Homework11

刘喆骐 2020013163 探微化01

35.3-3

算法如下：

遍历所有的子集S，来初始化如下的值：

list[k]:假设k是子集中包含元素最多的子集的大小。对于每个list[i]，其包含的是含有i个未被覆盖元素的子集的集合S_i，使用链表储存。对于每个子集S，都有一个S.num来记录其中尚未被覆盖的元素的个数。

v[n]：元素是否已经被访问。

P[n]：n为总元素个数，记录每个元素在哪些子集中。

max：子集中未被覆盖的元素的最大值。例如子集A中有4个未被覆盖的元素，子集B中为3个，那么max取4.

all：已经覆盖的元素个数。

随后进行循环：

取出并删除list[max]的表尾元素S_{max}，将其加入answer中，ans+=max，若ans=n则结束。若list[max]为空则更新max（减1）。

对于S_{max}中的每一个元素a_i，若其没被访问则更新v[a_i]。对于P[a_i]中的所有集合，把它们从list[num]移动到list[num-1]中，并将它们的num减1。

伪代码如下:

```
GREEDY-SET-COVER(X,F)
    list[k]
    v[n]={0}
    all=0
    max=size
    answer=[]
    for S in F:
        list(S.size).append(S)
        S.num=S.size
        for x in S:
            P[x].append(S)
    while all=!n:
        while not list[max]:
            max-=1
        S_max=list[max][-1]
        answer.append(S_max)
        list[max].delete(S_max)
        all+=max
        for x in S_max:
            if v[x]=0 v[x]=1
            for S in P[x]:
                list[S.num].delete(S)
                S.num-=1
                list[S.num].append(S)
```

时间复杂度的说明：

对于初始化，需要遍历所有的子集中的所有元素，故时间复杂度为$O(\sum|S|)$.

对于循环任务，max从k减小到了1，耗时$O(k)$。假设循环进行了m次(m<n)，那么提取最大的集合耗时$O(m)$。而更新每个元素对应的各种值最终效果等于访问所有的元素一遍，每次耗时$O(1)$，时间复杂度为$O(\sum|S|)$。故总时间复杂度为$O(\sum|S|+m+k)=O(\sum|S|)$。

35.5-5

在MERGE_LISTS的时候，储存如下的结构体，每个结构体有着两个数值num,sum和一个指针p。sum表示所取的数之和；num表示该数是否是加上x[i]得到的，若是，则num=x[i]，否则num=0；p指向sum-num。最后得到z^*时，根据z^*对应的指针一直寻找即可。

```
class data{
    s(num,sum,p):num(num),sum(sum),p(p)
}
merge_list(l,x)
    l1,l2=[]
    for a in l:
        l1.append(s(a.sum,a,0))
        l2.append(s(a.sum+x,a,x))
    i,j=0
    ret=[]
    while i+j<2*l.size():
        if l1[i]==l2[j]:j+=1
        if l1[i]<l2[j] or j==l2.size():
            ret.append(l1[i])
            i+=1
        if l1[i]>l2[j] or i==l2.size():
            ret.append(l2[j])
            j+=1
    return ret

extract-subset-num(S,t)
    n=|S|
    l[0]=[0]
    for i=1 to n:
        l[i]=merge_list(l[i-1],l[i-1]+x[i])
        j=i
        while l[j]>t:
            delete(l[j])
            j-=1
    ans=[]
    l=max_sum(L)
    while(l.p):
        if l.num:
            ans.append(l.num)
    return ans
```