算法第二次作业

17373126 刘萱

1.二进制串变换问题

给定两个长度均为 n 的仅由 0 和 1 组成的字符串 a 和 b，你可以对串 a 进行如下操作：

1. 对任意 $i, j (1 \le i, j \le n)$，交换 a_i 和 a_j，操作代价为 $|i - j|$；
2. 对任意 $i (1 \le i \le n)$，取反 a_i，操作代价为 1；

请你设计算法计算将串 a 变为串 b 所需的最小代价（只能对串 a 进行操作），并分析该算法的时间复杂度。

思路： 分析这两种操作，发现只有当`ai`与`aj`相邻的时候，操作1的代价才会与操作2的代价相当，否则执行交换操作的代价都会大于执行取反操作，故我们只需要考虑**相邻两个字符交换及单字符直接取反**两种情况的代价计算即可。

`Transform(n,a,b)`

```
Input : Two strings of length n consisting of only 0 and 1 a,b.
Output :The minimum cost of converting a to b.

v[n] <- {0};
record <- 0;
for i<-0 to n-1 do
    if a[i]==b[i] contune;
    else if a[i] == b[i+1] and b[i] == a[i+1] then
        record <- record + 1;
        i <- i + 1;
    end
    else record <- record + 1;
end
return record;
```

算法时间复杂度为：`T(n) = O(n)`.

2.最长递增子序列问题

递增子序列是指：从原序列中按顺序挑选出某些元素组成一个新序列，并且该新序列中的任意一个元素均大于该元素之前的所有元素。例如，对于序列 $< 5, 24, 8, 17, 12, 45 >$，该序列的两个递增子序列为 $< 5, 8, 12, 45 >$ 和 $< 5, 8, 17, 45 >$，并且可以验证它们也是原序列最长的递增子序列。请设计算法来求出一个包含 n 个元素的序列 $A =< a_1, a_2, \cdots, a_n >$ 中的最长递增子序列，并分析该算法的时间复杂度。

思路：先对给定序列进行排序，使其成为一个增序的序列，然后求原序列与排序后序列的最大公共子序列。

`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;
```

`Longest-Common-Subsequence(X,Y)`

```
Input: Two sequence X,Y.
Output: Longest common subsequence of X and Y.

n <- length(X);
Let d[0..n,o..n] and p[o..n,o..n] be two new w-dimension arrays;
for i<-0 to n do
    d[i,0] <- 0;
    d[0,j] <- 0;
end
for i<-1 to n do
    for j<-1 to n do
        if X[i]==Y[j] then
            d[i,j] <- d[i-1][j-1] + 1;
            p[i,j] <- "LU";
        end
        else if d[i-1,j] >= d[i,j-1] then
            d[i,j] <- d[i-1,j];
            p[i,j] <- "U";
        end
        else
            d[i,j] <- d[i,j-1];
            p[i,j] <- "L";
        end
    end
end
return d,p;
```

`Print-LCS(p,X,i,j)`

```
Input : Array p generated from Longest-Common-Subsequence, string X, index i and
j.
Output: Output the longest common subsequence of X[1..i] and Y[1..j].

if i == 0 or j == 0 then
    return NULL;
end
if p[i,j] == "LU" then
    Print-LCS(p,X,i-1,j-1);
    print x[i];
end
else if p[i,j] == "U" then
    Print-LCS(p,X,i-1,j);
end
else
    Print-LCS(p,X,i,j-1);
end
```

`Solve(n,A)`

```
Input : Array A, the length of the array n.
Output : Output the longeset increasing subsequence.

B <- QuickSort(A,0,n-1);
d,p <- Longest-Common-Subsequence(A,B);
Print-LCS(p,A,n,n);
```

快速排序时间复杂度为O(`nlogn`),求`LSP`时间复杂度为O(`n²`)

算法总的时间复杂度 `T(n) = O(n²)`

3.括号匹配问题

定义**合法的括号串**如下：

1. 空串是合法的括号串；
2. 若串 s 是合法的，则 (s) 和 $[s]$ 也是合法的；
3. 若串 a, b 均是合法的，则 ab 也是合法的。

现在给定由 ‘[’，‘]’ 和 ‘(’，‘)’ 构成的字符串，请你设计算法计算该串中合法的子序列的最大长度，并分析该算法时间复杂度。例如字符串 “([[(])])”，最长的合法子序列 “([()])” 长度为 6。

思路： 求原括号串的最长合法子序列，可以转化为先求最少添加n个符号可使得原括号串转化为合法序列，那么可得原括号串有n个括号是不匹配的，则最长合法子序列的长度即为原括号串长度`s-n`.

`match(a,b)`

```
Input : char a,char b.
Output : the result wheather a match b.

if a=='(' and b==')' then
    return 1;
end
if a=='[' and b==']' then
    return 1;
end
return 0;
```

`min(a,b)`

```
Input : int a, int b.
Output : the smaller of a and b.

if a<b then
    return a;
end
return b;
```

`Solve(str)`

```
Input : A string only consists of '('')''['']' str
Output : The length of the longest resultant subsequence

len <- str.length;
dp[len][len] <- {0};
for i<-0 to len-1 do
    dp[i,i] <- 1;
end
for k<-1 to len-1 do
    for i<-0 to len-k-1 do
        j <- i+k;
        dp[i,j] = INT_MAX;
        if match(str[i],str[j]) then
            dp[i,j] <- dp[i+1,j-1];
        end
        for x<- i to j-1 do
            dp[i,j] <- min(dp[i,j], dp[i,x]+dp[x+1,j]);
        end
    end
end
return len - dp[0,len-1];
```

算法的时间复杂度为 `T(n) = O(n²)`

4.分组可行性判定问题

给定按非降序排列的 n 个数 $a_1, a_2, \ldots, a_n$。现需将这 n 个数分组，满足：

1. 每个数 a_i 仅属于一个组；
2. 每个组中包含至少 k 个数；
3. 对属于同一组的任意两个元素 a_i, a_j，需满足 $|a_i - a_j| \le d$。

请你设计算法判断是否可以将给定的 n 个数按照上述要求分组，并分析该算法的时间复杂度。

(此题与赵梁煊、刘丽君讨论，感谢大佬)

思路： 使用一个**一维数组v[n]**来记录前n个数是否可以按照要求分组，可以设为1，不可以设为0

判断A[1]到A[i]是否可以按照要求分组，可以将问题拆分为分别判断`A[1]`到`A[m]`和`A[m+1]`到`A[i]`是否可以按要求分组（`0 < m < i-k`），若`m=0`，则认为不存在`A[1]到A[0]`段，故初始化`v[0] = 1`.

若存在m使得`v[m] == 1`并且`A[i] - A[m+1] <= d`，则令`v[i]=1`，否则`v[i]=0`.

序列可按照要求分组的充要条件是，`v[n] == 1`.

`Judge(n,A)`

```
Input : An array A of length n.
Output : Determine the feasibility of grouping.
Let v[0...n] be a new array(each element is set to false)
v[0] <- true;
for i <- 1 to n do
    if i < k then
        continue;
    end
    for m <- 0 to i-k do
        if v[j]==true and A[i]-A[m+1]<=d then
            v[i] <- true;
            break;
        end
    end
end
return v[n];
```

算法的时间复杂度为：`T(n) = O(n²)`.

5.最大分值问题

给定一个包含 n 个整数的序列 $a_1, a_2, \ldots, a_n$，对其中任意一段连续区间 $a_i..a_j$，其分值为

$$(\sum_{t=i}^{j} a_t)\%p$$

符号 % 表示取余运算符。

现请你设计算法计算将其分为 k 段 (每段至少包含 1 个元素) 后分值和的最大值，并分析该算法的时间复杂度。

例如，将 $3, 4, 7, 2$ 分为 3 段，模数为 $p = 10$，则可将其分为 $(3, 4), (7), (2)$ 这三段，其分值和为 $(3 + 4)\%10 + 7\%10 + 2\%10 = 16$。

`MaxSum(n,A,k,p)`

```
Input : A sequence a with n integers A,k segment,divisor p.
```

```
Output : The max Sum.

Let sum[n] be a new array.
Let molmax[n][n] be a new array.
sum[1] <- A[1];
for i<-2 to n do
    sum[i] <- sum[i-1] + A[i];
end
for i<-1 to n do
    molmax[i][1] = sum[i] % p;
end
for i<-2 to n do
    for j<-2 to min(i,k) do
        for m<-j-1 to i-1 do
            temp <- molmax[m,j-1] + (((sum[i] - sum[m]) % p) + p) % p;
            if temp > molmax[i,j] then
                molmax[i,j] <- temp;
            end
        end
    end
end
return molmax[n,k];
```

算法总时间复杂度为：`T(n) = O(n²)`.

讨论同学：赵梁煊，刘丽君，陈新月