

目录

1	无重复排序	2
1.1	本关任务	2
1.2	解法	2
2	最短路	3
2.1	本关任务	3
2.2	解法	3
2.3	分析与证明	4
2.4	样例输入与输出	4
3	最大括号距离	5
3.1	本关任务	5
3.2	解法	5
3.3	分析与证明	6
3.4	样例输入与输出	6
4	括号匹配	6
4.1	本关任务	6
4.2	解法	6
4.3	分析与证明	6
4.4	样例输入与输出	6
5	高精度整数	6
5.1	本关任务	6
5.2	解法	7
5.3	分析与证明	7
5.4	样例输入与输出	7
6	无重复排序	7
6.1	本关任务	7
6.2	解法	7

1 无重复排序

```

1  val N = getInt();
2  val A = getIntTable(N);
3
4  fun merge([], ys) = ys : int list
5    | merge(xs, []) = xs
6    | merge(x::xs, y::ys) =
7      if x < y then x::merge(xs, y::ys)
8      else y::merge(x::xs, ys);
9
10 fun mergesort xs =
11   let fun sort(0, xs) = ([], xs)
12       | sort(1, x::xs) = ([x], xs)
13       | sort(n, xs) =
14         let val (l1, xs1) = sort((n+1) div 2, xs)
15         val (l2, xs2) = sort(n div 2, xs1)
16         in (merge (l1,l2), xs2)
17         end
18       val (l, _) = sort (length xs, xs)
19   in l end;
20
21 val B = mergesort(A);
22 printIntTable(B);

```

1.1 本关任务

给出一个具有 N 个互不相同元素的数组，请对它进行升序排序。

1.2 解法

我们可以采用归并排序算法解决。将数据分为即复杂度为

$$W = O(n \log n) \quad S = O(\log^2 n)$$

Algorithm 1: MergeSort

```

1 MergeSort( $\langle a_n \rangle$ ) =
2 if  $|a_n| = 1$  then
3    $a_1$ 
4 else
5   let
6      $L \leftarrow \text{MergeSort}(\langle a_1, a_2, \dots, a_{(n+1)/2} \rangle);$ 
7      $R \leftarrow \text{MergeSort}(\langle a_{(n+1)/2}, \dots, a_n \rangle);$ 
8   in
9      $\text{Merge}(L, R)$ 
10  end
11 end

```

2 最短路

2.1 本关任务

给定一个带权无向图，一个源点，权值在边上。计算从源点到其他各点的最短路径。

2.2 解法

可以采用 Dijkstra 算法解决该问题。

利用二维数组保存两点之间的路径长度，即 $A[i][j]$ 即是编号为 $i-1$ 和 $j-1$ 点之间的路径长度。若无给定长度，则用 ∞ 来表示无法两点之间无法到达。创建一个一维数组 dis ，用以储存源点到每个点的距离。并且集合 B 为中只有源点 a 。

(1) 在 dis 数组中寻找一个长度最短的值 $\text{dis}[k]$ ，并且保证 k 点不在集合 B 中。

(2) 再对每个不属于 B 集合中的点进行长度收缩：如果 $\text{dis}[i]$ 的值 $\text{dis}[k] + A[k][i]$ 的大小，就将 $\text{dis}[i]$ 的值更新为 $\text{dis}[k] + A[k][i]$ 。将 k 也加入集合 B 中。

(3) 重复上述步骤，直至集合 B 中有所有点。得到的数组 dis 的值就是源点到所有对应点的最短路径值。

在 SML 实际编写程序中，我用 99999 来替代 ∞ ，在比较长度的过程中也方便运算。在最后输出最短路之前，再将所有长度为 99999 的数据转化成 1 说明没有此路径。

2.3 分析与证明

Dijkstra 算法使用的迭代不超过 $n - 1$ 次，在每次迭代的运算中，即比较与更新一共不超过 $2(n - 1)$ 次，因此 $W = O(n^2)$ 。每一次的迭代都不能并行运算，但是在每一次的迭代中的比较与收缩长度操作可以同时进行，因此 $S = O(n) + 1$ 。

2.4 样例输入与输出

样例输入：

1	7	11	5
2	2	4	2
3	1	4	3
4	7	2	2
5	3	4	3
6	5	7	5
7	7	3	3
8	6	1	1
9	6	3	4
10	2	4	3
11	5	6	3
12	7	2	1

样例输出：

1	4	6	7	7	0	3	5
---	---	---	---	---	---	---	---

3 最大括号距离

3.1 本关任务

我们定义一个串 s 是闭合的，当且仅当它由 '(' 和 ')' 构成，而且满足以下条件之一：

空串：这个串为空

连接：这个串由两个闭合的串连接构成

匹配：这个串是一个闭合的串外面包裹一个括号构成的

现在给你一个串，你需要找出所有这个串中匹配的子串（一个闭合的串，并且外侧由括号包裹）中最长的那个，输出它的长度。

3.2 解法

现在采用一个 `stack` 函数来储存括号的栈。

当栈为空时，判断下一个要入栈的括号的类型：如果是右括号，则必定不会有与之匹配的括号，直接丢弃该右括号；如果是左括号，就将左括号压栈。并在每次栈为空时更新当前最大的匹配括号长度。

当栈不为空时，如果栈顶元素是左括号，即将压栈的括号是右括号，就直接弹出栈顶元素，并增加当前括号距离 2。否则将新元素压栈。

由于这种解法只能解决括号串整体匹配和右括号多于左括号的情况，因此需要将原来的括号串每个元素置换并颠倒，得到新的最大匹配长度不变但括号冗余情况完全相反的括号串。两个串同时经过 `stack` 函数的处理，得到的最小的值就是括号匹配的最大的距离。

关键函数 `stack` 的代码如下：

```
1 fun stack([], _, maxpair, tmppair) = Int.max(maxpair, tmppair)
2 | stack(x::xs, [], maxpair, tmppair) =
3     if (x=0)
4         then stack(xs, [x], Int.max(maxpair, tmppair), 0)
5         else stack(xs, [], Int.max(maxpair, tmppair), 0)
6 | stack(x::xs, y::ys, maxpair, tmppair) =
7     if ((x=1) andalso (y=0))
8         then stack(xs, ys, maxpair, tmppair+2)
```

```
9 | else stack(xs, x::y::ys, maxpair, tmppair);
```

3.3 分析与证明

字符串元素的置换和反转都需要 $O(n)$ 的 W 和 $O(1)$ 、 $O(n)$ 的 S 。

stack 函数只需要扫描整个 list 一遍, 因此串行需要的 W 是 $O(n)$, S 也是 $O(n)$ 。

3.4 样例输入与输出

样例输入:

```
5010101100101111001011101010101110101110101
101111000
```

样例输出:

6

4 括号匹配

4.1 本关任务

给定一个括号序列, 判断它是否是匹配的。注意 $()()$ 在本题也当做匹配处理。

4.2 解法

4.3 分析与证明

4.4 样例输入与输出

5 高精度整数

5.1 本关任务

给定两个任意精度的整数 a 和 b , 满足 $a \geq b$, 求出 $a+b$, $a-b$, $a \times b$ 的值。

Algorithm 2: Addition

```

1 Addition([ ], [ ], c) = c
2 if  $|a_n| = 1$  then
3    $a_1$ 
4 else
5   let
6      $L \leftarrow \text{MergeSort}(\langle a_1, a_2, \dots, a_{(n+1)/2} \rangle)$ ;
7      $R \leftarrow \text{MergeSort}(\langle a_{(n+1)/2}, \dots, a_n \rangle)$ ;
8   in
9     Addition( $L, R$ )
10  end
11 end

```

5.2 解法**5.3 分析与证明****5.4 样例输入与输出****6 无重复排序****6.1 本关任务**

给出一个具有 N 个互不相同元素的数组，请对它进行升序排序。

6.2 解法

我们可以采用归并排序算法解决，即
复杂度为

$$W = O(n \log n) \quad S = O(\log^2 n)$$

Algorithm 3: MergeSort

```
1 MergeSort( $\langle a_n \rangle$ ) =  
2 if  $|a_n| = 1$  then  
3    $a_1$   
4 else  
5   let  
6      $L \leftarrow \text{MergeSort}(\langle a_1, a_2, \dots, a_{(n+1)/2} \rangle);$   
7      $R \leftarrow \text{MergeSort}(\langle a_{(n+1)/2}, \dots, a_n \rangle);$   
8   in  
9      $\text{Merge}(L, R)$   
10  end  
11 end
```
