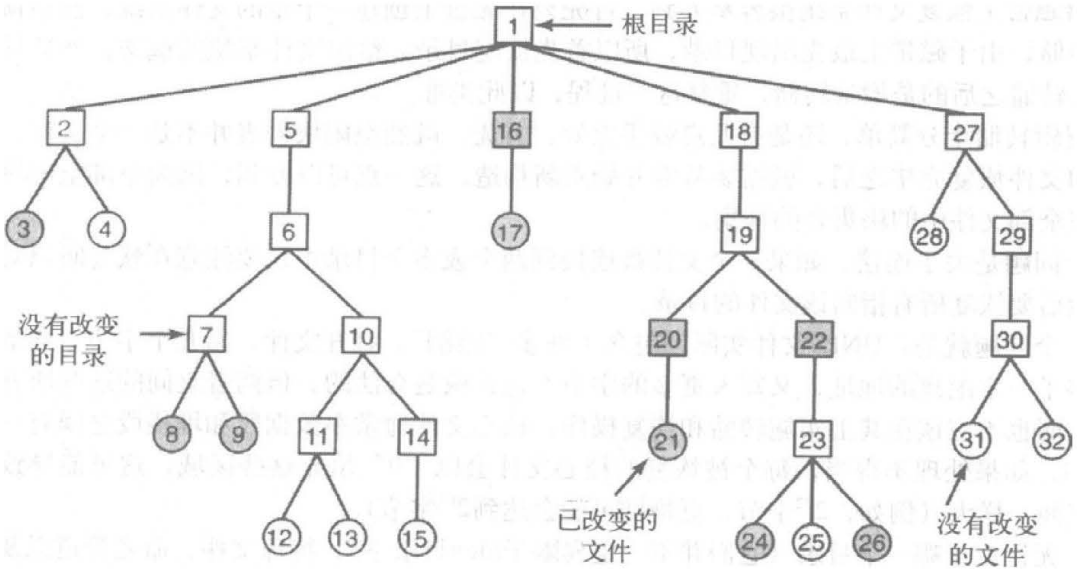

图4-25 待转储的文件系统，其中方框代表目录，圆圈代表文件。被阴影覆盖的项目表示自上次转储以来修改过。每个目录和文件都被标上其i节点号

转储被修改文件之上的未修改目录的第二个原因是为了可以对单个文件进行增量恢复（很可能是对愚蠢操作所损坏文件的恢复）。设想如果星期天晚上转储了整个文件系统，星期一晚上又做了一次增量转储。在星期二，/usr/jhs/proj/nr3目录及其下的全部目录和文件被删除了。星期三一大早用户又想恢复/usr/jhs/proj/nr3/plans/summary文件。但因为没有设置，所以不可能单独恢复summary文件。必须首先恢复nr3和plans这两个目录。为了正确获取文件的所有者、模式、时间等各种信息，这些目录当然必须再次备份到转储磁带上，尽管自上次完整转储以来它们并没有修改过。

逻辑转储算法要维持一个以i节点号为索引的位图，每个i节点包含了几位。随着算法的执行，位图中的这些位会被设置或清除。算法的执行分为四个阶段。第一阶段从起始目录（本例中为根目录）开始检查其中的所有目录项。对每一个修改过的文件，该算法将在位图中标记其i节点。算法还标记并递归检查每一个目录（不管是否修改过）。

第一阶段结束时，所有修改过的文件和全部目录都在位图中标记了，如图4-26a所示（以阴影标记）。理论上说来，第二阶段再次递归地遍历目录树，并去掉目录树中任何不包含被修改过的文件或目录的目录上的标记。本阶段的执行结果如图4-26b所示。注意，i节点号为10、11、14、27、29和30的目录此时已经被去掉标记，因为它们所包含的内容没有做任何修改。它们因而也不会被转储。相反，i节点号为5和6的目录尽管没有被修改过也要被转储，因为到新的机器上恢复当日的修改时需要这些信息。为了提高算法效率，可以将这两阶段的目录树遍历合二为一。

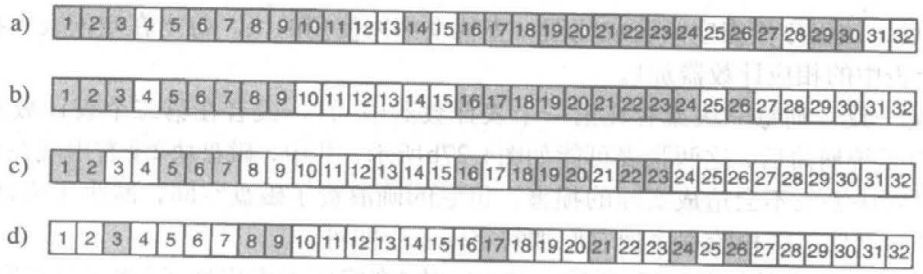

图4-26 逻辑转储算法所使用的位图

现在哪些目录和文件必须被转储已经很明确了，就是图4-26b中所标记的部分。第三阶段算法将以节点号为序，扫描这些i节点并转储所有标记的目录，如图4-26c所示。为了进行恢复，每个被转储的目录都用目录的属性（所有者、时间等）作为前缀。最后，在第四阶段，在图4-26d中被标记的文件也被转储，同样，由其文件属性作为前缀。至此，转储结束。