数据结构 hw9

刘良宇 PB20000180

6.73

```
Tree InitTree(char* s, int length) {
    // 这里拓展成构造森林的算法
    // 举例一个合理的输入: A(B(E,F),C),B,C
    Tree head = new TreeNode();
    head->data = s[0];
    // 先确定头结点的孩子
    if (length > 1 && s[1] = '(') {
        // 括号内部也是一个森林
        int lbrace = 1, i;
        for (i = 2; lbrace; i++) {
            if (s[i] == '(')
                lbrace++;
            if (s[i] == ')')
                lbrace--;
        }
        // 括号内部从 2 到 i - 2
        // 这里假设输入合法，也就是假设括号内部长度至少为 1
        head->firstChild = InitTree(s + 2, i - 3);
    } else {
        head->firstChild = nullptr;
    }
    // 再确定头结点的兄弟
    head->nextSibling = nullptr;
    for (int i = 0; i < length; i++) {
        if (s[i] == ',') {
            head->nextSibling = InitTree(s + i + 1, length - i - 1);
            break;
        }
    }
    return head;
}
```

7.16

假设有向图

```
Status InsertVex(Graph& G, VexNode v) {
    if (LocateVex(G, v) != -1 || G.vexnum = MAX_VERTEX_NUM)
        return ERROR;
    G.vertices[G.vexnum++] = v;
    return OK:
}

Status InsertArc(Graph& G, VexNode v, VexNode w) {
    int v_index = LocateVex(G, v), w_index = LocateVex(G, w);
    if (v_index == -1 || w_index == -1)
        return ERROR;
```

```
    ArcNode* arc = new ArcNode();
    arc->adjvex = w_index;
    arc->InfoType = nullptr;
    arc->nextarc = G.vertices[v_index]->firstarc;
    G.vertices[v_index]->firstarc = arc;
    return OK;
}

Status DeleteVex(Graph& G, VexNode v) {
    // 先遍历所有顶点，把 v 和 最后一个顶点交换
    // 考虑到这不是本题的重点，交换过程略去
    for (int i = 0; i < G.vexnum - 1; i++) {
        // 这里遍历除了 v 之外的所有顶点
        DeleteArc(G, G.vertices[i], v);
    }
    // 接下来删除 v 的所有边和 v 这个点
    int n = G.vexnum - 1;  // v 点
    while (G.vertices[n]->firstarc) {
        auto temp = G.vertices[n]->firstarc;
        G.vertices[n]->firstarc = G.vertices[n]->firstarc->nextarc;
        delete temp;
        G.arcnum--;
    }
    G.vexnum--;
}

Status DeleteArc(Graph& G, VexNode v, VexNode w) {
    int v_index = LocateVex(G, v), w_index = LocateVex(G, w);
    if (v_index == -1 || w_index == -1)
        return ERROR;
    if (G.vertices[v_index]->firstarc &&
        G.vertices[v_index]->firstarc->adjvex == w_index) { // 可能第一个弧就是
        auto temp = G.vertices[v_index]->firstarc;
        G.vertices[v_index]->firstarc = G.vertices[v_index]->firstarc->nextarc;
        delete temp;
        G.arcnum--;
    } else {
        for (auto p = G.vertices[v_index]->firstarc; p->nextarc;
             p = p->nextarc) {
            if (p->nextarc->adjvex == w_index) {
                auto temp = p->nextarc;
                p->nextarc = p->nextarc->nextarc;
                delete temp;
                G.arcnum--;
                break;
            }
        }
    }
    return OK;
}
```