

照地址排序的首次适配比 LIFO 排序的首次适配有更高的内存利用率，接近最佳适配的利用率。

一般而言，显式链表的缺点是空闲块必须足够大，以包含所有需要的指针，以及头部和可能的脚部。这就导致了更大的最小块大小，也潜在地提高了内部碎片的程度。

9.9.14 分离的空闲链表

就像我们已经看到的，一个使用单向空闲链表的分配器需要与空闲块数量呈线性关系的时间来分配块。一种流行的减少分配时间的方法，通常称为分离存储(segregated storage)，就是维护多个空闲链表，其中每个链表中的块有大致相等的大小。一般的思路是将所有可能的块大小分成一些等价类，也叫做大小类(size class)。有很多种方式来定义大小类。例如，我们可以根据 2 的幂来划分块大小：

$$\{1\}, \{2\}, \{3, 4\}, \{5 \sim 8\}, \dots, \{1025 \sim 2048\}, \{2049 \sim 4096\}, \{4097 \sim \infty\}$$

或者我们可以将小的块分派到它们自己的大小类里，而将大块按照 2 的幂分类：

$$\{1\}, \{2\}, \{3\}, \dots, \{1023\}, \{1024\}, \{1025 \sim 2048\}, \{2049 \sim 4096\}, \{4097 \sim \infty\}$$

分配器维护着一个空闲链表数组，每个大小类一个空闲链表，按照大小的升序排列。当分配器需要一个大小为 n 的块时，它就搜索相应的空闲链表。如果不能找到合适的块与之匹配，它就搜索下一个链表，以此类推。

有关动态内存分配的文献描述了几十种分离存储方法，主要的区别在于它们如何定义大小类，何时进行合并，何时向操作系统请求额外的堆内存，是否允许分割，等等。为了使你大致了解有哪些可能性，我们会描述两种基本的方法：简单分离存储(simple segregated storage)和分离适配(segregated fit)。


1. 简单分离存储

使用简单分离存储，每个大小类的空闲链表包含大小相等的块，每个块的大小就是这个大小类中最大元素的大小。例如，如果某个大小类定义为 $\{17 \sim 32\}$ ，那么这个类的空闲链表全由大小为 32 的块组成。

为了分配一个给定大小的块，我们检查相应的空闲链表。如果链表非空，我们简单地分配其中第一块的全部。空闲块是不会分割以满足分配请求的。如果链表为空，分配器就向操作系统请求一个固定大小的额外内存片(通常是页大小的整数倍)，将这个片分成大小相等的块，并将这些块链接起来形成新的空闲链表。要释放一个块，分配器只要简单地将这个块插入到相应的空闲链表的前部。

这种简单的方法有许多优点。分配和释放块都是很快的常数时间操作。而且，每个片中都是大小相等的块，不分割，不合并，这意味着每个块只有很少的内存开销。由于每个片只有大小相同的块，那么一个已分配块的大小就可以从它的地址中推断出来。因为没有合并，所以已分配块的头部就不需要一个已分配/空闲标记。因此已分配块不需要头部，同时因为没有合并，它们也不需要脚部。因为分配和释放操作都是在空闲链表的起始处操作，所以链表只需要是单向的，而不用是双向的。关键点在于，在任何块中都需要的唯一字段是每个空闲块中的一个字的 succ 指针，因此最小块大小就是一个字。

一个显著的缺点是，简单分离存储很容易造成内部和外部碎片。因为空闲块是不会被分割的，所以可能会造成内部碎片。更糟的是，因为不会合并空闲块，所以某些引用模式会引起极多的外部碎片(见练习题 9.10)。

 **练习题 9.10** 描述一个在基于简单分离存储的分配器中会导致严重外部碎片的引用模式。