选8、链表的概念与基本操作3–链表合并

1、任务：有两个链表a、b，都为具有10个节点长度的随机降序链表，现在要将这链表b合并到链表a，并保持链表a依然可以被降序遍历。

生成两个链表a、b：

```
#生成链表a、b
from random import randint
nodes_a=[]
head_a=-1
nodes_b=[]
head_b=-1
for i in range(10):
    if i==0:
        nodes_a.append([randint(95,100),head_a])
        head_a=0
        nodes_b.append([randint(95,100),head_b])
        head_b=0
    else:
        nodes_a.append([nodes_a[i-1][0]-randint(1,5),nodes_a[i-1][1]])
        nodes_a[i-1][1]=len(nodes_a)-1
        nodes_b.append([nodes_b[i-1][0]-randint(1,5),nodes_b[i-1][1]])
        nodes_b[i-1][1]=len(nodes_b)-1
nodes_a,nodes_b
```

2、链表合并：

基本思想：

遍历链表b，对链表b的每一个节点，用之前学过的插入方法插入到链表a中：

```
#链表合并1
kb=head_b
while kb!=-1:
    ka=head_a
    while ka!=-1:
        if nodes_a[ka][0]>nodes_b[kb][0]:
            qa=ka
            ka=nodes_a[ka][1]
        else:
            if ka==head_a:
                nodes_a.append([nodes_b[kb][0],head_a])
                head_a=len(nodes_a)-1
            else:
                nodes_a.append([nodes_b[kb][0],ka])
                nodes_a[qa][1]=len(nodes_a)-1
            break
    if ka==-1:
        nodes_a.append([nodes_b[kb][0],-1])
        nodes_a[qa][1]=len(nodes_a)-1
    kb=nodes_b[kb][1]
```

但是这样的合并效率并不高，因为每次的插入位置，都需要从链表a的头部开始搜索，考虑降序特性，可以在插入节点后，将qa更新，而ka可以不更新，这样就不用从头部开始搜索了。改进代码如下：

```
#链表合并2
kb=head_b
ka=head_a
while kb!=-1:
    while ka!=-1:
        if nodes_a[ka][0]>nodes_b[kb][0]:
            qa=ka
            ka=nodes_a[ka][1]
        else:
            if ka==head_a:
                nodes_a.append([nodes_b[kb][0],head_a])
                head_a=len(nodes_a)-1
                qa=head_a
            else:
                nodes_a.append([nodes_b[kb][0],ka])
                nodes_a[qa][1]=len(nodes_a)-1
                qa=nodes_a[qa][1]
            break
    if ka==-1:
        nodes_a.append([nodes_b[kb][0],-1])
        nodes_a[qa][1]=len(nodes_a)-1
        qa=nodes_a[qa][1]
    kb=nodes_b[kb][1]
```

当然，书本上p48页的代码，也能实现链表合并，对书本上的每句代码，请写出对应注释。