Zadanie 2
Symulacja algorytmów planowania dostępu do dysku.

- 'Dysk' to w naszym przypadku liniowo uporządkowany ciąg bloków o nr od 1 do MAX.
- Kryterium oceny algorytmów będzie suma przemieszczeń głowicy dysku, jak wiadomo proporcjonalna do czasu realizacji zleceń.

1.Sprawdzić algorytmy FCFS, SSTF, SCAN i C-SCAN.
2.Następnie założyć, że w systemie istnieją także aplikacje real-time, które musza być obsłużone za pomocą EDF i/lub FD-SCAN. Jak wpływa to na wyniki?
UWAGA!
Sformułowanie nie wymienionych powyżej warunków symulacji należy do Państwa. Mam na myśli:
-wielkość 'dysku' (ilość bloków)
-liczba i sposób generowania zgłoszeń (pełna kolejka od początku? zgłoszenia w trakcie? rozkład zgłoszeń- równomierny, inny?)
-sposób uwzględnienia obsługi zgłoszeń real-time
-mile widziana umiejętność uzasadnienia przyjętego rozwiązania.

Zadanie 2 - symulacja algorytmów planowania dostępu do dysku

Zgodnie z założeniami zadania dysk to uporządkowany liniowo ciąg bloków. W ramach założeń zadania proponuję, aby rozmiar tego bloku był konfigurowany poprzez podanie odpowiedniego parametru przy uruchomieniu symulacji.

W ramach zadania zwracam również uwagę, że w trakcie symulacji algorytmy będą oceniane poprzez określenie sumy przemieszczeń głowicy (dla uproszczenia możemy pominąć czas odczytu sektora na potrzeby obsługi żądania). W związku z tym, w praktyce, wygodnie będzie jeżeli w przypadku uwzględnienia aplikacji real-time przyjmą Państwo założenie, że jeden kwant czasu odpowiada przemieszczeniu głowicy o jeden cylinder dysku. W praktyce będzie to oznaczało, że jeżeli mamy deadline na realizację żądania zdefiniowany jako 72, to jesteśmy w stanie go zrealizować jedynie wtedy gdy odległość poszukiwanego cylindra na dysku od głowicy jest mniejsza lub równa 72.

Istotnym elementem, który powinien zostać zaimplementowany jest możliwość parametryzowania symulacji. W związku z tym parametry takie jak rozmiar dysku, liczba żądań real-time (w scenariuszach odpowiednich dla tego) lub ich rozkład powinny być możliwe to zdefiniowania, chociażby w ograniczonym zakresie. Przypominam również, że na zajęciach skupiamy się na symulacji działania algorytmów, a nie na ich samej implementacji, w związku z tym zbieranie statystyk (w szczególności kryterium oceny algorytmów) z działania algorytmów jest równie ważne co ich implementacja. Z tego powodu **istotne jest również to, aby każdy z algorytmów operował na tym samym ciągu żądań**, aby możliwe było ich porównanie.

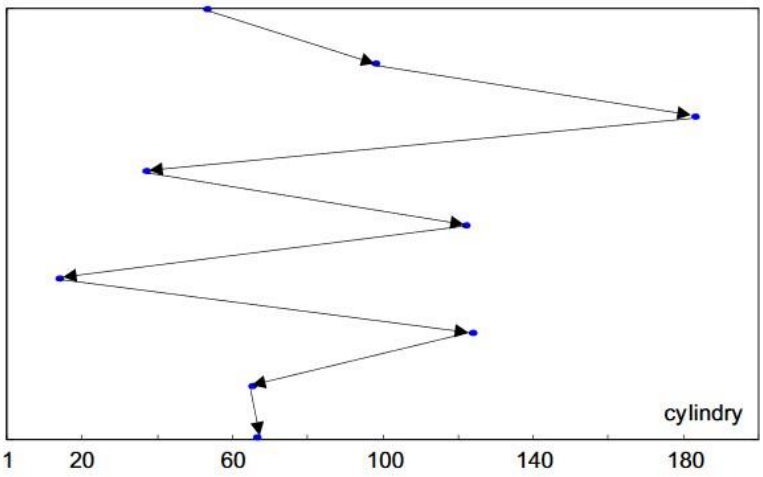
Jeżeli chodzi o same algorytmy, to w ramach zadania zostały zdefiniowane 4 algorytmy podstawowe oraz 2 strategie obsługi żądań real-time, które powinny zostać zaimplementowane w ramach symulacji. Zostały one krótko omówione poniżej, natomiast w celu poznania szczegółów ich działania odsyłam Państwa do literatury oraz do materiałów z wykładu.

1) FCFS (First Come First Serve)

Jest to najprostszy algorytm dostępu do dysku, ale jednocześnie jest to algorytm sprawiedliwy. W ramach algorytmu żądania odczytu z dysku są realizowane w kolejności nadejścia. Oznacza, to, że

w celu obsługi żądań są one kolejkowane w kolejności nadejścia, a do obsługi po kolei wybierane są żądania, które są na początku kolejki.

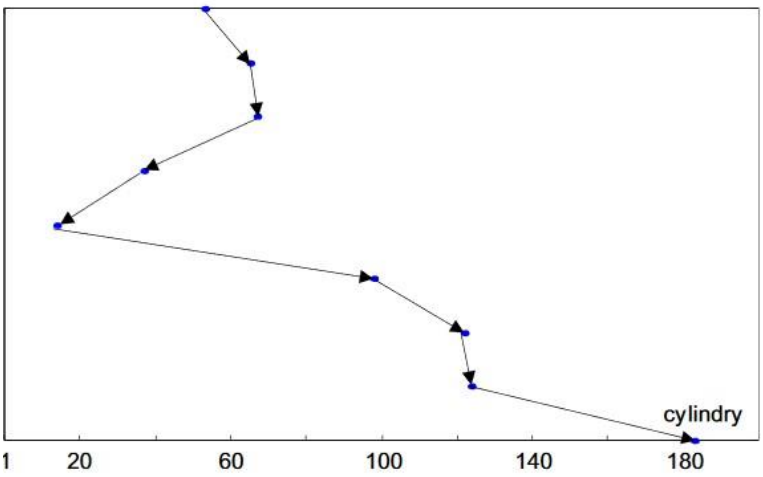
Poniżej zamieszczony został obrazek, który wizualizuje działanie tego algorytmu. W analizowanym momencie głowica dysku znajduje się nad cylindrem **53**, a kolejka żądań jest następująca: **[98,183,37,122,14,124,65,67]** (i nie napływają do niej nowe żądania). Suma przewiniętych cylindrów dla tego algorytmu w tym przypadku to: **640**.



2) SSTF (Shortest Seek Time First)

Omawiany algorytm stara się zminimalizować przemieszczenia głowicy poprzez obsługiwanie w pierwszej kolejności żądań, które są najbliżej obecnej pozycji głowicy. W praktyce oznacza to, iż każde kolejne żądanie napływające do systemu jest dodawane do kolejki, a następnie kolejka ta jest sortowana rosnąco względem odległości poszczególny żądań od aktualnej pozycji głowicy.

Poniżej zamieszczony został obrazek, który wizualizuje działanie tego algorytmu. W analizowanym momencie głowica dysku znajduje się nad cylindrem **53**, a kolejka żądań jest następująca: **[98,183,37,122,14,124,65,67]** (i nie napływają do niej nowe żądania). Suma przewiniętych cylindrów dla tego algorytmu w tym przypadku to: **236**.

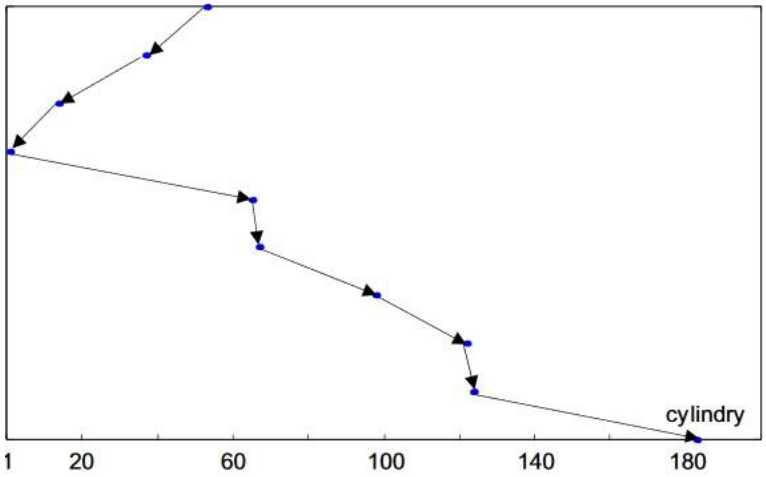


3) SCAN

W metodzie SCAN ramię dysku rozpoczyna pracę od jednej krawędzi dysku i przemieszcza się w kierunku krawędzi przeciwległej obsługując zamówienia z kolejki znajdujące się po drodze. Po

dotarciu do krawędzi dysku zmienia się kierunek ruchu głowicy. W praktyce oznacza to, że głowica nieustannie przeszukuje (skanuje) dysk tam i z powrotem. Oznacza to, że po osiągnięciu dowolnej z krawędzi dysku żądania w kolejce do obsługi są sortowane zgodnie z kierunkiem, w którym będzie poruszała się głowica, a w przypadku pojawienia się nowych żądań, które znajdują się „przed" głowią są on wstawiane na odpowiednie miejsce w kolejce.

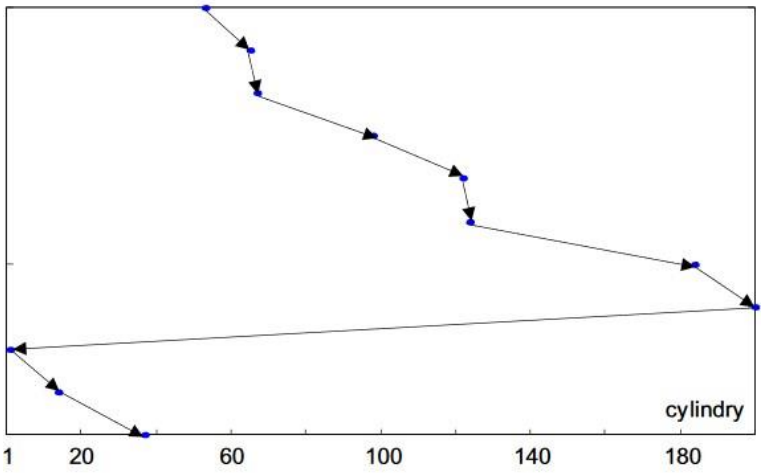
Poniżej zamieszczony został obrazek, który wizualizuje działanie tego algorytmu. W analizowanym momencie głowica dysku znajduje się nad cylindrem **53**, a kolejka żądań jest następująca: **[98,183,37,122,14,124,65,67]** (i nie napływają do niej nowe żądania). Suma przewiniętych cylindrów dla tego algorytmu w tym przypadku to: **236**.



4) C-SCAN

Algorytm C-SCAN jest wariantem algorytmu SCAN. Algorytm działa podobnie jak metoda SCAN, z tą różnicą, że po dojściu głowicy do skrajnego położenia wraca ona natychmiast do przeciwległego położenia (bez obsługi zamówień znajdujących się po drodze).

Poniżej zamieszczony został obrazek, który wizualizuje działanie tego algorytmu. W analizowanym momencie głowica dysku znajduje się nad cylindrem **53**, a kolejka żądań jest następująca: **[98,183,37,122,14,124,65,67]** (i nie napływają do niej nowe żądania). Suma przewiniętych cylindrów dla tego algorytmu w tym przypadku to: **183**.



I dodatkowo, krótkie omówienie dwóch strategii obsługi żądań dla aplikacji czasu rzeczywistego:

5) EDF (Earliest Deadline First)
   Algorytm EDF działa w sposób analogiczny do algorytmu SSTF, z tym, że zamiast wybierania do realizacji żądania najbliższego wybiera on w sposób zachłanny do realizacji jako następne to żądanie, którego deadline na realizację jest najkrótszy i zmierza bezpośrednio do cylindra, w którym znajduje się poszukiwany sektor. Po obsłużeniu wszystkich tego typu żądań system obsługi dysku przestawia się na wybrany standardowy algorytm obsługi żądań do momentu pojawienia się kolejnego żądania real-time.

6) FD-SCAN (Feasible Deadline SCAN)
   Algorytm FD-SCAN działa w nieco mniej „bezmyślny" sposób niż algorytm EDF. W przypadku tego algorytmu realizowane w pierwszej kolejności są żądania o najkrótszym, **możliwym do spełniania** (stąd „feasible" w nazwie). Żądania, których deadline jest krótszy niż czas potrzebny na dotarcie do nich są po prostu odrzucane przez algorytm. Dodatkowo, w trakcie przemieszczania głowicy realizowane są wszystkie żądania (zwykłe i real-time), które znajdują się po drodze (stąd „SCAN" w nazwie).

Warto zwrócić uwagę, że uwzględnienie strategii czasu rzeczywistego w symulacji wymaga rozbudowania struktury opisującej żądania o flagę wskazującą, że jest to tego typu żądanie (obsługiwane priorytetowo) oraz o pole opisujące deadline na realizację tego żądania.

W przypadku wystąpienia problemów w implementacji algorytmów lub pojawienia się pytań raz jeszcze odsyłam do polecanej literatury oraz zachęcam do kontaktu w ramach odbywających się ze mną konsultacji.

Źródło obrazków:

http://www.issi.uz.zgora.pl/pl/didactic/kp/so/wyk7.pdf