我保证没有抄袭别人作业

1

记原表为 list

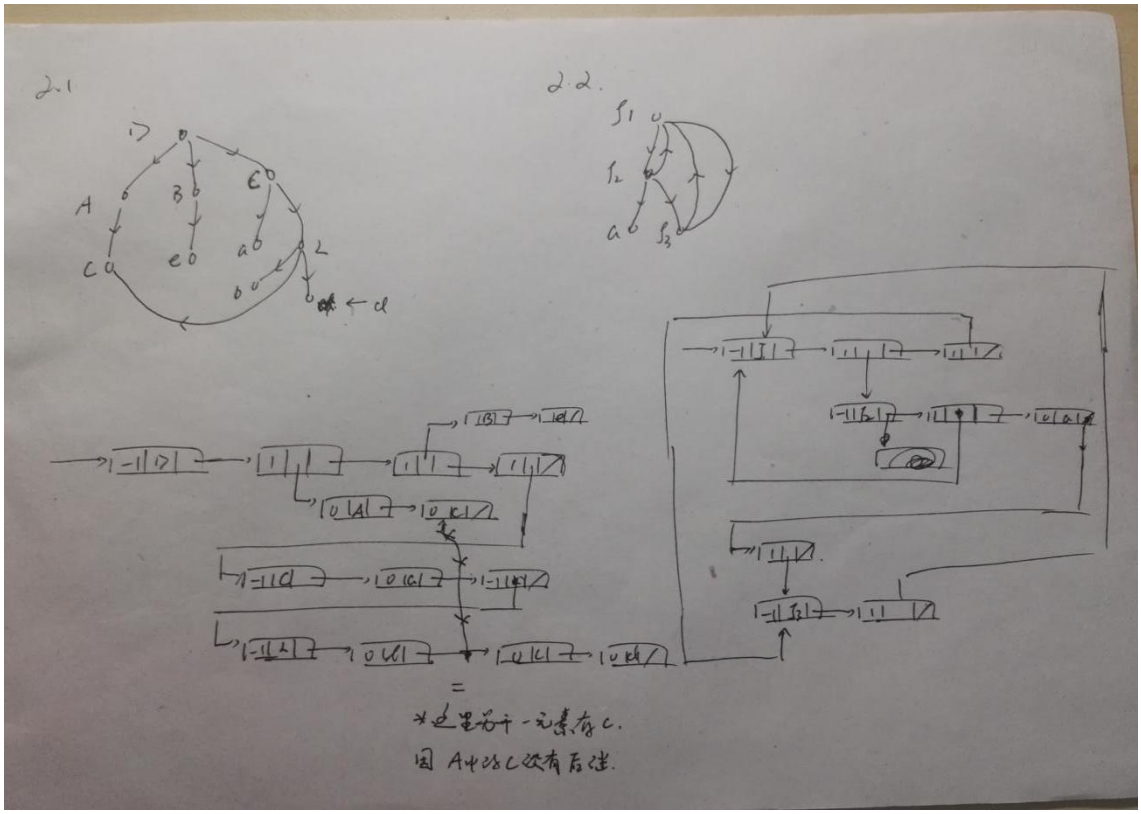
(1) banana = GetHead(GetTail(GetTail(list)))

(2) banana = GetHead(GetHead(GetTail(list)))

(3) banana = GetHead(GetHead(GetTail(GetTail(list))))

(4) banana = GetHead(GetHead(GetTail(GetHead(GetTail(list)))))

2



3

(1) 300 300 300 1500 700:

首次试配	最佳匹配	最差匹配
600 1500 700	900 1500 400	900 1200 700
300 1500 700	900 1500 100	900 900 700
0 1500 700	600 1500 100	600 900 700
0 0 700	600 0 100	无法匹配
0 0 0	无法匹配	

(2) 700 900 1500

首次试配	最佳匹配	最差匹配
200 1500 700	900 1500 0	900 800 700
200 600 700	0 1500 0	0 800 700
无法匹配	0 0 0	无法匹配

(3) 800 900 700 700

首次试配	最佳适配	最差匹配
100 1500 700	100 1500 700	900 700 700
100 600 700	100 600 700	0 700 700
100 600 0	100 600 0	0 0 700
无法匹配	无法匹配	0 0 0

4

将字符串的所有后缀插入一棵初始为空的 Trie 树,则 Trie 树的每一个结点对应一个不同的后缀.同时维护一个计数变量,若增加了一个结点,则对此变量加一.

插入 Trie 树的时间复杂度为 $O(n^2)$,因为每一个字符串的长度为 O(n),每一次插入时间为 O(n)

则算法的时间复杂度为 $O(n^2)$

5

正常删除:

每一次插入节点时间复杂度为 O(l),l 为结点的深度(=O(logk), k = 1,2,...,n),建树时间复杂度为 O(nlogn)

删除一个结点时间复杂度为 O(l),l 为结点的深度,类似的,全部删除时间复杂度为 O(nlogn)

重复 m 次,则总的时间复杂度为 O(mnlogn)

懒惰删除:

注意到此时删除全部结点并不会使树变成空树,

则插入节点时间复杂度为 O(l),l 为结点的深度,建树的时间复杂度为 O(sum(l)) (其中 l = log(i), i = 1, 2,...,nm), 则有时间复杂度为 O(mnlogmn)

类似的, 删除节点的时间复杂度为 O(mnlogmn)

懒惰删除的时间复杂度比正常删除还要多上 O(mnlogm)一项