1995年4月 第25卷 第2期 西北大学学报(自然科学版) Journal of Northwest University(Natural Science Edition) Apr. 1995 Vol. 25 No. 2

超文本结点粒度划分问题的研究*

赵 慧 党华锐 罗景仁

(西北大学计算机科学系,710069,西安太白北路1号;第一作者26岁,女,硕士)

摘 要 讨论了超文本结点的粒度问题和粒度划分的原则,提出了动态增加超文本链的算法,并叙述了一个原型系统的设计与实现。

关键词 超文本;粒度;原型系统

分类号 TP311.56

超文本(Hypertext)是一种新颖的信息表示方法和管理技术。它改以往线性方式组织信息为非线性方式,从而提供用户对信息快速、灵活,而且符合人们阅读习惯的检索和查询方式。同时,用户可以动态地重构信息的链接以组成新的、满足自己需要的阅读文本。由于上述特点,它特别适合于大型软件设计中文档的动态管理,以及文献的管理。因此,近年来超文本技术应用越来越广泛,MS-WINDOWS中的HELP窗口即为一超文本系统。地理信息系统中地图的处理也常采用超文本技术。目前,超文本在INTERNET中也被作为一种传输对象。

人们在实际阅读文本时,通常不是以一章或一个图集为单位,而常常针对一章中的某一句话、某一个词、某一地区图、某一点图进行检索查询。所以,我们在进行文档管理、地图检索管理时,采用超文本技术,同时引入超文本粒度的概念,确定了粒度划分原则,设计了数据结构,并利用C++语言实现了一个原型系统。本文就超文本结点粒度划分问题进行了研究,同时就设计和实现细节进行详细的阐述。

1 超文本基本概念

1.1 超文本是一种新颖的信息表示方法和管理工具

一个超文本系统的基本成分是结点和链,结点用于存储信息,链用于形成不同结点之间的相互关系,其基本特征为:①超文本是以结点为单位,用链联接起来的网状结构;②结点与窗口一一对应,每一结点有一名字或标题,在窗口显示。

1.2 超文本粒度

实现超文本系统时,考虑到人们的阅读习惯,表示信息的结点不能太大,否则,

- 翻阅较慢。
- 在阅读结点内的信息时,仍是线性浏览。

如果粒度太小,则造成

- 结点太多,结点库庞大,难于管理。例如,若超文本中有如图1的存储结构,那么,要得到完整的A信息,需要调用多次B,消耗时间。
- 信息不完整,每个结点所示信息有限,极可能使其含义不完整。

* 陕西省自然科学基金资助课题

收稿日期:1994-06-27

因此,实现超文本系统时,每一个结点表示的信息在含义完整的前提下,应是多大范围,是一完整的章节,还是一个标题、一个图素,亦即超文本的粒度是多大?这是成功地实现超文本系统的关键。

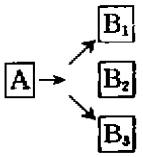


图 1 存储结构图

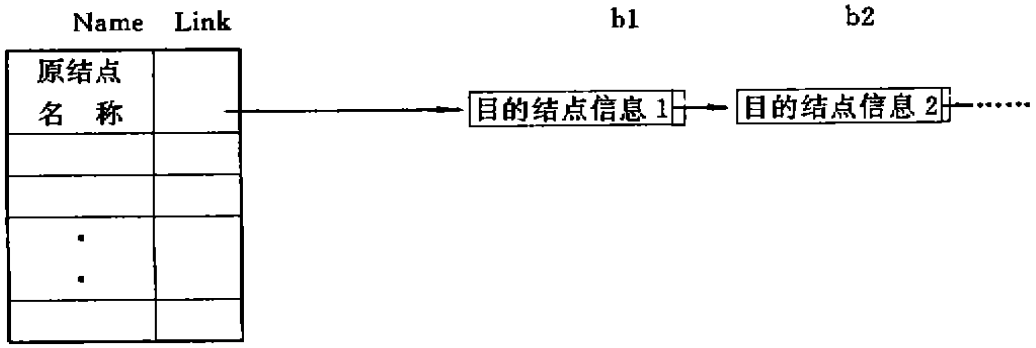
2 超文本粒度的划分

2.1 划分原则

结点为一完整的信息单位,例如书的一个章节、地图集中的一个地区图或一个城区图。结点内可根据阅读需要划分成块,一块可为一段文字,或一个单词。当结点内不划分块时,即默认块为该结点本身。

2.2 数据结构

实现时,我们采用邻接表的存储结构,如下所示:



具体描述采用 C++语言,链结点结构为

```
struct Node{
        char Name _Dest[3];              //目的结点名称
        int Sourse _block _begin;        //源结点索引块的相对起始位置
        int Sourse _block _end;          //源结点索引块的相对终止位置
        int Dest _block _begin;          //目的结点索引块的相对起始位置
        int Dest _block _end;            //目的结点索引块的相对起始位置
        struct node *next;               //指针
    }
struct hyper _link{
        char *Name _Sourse;              //源结点名称
        struct Node *link;               //指针
```

图 2 示例采用上述数据结构可表示为

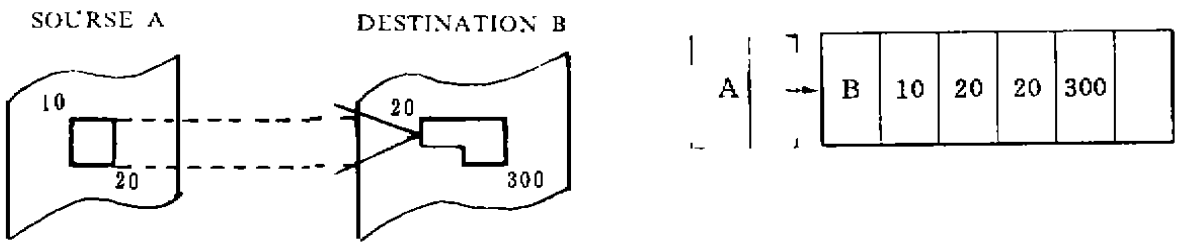


图 2 数据结构示例图

按照上述数据结构,当 Dest _ block _ begin=1 && Dest _ block _ end=EOF 时,目的块即为目的结点的全部,此时,系统与通常的超文本系统完全一致起来。

根据上述的数据结构,我们设计了超文本动态增加文本链和调用的算法。

2.3 算　法

算法 I ,增加超文本链:

(1)确定源结点(设为 A)

(2)确定源结点内的块(利用功能键完成块的定义),记下起始与终止位置 Bs,Es(以字符为单位)

(3)确定目的结点(设为 B)

(4)确定目的结点内的块(同(2)),记下起始与终止位置 Bd,Ed

(5)确认块(Y/N)

Y:做(6)

N:做(1)

(6)生成一新结点 New _ Node,使

New _ node→Name _ Dest=B

New _ Node→Sourse _ block _ begin=Bs

New _ Node→Sourse _ block _ end=Es

New _ Node→Dest _ block _ begin=Bd

New _ Node→Dest _ block _ end=Ed

New _ Node→Next _=NULL

(7)在邻接表中查找 Name=A 的一项,遍历链表,将 New _ Node 插入适当的位置。

算法 I :超文本的使用,设从结点 A 中块 i 指向结点 B 中的块 j(块 j 不止一个)

(1)将结点 A 的文本显示于窗口内

(2)在窗口内移动文本,直至块 i 出现

(3)选中块 i 通过文本属性来判断,例如颜色

(4)计算块 i 首地址在结点 A 中的绝对位置 Bs 和 Es

(5)在链接表中 A 链中检索所有

Sourse _ block _ begin=Bs

Sourse _ block _ end=Es

的结点,根据 Name _ Dest 和 Dest _ block _ begin, Dest _ block _ end 的值依次显示即可。

依据算法 I 和算法 I ,用上述数据结构,我们实现了具有动态增、删、改超文本链功能的原型系统。

3　原型的设计与实现

我们采用面向对象程序设计语言 C++,在 DOS 环境下实现原型系统。利用 C++语言的特点,将超文本中的结点及其对结点的操作,如生成一新结点、插入一条新的链等操作封装成一个类,并利用面向对象语言特有的封装和继承特性,使整个系统更易于动态扩充,代码的效率更高。

原型类定义如下:

```
class Hypertext {
        struct node {
                (同 2.2 处的定义)
                } Node[N];                    //链结点结构
        struct cood {
                int begin;
```

```
                int end;
                    }                          //块首尾位置
    public:                                    //成员函数说明
        struct cood *Def_Block();              //确认块
        struct cood *Def_Node();               //确认结点
        struct *Gen_New_Node();                //生成新结点
        Insert_Node(Sourse, Node);             //插新结点 Node 至 Sourse 链中
        Disp_Node(Node, WIN);                  //显示 Node 信息于 WIN 窗口
        Disp_Block(Block,WIN);                 //显示 Block 信息于 WIN 窗口
        //其他成员函数说明
}
//成员函数描述
以增加超文本链为例,说明实现过程
void Hyper text::Insert_Node()
    {
    Sourse_Node=Def_Node();
    Sourse_Block=Def_Block();
    Dest_Node=Def_Block();
    confirm=getchar();
    if(confirm=="Y")
      {
      New_node=Gen_New_Node();
      for(i=0;i<N;i++)
          if(strcmp(Node[i].name,A)==0)
            //将 New_Node 插入 A 的链中 (代码略)
        }
    }
```

参 考 文 献

1 Conklin Jeff. Hypertext: an introduction and survey. IEEE Computer, 1987(9):17～41

2 Schildt H. Using Turbo C++. California: McGraw-Hill, 1990

The Research of Partitioning Granularity of Node in Hypertext

Zhao Hui Dang Huarui Luo Jingren

(Department of Computer Science, Northwest University, 710069, Xi'an)

Abstract The concept of granularity of node in hypertext and the rules of partitioning the granularity of node are discussed, and a prototype is discribed.

Key words granularity; hypertext; prototype system