算法导论　第八周作业　11 月 5 日　周四

PB18151866 龚小航

5.1 使用链表表示和加权合并启发式策略，写出 MAKE－SET，FIND－SET 和 UNION 操作的伪代码。

解：加权合并启发式策略保证每次合并时将较短的链表合并入较长的链表，以减少指针改动次数。

假定之前已经定义了两种结构体：

```
struct NODE{
    struct ROOT *root;
    ELEMENTTYPE  key;
    struct NODE *next;
}
```

```
struct ROOT{
    struct NODE *head;
    struct NODE *tail;
    int  length;
}
```

再写出 MAKE－SET:

```
MAKE-SET(x):
S=(struct ROOT *)malloc(sizeof(struct ROOT));
p=(struct NODE *)malloc(sizeof(struct NODE));
p->root = S;
P->next = NULL;
p->key = x;
S->head = p;
S->tail = p;
```

以下是 FIND－SET 的实现：（已经有待查元素的指针）：

```
FIND-SET(struct NODE *x):
return x->root->head;
```

再写出合并操作：

```
UNION(struct ROOT *p1,struct ROOT *p2):
if(p1->length <= p2->length){
   for each node in p2{
      root = p1;
   }
   p1->tail->next = p2->head;
   p1->tail = p2->tail;
}
else Similar to the one described above
```

5.2 设定动态规划算法求解 $0-1$ 背包问题，要求运行时间为 $O(nW)$，n 为商品数量，W 是小偷能放进背包的最大商品总重量。

解：声明一个$c[n+1][w+1]$ 的二维数组，$c[i][j]$ 表示 在面对第 i 件物品，且背包容量为 j 时所能获得的最大价值 ，那么我们可以很容易分析得出 $c[i][j]$ 的计算方法，

（1）$j < w[i]$ 的情况，这时候背包容量不足以放下第 i 件物品，只能选择不拿

$$c[i][j] = c[i-1][j]$$

（2）$j >= w[i]$ 的情况，这时背包容量可以放下第 i 件物品，我们就要考虑拿这件物品是否能获取更大的价值。

如果拿取，$c[i][j] = c[i-1][j-w[i]] + v[i]$。 这里的 $c[i-1][j-w[i]]$ 指的就是考虑了$i-1$件物品，背包容量为 $j-w[i]$ 时的最大价值，也是相当于为第 i 件物品腾出了$w[i]$ 的空间。

如果不拿，$c[i][j] = c[i-1][j]$ ，同（1）

需要比较这两种情况那种价值最大。

具体函数代码如下：

```
//为使下标与物品标号一致，多开辟一个空间
int v[N+1];           //物品价值
int w[N+1];           //物品重量
int c[N+1][W+1];       //c[i][j]表示在剩余容量为j的情况下，对于物品1~i所能达到的最大价值
int x[N+1]={0};        //用于记载哪些物品被选中

void Dynamic_select(int v[],int w[],int n,int weight){  //求解最大价值,动态选择
    for(int j=0;j<=weight;++j)
        c[0][j]=0;
    for(int i=1;i<=n;++i){
        c[i][0]=0;
        for(int j=1;j<=weight;++j){
            if(w[i]<=j){                                //当还有还有剩余容量时
                if(v[i]+c[i-1][j-w[i]]>c[i-1][j])//此时选择装入物品i
                    c[i][j]=v[i]+c[i-1][j-w[i]];
                else                                    //不装入物品i
                    c[i][j]=c[i-1][j];
            }
            else
                c[i][j]=c[i-1][j];
        }
    }
}
void Showchoise(int c[][W+1],int w[],int weight,int n,int x[]){//选择被选中的物品
    for(int i=n;i>=1;--i){
        if(c[i][weight]==c[i-1][weight]) {  //物品i未被选中
            x[i]=0;
        }
        else{                               //物品i被选中
            x[i]=1;
            weight=weight-w[i];
        }
    }
}
```