CS.4310.01 Homework 2

1. Consider a swapping system in which memory consists of the following hole sizes in memory order: 7KB, 4KB, 23KB, 9KB, 6KB, 18KB, 11KB, and 2KB. Which hole is taken for successive segment requests of

• 6KB • 15KB • 9KB • 10KB • 2KB

First fit

6KB – 7KB hole

15KB – 23KB hole

9KB – 9 KB hole

10 KB - 18KB hole

2KB – 4KB hole

Best fit

6KB – 7KB hole

15KB – 18KB hole

9KB - 9KB hole

10KB – 11KB hole

2KB – 2KB hole

Worst fit

6KB – 23KB hole

15KB – 18KB hole

9KB – 17KB hole (remaining from 23KB hole taken by 6KB)

10KB – 11KB hole

2KB – 9KB hole

Next fit

6KB – 7KB hole

15KB – 23KB hole

9KB – 9KB hole

10KB – 18KB hole

2KB – 11KB hole

1. For 2KB page size – 11 bits is needed so the offset will be 11 bits. The 11 rightmost bit will be the offset and the remaining will be the virtual page number

* 3002 in binary is 101110111010 which is 12 bits so the virtual page number will be 1 and the offset is 01110111010
* 1097 in binary is 10001001001 which is 11 bits so the virtual page number will be 0 and the offset will be 10001001001
* 28127 in binary is 110110111011111 which is 15 bits so the virtual page number will be 1101 and the offset is 10111011111
* 14550 in binary is 11100011010110 which is 14 bits so the virtual page number will be 111 and the offset is 00011010110

For 4KB page size – 12 bits is needed so the offset will be 12 bits. The 12 rightmost bit will be the offset and the remaining will be the virtual page number

* 3002 in binary is 101110111010 which is 12 bits so the virtual page number will be 0 and the offset is 101110111010
* 1097 in binary is 10001001001 which is 11 bits so the virtual page number will be 0 and the offset will be 10001001001
* 28127 in binary is 110110111011111 which is 15 bits so the virtual page number will be 110 and the offset is 110111011111
* 14550 in binary is 11100011010110 which is 14 bits so the virtual page number will be 11 and the offset is 100011010110

1. The offset will be 64 – 14 – 16 = 34 bits which correspond to 2^34 = 16GiB for the page size

The number of pages will be 2^14 x 2^16 = 2^30 number of pages in the address space

1. There will be 8 page faults for the reference string 236571345157245 for five page frames that are initially empty when using FIFO

A screenshot of a computer code

Description automatically generated

1. There will be 8 page faults for the reference string 236571345157245 for five page frames that are initially empty when using LRU

A screenshot of a computer program

Description automatically generated

1. The beginning of a free space bitmap looks like this after the disk

partition is first formatted: 1000 0000 0000 0000 0000 0000 0000 0000 (the first block is used by the root directory). The system always searches for free blocks starting at the lowest numbered block, so after writing file A, which uses 8 blocks, the bitmap looks like this 1111 1111 1000 0000 0000 0000 0000 0000. Show the bitmap after each of the following actions:

1. File B is written, using 12 blocks

1111 1111 1111 1111 1111 1000 0000 0000

1. File C is written, using 9 blocks

1111 1111 1111 1111 1111 1111 1111 1100

1. File A is deleted

1000 0000 0111 1111 1111 1111 1111 1100

1. File B is deleted

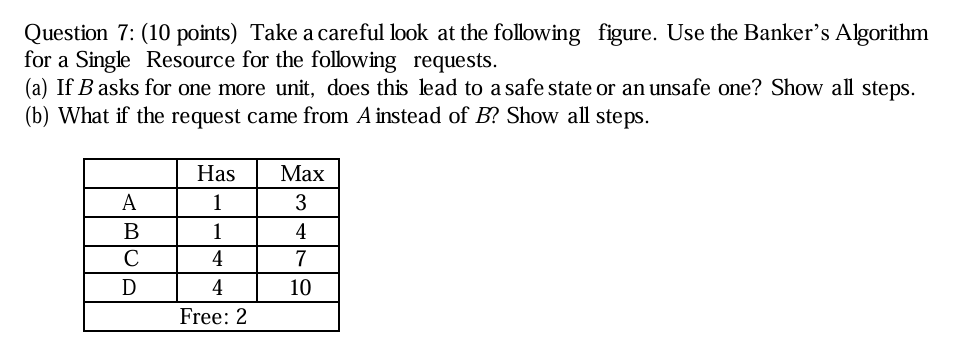
1000 0000 0000 0000 0000 0111 1111 1100

1. File D is written, using 10 blocks

1111 1111 1110 0000 0000 0111 1111 1100

1. File E is written, using 3 blocks

1111 1111 1111 1100 0000 0111 1111 1100



7a. Has Max Need

A 1 3 2

B 1 4 3

C 4 7 3

D 4 10 6

Free: 2

Currently, the system is safe because A can finished if the 2 units were assigned to A. However, by giving B 1 unit, which is possible since B still needs 3 units to complete and the requested amount is less than or equal to 2, but it would lead to an unsafe state where

Has Max Need

A 1 3 2

B 2 4 2

C 4 7 3

D 4 10 6

Free: 1

None of the process can finish.

7b. If the request came from A, it would still be in a safe state since process A can finish with the remaining free resource unit

Has Max Need

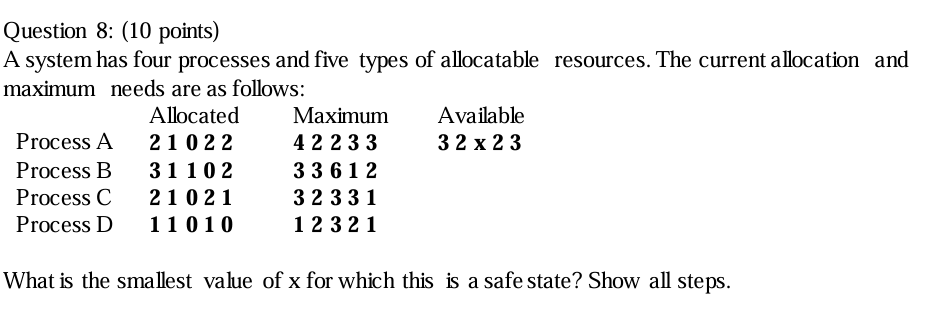
A 2 3 1

B 1 4 3

C 4 7 3

D 4 10 6

Free: 1



8. From the given data, we can compute the following table

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Allocated | | | | | Max | | | | | Need | | | | |
| A | 2 | 1 | 0 | 2 | 2 | 4 | 2 | 2 | 3 | 3 | 2 | 1 | 2 | 1 | 1 |
| B | 3 | 1 | 1 | 0 | 2 | 3 | 3 | 6 | 1 | 2 | 0 | 2 | 5 | 1 | 0 |
| C | 2 | 1 | 0 | 2 | 1 | 3 | 2 | 3 | 3 | 1 | 1 | 1 | 3 | 1 | 0 |
| D | 1 | 1 | 0 | 1 | 0 | 1 | 2 | 3 | 2 | 1 | 0 | 1 | 3 | 1 | 1 |

We see that for type 3 (the third column) that the biggest space needs to be at least 5 so that process B won’t cause the system to be in an unsafe state

Therefore, the smallest value of x for which this is a safe state is x = 5