HW5

姓名: 陈锐林, 学号:21307130148

2023 年 10 月 14 日

Question1

(1) 已知基址为 1000，空间大小为 100，头部大小为 0，根据题目的要求，进行一系列的 malloc 和 free 后，结果如下表：

操作	返回	内存
Alloc(3)	1000	(1003,97)
free	0	(1000,3),(1003,97)
Alloc(5)	1003	(1000,3),(1008,92)
free	0	(1000,3),(1003,5),(1008,92)
Alloc(8)	1008	(1000,3),(1003,5),(1016,84)
free	0	(1000,3),(1003,5),(1008,8),(1016,84)
Alloc(8)	1008	(1000,3),(1003,5),(1016,84)
free	0	(1000,3),(1003,5),(1008,8),(1016,84)
Alloc(2)	1000	(1002,1),(1003,5),(1008,8),(1016,84)
Alloc(7)	1008	(1002,1),(1003,5),(1015,1),(1016,84)

可以注意到最后 free list 被分成越来越多的小块；并且不可逆。

Question2

(1) 如果换成 WORST 策略，结果如下表：

操作	返回	内存
Alloc(3)	1000	(1003,97)
free	0	(1000,3),(1003,97)
Alloc(5)	1003	(1000,3),(1008,92)
free	0	(1000,3),(1003,5),(1008,92)
Alloc(8)	1008	(1000,3),(1003,5),(1016,84)
free	0	(1000,3),(1003,5),(1008,8),(1016,84)
Alloc(8)	1016	(1000,3),(1003,5),(1008,8),(1024,76)
free	0	(1000,3),(1003,5),(1008,8),(1016,8),(1024,76)
Alloc(2)	1024	(1000,3),(1003,5),(1008,8),(1016,8),(1026,74)
Alloc(7)	1026	(1000,3),(1003,5),(1008,8),(1016,8),(1033,67)

同样的 free list 被分成越来越多的小块，且不可逆；且因为 WORST 策略导致前面的小块不再被使用到，最后剩的块数也多起来了。

Question3

如果用 FIRST 策略，因为选择最小的合适块即可，所以会减少搜索次数；这加速了查找。结果如下：

操作	返回	内存
Alloc(3)	1000	(1003,97)
free	0	(1000,3),(1003,97)
Alloc(5)	1003	(1000,3),(1008,92)
free	0	(1000,3),(1003,5),(1008,92)
Alloc(8)	1008	(1000,3),(1003,5),(1016,84)
free	0	(1000,3),(1003,5),(1008,8),(1016,84)
Alloc(8)	1008	(1000,3),(1003,5),(1016,84)
free	0	(1000,3),(1003,5),(1008,8),(1016,84)
Alloc(2)	1000	(1002,1),(1003,5),(1008,8),(1016,84)
Alloc(7)	1008	(1002,1),(1003,5),(1015,1),(1016,84)

Question4

(1)FIRST 策略是选择找到的第一个足够大的空间，所以很难去判断会怎样受到排序策略的影响。能确定的是 SIZESORT-下是最快的，因为第一块应是最能满足的；其余排序则要考虑所需块的大小。(2)BEST 策略在 SIZESORT+ 下表现最好，因为这样遍历过去，第一个满足空间大小的就是所求。但是因为 BEST 是找满足大小的最小块，所以实际需要的时间会因为具体的 free list 和所需大小而浮动。只是 SIZESORT+ 是最顺的。(3)WORST 在 SIZESORT-下最好，第一个一定就是满足要求的 (或者再也找不到了)。(4) 如果要考虑合并的话 ADDSORT 可能会是一个好策略。

Question5

(1) 较大的分配需求可能会失败，因为不合并的话最后 free list 是支离破碎的小块。(2) 在有-C 标志的情况下，相邻的空闲块得到合并；能应付（不合并时无法应付的）较大的分配需求。(3) 沿用前面的命令，只修改-n(FIRST 策略)。在允许合并时，每行的 free list 都很简洁；最后剩下一块 90 的大小；不允许合并时，中间出现了分配失败的情况；最后 free list 冗长且每块很小，最后大小是不到 90 的。(4) 这时，列表的顺序只有当采用 FIRST 策略时才会对 free list 有影响。其他策略不受影响。

Question6

当改变 -P 时，高于 50，说明分配操作要比释放多；接近 100，意味着几乎所有操作都是分配操作；接近 0 时，仍近乎是一半分配一半释放，因为只有分配了才能释放。

Question7

(1) 指令：python3 ./malloc.py -c -A +10,+15,+20,+25,+30,-0,-1,-2,-3,-4; 最后 free list 为：[addr:1000 sz:10][addr:1010 sz:15][addr:1025 sz:20][addr:1045 sz:25][addr:1070 sz:30]。

(2) 指令：python3 ./malloc.py -c -A +10,+15,+20,+25,+30,-0,-1,-2,-3,-4 -l SIZESORT+ -C; 最后 free list 为：[addr:1025 sz:20][addr:1000 sz:25][addr:1045 sz:55]。

(3) 指令：python3 ./malloc.py -c -A +10,+15,+20,+25,+30,-0,-1,-2,-3,-4 -l SIZESORT--C; 最后 free list 为：[addr:1070 sz:30][addr:1045 sz:25][addr:1025 sz:20][addr:1010 sz:15][addr:1000 sz:10]。