Tobiasz Kownacki 331391

# Algorytmy zastępowania stron

W trakcie rozwoju komputerów problem programów o rozmiarach większych od pamięci stawał się coraz bardziej powszechny. Metodą, która rozwiązała ten problem, jest pamięć wirtualna. Dzięki niej każdy program może korzystać z własnej przestrzeni adresowej, która jest większa niż pamięć RAM. Wirtualna przestrzeń adresowa jest podzielona na strony o stałym rozmiarze. Odpowiadające im jednostki o takim samym rozmiarze w pamięci RAM są nazywane ramkami stron. Pamięć wirtualna umożliwia przechowywanie części danych w pamięci RAM, a reszty np. na dysku twardym.
Jak program odwołuje się do wirtualnego adresu, to specjalna jednostka zarządzania pamięcią sprawdza, do której strony należy podany adres. Następnie kontroluje czy dana wirtualna strona ma odwzorowanie w pamięci RAM (ramce strony). Jeżeli tak to adres wirtualny jest mapowany na adres fizyczny pamięci. W przeciwnym wypadku procesor wykona rozkaz pułapki do systemu operacyjnego. Ta pułapka nazywa się **brak strony (**page fault). System operacyjny za pomocą pewnego algorytmu wybiera ramkę strony, która była rzadko używana i zapisuje jej zawartość na dysk (jeśli była modyfikowana). Następnie w miejsce zwolnionej ramki strony załaduje ramkę z dysku, do której odwoływał się program. System operacyjny również zmieni odpowiednio mapowanie z wirtualnej strony na ramkę strony, a na sam koniec wznowi przerwaną operację.
Algorytmy, które wybierają ramkę strony do usunięcia, nazywamy algorytmami **zastępowania stron**. Celem tych algorytmów jest wybranie takich stron, które są najmniej używane z wszystkich znajdujących się w pamięci RAM, po to, aby jak najrzadziej przenosić dane między pamięcią RAM a dyskiem. Problem zastępowania stron występuje również w innych obszarach branży komputerowej takich jak na przykład pamięć cache.

## Optymalny algorytm zastępowania stron

Optymalny algorytm zastępowaniu stron polega na dodaniu etykiety do każdej ramki strony. W każdej etykiecie znajduje się informacja, za ile instrukcji będzie odwołanie do tej strony. Optymalny algorytm zastępowania stron usuwa stronę o najwyższej wartości etykiety. Jest to tak naprawdę odkładanie możliwości wystąpienia błędu braku strony, jak najdalej w przyszłość. Jego największą wadą jest to, że jest on niewykonalny. System operacyjny nie ma wiedzy o przyszłych odwołaniach do stron. Można jednak uruchomić program na symulatorze i prześledzić wszystkie odwołania do stron. Następnie przy drugim przebiegu zaimplementować optymalny algorytm przy użyciu informacji uzyskanych przy pierwszym przebiegu. Należy podkreślić, że algorytm zaimplementowany tą metodą dotyczy tylko jednego programu i jednego zestawu danych wejściowych. Nie ma on zastosowania w systemach operacyjnych, ale jest bardzo użyteczny przy porównywaniu wydajności z algorytmami wykonanymi.

## Algorytm NRU

Większość algorytmów zastępowania stron potrzebuje pewnych statystyk wykorzystywania poszczególnych ramek stron. Dlatego większość komputerów używających pamięci

wirtualnych wykorzystuje dwa bity R i M, które są powiązane do każdej strony. Bit R jest ustawiamy za każdym razem, jak występuje odwołanie do danej strony. Natomiast bit M jest ustawiany za każdym razem, kiedy występuje modyfikacja danej strony. Początkowo system operacyjny zeruje oba bity dla wszystkich stron. Bity trzymają swoją wartość tak długo, jak system operacyjny sam ich nie zmieni. Co jakiś czas bit R jest zerowany (np. przy każdym takcie zegara), po to, aby odróżnić strony, do których nie było odwołań, od tych, do których były.
Kiedy występuje brak strony algorytm NRU (Not Recently Used) przyporządkowuje każdą stronę do jednej z czterech kategorii, na podstawie bitów R i M:

- Klasa 0: strony, do których nie było odwołań, niemodyfikowane.
- Klasa 1: strony, do których nie było odwołań, modyfikowane.
- Klasa 2: strony, do których było odwołanie, niemodyfikowane.
- Klasa 3: strony, do których były odwołania, zmodyfikowane.

Algorytm NRU usuwa losową stronę z niepustej klasy o najniższym numerze. Głównymi zaletami algorytmu jest to, że jest on łatwy do zrozumienia i w miarę wydajny.

## Algorytm FIFO

Algorytm FIFO jak sama nazwa wskazuje, bazuje na strukturze danych zwana kolejką. Ten prosty algorytm o niskich kosztach obliczeniowych utrzymuje listę wszystkich stron będących w pamięci RAM. Jeśli występuje brak strony, algorytm FIFO usuwa stronę będącą na przodzie kolejki, a nową dodaje na jej koniec. W konsekwencji na przodzie kolejki znajduje się strona, która jest w pamięci RAM najdłużej. Istnieje jednak poważny problem z tym algorytmem: najstarsza strona może być częściej potrzebna w przyszłości niż strona, która jest na przykład w środku kolejki.

## Algorytm drugiej szansy

Jest to prosta modyfikacja algorytmu FIFO, która rozwiązuje problem usuwania często używanej ramki strony. Gdy występuje brak strony, algorytm patrzy na wartość bitu R strony, która jest na początku kolejki. Jeżeli wartość bitu R wynosi 1 to, ta strona jest przerzucana na koniec kolejki, a algorytm zmienia wartość bitu R na 0 i sprawdza kolejną stronę. Jak znaleziono stronę o wartości bitu R równej 0, to ta strona jest zapisywana na dysk (jeśli dane były modyfikowane), a następnie usuwana z pamięci RAM. Po tych czynnościach na koniec kolejki wskakuje nowa ramka strony. Nawet jeśli wszystkie strony mają bit R równy 1, to po przeniesieniu po kolei wszystkich stron z przodu kolejki na tył i wyzerowaniu bitów R, kolejka powróci do stanu początkowego i algorytm zachowa się jak klasyczny algorytm FIFO. Algorytm zawsze się zakończy, ale jego wadą jest to, że ciągłe przemieszczanie stron z przodu kolejki na jej tył jest wymagające obliczeniowo.

## Algorytm zegarowy

Aby uniknąć przemieszczania stron na liście, można trzymać wszystkie ramki stron na liście cyklicznej w formie zegara. Wskazówka wskazuje na stronę, która jest w pamięci RAM najdłużej. Algorytm ten działa tak samo, jak algorytm drugiej szansy. Różnicą jest to, że

strony stoją w miejscu, a wskazówka zegara przesuwa się na kolejne pozycje. Jeśli bit R jest równy 0, to na jej miejsce wchodzi nowa strona, a wskazówka się przesuwa. Natomiast jeśli bit R wynosi 1, to jest on resetowany, a wskazówka również przechodzi na kolejną pozycję. Algorytm Zegarowy ma takie same właściwości wydajnościowe co algorytm drugiej szansy, aczkolwiek wykonanie go zajmuje nieco mniej czasu.

## Algorytm LRU

Algorytm LRU (Least Recently Used) opiera się na założeniu, że strony, które były używane niedawno, prawdopodobnie będą używane w bliskiej przyszłości. Z kolei strony, których nie używano od dłuższego czasu, raczej nie będą nam potrzebne w bliskiej przyszłości. Algorytm LRU jest teoretycznie wykonalny, ale nie jest tani, ponieważ aby go w pełni zaimplementować, potrzebna jest lista wszystkich ramek stron. Ponadto ta lista musi być uporządkowana względem czasu od ostatniego odwołania do danej strony. Utrzymanie tak posortowanej listy wymaga jej aktualizacji co każde odwołanie do pamięci RAM. Jest to bardzo wymagające obliczeniowo. Istnieją również inne sposoby implementacji algorytmu LRU, ale potrzebują one specjalnego sprzętu. Jednym z takich rozwiązań jest wykorzystanie 64-bitowego licznika, który jest automatycznie inkrementowany po każdej instrukcji. Co więcej, każda ramka strony musi zawierać pole o tej samej wielkości do zapisywania wartości licznika. Przy każdym odwołaniu do pamięci bieżąca wartość tego licznika jest zapisywana do ramki strony, do której jest odwołanie. Kiedy występuje brak strony system operacyjny usuwa ramkę strony o najmniejszej wartości zapisanej wartości licznika, czyli tą, do której odwołanie było najdawniej.

## Algorytm postarzania

Algorytm postarzania to wydajny algorytm, który przybliża algorytm LRU. Każda ramka strony ma swój licznik, składający się z np. 8 bitów. Przy każdym przerwaniu zegara, system operacyjny przesuwa każdy licznik strony w prawo o jeden bit, a następnie dodaje do skrajnie lewego bitu licznika wartość bitu R. Dzięki temu system operacyjny ma informacje o ważności poszczególnej ramki strony w ostatnich 8 taktach zegara. Dzięki przesuwaniu licznika w prawo, strona, do której odwołano się jedynie w poprzednim takcie zegara, uzyskuje większą wartość licznika niż strona, która miała więcej odwołań w ostatnich 8 cyklach, ale były one dawniej, na przykład w 3, 4 i 5 cyklu zegara. Kiedy wystąpi błąd braku strony, algorytm postarzania usuwa taką stronę, której licznik ma najmniejszą wartość. Algorytm postarzania to bardzo dobre przybliżenie algorytmu LRU, które można wydajnie zaimplementować.

## Algorytm bazujący na zbiorze roboczym

Zauważono, że procesy charakteryzują się lokalnością odwołań, co oznacza, że podczas dowolnej fazy wykonania, proces odwołuje się niewielkiej części jego stron. Zbiorem roboczym nazywamy zbiór stron, które są aktualnie wykorzystywane przez proces. Algorytmy bazujące na zbiorze roboczym usuwają ramkę strony spoza tego zbioru, gdy występuje brak strony. Jednym ze sposobów implementacji tej idei jest zdefiniowanie zbioru roboczego jako zbiór stron, do których proces odwoływał się w czasie ostatnich β sekund czasu wirtualnego, gdzie czas wirtualny nazywamy czas procesora zużyty przez proces od

momentu jego uruchomienia. Zakłada się, że β obejmuje kilka taktów zegara. Podstawowa idea polega na znalezienie strony, której nie ma w zbiorze roboczym. Każda ramka strony musi posiadać pole zawierające przybliżony czas od ostatniego użycia strony oraz bit R. Przy każdym wystąpieniu błędu braku strony następuje przeszukiwanie ramek stron w celu znalezienia strony do usunięcia. Podczas przetwarzania każdej pozycji badany jest bit R. Jeśli ma on wartość 1, bieżący czas wirtualny jest zapisywany do pola ramki strony. Natomiast jeżeli w bieżącym takcie zegara nie było odwołania do danej strony i R wynosi 0, algorytm wylicza wiek ramki strony. Robi się to, odejmując od bieżącego czasu wirtualnego, czas od ostatniego użycia ramki strony. Jeżeli wiek jest większy niż β, strona nie należy już do zbioru roboczego i jest ona usuwana. Pomimo zastąpienia strony, algorytm jest kontynuowany, aby zaktualizować czas ostatniego użycia strony dla pozostałych pozycji. Jeśli jednak wiek jest mniejszy od β, strona dalej należy do zbioru roboczego. System operacyjny zapamiętuje stronę o największym wieku na wypadek, gdyby wszystkie strony były w zbiorze roboczym. W takim przypadku usuwana jest strona o największym wieku. Algorytm ten jest trudny do zaimplementowania, ponieważ musi przeskanować wszystkie ramki stron.

## Algorytm WSClock

Algorytm WSClock jest to algorytm, który łączy algorytm zegarowy z ideą zbioru roboczego. Potrzebna jest struktura danych taka sama jak w algorytmie zegarowym, czyli cykliczna lista ramek stron, bit R i M oraz wskazówka. Dodatkowo każda ramka strony zawiera pole: czas ostatniego użycia strony. W przypadku błędu braku strony badana jest ramka strony, na którą wskazuje wskazówka. Gdy bit R równy jest 1, algorytm zachowuje się jak przy algorytmie zegarowym. Natomiast gdy bit R równy jest 0, to badany jest wiek strony. Jeżeli wiek jest mniejszy od β, to strona należy do zbioru roboczego i algorytm szuka dalej. Sytuacja się komplikuje, gdy wiek jest większy od β. Jeżeli strona nie była modyfikowana, to jest ona usuwana. W przeciwnym wypadku system planuje zapis zmienionych danych na dysk, ale wskazówka jest przesuwana dalej i dla niej jest realizowany algorytm. Na dalszych pozycjach może być strona, która nie jest w zbiorze roboczym i której nie trzeba zapisywać na dysk. Jeżeli wszystkie strony należą do zbioru roboczego, najprostsze co może zrobić algorytm to usunięcie jakiejkolwiek ramki, której dane nie były modyfikowane. Algorytm WSClock jest powszechnie używany w praktyce ze względu na swoją wydajność oraz względną prostotę.

## Podsumowanie

Najlepsze algorytmy zastępowania stron to bez wątpienia algorytm postarzania i WSClock. Stanowią one odpowiednio zgrabną implementację idei algorytmu LRU i zbioru roboczego. Oba zapewniają dobrą wydajność.

### Bibliografia


1. Systemy operacyjne, Andrew S. Tanenbaum, Herbert Bos. Wydanie 4
2. https://en.wikipedia.org/wiki/Page_replacement_algorithm