WYŻSZA SZKOŁA INFORMATYKI STOSOWANEJ I ZARZĄDZANIA
WYDZIAŁ INFORMATYKI

WIELODOSTĘPNE SYSTEMY OPERACYJNE II

Zagadnienia zaawansowane

CZĘŚĆ 2

BLOKADY – ZAKLESZCZENIA PROCESÓW

Semestr 4

Lech Kruś,

BLOKADY - ZAKLESZCZENIA (DEADLOCKS)

Blokada: sytuacja, w której procesy wzajemnie przetrzymują zasoby, do których chciałyby uzyskać dostęp. W sytuacji takiej procesy są w stanie oczekiwania, z którego nie mogą wyjść.

OPIS SYSTEMU

Zasoby

System zawiera skończoną liczbę zasobów różnego typu.
Mogą występować grupy równoważnych zasobów.

Zasady korzystania z zasobów przez procesy

Zamówienie zasobu
Użycie
Zwolnienie zasobu

Def. Wzajemnego blokowania, zakleszczenia procesów (deadlock):

Zbiór procesów jest w stanie blokady, jeśli każdy proces z tego zbioru czeka na zdarzenie spowodowane przez inny proces z tego samego zbioru.

Przykłady zasobów:
zasoby fizyczne: drukarki, napędy taśmy, cykle procesora, pamięć
zasoby logiczne: pliki, semafory

WARUNKI KONIECZNE WYSTĄPIENIA BLOKADY

Wzajemne wyłączanie

Co najmniej jeden zasób jest niepodzielny.
Tylko jeden proces może korzystać z tego zasobu, inne procesy zamawiające ten zasób są opóźniane.

Przetrzymywanie i oczekiwanie

Musi istnieć proces mający przydzielony pewien zasób (co najmniej jeden) i oczekujący na przydział dodatkowego zasobu, przetrzymywanego przez inny proces.

WARUNKI KONIECZNE WYSTĄPIENIA BLOKADY (c. d.)

Brak wywłaszczeń

Tylko proces przetrzymujący określony zasób, może ten zasób zwolnić.

Czekanie cykliczne

Musi istnieć zbiór oczekujących procesów $\{P_0, P_1, \ldots, P_{n-1}\}$, takich, że P_0 czeka na zasób przetrzymywany przez P_1, P_1 czeka na zasób przetrzymywany przez P_2, itd. . . . , aż P_{n-1} czeka na zasób przetrzymywany przez P_0 .

GRAF PRZYDZIAŁU ZASOBÓW

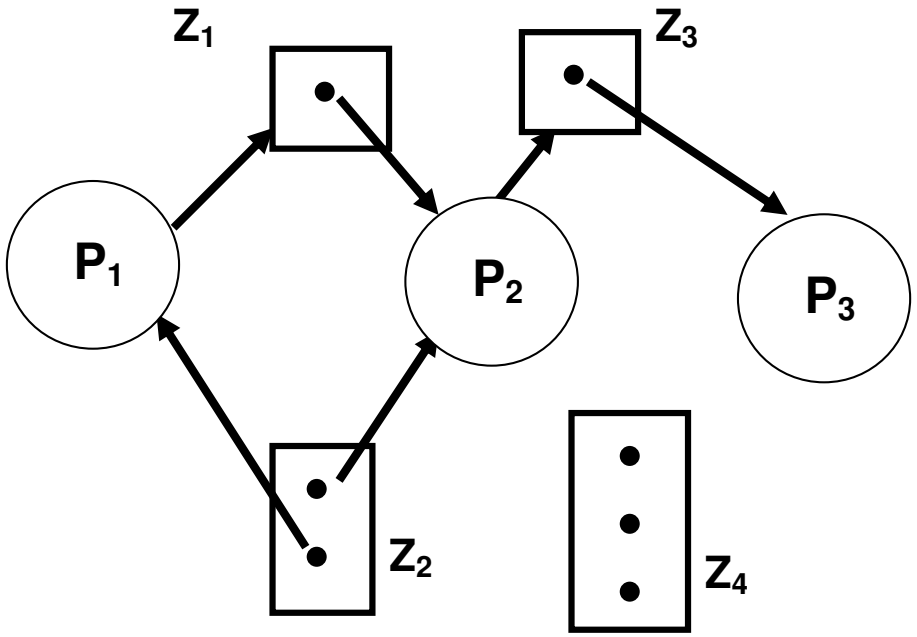
Graf skierowany opisujący stan systemu przydziału zasobów, umożliwia analizę sytuacji blokad.

Zbiór wierzchołków W składający się z podzbiorów:

$P=\{ P_1 , P_2 , \ldots , P_n\}$ podzbiór procesów,
$Z=\{ Z_1 , Z_2 , \ldots , Z_m\}$ podzbiór typów zasobów.

Zbiór krawędzi skierowanych K:

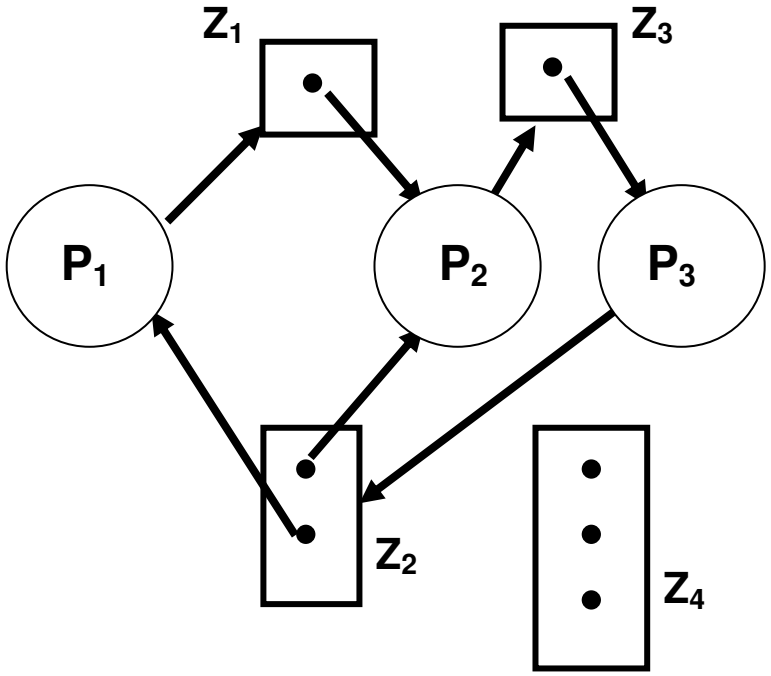
$P_i \rightarrow Z_j$ oznacza, że proces P_i zamówił zasób Z_j,
$Z_i \rightarrow P_j$ oznacza, że zasób Z_i został przydzielony procesowi P_j .



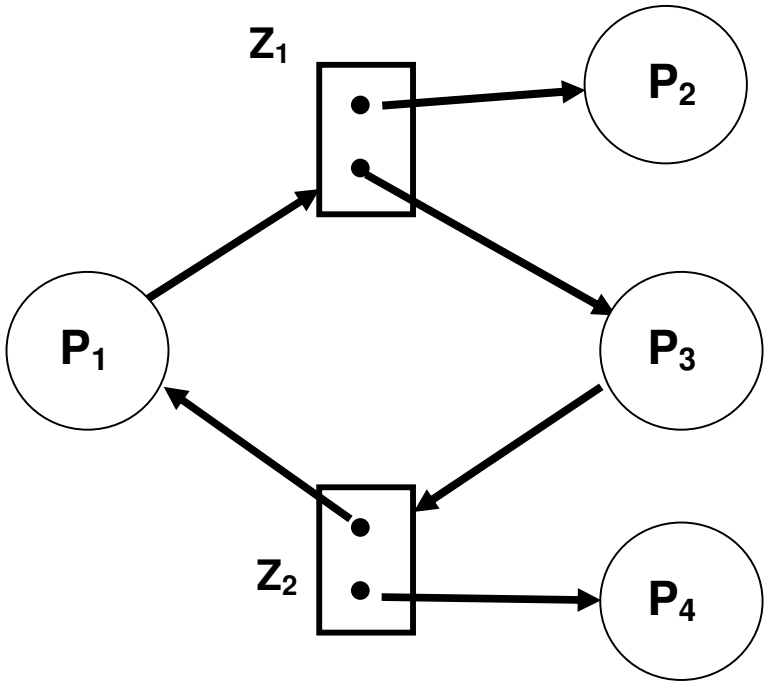
Przykład grafu (oznaczenia na rys. : proces - ❍ , zasób - ☐).

Warunek konieczny wystąpienia blokowania:
graf przydziału zasobów zawiera cykl.

W przypadku, gdy każdy typ zasobów zawiera tylko jeden egzemplarz, to cykl zawarty w grafie przydziału zasobów jest warunkiem koniecznym i dostatecznym.



Przykład grafu z blokadą



Przykład grafu z cyklem, ale bez blokady

SPOSOBY ROZWIĄZYWANIA PROBLEMU BLOKADY

- Zapobieganie wystąpieniu blokady.
- Unikanie blokad.
- Pozwolić na wejście w stan blokady i spowodować jej usunięcie.

ZAPOBIEGANIE WYSTĄPIENIU BLOKADY

Wyeliminowanie co najmniej jednego z warunków koniecznych.

Wzajemne wyłączanie

Warunek ten można wyeliminować tylko w przypadku zasobów podzielnych.

Przetrzymywanie i oczekiwanie

idea: zastosować protokół zapewniający, że proces zamawiający określony zasób nie powinien przetrzymywać innych.

Brak wywłaszczeń

idea: zastosować odpowiedni protokół wywłaszczeniowy

Czekanie cykliczne

idea: wyeliminować czekanie cykliczne przez uporządkowanie zasobów i zastosowanie odpowiedniego protokołu przydzielania zasobów procesom.

$Z=\{Z_1, \ldots Z_m\}$ zbiór zasobów

F: Z -> N funkcja przyporządkowująca każdemu zasobowi liczbę naturalną ze zbioru **N**.

Przykład:
Z={nap. taśmy, nap. dysku, drukarki},
N={ 1, 5, 7},
F(nap. taśmy)=1, F(nap. dysku)=5, F(drukarki)=7,

Protokół: każdy proces może zamawiać zasoby tylko we wzrastającym porządku ich numeracji, tzn. na początku proces może zamówić dowolną dostępną liczbę egz. zasobu np. Z_l.

Następnie jednak każdy egz. zasobu Z_k tylko wtedy, gdy $F(Z_k) > F(Z_l)$.

Wymaga się, aby proces zamawiający zasób Z_l miał wcześniej zwolnione zasoby Z_k takie, że $F(Z_k) \geq F(Z_l)$.

UNIKANIE BLOKAD

idea: przy każdym zamawianiu zasobów przez proces, system operacyjny decyduje czy ten proces ma czekać czy nie. Wymagana jest wcześniejsza informacja jak procesy będą zamawiać i zwalniać zasoby.

Na podstawie deklaracji procesów o maksymalnej liczbie potrzebnych zasobów konstruuje się algorytmy przydzielania zasobów, tak aby system nie wszedł w stan blokady. Algorytm dynamicznie sprawdza stan przydziału zasobów i decyzja o przydziale podejmowana jest tak aby nie dopuścić do spełnienia warunku czekania cyklicznego. Różne algorytmy wymagają różnych ilości i typów informacji.

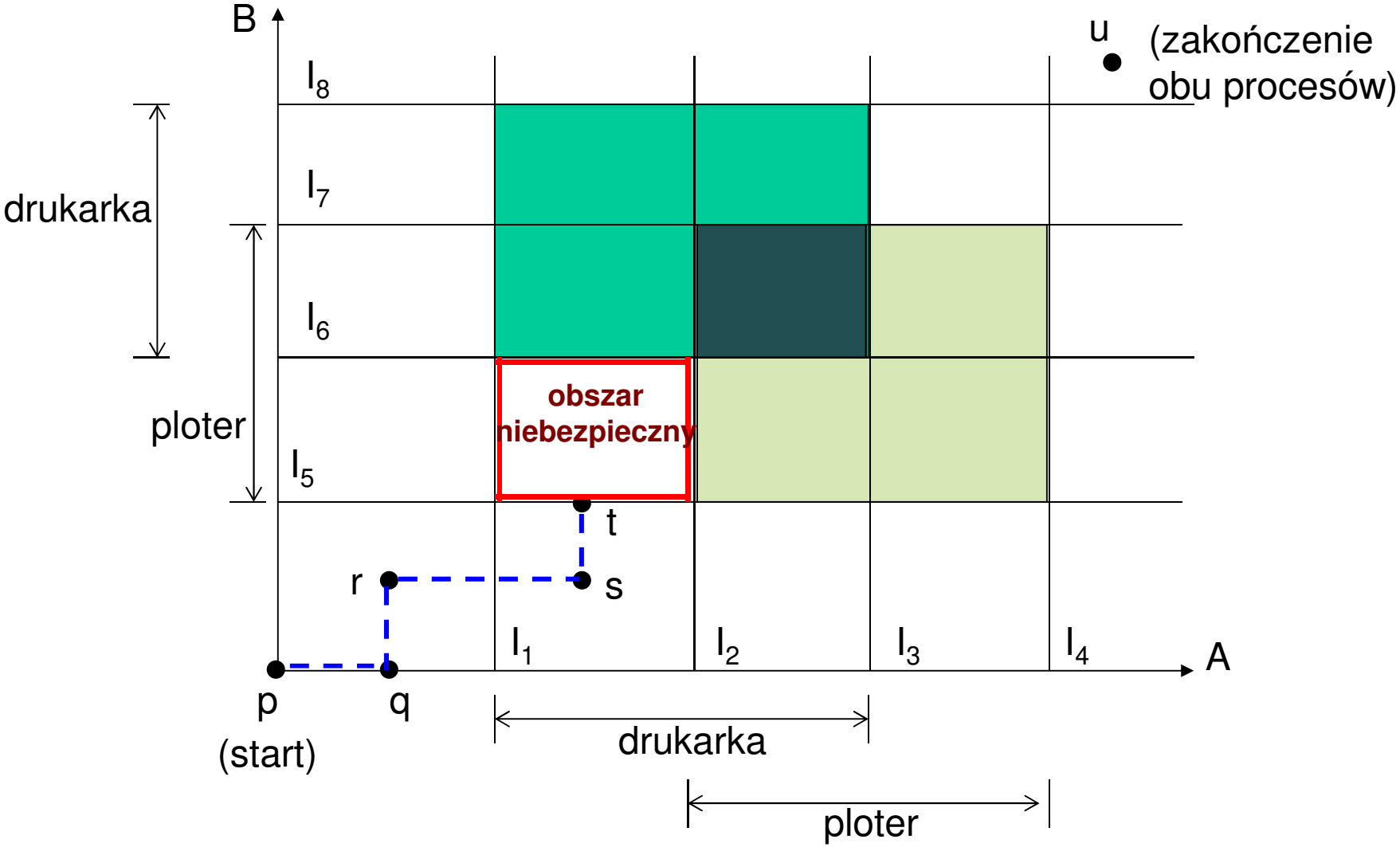
Stan przydziału zasobów określony jest przez liczbę zasobów dostępnych, przydzielonych oraz przez maksymalne zapotrzebowania procesów.

Stan bezpieczny:
Stan przydziałów dla którego istnieje **ciąg bezpieczny procesów P_1 , . . ., P_n ,**
tzn. taki ciąg, że dla każdego procesu P_i jego potencjalne zapotrzebowanie na zasoby musi być zaspokojone przez zasoby aktualnie dostępne oraz przez zasoby użytkowane przez procesy P_j, **j<i.**

Stan zagrożenia - gdy żaden taki ciąg bezpieczny nie istnieje.

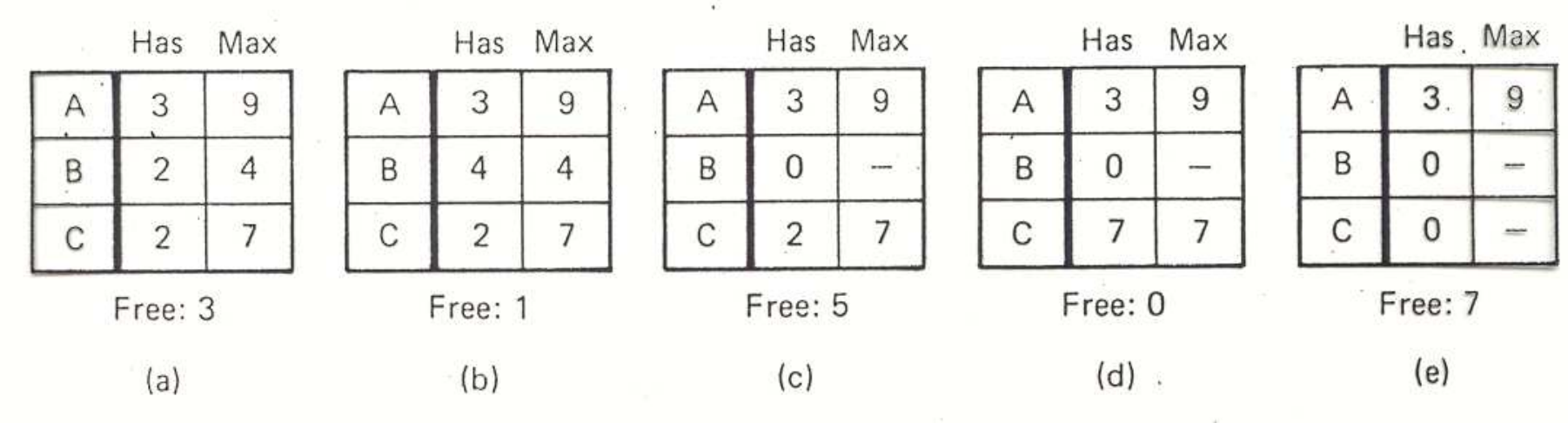
Ilustracja stanu bezpiecznego i stanu niebezpiecznego.

Przykład 1: trajektoria realizacji dwóch procesów: A i B wykorzystujących dwa zasoby

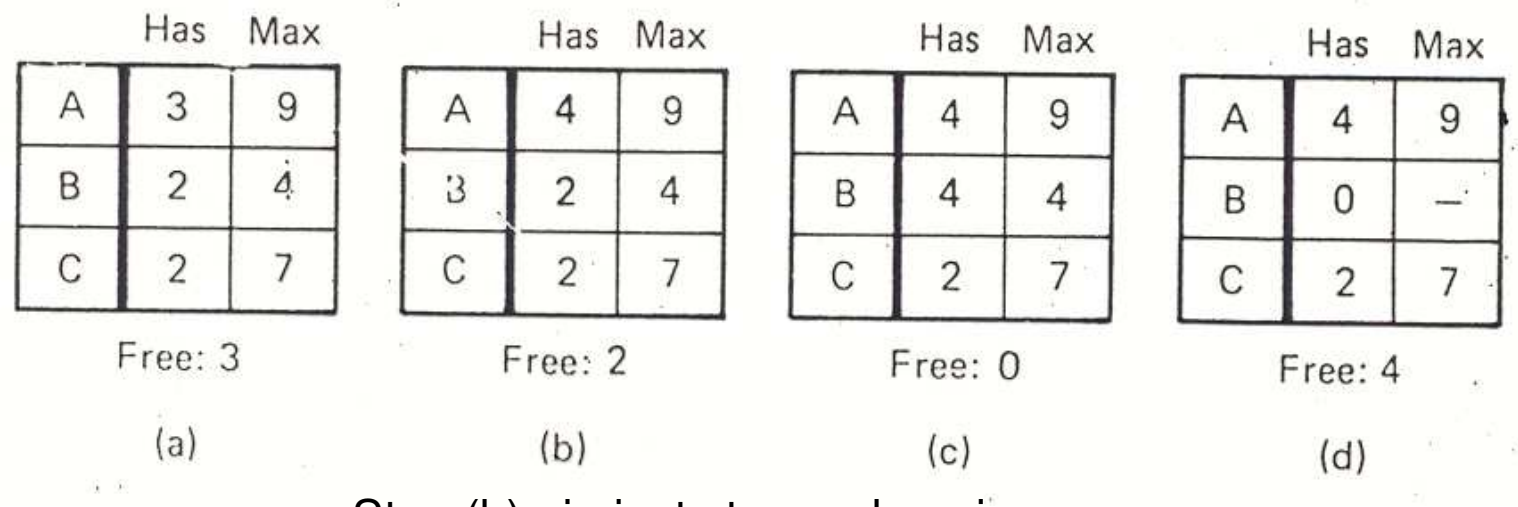


Ilustracja stanu bezpiecznego i stanu zagrożenia. Przykład 2.

Jeden zasób dostępny w 10-u egzemplarzach, 3 procesy: A, B, C.



Stan (a) jest stanem bezpiecznym



Stan (b) nie jest stanem bezpiecznym

Rozwiązanie dla zasobów reprezentowanych wielokrotnie: algorytm „bankiera”

Proces wchodzący do systemu musi zadeklarować maksymalną liczbę potrzebnych egzemplarzy każdego zasobu.

Kiedy proces w trakcie wykonywania zamawia potrzebne zasoby - system operacyjny decyduje czy te zasoby udostępnić, czy proces musi poczekać - tak aby system pozostał w stanie bezpiecznym

Struktury danych przechowujące stan systemu przydziałów.

dla n procesów, m typów zasobów

Dostępne: array[0..m-1] of integer

określa liczbę dostępnych egz. każdego zasobu, np.
Dostępne[j]=k ozn., że jest dostępnych **k** egzemplarzy zasobu typu **j.**

Maksymalne: array[0..n-1,0..m-1] of integer

określa maksymalne żądania procesów względem zasobów poszczególnych typów.

Przydzielone: array[0..n-1,0..m-1] of integer

określa liczbę zasobów poszczególnych typów już przydzielonych do każdego z procesów.

Potrzebne: array[0..n-1,0..m-1] of integer

określa pozostałe do spełnienia zamówienia każdego procesów.

Uwagi:

Potrzebne[i,j]=Maksymalne[i,j] - Przydzielone[i,j]

Struktury te są dynamiczne - zmieniają w czasie wymiary i wartości.

W formułowaniu algorytmów korzystamy dla uproszczenia zapisu z wektorów: **Dostępne, $\mathbf{Maksymalne_i}$, $\mathbf{Przydzielone_i}$, $\mathbf{Potrzebne_i}$** (wektory względem zasobów dla ustalonego procesu i).

Wprowadzamy relację dominacji między wektorami X, Y o wymiarach m:

$X \leq Y$ wtedy i tylko wtedy gdy $X[j] \leq Y[j]$ dla każdego j=1,..,m.

Zamówienia procesu **i** na zasoby oznaczane są przez:
$Zamówienia_i$: array[0..m-1] of integer

ALGORYTM DZIAŁAŃ

podejmowanych, gdy proces **i** wykonuje zamówienia

Krok 1: Sprawdź czy **$Zamówienia_i \leq Potrzebne_i$**
tak: wykonaj **krok 2,**
nie: warunek błędu**.**

Krok 2: Sprawdź czy **$Zamówienia_i \leq Dostępne$**
tak: wykonaj **krok 3,**
nie: proces **i** musi czekać**.**

Krok 3: Wykonaj próbę przydziału zasobów**:**
$Dostępne := Dostępne - Zamówienia_i$;
$Przydzielone_i := Przydzielone_i + Zamówienia_i$;
$Potrzebne_i := Potrzebne_i - Zamówienia_i$;

Sprawdź czy **stan jest bezpieczny** (algorytm bezpieczeństwa).
tak: przydziel zasoby procesowi **i** zgodnie z zamówieniem**,**
nie: przywróć poprzedni stan zasobów, proces **i** musi czekać na realizację zamówienia: **$Zamówienia_i$.**

ALGORYTM BEZPIECZEŃSTWA

Wprowadzamy wektory **Praca** o wymiarze m, **Koniec** o wymiarze n .

Krok 1: Przypisania początkowe:

Praca:= Dostępne;

Koniec[i]:= false; dla i=0,...,n-1.

Krok 2: Znajdź takie **i ,** że jednocześnie**:**

Koniec[i]=false and

$Potrzebne_i \leq Praca$

czy istnieje takie **i?**

tak: wykonaj **krok 3**

nie: wykonaj **krok 4.**

Krok 3: Praca := Praca + $Przydzielone_i$;

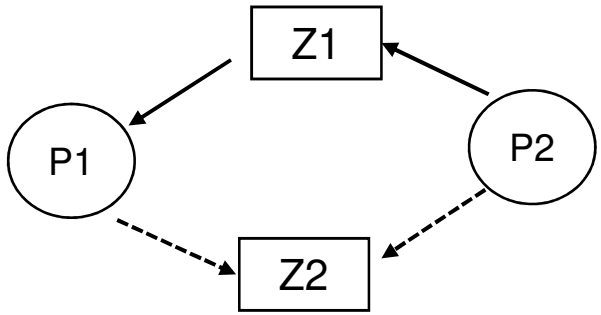
Koniec[i]:= true;

wykonaj krok 2.

Krok 4: Jeśli **Koniec[i] = true** dla wszystkich **i** to system jest w stanie bezpiecznym.

UNIKANIE ZAKLESZCZEŃ W PRZYPADKU ZASOBÓW POJEDYNCZYCH

Algorytm bankiera jest ogólny, jest jednak pracochłonny.
Wymaga do mxn^2 operacji, gdzie m – liczba zasobów, n – liczba procesów.
W przypadku zasobów pojedynczych można zdefiniować algorytm efektywniejszy.
Stosuje się wariant grafu przydziału zasobów, w którym są krawędzie zamówień i przydziałów, a dodatkowo krawędzie deklaracji.
Krawędź deklaracji oznacza, że proces zamówi określony zasób w przyszłości.
Przykład:



Zanim proces zacznie działać, wszystkie deklaracje muszą być znane.
Gdy proces zamawia deklarowany zasób, krawędź deklaracji zastępowana jest krawędzią zamówienia.

Idea algorytmu

Niech proces P_i zamawia zasób Z_j. Zamówienie jest realizowane tylko wtedy, gdy zamiana krawędzi zamówienia na krawędź przydziału, nie spowoduje wystąpienia cyklu.

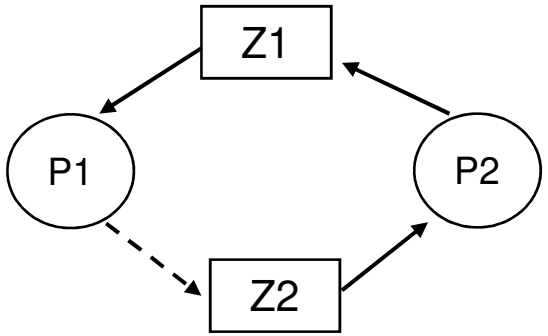
Sprawdzenie bezpieczeństwa polega na wykryciu cyklu w grafie - są odpowiednie algorytmy.

Jeśli nie ma cyklu, to realizacja zamówienia pozostawi system w stanie bezpiecznym.

Jeśli jest cykl, to realizacja zamówienia wprowadzi system w stan niebezpieczny. Proces P_i musi poczekać.

Ilustracja:

P1 i P2 zadeklarowały wcześniej zamówienie zasobu Z2. Gdy P2 go zamówił następuje zamiana deklaracji zamówienia na krawędź przydziału. Jest cykl w grafie. Nie należy realizować tego zamówienia.



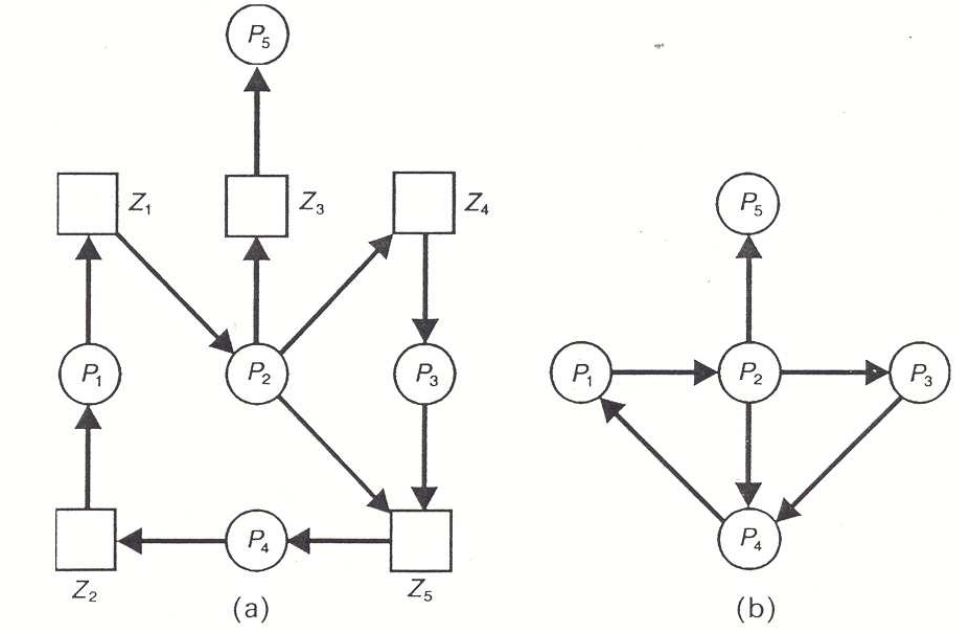
WYKRYWANIE I WYCHODZENIE Z BLOKADY

WYKRYWANIE BLOKAD

Algorytmy wykrywania blokad dla zasobów reprezentowanych wielokrotnie i jednokrotnie.
Przykład algorytmu dla pierwszej klasy - patrz Silberschatz i inni, Podstawy systemów operacyjnych.WNT, W-wa, 1993.

Zasoby reprezentowane pojedynczo

idea algorytmu:
Budowa grafu przydziałów
Budowa grafu oczekiwań (przez usunięcie węzłów zasobów)
Krawędź P_i -> P_j oznacza, że proces P_i czeka na zwolnienie zasobów przez proces P_j.



Wykrywanie blokady polega na wykrywaniu pętli w grafie oczekiwań.

Niezbędne jest przechowywanie i aktualizowanie „na bieżąco” grafu oczekiwań oraz wykonywanie algorytmu szukającego pętli w grafie.

Problem: kiedy i jak często wykrywać blokady?

WYCHODZENIE Z BLOKADY

Usunięcie jednego (lub kilku) procesów w celu przerwania cyklicznego czekania

usunąć wszystkie procesy w blokadzie
znaczny koszt

usuwać procesy pojedynczo, aż do usunięcia blokady
Sposób działania: usunąć proces,
wykonać algorytm wykrywania blokady.
Problem wyboru procesu do usunięcia

Wywłaszczenie procesów z zasobów
Wybór wywłaszczanego procesu
Wznawianie wycofywanego procesu
Głodzenie procesu

ROZWIĄZYWANIE PROBLEMU BLOKAD W PRAKTYCE

Zastosowanie różnych metod dla różnych klas zasobów.

Zasoby wewnętrzne systemu (np. bloki kontrolne procesów)
Zapobieganie powstawaniu blokad przez uporządkowanie zasobów (nie trzeba dokonywać wyborów między realizowanymi zamówieniami.

Pamięć główna
Wywłaszczanie.

Zasoby zadania (przydzielane urządzenia, pliki, ..)
Unikanie blokad.

Obszar wymiany
Zastosowanie wstępnego przydziału.

Pytania podstawowe z zakresu blokad - zakleszczeń (deadlock) procesów przykłady

1. Co to jest blokada – zakleszczenie (deadlock) procesów?
2. Jakie są warunki konieczne wystąpienia blokady - zakleszczenia?
3. Co oznacza warunek czekania cyklicznego?
4. Na czym polega warunek przetrzymywania i oczekiwania?
5. Na czym polegają metody zapobiegania blokadom-zakleszczeniom?
6. W jaki sposób można wyeliminować warunek przetrzymywania i oczekiwania?
7. W jaki sposób można wyeliminować warunek braku wywłaszczeń?
8. W jaki sposób można wyeliminować warunek czekania cyklicznego?
9. Na czym polegają metody unikania blokad?
10. Jakie informacje są niezbędne do opisania stanu systemu przydziału zasobów?
11. Co to jest stan bezpieczny?
12. Co to jest stan zagrożenia?
13. Podać ideę algorytmu bankiera.
14. Na czym polegają metody wykrywania i wychodzenia z blokady?
15. W jaki sposób można zidentyfikować stan blokady?
16. W jaki sposób można wyjść z istniejącej blokady i jakie wiążą się z tym koszty?