A Persistent Memory Management in Java Card

IM Y. JUNG SUNG I. JUN KYO I. CHUNG

Electronics and Telecommunications Research Institute
161, Gajeong-Dong, Yuseong-Gu
Daejeon, 305-350
REPUBLIC OF KOREA
{imyoung, sijun, kyoil}@etri.re.kr

Abstract: - EEPROM as persistent memory in smart card has some different features with other memories. It consists of the pages with a fixed size and has the characteristics due to them. It is more efficient to consider them and to handle EEPROM by page than to think of it as a seamless space and to deal with it as one large chunk. Because smart card is categorized as the device closely interactive with users, the faster response time is important. Accordingly, the time spent in managing the card memory should be in the acceptable range. The necessity to manage the card memory, EEPROM, to be reused efficiently, arises from its small capacity due to the physical size of smart card itself. In this point, the memory management schemes such as memory allocation, withdrawal and garbage collection get to have importance concretely. In this paper, we propose an efficient persistent memory management scheme in smart card based on the model of our Next Generation Integration Circuit(NGIC) Card.

Key-Words: - Memory Management in Java Card, EEPROM, Page handing, Page Manager, NGIC Card

1 Introduction

EEPROM as persistent memory in smart card has some qualities. If we do ignore them, the utilization of EEPROM may not be high and it may take much time to handle it.

There are some trade offs between the space utilization in memory and the time required to enhance it. The garbage collection is one example. If we take advantage of it, we can enhance the memory utilization. But, the time to do it may exceed the margin we can endure. If we use it in smart card, we may be disappointed of the processing time. It arises from the weak processing power and the small memory sizes in smart card including RAM as volatile memory as well as EEPROM. The physical RAM size in smart card is four times larger than EEPROM if compared in the same capacity[8]. But, EEPROM has some handicaps that RAM dose not have. One thing is the elapsed time to write on. The writing on EEPROM is slower than that on RAM from 30 times[1] to 100 times[8] according to the manufactories of EEPROM. So, we take RAM management into less consideration than EEPROM management in the viewpoint of the card response time.

In this paper, we discuss the EEPROM management scheme including memory allocation and withdrawal and the garbage collection. We process our research with the model of our Next Generation Integrated Circuit(NGIC) card, which is the product name of our project developing the next generation smart card.

2 Next Generation Integrated Circuit Card

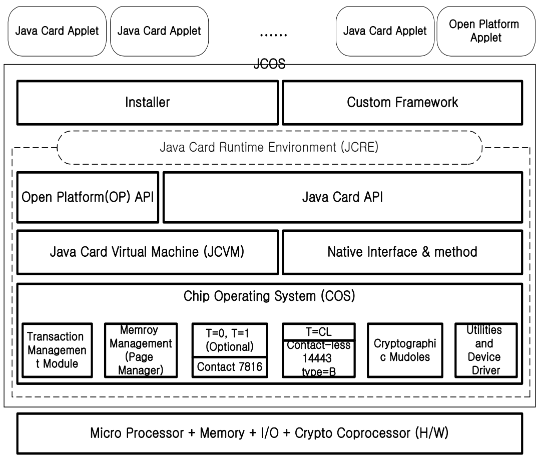
Java Card Applet
Java Card Applet
......
Java Card Applet
Open Platform Applet
JCOS
Installer
Custom Framework
Java Card Runtime Environment (JCRE)
Open Platform(OP) API
Java Card API
Java Card Virtual Machine (JCVM)
Native Interface & method
Chip Operating System (COS)
Transaction Management Module
Memroy Management (Page Manager)
T=0, T=1 (Optional)
Contact 7816
T=CL
Contact-less 14443 type=B
Cryptographic Mudoles
Utilities and Device Driver
Micro Processor + Memory + I/O + Crypto Coprocessor (H/W)

Fig. 1. NGIC Card Architecture

Our NGIC card is Java Card. As licensee of Java Card, we have referenced and followed the Java Card specification from SUN Microsystems Inc. We have developed our NGIC card from its Card or Chip Operating System(COS) to its applications for some years. Java Card is the smart card to execute Java application on its platform. As a subset of Java language, but not an exact one, the Java language for smart card has the properties adapted to the execution platform with the insufficient resources of smart card. The architecture of our card is shown in Fig.1. It consists of some layers to execute Java Card applications and to support Java Card Virtual Machine(JCVM) and the data securities. The memory management module we are to discuss is a component included in COS level and is named as Page Manager. The Java Card including a lot of cryptography modules and Java Card Remote Method Invocation(JCRMI) module may be general in the world of smart card to activate its merits such as the dynamic downloading and deletion of applications and the security support by applet firewall. Nowadays, we dare to realize the applications that we couldn't have sketched owing to the resource limitation of smart card; the processing power and the memory capacity closely associated with the physical chip size of smart card. It results from the increased processing power with the processing unit more than 32bit, and the enlarged RAM and EEPROM more than 4KB and 64KB each. The larger memory Java Smart Card adopts so as to execute the larger applications on it, the more necessary it is to manage the memory not to waste time and memory space.

2.1 Heap in Java Card

We have used a heap structure for memory handling in our smart card. The whole EEPROM may be divided into several heaps. The memory allocation and withdrawal is managed in the heaps. Each heap has a table to maintain and manage the information of the objects allocated and freed within it. Memory allocation to some objects and the withdrawal of them are checked in the table. When the allocated space is returned as free one, it can be reused. When the cycles of allocation and withdrawal do over, there exist some gaps among the adjacent allocated memory spaces. We name them as heap fragments or fragments and the process to produce them as heap fragmentation or fragmentation. To gather and reuse the fragments, Java Language provides the mechanism of the garbage collection apart from coalescing the adjacent available spaces to make a large usable one. But, in case of Java Card, only the memory coalescence is recommended and the garbage collection is considered to gather the spaces not used any more by any objects only when there is no more memory space to allocate. In Java Card, the objects do not return or free its memory that they have occupied spontaneously until it is inevitable to do. The memory compaction that piles up the unoccupied spaces to one side and the occupied ones to the other side, is not supported even in the garbage collection or in other modules implemented on Java Card 2.2[a]. The processing time may be one major factor to avoid the memory compaction. At most, in memory withdrawal, some adjacent memory spaces unoccupied may be connected to one space available to be reused. In brief, Java Card has the aspect that to call the garbage collection is a special event that is not included in a part of the general memory management mechanism.

3 EEPROM As Persistent Memory In Smart Card

EEPROM has its limit in the number of writes on it. That is to say, if we write on one spot in EEPROM more than the cycle of 100000, we can not use the EEPROM any more[6][7][8][b]. And, when we write over two pages in EEPROM consecutively, we should wait about 10msec[6][7][c] before writing to the second page. This time blocked exists even when writing in the same page, if the time gap between consecutive writings exceeds about 15usec[6][7][d]. Therefore to minimize this time blocked, it may be much efficient to handle EEPROM by the unit of page. Besides, in order to avoid concentrated writings, it stimulates the balanced uses of EEPROM memory.

4 Memory Management By Page Handling Scheme

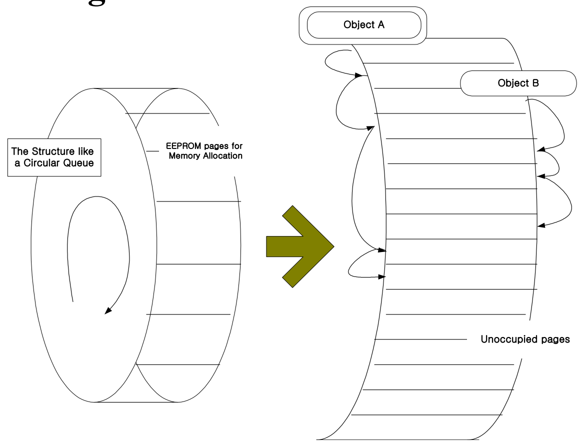
Object A
Object B
The Structure like a Circular Queue
EEPROM pages for Memory Allocation
Unoccupied pages

Fig. 2. The Memory Architecture for the allocation in EEPROM

[a] Java Card 2.2 is the latest version published by SUN.
[b,c,d] The characteristics of EEPROM are similar, although there are some ranges on the value.

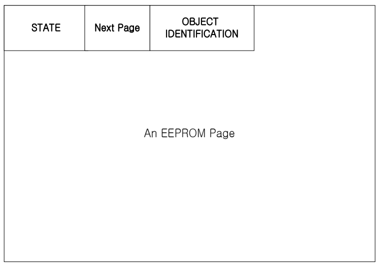

Fig. 3. An EEPROM page to be allocated

The EEPROM memory structure and the memory management scheme we propose are shown as Fig 2. And one page structure in our proposal is shown as Fig 3. In this paper, we assumed the whole EEPROM available to be allocated dynamically as one heap without having its nested heaps or other neighbor heaps. But, multiple heaps may be possible. In that case, each heap is managed by the unit of page. But, if the heap size is small, our scheme may not be effective.

The pages available to be allocated are made to a list named as the free page list and the list is handled as a kind of queue. The pages occupied by some objects are also managed as the shape of linked lists, one list per one object. In our memory management named as the Page Handling Scheme, all memory allocations for new objects in EEPROM are initiated with new pages not with the used pages in that other objects have occupied, but some space are left. So, it is very simple to free the memory occupied by an object because the allocated space initiates from an independent page and ends in another unrelated page with the spaces for other objects. Even though there may be some memory fragments unused, but each size of fragments can't exceed the size of one EEPROM page. If the space for an object that spans several EEPROM pages can't be obtained because the pages are not adjacent or because there are several adjacent pages unoccupied but they are not large enough, those pages may be useless. But, they can be useful in our Page Handling scheme only if they are unoccupied pages. There may be the spaces unavailable to use within pages, but they are swept away naturally when the allocated pages are withdrawn. So, the real fragments can be reduced, and we can even get the memory compaction effect without another overheads. In our scheme, we don't consider the physical position of pages in EEPROM but take the front and rear relation among the pages represented by their link into consideration according to their contents, objects. If we are to delete the object on EEPROM dynamically and withdraw its memory, we are only to adjust the pages' links that have shown the memory space of the deleted object to the free page list. The state of each page is recorded in its header, so it needs not to keep the information for all pages on another EEPROM space separately. The fields each page header keeps are "NEXT PAGE", "OBJECT IDENTIFICATION" and "STATE". The field of "NEXT PAGE" means the page that the current page's link is connected to. And, "OBJECT IDENTIFICATION" shows which object this page belongs to, "STATE" says whether this page is the front, end or middle area of the occupied memory space. All the management for pages including whether they are occupied or not is dependent on the Page Manager.

4.1 Page Manager

The Page Manager implemented on the COS part in Fig. 1 is the module to handle the pages available to be allocated and freed on EEPROM. It is irrelevant whether to manage the objects created with the information table on EEPROM or on RAM. But, it is useful to get tables for occupied pages and unoccupied page on RAM for rapid processing and garbage collection. This information is small because it is enough to keep only the first pages for all meaningful areas. Only if we know the start page of any allocated or unoccupied areas, we can trace the other pages by their links. After ending one Card Acceptance Device(CAD) session-the period from when the card is inserted in the CAD to when the card is removed from the CAD-, new CAD sessions get to use the pages used earlier or unused first. It enhances the memory utilization with the evenly distributed uses. The relative ages of pages can be judged by the information on their headers.

4.2 Memory Allocation

The pages on EEPROM do not have their absolute ages. But, we can know the relative ones by their links. Even though the information for the pages kept on RAM disappears due to a sudden power off or card tear, the relations among the pages can be perceived. If there are unused pages, they are preferable to be allocated. When there are only used pages, the first page in the linked lists for the objects deleted is used first. Each linked list is differentiated by the field of "OBJECT IDENTIFICATION".

4.3 Memory Withdrawal

When the linked lists are withdrawn, the fields of "OBJECT IDENTIFICATION" on their first pages are set to NULL. When these pages are reallocated to another object, they can be differentiated by these NULL values or by the middle page's field of "STATE". If the first page of the linked list freed has already been used, the rest pages can be judged by its "STATE" as free pages. But, the states of the EEPROM pages can be traced more easily with the help of the information tables for free pages and allocated pages on RAM. These information tables are not indispensable because they can be constructed at anytime with the headers of pages.

5 Garbage Collection

When the garbage collection is called independently, it checks all allocated pages by objects and collects the ones unused to add them to the free page list. All allocated pages are sensed with the objects registered in the object table by the method of mark and sweep. The pages unused are put in order by checking whether its first page's "OBJECT IDENTIFICATION" is NULL. Our Page Handling Scheme facilitates the garbage collection. Only by dint of adjusting the pages' links and the field of "OBJECT IDENTIFICATION", we can reuse the memory areas cast aside.

6 Simulation

6.1 Comparison of the Best-Fit Algorithm and the Page Handling Scheme

The representative techniques in memory allocation are based on the algorithm of Best-Fit, First-Fit and Worst-Fit. Among them, the Best-Fit algorithm has some advantages in the aspect of memory utilization. But, if it is applied to the large memory, it may take much time to search the best-fit space. So, even though the other algorithms produce many fragments in memory space to reduce space utilization, they may be used to shorten the time to seek the suitable memory fragment to be allocated. In case of smart card, the memory area to search is small, the Best-Fit algorithm may be the best among them in the aspect of space utilization and time consumption if we do not consider the characteristics of EEPROM. Therefore, in this section, we compare the case applying our Page Handling Scheme with the one using the Best-Fit algorithm in memory allocation and withdrawal and discuss the efficiency of space and time.

6.2 Assumptions and Environment Settings

To affirm in what aspect our scheme is effective compared with the Best-Fit algorithm, we prepare several page sizes for our Page Handling Scheme: 4B, 8B, 16B, and 32B. These page sizes selected are based on the data sheets provided by the memory manufactory, ARTMEL[6][7] . We generated the 3,000 requests for memory allocation and withdrawal to be used in our simulation. And, we tried them 10 times. The portions to be freed in the requests are chosen randomly among the memory segments occupied by prior requests for allocation. The memory sizes to be allocated are chosen in the range from 50B to 2500B in random. We took this range based on the average size of applets to be downloaded on smart card and the sizes of the objects to be created dynamically in our NGIC card. And, the EEPROM size available to be allocated and freed dynamically is assumed to be 32KB. We obtained our simulation results with the establishment that the success or failure cases in memory allocation when the Best-Fit algorithm is used are applied to our Page Handling Scheme in the same way.

Table 1 at Appendix shows the number of successes and failures in memory allocation and that of memory withdrawal in the 3000 requests for 10 trials. These data are the same in our Page Handling Scheme with the page size of 4B, 8B, 16B and 32B as well as in the Best-Fit algorithm.

6.3 Simulation Results

We have an intention by the simulation to affirm the saved number of page transfer in our Page Handling Scheme compared with in Best-Fit algorithm and the size distribution of memory fragments showing the memory utilization.

Table 2 at Appendix shows the memory fragments after processing the 3000 requests for 10 trials. Because we pick up the memory size to be allocated randomly, the fragments in each trial show some different distributions. But, we can verify that all fragments from the Best-Fit algorithm are much smaller than the largest one from the Page Handling Scheme. As the page size goes to be small, the chances not to reject the request of memory allocation for some large objects become high. But, as the page size in our Page Handling Scheme becomes large, the fragments get not to contribute to the memory utilization largely. In our Page Handling Scheme, the physical continuity of memory spaces is not important. Only the pages freed or allocated are meaningful. So, we can ensure the larger space available to be allocated in our Scheme than in the Best-Fit algorithm. And, we can manage EEPROM more easily by the unit of page. We allocate rough amount of EEPROM pages not the exact amount of space, so we need not calculate or know the exact memory position or address to be allocated. Only to know the first page number or address will do. And it is needless to spend much time and processing power to compact memory fragments. In the Page Handling Scheme, the major space can be used in new allocation except the fragments within the last pages in the allocated memory spaces. Other memory management schemes including the Best-Fit algorithm, even though they need compact memory fragments, they can't afford to do it due to a lot of overheads especially in smart card. Because the processing power and resources in smart card are poorer than most devices.

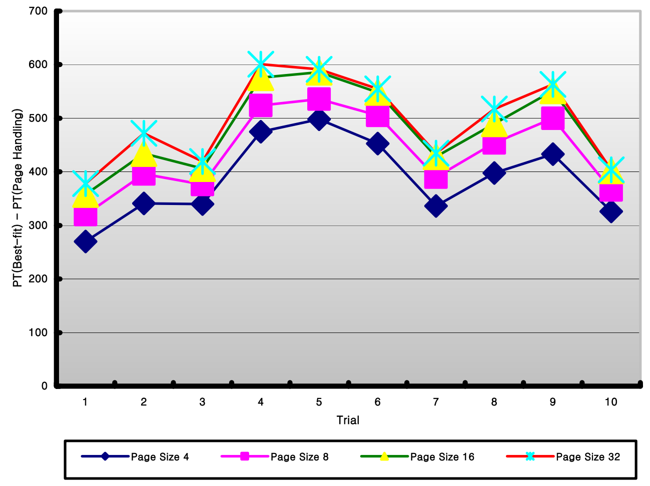

Fig. 4. The Difference of Page Transfer

Fig 4 shows the subtraction of the number of page transfer(PT) in the Page Handling Scheme from that in the best-fit algorithm for 10 trial. The number of page transfer is meaningful in that there is the time blocked of 10msec for the consecutive writing spanning two pages. In our simulation, the page transfer is measured with the assumption that we write onto the whole allocated memory segment once. The numbers of page transfers are related only with the succeeded allocation, not with the 3000 requests. So, if there is more than one writing onto the allocated area, the difference of page transfers will be larger. This result arises from the fact that the Best-Fit algorithm requires more pages than the Page Handling Scheme in the allocation of the same memory size.

7 Related Works

There was a research to overcome the small memory size of smart card[2]. To come over the small capacity, it assumed a virtual memory in off-card part. So, if there are some requests to ask big memory space, the off-card memory was utilized and the allocated areas are mapped onto the on-card part. There were rare the studies on memory handling scheme in smart card. The memory for smart card gets better on its size and access way, and its processing power. It is valuable to think over its structure and its management scheme especially at the viewpoint of implementation, because the smart card market is and will be enlarged and so do the customers' requests.

8 Conclusion

By using the page characteristic of EEPROM, the allocated and freed memory area can be managed easily. Because the state information for each page can be maintained on RAM it can be very fast to access and use it. But, this information can be reconstructed at anytime, there is no burden to worry about its disappearance against sudden power offs or card tears. The garbage collection including memory compaction does not require much time and labor. Also, the possibility to use the whole EEPROM evenly gets to be high by our scheme. Our Page Handling Scheme can be applied to other devices with the memory composed of pages as EEPROM.

References:

[1] Marcus Oestreicher, *Transactions in Java Card*, In 15th Annual Computer Security Applications Conference (ACSAC'99), pp. 291-298. IEEE, 1999.

[2] Clemens H. Cap, Nico Maibaum, Lars Heyden, *Extending the Data Storage Capabilities of a Java-based Smartcard*, Chair for Information and Communication Services, University of Rostock, Germany, unpublished.

[3] Sun MicroSystems, Inc., *Java Card^{TM} 2.1 Application Programming Interface*, Final Revision, June 7, 1999.

[4] Sun Microsystems, Inc., *Java Card^{TM} 2.1 Runtime Environment Specification*, Final Revision, June 7, 1999.

[5] Sun Microsystems, Inc., *Java Card^{TM} 2.1 Virtual Machine Specification*, Final Revision, June 7, 1999.

[6] Atmel, *AT28BV256 Specification*, http://www.atmel.com, 1999.

[7] Atmel, *AT28LV010 Specification*, http://www.atmel.com, 1998.

[8] Zhiqun Chen, *Java Card Technology for Smart Cards* Addison-Wesley, June 2000.

[9] Marek Rusinkiewicz and Amit Sheth, *Specification and execution of transactional workflows*, In W. Kim, editor, Modern Database Systems : The Object Model, Interoperability, and Beyond, ACM Press : Cambridge, Messachussetts, 1994.

Appendix

Table 1 Simulation Establishment

Trial	1	2	3	4	5	6	7	8	9	10
Free	1443	1447	1440	1482	1478	1479	1456	1459	1478	1444
Allocation Success	804	1029	881	1356	1464	1334	1028	1124	1366	963
Allocation Failure	753	494	676	162	58	187	516	417	156	593

Table 2 Fragment Distribution

Trial	Approach	Fragments (B)
1	Best-Fit	20, 42, 61, 65, 69, 85, 111, 118, 126, 131, 159, 672, 1093, 1095, 1209
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4992
	Page SIZE 8	1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 4896
	Page SIZE 16	1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 8, 8, 9, 10, 10, 10, 10, 12, 12, 13, 14, 14, 15, 15, 4784
	Page SIZE 32	1, 2, 2, 2, 2, 4, 4, 5, 5, 5, 7, 7, 9, 10, 10, 10, 12, 13, 14, 14, 15, 16, 16, 17, 17, 17, 19, 19, 19, 20, 20, 21, 21, 21, 22, 22, 22, 22, 23, 24, 24, 26, 28, 31, 4416
2	Best-Fit	32, 60, 116, 126, 351, 523, 999, 1131, 1276
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4564
	Page SIZE 8	1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 7, 7, 4496
	Page SIZE 16	2, 2, 3, 4, 4, 4, 4, 5, 5, 5, 6, 7, 8, 8, 9, 9, 10, 10, 11, 11, 11, 11, 11, 11, 12, 13, 13, 13, 13, 14, 14, 15, 4336
	Page SIZE 32	2, 4, 4, 5, 5, 6, 8, 8, 9, 10, 11, 11, 11, 11, 12, 13, 13, 14, 14, 16, 16, 18, 19, 20, 20, 21, 23, 25, 26, 27, 27, 29, 29, 31, 4096
3	Best-Fit	9, 18, 32, 35, 54, 82, 120, 184, 220, 356, 358, 492, 553, 738, 828, 919, 1146, 2655
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 8744
	Page SIZE 8	1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 7, 8680
	Page SIZE 16	1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 4, 6, 7, 7, 7, 7, 8, 8, 8, 10, 11, 12, 13, 13, 13, 13, 13, 13, 14, 14, 15, 8560
	Page SIZE 32	1, 1, 1, 2, 2, 2, 3, 3, 4, 7, 7, 8, 8, 10, 11, 12, 13, 14, 14, 16, 16, 16, 17, 18, 18, 19, 22, 23, 23, 24, 29, 29, 29, 29, 29, 31, 8288
4	Best-Fit	25, 132, 135, 190, 580, 659, 1661
	Page SIZE 4	1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3340
	Page SIZE 8	1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 3272
	Page SIZE 16	2, 3, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 9, 9, 9, 10, 10, 11, 11, 12, 12, 13, 14, 14, 15, 3168
	Page SIZE 32	2, 4, 4, 5, 6, 6, 7, 7, 9, 10, 11, 12, 14, 14, 16, 16, 19, 20, 21, 22, 22, 25, 25, 26, 27, 28, 29, 31, 2944
5	Best-Fit	28, 43, 148, 211, 285, 362
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 1036
	Page SIZE 8	1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 976
	Page SIZE 16	1, 2, 3, 3, 3, 4, 4, 5, 5, 6, 6, 8, 9, 9, 10, 10, 10, 11, 12, 12, 13, 13, 13, 14, 14, 14, 15 , 848
	Page SIZE 32	1, 3, 5, 9, 10, 11, 13, 13, 14, 14, 16, 16, 18, 19, 19, 20, 20, 21, 22, 22, 24, 25, 26, 26, 28, 28, 29, 30, 31, 544
6	Best-Fit	2, 77, 138, 547, 846, 940, 1350, 1395, 1688, 1766
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 8704
	Page SIZE 8	1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 7, 8648
	Page SIZE 16	1, 1, 1, 3, 3, 4, 5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11, 13, 14, 14, 15, 15, 15, 8544
	Page SIZE 32	1, 1, 3, 5, 6, 8, 8, 9, 9, 10, 10, 11, 13, 14, 14, 15, 15, 16, 17, 19, 20, 21, 21, 22, 23, 23, 31, 8384
7	Best-Fit	19, 27, 51, 90, 138, 242, 292, 331, 342, 361, 361, 623, 643
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3460
	Page SIZE 8	2, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 3360
	Page SIZE 16	2, 2, 2, 2, 3, 4, 6, 6, 6, 7, 7, 8, 8, 8, 10, 10, 10, 11, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 15, 15, 15, 15, 15, 3152
	Page SIZE 32	2, 3, 6, 6, 7, 7, 10, 10, 12, 12, 13, 13, 13, 13, 14, 15, 15, 15, 18, 18, 18, 20, 22, 24, 24, 24, 26, 27, 28, 29, 29, 29, 29, 29, 30, 31, 31, 2848
8	Best-Fit	3, 3, 10, 12, 14, 24, 28, 68, 173, 279, 808, 857, 1098, 1178
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4496
	Page SIZE 8	1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 4424
	Page SIZE 16	1, 1, 2, 2, 2, 2, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 11, 11, 12, 13, 14, 14, 14, 15, 15, 4272
	Page SIZE 32	2, 2, 2, 6, 6, 7, 7, 8, 8, 9, 9, 11, 12, 14, 15, 16, 16, 17, 17, 18, 20, 21, 21, 21, 22, 23, 23, 24, 25, 25, 25, 26, 26, 27, 29, 30, 30, 31, 3904
9	Best-Fit	12, 65, 78, 85, 92, 168, 232, 249
	Page SIZE 4	1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 936
	Page SIZE 8	1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 6, 6, 6, 7, 7, 7, 7, 7, 864
	Page SIZE 16	1, 1, 1, 2, 3, 3, 4, 4, 4, 4, 4, 4, 5, 6, 7, 8, 8, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 14, 14, 15, 15, 15, 15, 720
	Page SIZE 32	1, 2, 3, 3, 4, 4, 5, 7, 8, 10, 11, 12, 12, 13, 14, 14, 15, 15, 16, 16, 17, 17, 20, 20, 20, 20, 22, 24, 24, 24, 25, 26, 27, 31, 31, 448
10	Best-Fit	21, 22, 28, 42, 62, 102, 141, 255, 489, 739, 914, 1180, 1368
	Page SIZE 4	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 5312
	Page SIZE 8	1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 7, 7, 5224
	Page SIZE 16	1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 11, 12, 12, 12, 13, 13, 13, 13, 14, 14, 14, 15, 15, 5072
	Page SIZE 32	1, 1, 2, 2, 2, 3, 4, 4, 4, 5, 5, 10, 10, 10, 12, 13, 14, 15, 15, 16, 17, 18, 18, 18, 20, 21, 21, 21, 21, 24, 24, 26, 27, 28, 28, 29, 29, 29, 30, 30, 4736