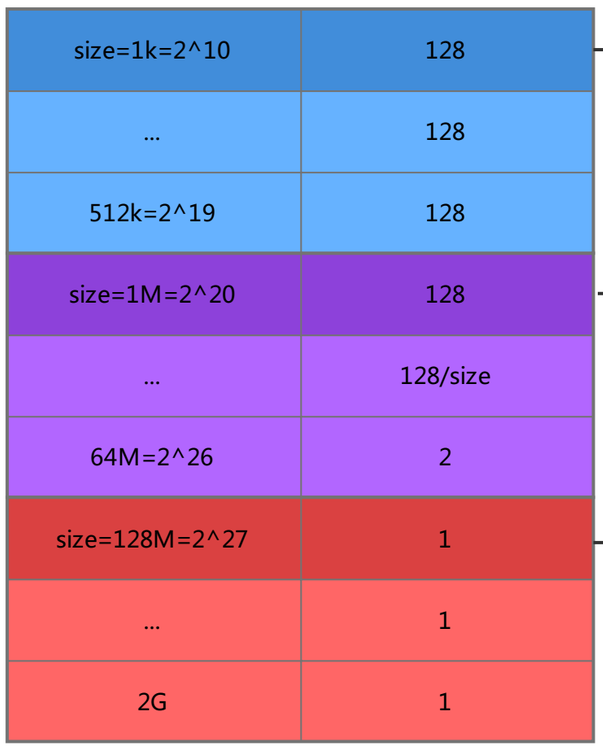
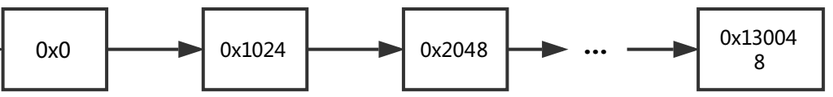
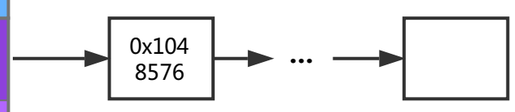
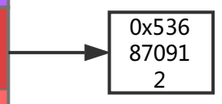
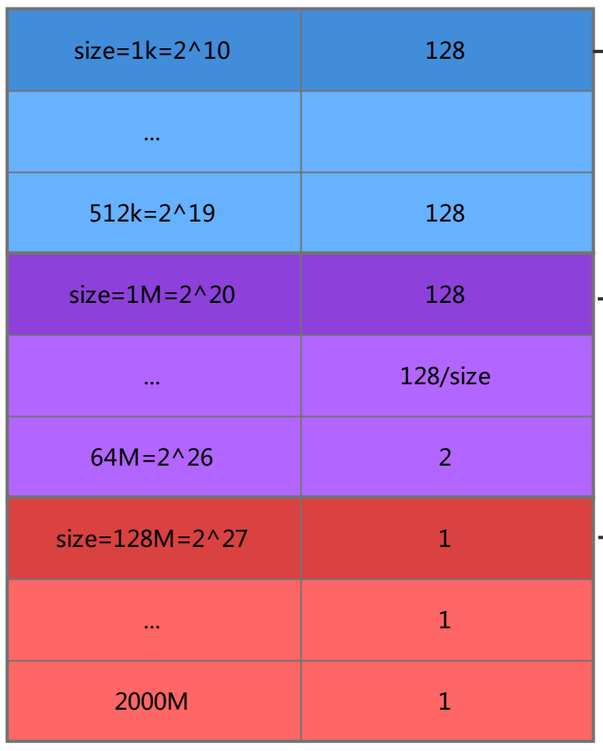


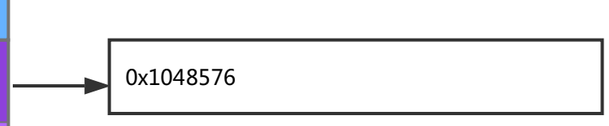


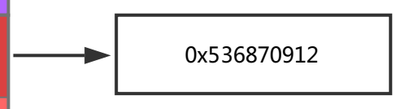




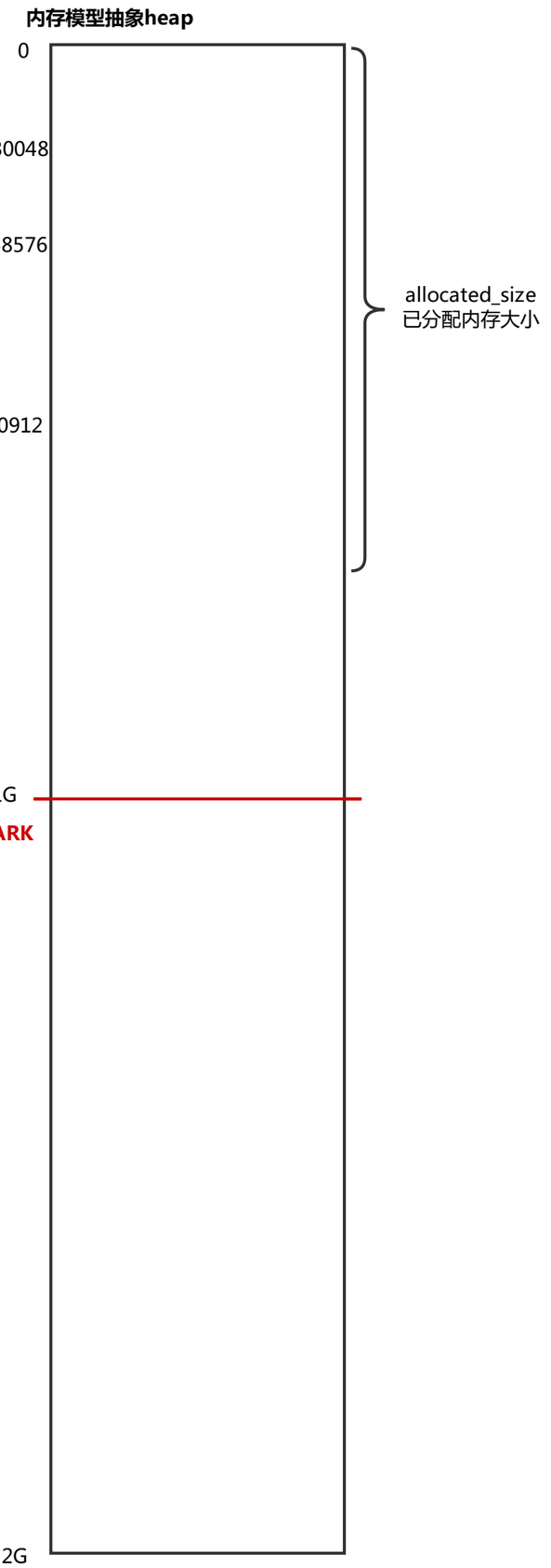
free_map空闲内存表 记录当前空闲的内存块







stub_map存根表 记录所有分配的内存块



当前内存池提供3种粒度内存大小:
1k 2k 4k 8k 16k 32k 64k ... 512k (指数增长)
1M 2M 4M 8M ... 64M
128M ... 2000M

list 采用头部插入,头部删除,避免尾部操作的拷贝.
set 只保存key值,内部可自动去重,当前用来保存内存地址.

初始状态:free_map表表项为空,stub_map表表项为空。
第一次申请内存:假设申请内存大小为9000Byte,实际分配的内存被调整为1024Byte(1k)的整数倍为1M.分别申请一个表项挂载到stub_map和free_map中,根据当前配置策略,从heap空间中new128次1M字节内存,将128个内存地址分别push到stub_map和free_map对应的size(1M)项中.从free_map的size(1M)项中取出一个1M内存节点,提供给用户.从free_map表中删除该节点.

第N次申请内存:假设申请内存大小为500Byte,被调整为1k.假设之前已经申请过1k表项,已存在1k的1链表。若链表还有可用的空闲节点,则直接取出并删除该节点返还给用户.若链表没有空闲的节点,则根据配置策略,从heap空间中申请128次1k字节的内存,全部存储到链表中,并返回一个节点给用户然后删除该节点.

释放内存:1、检测pMem内存大小memSize是否存在于stub_map表中,memSize查表,pMem查set.2、若存在于stub_map中,则查找memSize的free_map表项,将pMem插入该项的list中.3、检查释放内存,若当前memSize的free_map表项的list中节点个数大于同memSize的stub_map表项中节点个数的一半,则释放free_map的memSize表项中空闲节点个数的一半.若当前memSize的free_map表项的list中节点个数所占的总内存超过高水位标记HIGH_WATER_MARK,则释放free_map的memSize表项中空闲节点个数的一半.

申请内存nSize但是allocated_size已分配内存大小 + 被调整后的申请内存大小 * 当前配置策略需要配置的节点个数 > MAX_POOL_SIZE,则需要扩展内存池.第一步遍历两张表,清除free_map中每一个size对应的链表中可清除的内存块,规则同上3,第二步每清除一项free_map同时需要清除stub_map中对应的一项.第三步delete[]释放内存.第四步，调整高水位标记到MAX_POOL_SIZE,防止上述释放内存到HIGH_WATER_MARK位置错误释放内存.第五步,在heap上new nSize调整后的字节数并按照配置策略,申请n次.第六步将N个节点存储到stub_map和free_map中.