Report

固定大小的优缺点：

优点：不容易产生碎片。

缺点：malloc 和 free 速度不快，内存使用率较低。

适用场合：对分配速度要求不高，内存较大。

链表实现的优缺点：

优点：free 很快，内存使用率较高。

缺点：malloc 比较慢，容易产生碎片，对小内存分配效率较低。

适用场合：对分配实时性要求不高，内存较小。

哈希表实现的优缺点：

优点；malloc 速度块，空间使用率高。

缺点：free 速度比较慢，比较浪费空间。

适用场合：对分配实时性要求比较高，内存较小。

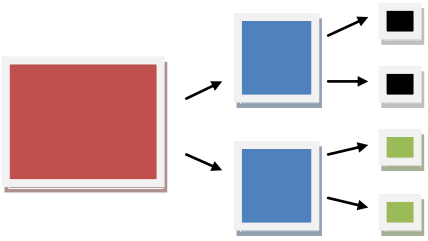
伙伴系统设计：

结合了哈希表，链表，固定大小的一些优点， malloc 和 free 很快，

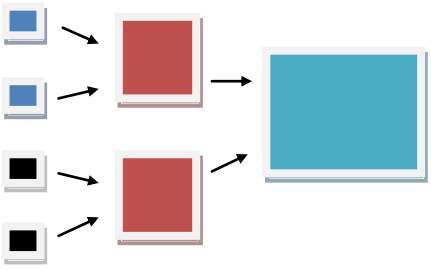
优点：malloc 和 free 都很快，不容易产生碎片。

缺点；内存使用率比固定大小高，比链表低。

思路：内存块的大小是 2 的倍数，它可以被分割成两个同样也是 2 的倍数的内存块。这样内存使用率比固定大小要高，而且不容易产生碎片。



Malloc 过程：寻找空闲块，进行分割直至大小刚好满足需求。



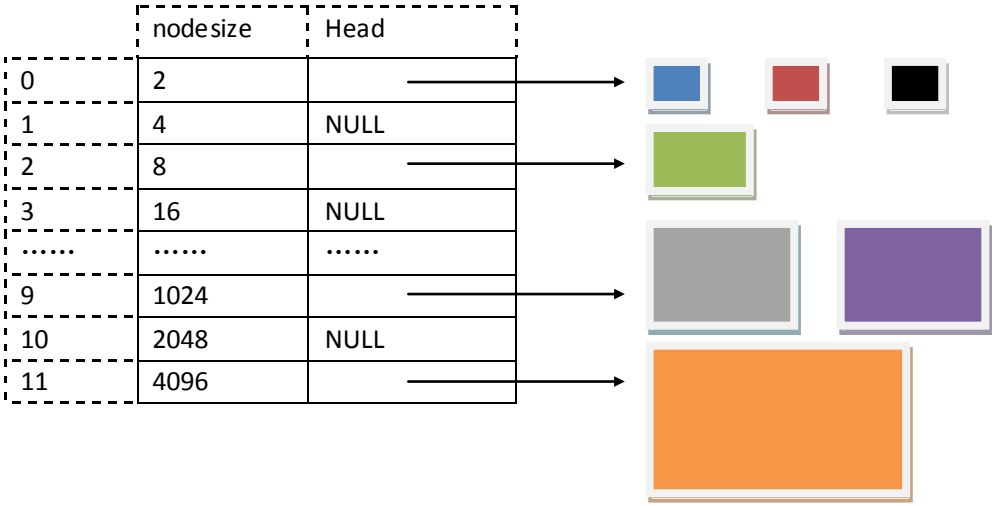
Free 过程：寻找伙伴，但伙伴存在且是空闲块时进行合并，成为更大的空闲块。

伙伴：因需要将空闲内存（2^m）块分割两个相同的内存块（2^{m-1}），它们就是伙伴关系，每个内存块有且只有最多一个伙伴内存块。且他们在物理地址上是相邻的。

Report

类似哈希表的空闲块列表：FreeList [M]

空闲块列表的每个成员包含两个信息，一个是该内存块的大小 nodesize ，和空闲块内存的双向循环链表，若没有则是 NULL。

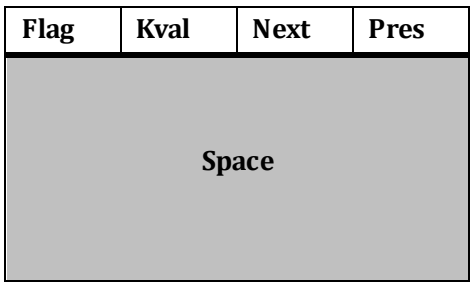


内存块数据结构(Node)：

Flag：标记该内存块是否被使用(used /unused)

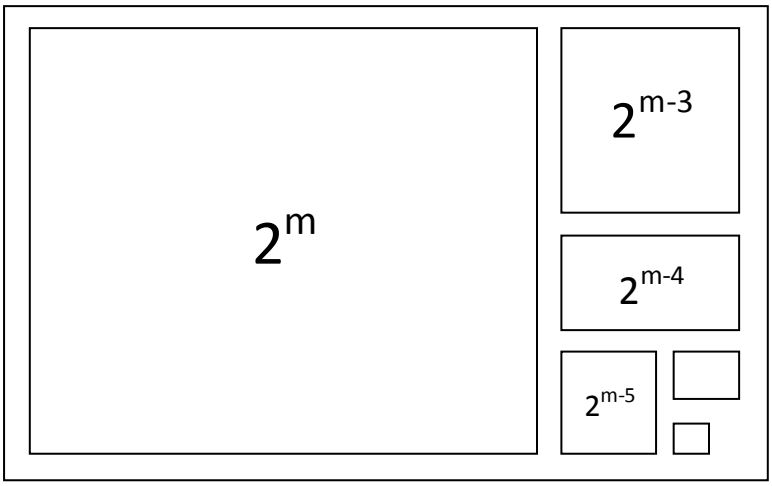
Kval: 该内存块的大小标记，假设该内存块大小是 1024，那么在 FreeList[9]中，所以 Kval 为 9。

Next&Pres：指针指向空闲块，用于建立双向循环链表。



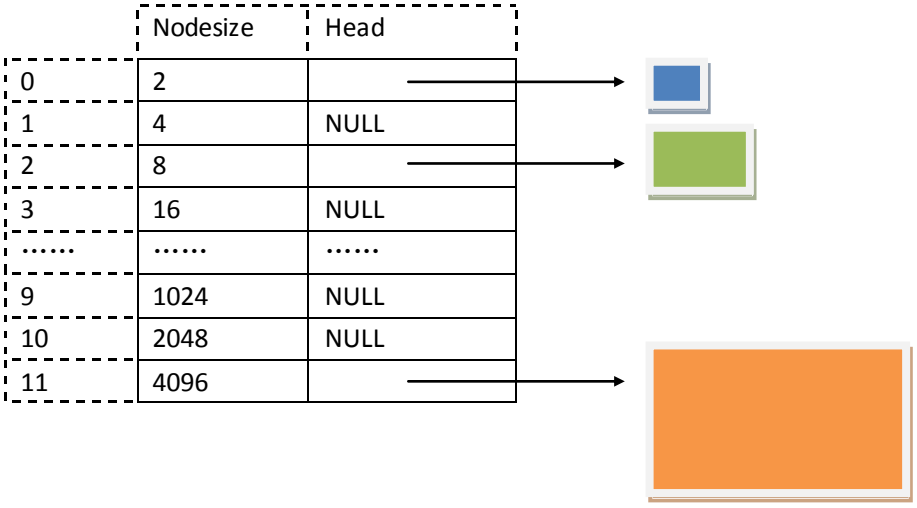
Init 内存初始化：

由于需要管理的内存可能不刚好是 2 的倍数，我们尽量从最大的 2 的倍数进行内存的分割。先分割出最大的 2^m 的内存块插入到对应的 Freelist[M]位置中，然后在剩余的空间中分出最大的 2^{m-3}，直到剩余空间小于 Freelist[0].nodesize 位置。

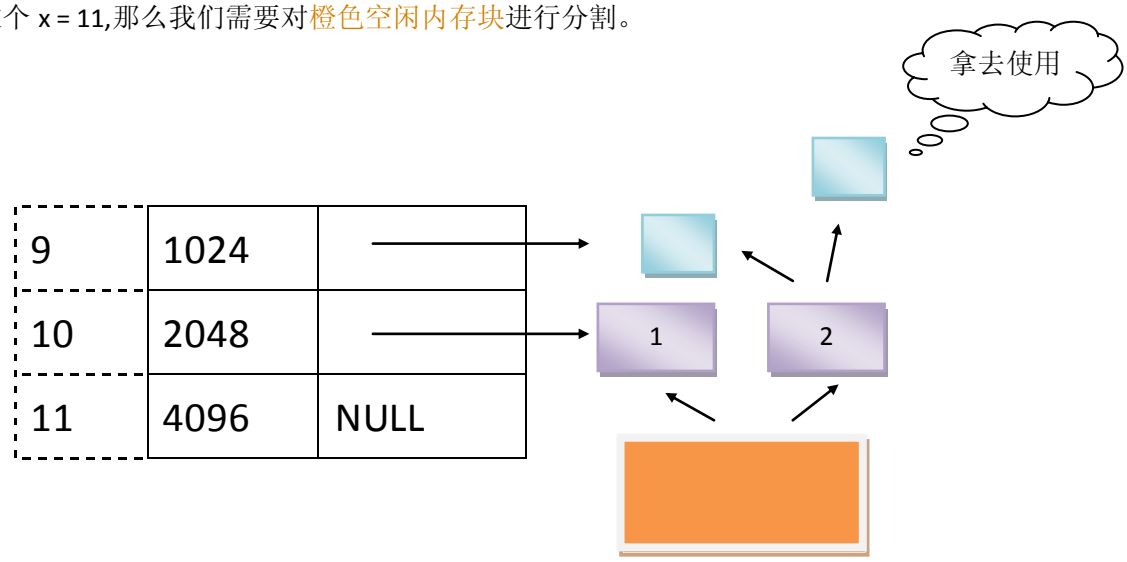


Report

初始化后的 Freelist[M]可能的情况为：
每一个 head 最多有一个空闲块，也可能没有，那么设置为 NULL；



Malloc 算法：
当需要分配内存块大小是 Size=1000 时，现实在 Freelist[0 ~ M].nodesize 中寻找到满足 Freelist[x].nodesize - sizeof(Node)> Size，且存在空闲内存块(Freelist[x].head ! = NULL)。当前这个 x = 11,那么我们需要对橙色空闲内存块进行分割。



由于我们需要的 Size 在 Freelist[9].nodesize 就可以满足，但是由于没有该大小的空闲块。所以寻找到更大的空闲块。当然我们不可能将过大的内存块使用，所以需要进行分割。

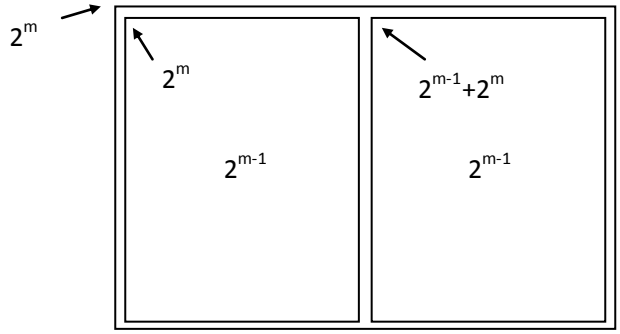
首先删除 Freelist[11]上的橙色空闲内存块地址，分割内存块 1,2，将 1 插入 Freelist[10].head 上(采用头结点插入)，将 2 再次进行分割，直至大小刚好基本满足需求为止。

Report

Free 算法（回收）
回收算法和 Malloc 刚好相反，不断的合并伙伴，直至伙伴不存在或者不可用为止。
计算出伙伴的地址。

假设需要 Free 的空间地址是 p，那么它伙伴的地址计算如下：

$$\mathrm{buddy(p,k)} = \begin{cases} \mathrm{p}+2^{\mathrm{k}} & (\text{当 p mod } 2^{\mathrm{k}+1}=0) \\ \mathrm{p}-2^{\mathrm{k}} & (\text{当 p mod } 2^{\mathrm{k}+1}=2^{\mathrm{k}}) \end{cases}$$



计算出伙伴后，然后判断该伙伴的 kval 和 Flag 是否满足合并的条件，即 kval==p->kval，Flag ==unused，若不满足这两个条件，则将 p 插入到 Freelist[p->kval].head 所指向的链表中(采用头结点插入)，满足条件时进行合并成新的空闲块，并将 Freelist[p->kval].head 指向的链表中删除伙伴，新的空闲块地址是他们其中地址小的那个，kval ++ ，Flag =unused。然后计算这个新空闲块的伙伴。如此循环直至 kval==M 为止。