算法第三次作业

170617 17373126 刘萱

一、

1 最长空位问题 (20 分)

给定一长度为 n 的 01 串 $S = < s_1, s_2, \cdots, s_n >$，你仅有一次机会挑选其中两个元素 $s_i, s_j (1 \le i, j \le n)$ 并交换他们的位置。请你设计算法求出交换之后 S 中最多有几个连续的 0，并分析该算法的时间复杂度。

例如，串 S =“10010101” 通过交换 s_4 和 s_7 可以变为 “10000111”，连续的 0 的数量为 4。

思路：要使S中交换两个元素的位置后，最多连续0的个数发生改变，则这两个被交换的元素必须为0和1。设置一个数组*v*，如果S[*k*] = 0，则*v*[*k*] = -1；如果S[*k*] = 1，则*v*[*k*] 表示S[*k*]前后连续0个数之和。现交换两个元素，这两个元素在原01串S中的下标分别为*i*和*j*,那么S[*i*]随对应的*v*[*i*] = *MAX*{ *v*[*k*] },(1<= *k* <= n)，其中S[k] = 1。

```
Input : 01串S
Output : 交换S中两个元素后S中连续0的最长个数

Let V[n] be a new array,each element is set to 0
sum_pre <- 0;
sun_aft <- 0;
index_pre <- -1;//记录前一个1的下标
max <- 0;//记录MAX{v[k]}
for i<-1 to n do
    if S[i] == 1 then
        if index_pre != -1 then
            V[index_pre] <- sum_pre + sum_aft;
            if V[index_pre] > max then//更新MAX{v[k]}
                max = V[index_pre];
            end
            sum_pre <- sum_aft;
            sum_aft <- 0;
        end
        index_pre <- i;
    end
    else then
        V[i] <- -1;
        sum_aft <- sum_aft + 1;
    end
    if index_pre == n then
        V[index_pre] <- sum_pre;
    end
end
return max + 1;
```

由于算法只需要遍历一遍S串，所以算法的时间复杂度`T(n) = O(n)`.

二、

2 最大收益问题 (20 分)

某公司有一台机器，在每天结束时，该机器产出的收益为 X_1 元。在每天开始时，若当前剩余资金大于等于 U 元，则可以支付 U 元来升级该机器（每天最多只能升级一次）。从升级之日起，该机器每天可以多产出 X_2 元的收益。即是说，在执行 K 次升级之后，这台机器每天的产出为 $X_1 + K \times X_2$ 元。

该公司初始资金为 C 元，请你设计算法求出 n 天之后该公司拥有的总资金的最大值并分析该算法的时间复杂度。

思路：每次升级机器的成本为U，之后从当天起每天多获得的收益为`x2`。那么，当总的收益大于升级所需要的成本时选择升级，否则不升级。设升级之后还可以生产k天，若`x2`*k > U，则选择升级。要使得收益最大，在满足条件且剩余资金大于等于U元时，越早升级所获得的收益越多。

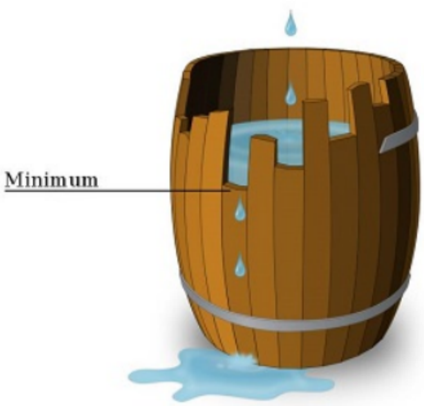
```
Input : X1,X2,U,K,C, n
Output : n天之后公司所拥有的总资金的最大值

sum <- C;//初始化
K <- 0;//升级次数
for i<-n to 1 do
    if i > U/X2 && sum >= U then
        K <- K + 1;
        sum <- sum - U;
    end
    sum <- sum + X1 + K*X2;
end
return sum;
```

时间复杂度`T(n) = O(n)`.

三、

3 水桶问题 (20 分)

给定 $m = n \times k$ 块木板，第 i 块木板的长度为 a_i，现需要用它们围成 n 个水桶，每个水桶使用 k 块木板。第 j 个水桶的体积 V_j 等于其中最短的那块木板的长度。



为了保持水桶的体积均衡，任两个水桶的体积之差不能超过 l。即是说，这 n 个水桶应满足：

$$|V_x - V_y| \le l \qquad \forall 1 \le x, y \le n$$

请你设计算法判断能否围成这样的 n 个水桶并分析该算法的时间复杂度。

思路： 由题意可知，每个木桶的体积大小取决于长度最小的那块木板，判断能否围成符合题意的水桶，即为判断最短的木板与长度第n小的木板之间的长度之差是否小于等于*l*，`|A[n] - Amin| <= l`

`Partition(A,p,r)`

```
Input : An array A waiting to be sorted, the range of index p,r.
Output : Index of the pivot after partition.

x <- A[r];
i <- p-1;
for j <- p to r-1 do
    if A[j] <= x then
        i <- i+1;
        exchange A[i] and A[j];
    end
end
exchange A[i+1] and A[r];
return i + 1;
```

`QuickSort(A,p,r)`

```
Input : An array A waiting to be sorted, the range of index p,r.
Output : Sorted array A.

if p < r then
    q <- Partition(A,p,r);
    QuickSort(A,p,q-1);
    QuickSort(a,q+1,r);
end
return A;
```

```
Input: m块木板的长度集合A, l
Output: 能否围成符合题意的n个木桶

A <- QuickSort(A,0,m);
if  A[n] - A[1] > l then
    return flase;
end
else
    return true;
end
```

快速排序的时间复杂度为`O(mlogm)`,故算法总的时间复杂度为`O(mlogm)`。

四、

4 无向图定向问题 (20 分)

给定一个连通无向图 $G = (V, E)$，满足 $|E| = |V|$。

1. 请证明总是存在一种方法对该无向图的每条边进行定向，使得每个点的出度均为 1。(5 分)
2. 请设计一种算法来完成该定向过程并分析该算法的时间复杂度。(15 分)

1.证明：由$|E| = |v|$可知，图G中存在回路

那么分两种情况进行讨论：

1）：图G中每个点的度均为2，且形成回路，则按照回路，以$v_1->v_2->...->v_n->v_1$的方向，依次对添$e_1,e_2,...e_n$加方向即可构造出每个点出度为1的有向图。

2）：图中存在度数为1的结点，则对每个度数为1的结点v_i，从图中删去结点v_i和v_i所连接的边e_i，此时图G'依然是连通无向图，且满足条件$|E|_{G'}=|V|_{G'}$,重复上述操作，直至G'中不存在度数为1的结点，那么此时G'中每个点的度数均为2，与(1）中情况相同。先对G'执行（1）中的操作，构造出G'对应的有向回路，接着逆序恢复删除的结点，即把度数为1的结点v_k添加到G'中，恢复边e_k，连接v_k在图G中所连接的结点，且以v_k为起点，即v_k是e_k的起点，重复上述步骤，直至恢复到原有的结点数和边数，此时得到G对应的每个点出度均为1的有向图。

综上所述，结论得证。

2.

```
Input : 存有图G的邻接矩阵G[ij]
Output: 定向之后的图G'

for i<-1 to n do
    for j←1 to n do
        G'[i,j] <- G[i,j];
    end
end


for i<-1 to n do
    degree <- 0;
    point <- 0;
    for j <- 0 to n do
        if G[i,j] == 1 then
            degree <- degree + 1;
            point <- j;
        end
    end

    if degree == 1 then
        G'[j,i] <- 0;
        G[i,j] <- 0;
        G[j,i] <- 0;
        judge <- 1;
    end
end

for i<-1 to n do
    for j <- 0 to n do
        if G[i,j] == 1 then
            G[j,i] <- 0;
            G'[j,i] <- 0;
            break;
        end
    end
end

return G';
```

算法时间复杂度为$T(n)=O(n^2)$.

五、

5 最短路径问题 (20 分)

在二维平面上有 n 个点，第 i 个点的坐标为 $(x_i, y_i)(1 \le i \le n)$。从第 i 个点到第 j 个点的距离为这两点的曼哈顿距离：$d(i,j) = |x_i - x_j| + |y_i - y_j|$。此外，这 n 个点中某些点为传送点，用 $t[i]$ 来表示第 $i(1 \le i \le n)$ 个点是否为传送点。$t[i]$ 为 0 表示该点不是传送点，$t[i]$ 为 1 表示该点为传送点。任意两个传送点之间的距离均为 0。请设计算法求出在此情况下从点 x 到点 y 的最短路径长度并分析该算法的时间复杂度。

分两种情况进行讨论：

设从点x到点y的路径经过n个传送点

1） $n <= 1$:

此时，两点之间的距离即为他们之间的曼哈顿距离$d(i,j) = |x_i - x_j| + |y_i - y_j|$，不存在传送点之间的传送，两点之间的最短距离即为$d(i,j)$;

2) $n > 1$:

找到x和y分别距离最近的传送点x',y',两点之间最短距离为$d(x,x') + d(y,y')$

> `Input: T[n]`, x , y
>
> `Output: 最短路径长度min_distance`
>
> $min_{xx'} \leftarrow \infty$;
>
> $min_{yy'} \leftarrow \infty$;
>
> `for i<-1 to n do`
>
> `if t[i] == 1 then`
>
> $d_x = |x_x - x_i| + |y_x - y_i|$;
>
> $d_y = |x_i - x_y| + |y_i - y_y|$;
>
> $if\ d_x < min_{xx'}\ then$
>
> $min_{xx'} \leftarrow d_x$;
>
> end
>
> $if\ d_y < min_{yy'}\ then$
>
> $min_{yy'} \leftarrow d_y$;
>
> end
>
> `end`
>
> `end`
>
> $d_{xy} = |x_x - x_y| + |y_x - y_y|$;
>
> $if\ d_{xy} < min_{xx'} + min_{yy'}\ then$
>
> $min\_distance \leftarrow d_{xy}$;
>
> end

> $else$
>
> $min\_distance \leftarrow min_{xx'} + min_{yy'};$
>
> end
>
> $return\ min\_distance;$

时间复杂度$T(n) = O(n)$.

讨论同学：刘丽君