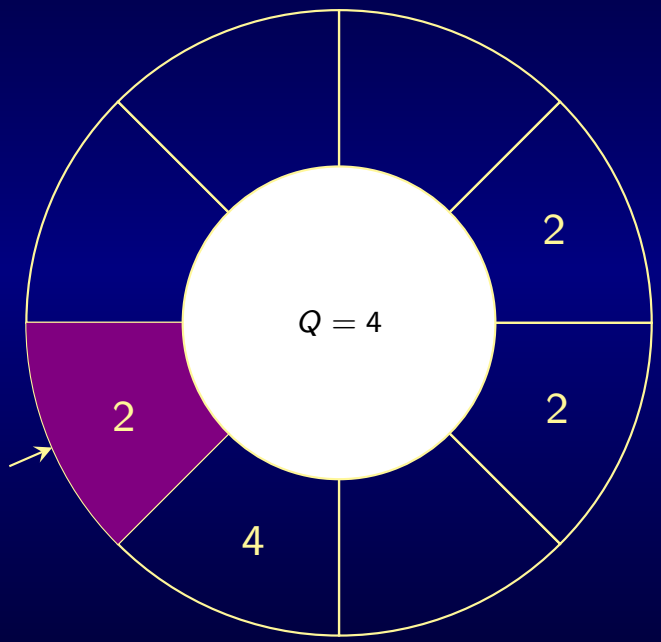
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



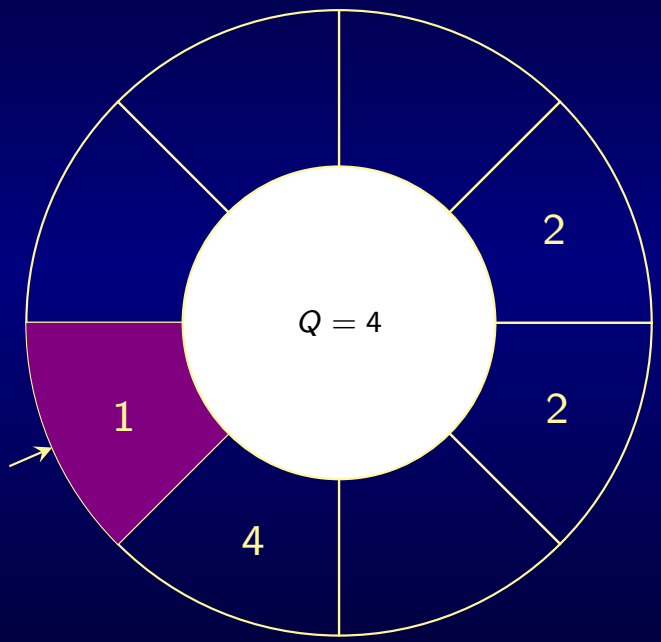
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



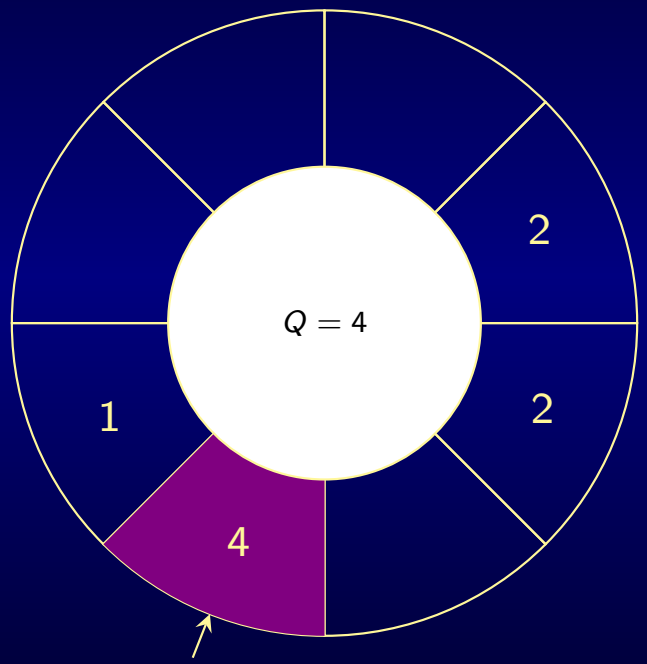
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



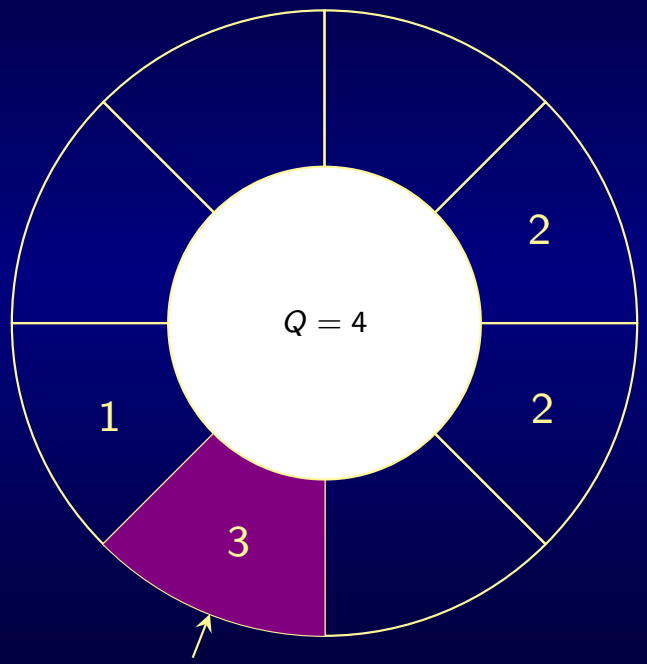
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



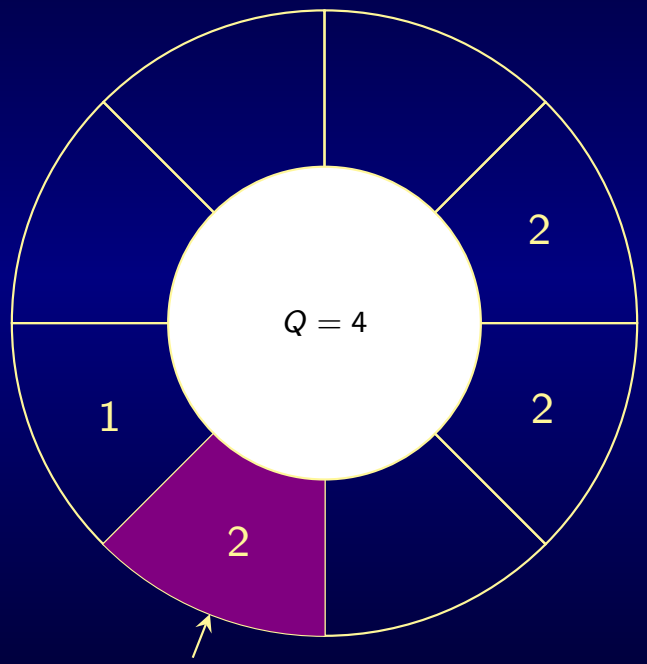
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



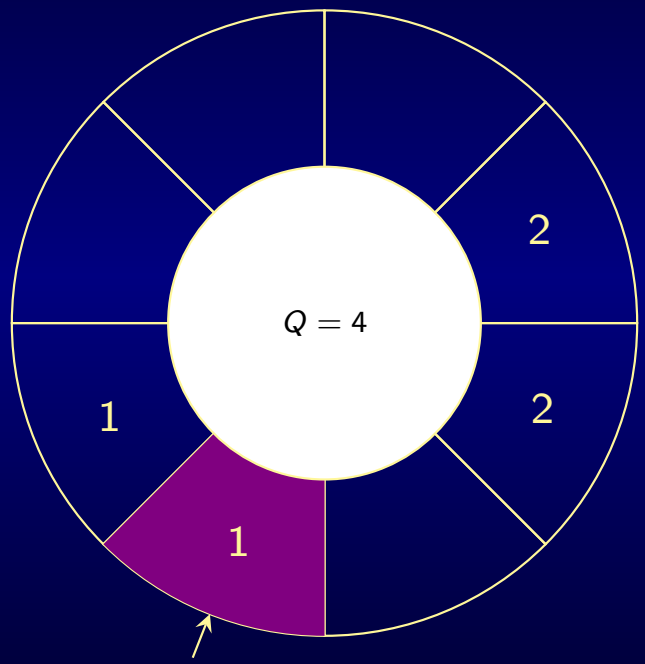
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



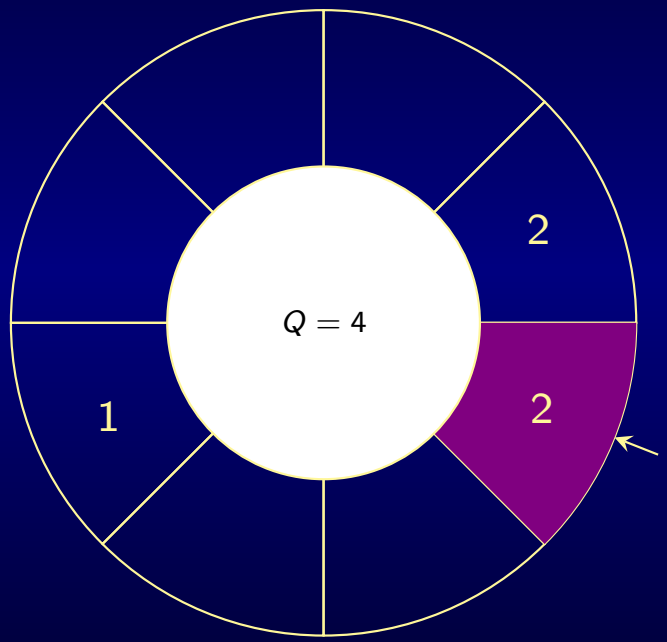
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



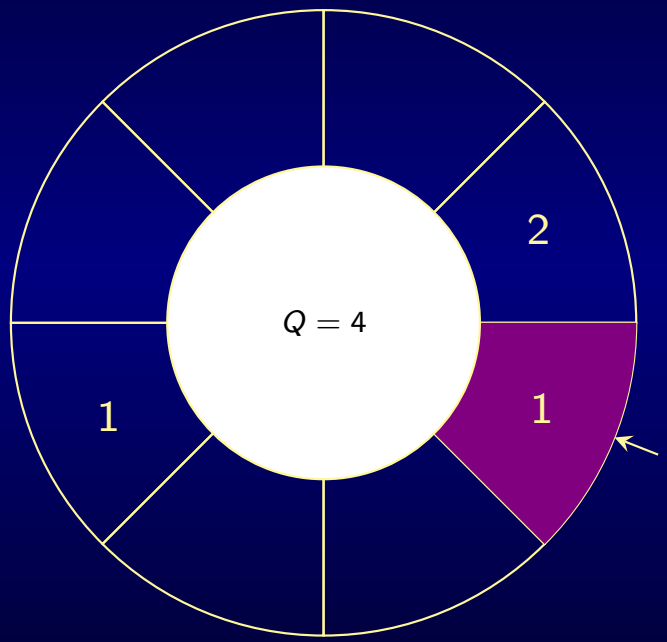
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



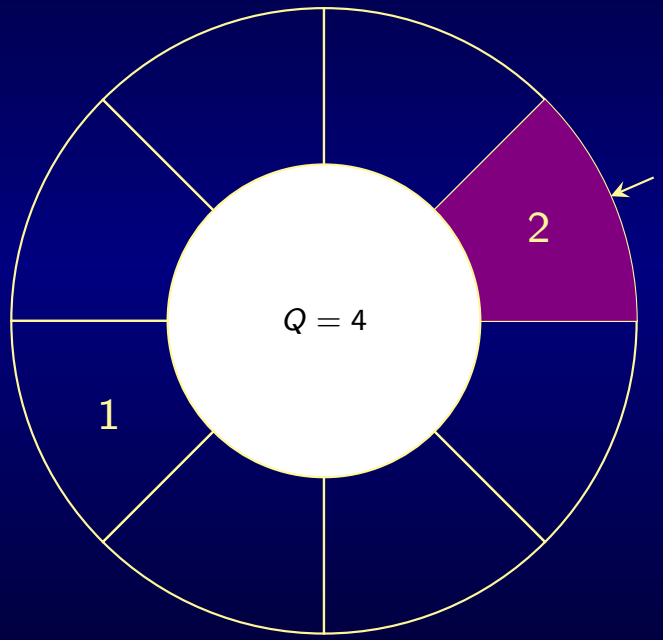
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



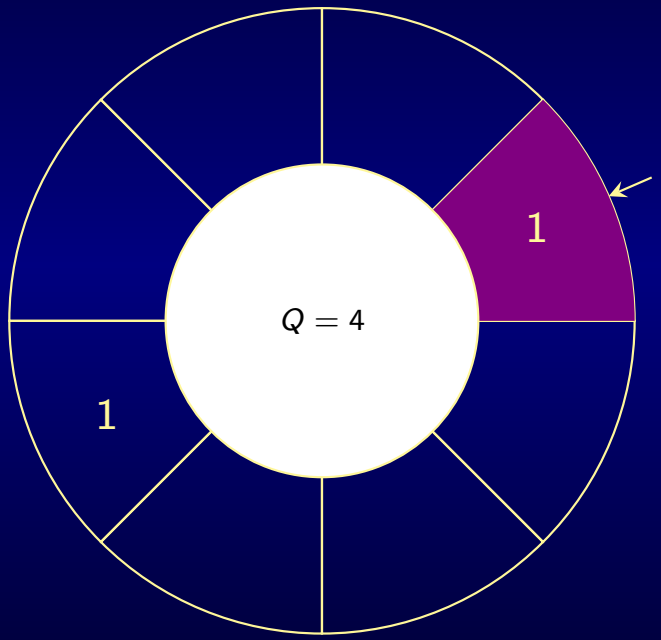
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



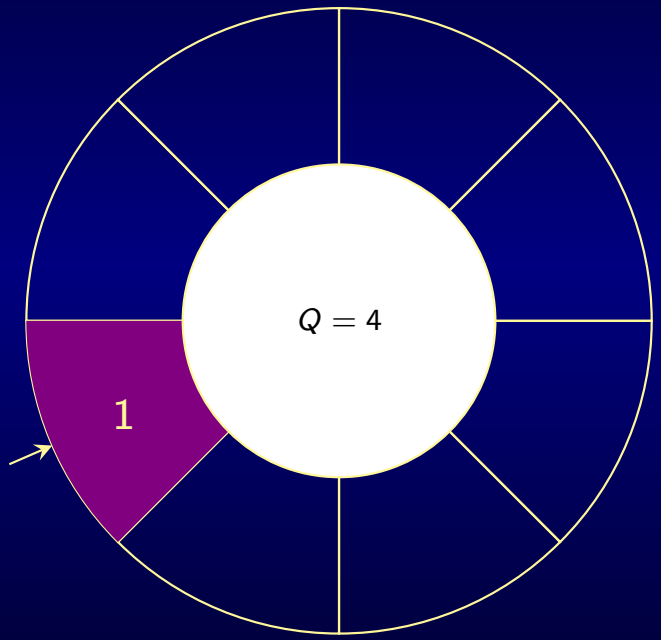
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



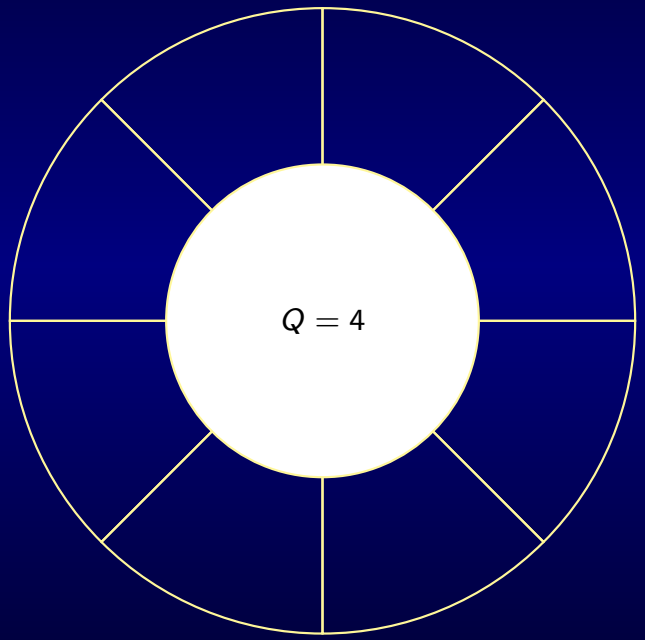
Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



Rysunek: Szeregowanie rotacyjne

Algorytm rotacyjny



Rysunek: Szeregowanie rotacyjne

Średni czas oczekiwania

Proces	Czas trwania fazy
P_1	24 ms
P_2	3 ms
P_3	3 ms

$$Q = 4ms$$

Licząc średni czas oczekiwania procesów dla algorytmu rotacyjnego należy uwzględnić czasy częściowych wykonań faz procesora poszczególnych procesów (może ich być kilka) i odjąć je od czasów ich oczekiwań. Obliczenia przeprowadzamy w następujący sposób: $((26 - 5 \cdot 4) + 4 + 7)/3 = 17/3 \approx 5,67ms$.

Kolejki wielopoziomowe

Działanie opisanych wcześniej algorytmów można połączyć dzieląc kolejkę procesów gotowych na kilka. Procesy umieszczane są w tych kolejkach w zależności do jakiej kategorii należą lub jaki mają priorytet. Szeregowanie w obrębie poszczególny kolejek odbywa się różnymi algorytmami lub też tym samym algorytmem, ale z różnymi parametrami dla tego algorytmu (np. algorytm rotacyjny z różnym kwantem czasu).

Kolejki wielopoziomowe ze sprzężeniami zwrotnymi

Opisany wcześniej schemat można uzupełnić o migrację procesów między kolejkami. Otrzymujemy w ten sposób wielopoziomową kolejkę ze sprzężeniami zwrotnymi (ang. Mulitlevel Feedback Queue). Autorem tego rozwiązania jest Leonard Kleinrock. Przechodzenie procesów między kolejkami jest zależne od czasu trwania ich poprzedniej fazy procesora. Na tym schemacie opiera się szeregowanie procesów w oryginalnym systemie Unix.

Szeregowanie w Windows 2000/XP

W systemach rodziny Windows[1] szeregowaniu podlegają nie procesy lecz wątki. W Windows 2000 i XP zastosowano algorytm bazujący na wielopoziomowych kolejkach ze sprzężeniami zwrotnymi. Istnieją trzy kategorie priorytetów, najniższy oznaczany przez 0, jest priorytetem systemowym i jest przeznaczony dla wątku bezczynności. Kolejne priorytety, z zakresu $1 - 15$ są priorytetami, które ulegają dynamicznym zmianom. Ostatnia kategoria to priorytety wątków (tolerancyjnego) czasu rzeczywistego , które są statycznie i odpowiadają im wartości z zakresu $16 - 31$. W przypadku wątków o dynamicznym priorytecie, każdy z nich dziedziczy po procesie, który w systemach Windows jest traktowany wyłącznie jako kontener na wątki, *priorytet bazowy*. W zależności od historii wykonania wątku planista dodaje do tego priorytetu odpowiedni modyfikator. Modyfikatory mają zakres wartości (-7) – (-3) i $(+3) - (+7)$. Na podstawie tak obliczonego priorytetu wątki są przypisywane do odpowiednich kolejek. Jeśli w wyniku dodawania wyjdzie wartość mniejsza od 1 lub większa niż 15, to wynik jest odpowiednio zaokrąglany. Szeregowanie w systemach Windows NT opiera się na podobnym schemacie.

[1] Materiał bazuje na informacjach ze strony http://wazniak.mimuw.edu.pl/

Szeregowanie w systemach wieloprocesorowych

Jeśli w systemie rozproszonym każdy procesor jest innego typu, to planowanie jest stosunkowo proste - każdy procesor otrzymuje sobie właściwe zadania. Jeśli jednak procesory są jednakowe, to planowanie może przebiegać na kilka sposobów. Każdy z procesorów może mieć osobną kolejkę zadań, a system operacyjny powinien dbać o to, aby liczby elementów tych kolejek były takie same lub porównywalne. Procesory mogą też mieć wspólną kolejkę zadań. W takim rozwiązaniu wyznacza się najczęściej jeden z procesorów do wykonywania kodu planisty. Jest to forma wieloprzetwarzania asymetrycznego.

Ocena algorytmów

Zanim algorytm szeregowania zostanie zaimplementowany w pracującym systemie operacyjnym, to należy (a przynajmniej warto) sprawdzić jego skuteczność. Wstępnie można dokonać *oceny analitycznej* algorytmu zakładając pewne statyczne, uśrednione obciążenie procesora. To oszacowanie wykonywane jest za pomocą diagramów Gantta. Mamy wtedy do czynienia z *modelowaniem deterministycznym*. To oszacowanie można poprawić stosując metodę *modeli kolejkowych*. Głównym elementem tej metody jest wzór Little'a $n = \lambda \cdot W$, gdzie λ to tempo przybywania nowych procesów do systemu, W to średni czas oczekiwania w kolejce, a n to średnia długość kolejki. Prostszą, ale równie wiarygodną metodą oceny algorytmów szeregowania jest symulowanie ich działania w komputerze. Dane dla symulatora na temat procesów może dostarczać generator liczb pseudolosowych o rozkładzie wykładniczym, lub można je pobrać z rzeczywistego systemu. Ostateczną oceną jest jednak zawsze implementacja algorytmu w prawdziwym systemie operacyjnym. W takim przypadku programista systemowy może zmieniać parametry algorytmu lub sam algorytm w oparciu o opinie użytkowników. Może się również okazać, że to użytkownicy (programiści) przystosują swoje aplikacje do nowego szeregowania.

Pytania

?

Koniec

Dziękuję Państwu za uwagę!