JEGYZŐKÖNYV

Operációs rendszerek BSc

2025. tavasz féléves feladat

Memóriafoglalási igények,
Lapozó algoritmusok

Készítette: **Tán Gergő**

Neptunkód: **BLCL2O**

Dátum: **2025. 05. 13.**

Sárospatak, 2025

Tartalomjegyzék

1. Feladat

Adott egy számítógépes rendszer, melyben a következő,

- szabad memória területek: 50k, 30k, 200k, 16k, 30k, melynek
- foglalási igénye: 20k, 30k, 10k, 100k, 60k.

Határozza meg változó méretű partíció esetén a következő algoritmusok felhasználásával: first fit, worst fit a foglalási igényeknek megfelelő helyfoglalást – táblázatos formában!

1.1 A feladat elkészítésének lépései

First Fit		Szabad területek					
Igény	Foglalható	50	30	200	16	30	
20	20	20, 30	30	200	16	30	
30	32	30	30	32, 168	16	30	
10	12	12, 18	30	168	16	30	
100	100	18	30	100, 68	16	30	
60	60	18	30	60, 8	16	30	
Szabad partíciók:		18	30	8	16	30	
Nem sikerült		0 igény					
Legkisebb eddigi foglalás		12					
							Össz
Maradék		2	2	0	0	2	6
Túl kicsi blokkok		0	0	8	0	0	8
Blokk. Miatti tör.		0					
		2					
		2					
		0					
		0					
							4
							18
Összes veszteség		6%					

Worst Fit		Szabad területek					
Igény	Foglalható	50	30	200	16	30	
20	20	50	30	20, 180	16	30	
30	32	50	30	32, 148	16	30	
10	12	50	30	12, 136	16	30	
100	100	50	30	100, 36	16	30	
60	60	50	30	36	16	30	
Szabad partíciók		50	30	36	16	30	
Nem sikerült		1 igény					
Legkisebb eddigi foglalás		12					
Maradék		2	2	0	0	2	6
Túl kicsi blokkok		0	0	0	0	0	0
Blokkok miatti töredezettség		0					
		2					
		2					
		0					
		0					
							4
							10
Összes veszteség		3%					

1.2 Futtatás eredménye, magyarázat

A **First Fit** algoritmus minden igényt sikeresen képes volt kiszolgálni. Ennek az az előnye, hogy a rendelkezésre álló memória területet a lehető legkorábban igyekezett felhasználni, így a kérések gyorsan teljesültek.

Ezzel szemben a **Worst Fit** algoritmus nem tudta kiszolgálni az 5. igényt, amely 60k memóriát igényelt. Ennek oka, hogy a legnagyobb elérhető blokkokat foglalta le előzőleg, így a nagyobb igények számára nem maradt elegendő hely. Az algoritmus úgy próbált gazdálkodni a memóriával, hogy mindig a legnagyobb szabad blokkot választotta ki, de a túl nagy blokkok lefoglalása azt eredményezte, hogy a későbbi nagyobb igények már nem fértek el.

Ez a megközelítés az üzleti szintű szemléletben azt mutatja, hogy bár a **First Fit** gyors és hatékony a kisebb igények kezelésében, a **Worst Fit** hosszú távon nem biztosít optimális memóriahasználatot, mivel a túl nagy blokkok lefoglalása miatt a később érkező, de kisebb igények már nem tudják elérni a szükséges erőforrást.

2. Feladat

Adott egy *igény szerinti lapozást* használó számítógéprendszer, melyben futás közben egy processz számára a következő laphivatkozással lehet hivatkozni:

6, 5, 4, 3, 5, 6, 2, 8, 5, 6, 5, 4, 7, 8, 4, 5, 6, 5, 5, 8

Memóriakeret (igényelt lapok): 3, ill. 4 memóriakeret.

Készítse el a laphivatkozások betöltését külön-külön táblázatba 3, ill. 4 memóriakeret esetén.

2.1 A feladat elkészítésének lépései

OPT:

Memóriakeret	Laphivatkozások																			
Igényelt lap	6	5	4	3	5	6	2	8	5	6	5	4	7	8	4	5	6	5	5	8
1. lap	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7
2. lap		5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3. lap			4	4	4	4	4	4	4	4	4	4	4	4	4	4	6	6	6	6
4. lap				3	3	3	2	8	8	8	8	8	8	8	8	8	8	8	8	8
Laphibák	*	*	*	*			*	*					*				*			
Laphibák Össz:	4+4=8																			

Memóriakeret	Laphivatkozások																			
Igényelt lap	6	5	4	3	5	6	2	8	5	6	5	4	7	8	4	5	6	5	5	8
1. lap	6	6	6	6	6	6	6	6	6	6	6	4	4	4	4	5	5	5	5	5
2. lap		5	5	5	5	5	5	5	5	5	5	5	7	7	7	7	6	6	6	6
3. lap			4	3	3	3	2	8	8	8	8	8	8	8	8	8	8	8	8	8
Laphibák	*	*	*	*			*	*				*	*			*	*			
Laphibák össz:	4+6=10																			

FIFO:

Memóriakeret	Laphivatkozások																			
Igényelt lap	6	5	4	3	5	6	2	8	5	6	5	4	7	8	4	5	6	5	5	8
1. lap	6	6	6	6	6	6	2	2	2	2	2	4	4	4	4	4	6	6	6	6
2. lap		5	5	5	5	5	5	8	8	8	8	8	7	7	7	7	7	7	7	7
3. lap			4	4	4	4	4	4	5	5	5	5	5	8	8	8	8	8	8	8
4. lap				3	3	3	3	3	3	6	6	6	6	6	6	5	5	5	5	5
Laphibák	*	*	*	*			*	*	*	*		*	*	*		*	*			
FIFO vége	6	5	4	3	2	8	5	6	4											
Laphibák össz:	4+9=13																			

Memóriakeret	Laphivatkozások																			
Igényelt lap	6	5	4	3	5	6	2	8	5	6	5	4	7	8	4	5	6	5	5	8
1. lap	6	6	6	3	3	3	3	8	8	8	8	4	4	4	4	5	5	5	5	5
2. lap		5	5	5	5	6	6	6	5	5	5	5	7	7	7	7	6	6	6	6
3. lap			4	4	4	4	2	2	2	6	6	6	6	8	8	8	8	8	8	8
Laphibák	*	*	*	*		*	*	*	*	*		*	*	*		*	*			
FIFO vége	6	5	4	3	6	2	8	5	6	4	7									
Laphibák össz:	4+11=15																			

2.2 Futtatás eredménye, magyarázat

Mennyi laphiba keletkezik az alábbi algoritmusok esetén: FIFO, OPT?

3 memória keret esetén OPT algoritmust használva összesen 10 laphiba keletkezett.

4 memória keret esetén OPT algoritmust használva összesen 8 laphiba keletkezett.

3 memória keret esetén FIFO algoritmust használva összesen 15 laphiba keletkezett.

4 memória keret esetén FIFO algoritmust használva összesen 13 laphiba keletkezett.

3. Összehasonlítás

Az **OPT (Optimal)** algoritmus úgy működik, hogy minden egyes laphiba esetén azt a lapot távolítja el a memóriából, amelyre a legtovább nem lesz szükség a jövőben. Ennek eredményeként az elérhető legkevesebb laphibát produkálja, hiszen mindig a „legjobb” döntést hozza a jövő ismeretében. A 4 memóriakeretes verzióban összesen 8 laphiba keletkezett, míg 3 keret esetén ez a szám 10-re emelkedett. Ez jól szemlélteti, hogy a memóriakeret méretének csökkentése negatív hatással van a teljesítményre, még egy optimális algoritmus esetén is.

A **FIFO (First-In, First-Out)** algoritmus ezzel szemben nem veszi figyelemben a jövőben zajló eseményeket, és a legrégebben betöltött lapot cseréli ki akkor is, ha az éppen most válna újra szükségessé. Ez a stratégia gyakran vezet rossz döntésekhez, különösen olyan helyzetekben, amikor a gyakran használt lapok kerülnek először betöltésre. A 4 keretes FIFO esetén 13 laphiba történt, míg 3 keret esetén ez a szám 15-re nőtt. Ez az eredmény jelentősen elmarad az OPT algoritmus teljesítményétől, különösen a kisebb memóriakeret esetén, ahol minden helytelen döntés jobban érezteti a hatását.

A vizsgálat alapján egyértelműen megállapítható, hogy az OPT algoritmus produkálja a legkevesebb laphibát, mivel minden döntését a jövőbeli lapreferenciák alapján hozza meg. Ez az algoritmus ideális esetet feltételez, hiszen a jövőt a valóságban nem ismerjük, így ez leginkább elméleti összehasonlítási alapként szolgál. A FIFO algoritmus teljesítménye jelentősen elmarad az OPT mögött, mivel nem képes alkalmazkodni a lapok tényleges használati gyakoriságához vagy időzítéséhez. Az algoritmus egyszerűsége miatt gyakran vezet nagyobb számú laphibához, különösen akkor, ha a memóriakeret szűkös és gyakori a lapcsere.

Általánosságban elmondható, hogy a memóriakeret méretének növelése minden algoritmus esetén csökkenti a laphibák számát, mivel több lap fér el egyszerre a memóriában, így kisebb az esélye annak, hogy egy szükséges lap épp ne legyen elérhető. Ugyanakkor a választott algoritmus típusa sokkal nagyobb hatással van a teljesítményre, mint önmagában a keret mérete. Míg az OPT intelligens döntései révén még kisebb memóriakerettel is hatékony marad, a FIFO gyakran túl korán törli a később újra szükséges lapokat.

4. Összefoglalás

Összefoglalva a lapozási algoritmusok teljesítménye erősen függ attól, hogy mennyire képesek előre látni vagy előnyben részesíteni a gyakori lapokat. Az optimális döntéshozatalhoz hasonló viselkedés jelentős laphibacsökkenést eredményezhet, míg egyszerű, nem adaptív stratégiák – mint a FIFO – megnövelik a rendszerre nehezedő terhelést.