贪心算法作业

2021K8009918003 王静初

第一题 Commando War

思路

对士兵的讲解是顺序进行的，而士兵们的行动是并行的。
因此，从一开始，对士兵讲解所花时间的总和就是确定的，我们假定为m。假设最后一个士兵需要执行的任务耗时k秒，那么总时间至少为m+k秒。
至此，一个贪心的想法产生了：不妨让最后那个士兵执行耗时最短的任务，让总时间的下限尽量低一些。如果存在多个执行时间同样最短的任务，那么选择哪一个都可以：不论怎么选，总讲解时间是相同的，最后一个任务执行完成的时间也是相同的。
用数组brief和exec分别表示讲解和执行的时间，且执行时间最短的任务标号为i。既然已经确定了最后一个任务，假设倒数第二个要执行的任务标号为j，那么可以知道结束讲解这个任务的时间为m-brief[i]，任务完成的时间为m-brief[i]+exec[j].让任务完成的时间尽量短即可。
由此可以看出，只要按执行时间长短排序，使执行时间越长的任务越先执行即可。

伪代码

```
// 为方便排序，使用数据结构
typedef struct{
    brief;
    exec;
}task;

// 排序判断
function cmp(a,b)
return a.exec>b.exec;
end function

// 输入为n个task的数组tasks，输出为最短完成时间
function CommandoWar(n,tasks)
b_time=0;
sort(tasks,tasks+n,cmp);
done_time=INT_MAX;
for i=0 to n-1 do:
    b_time+=tasks[i].brief;
    done_time=max(done_time,b_time+tasks[i].exec);
end for
return done_time;
end function
```

正确性证明

现在tasks数组以按exec从大到小的顺序排序好，并按上述伪代码算出了结果done_time。假设tasks[n-1]不是在最后一个执行：
1.如果$done_time=tasks[n-1].exec+\sum_{i=0}^{n-1}tasks[i].brief$，显然更改位置后的时间会延长。
2.如果$done_time=tasks[j-1].exec+\sum_{i=0}^{j-1}tasks[i].brief$，那么tasks[n-1]必须换到tasks[j]及之前的位置才可能使结果改变。假设换到了位置k（k<=j），这样的话，位置k之前的完成时间不变，位置k后的完成时间增加tasks[n-1].brief。总完成时间延长。
故tasks[n-1]最后一个执行是最优解之一。递归来看tasks[n-2]、tasks[n-3]...可知按排序顺序排列即为最优解之一（最优解可能不唯一）。

算法复杂度

时间复杂度为$O(logn)$：排序需$O(logn)$，计算done_time需$O(n)$.
空间复杂度为$O(n)$：仅需保存tasks数组。如果不算tasks数组的话空间复杂度为$O(1)$.

第二题 DNA Consensus String

思路

似乎也没有什么特别好的办法，直接模拟即可。
对二维数组dna[m][n]，其中每一行表示一组DNA。对dna的每一列，找到重复最多的碱基并记录下来，如果有重复次数相同的，选择字母序较小的那一个。将得到的碱基按序拼接即可得到答案。

伪代码

```
function char2num(c)
if c=='A' then return 0;
if c=='C' then return 1;
if c=='G' then return 2;
else return 3;
end function

function num2str(num)
static const string arr[4]={"A","C","G","T"};
return arr[num];

function DNAConsensusString(dna,m,n) //dna[m][n]
cnt[4]={0};
str="";
dist=0;
for i=0 to n-1 do:
    for j=0 to 3 do:
        cnt[j]=0;
    end for
    max=0;
    min_k=4;
    for j=0 to m-1 do:
        k=char2num(dna[j][i]);
        cnt[k]++;
        if cnt[k]>max then
            max=cnt[k];
            min_k=k;
```

```
        end if
        else if cnt[k]==max&&k<min_k then
            min_k=k;
        end else if
    end for
    str+=num2str(min_k);
    dist+=m-max;
end for
print(str);
print(dist);
end function
```

正确性证明

整个dna序列的汉明距离为每个位置的汉明距离之和。因此，只要能保证每个位置的汉明距离最小，就能使整个dna序列的汉明距离最小。

算法复杂度

时间复杂度为$O(m*n)$：对dna数组遍历一遍即可。
不算dna数组的话，空间复杂度为$O(1)$.
因为至少也得把输入遍历一遍，应该没有比本算法复杂度更低的算法了。

第三题 Opponents

思路

只有对手都出席才可能被打败。每天检查对手是否都来，如果都来了则标记为“失败”，反之为成功。
用双指针保存成功的最长持续时间（长度）。j指针始终向后走一天，如果该天成功则i指针不动，否则i指针指向j指针的后方。

伪代码

```
function Opponents(opnts,m,n)
fail=1;
maxlen=0;
i=j=0;
for j=0 to n-1 do:
    fail=1;
    for k=0 to m-1 do:
        fail&=opnts[k][j]; //有对手不出席则不会被打败
    end for
    if fail==1 then:
        i=j+1; //下一次成功必须在j之后
    end if
    else then:
        maxlen=max(maxlen,j-i+1); //更新长度
    end else
end for
```

```
return maxlen;
end function
```

正确性证明

失败时令i=j+1是为了计算可能的最长时间时i指向成功序列。且，i指向的位置的前一天失败了，所以i指向的位置是成功序列的第一天。不断更新最长成功序列的长度即可得到答案，故而正确。

算法复杂度

时间复杂度为$O(m*n)$，空间复杂度（不算输入）为$O(1)$.

第四题 Minimum Varied Number

思路

要想让答案最小，前面的数字越小越好，位数越少越好。
故，先确定最后一位数字是9，然后让输入数字减去9。如果输入数字小于9则把这个数字放到最后一位，计算结束。接着按8、7、6、5、4、3、2、1的顺序对十位、百位一直到亿位重复此过程直到计算结束。这样算出来的数满足最高位最小且位数最少，所得即为所求。

伪代码

```
function MinimumVariedNumber(num)
dgt=9;
ans=0;
pow=1;
while num>0&&dgt>0 do:
    if num>=dgt then:
        num-=dgt;
        ans+=pow*dgt;
        pow*=10;
        dgt--;
    end if
    else then:
        ans+=pow*num;
        num=0;
    end else
end while
if num>0 then return -1;
return ans;
end function
```

正确性证明

从后往前进行判断，按9、8、7...这个最大的递减序列取数，可以保证剩下的数字和最小。每一位都取到可以取得的最大，使输出结果位数最小。故最终结果是最小的。

算法复杂度

时间复杂度为$O(1)$，因为最多需要执行9次。空间复杂度为$O(1)$.

第五题 Joey Takes Money[^1]

思路

我们都知道，乘积一定时，两个数相差越大其和越大。所以不妨每次交换时把a_i和a_j换成1和a_i*a_j。
则最终的数组为n-1个1和数组累乘之积。

伪代码

```
function JoeyTakesMoney(money,n)
mul=1;
for i=0 to n-1 do:
    mul*=money[i];
end for
return (mul+n-1)*2022;
end function
```

正确性证明

假设我们找到了一组a_i和a_j，使将其替换为1和a_i*a_j后数组最大。
那么接下来，被换成1的数就不需要和别的数一起被选中了，因为1已经是最小值，换数最好的结果就是互换，没有意义。
还剩下n-1个数需要进行换数。重复过程，最终的数组会是n-1个1和数组累乘之积。这时数组的和最大。

算法复杂度

时间复杂度为$O(n)$，空间复杂度（不算输入）为$O(1)$.
[^1]:RIP Chandler Muriel Bing