NOIp Senior Day1 Solution

zhzh2001

1 阶乘

1.1 直接计算

用 64 位整数、双精度浮点数或扩展精度浮点数保存结果，时间复杂度 $O(n)$。

期望得分：5/15/30

1.2 高精度

直接用高精度计算，时间复杂度大约 $O(n^2)$，用压位可以加速。

期望得分：50(使用 huge 模板)~65(使用 gmp 库)

1.3 计算对数

使用`log10` 函数即可计算出 $n!$ 以 10 为底的对数，取整作为指数，小数部分作为尾数 (用`pow` 计算)，时间复杂度 $O(n)$。

然而即使用扩展精度浮点数也有误差。

期望得分：71

1.4 正解

其实也很简单，由于只要保留 k 位有效数字，后面很多数字显然都没有用。考虑仍然用浮点数保存结果，但是这次，每次算完后都判断结果是否大于一个特定值 (10^k)，如果是则除以这个值，并累加计数器。经过实验，这种方法基本上是正确的 (用高精度浮点数库`mpfr` 对拍)。时间复杂度 $O(n)$。

然而，用双精度浮点数保存结果精度不够，只能得到 60 分。应该用扩展精度浮点数保存才能得到 95 分。最后一个点需要分段打表。

1.5 使用 lgamma 函数计算

gamma 函数定义为 $\Gamma n = (n-1)!$，有一个函数lgamma 可以计算gamma 函数的自然对数，换底之后就是答案了。但是即使用扩展精度浮点数精度也不够，要用四精度浮点数，在 GCC 中为__float128。但是四精度的数学运算需要特殊的quadmath.h 数学库，而且在连接时要加上-lquadmath 选项。唯一的方法是把源代码复制到代码中，但是这样代码就超过长度限制了，需要进行精简。用这种方法的标程，有大约 18KB 长。时间复杂度是 $O(1)$ 的。

1.6 总结

这题是我的原创题，方法也比较多。但是数据范围可能比较神奇。

主要考察了创新能力和分段打表。

2 激光和镜子

2.1 思路

在这个问题中，我们想要把激光从源点发射到终点。激光可以从水平或竖直方向开始，并且镜子可以放置在特定位置来改变激光的方向。我们想要放置最少的镜子来到达终点。

我们可以发现在任何激光的最优路径中，任意一条水平或竖直的直线，激光最多只会覆盖直线上连续的一段。如果激光覆盖了分离的两段，那么我们可以跳过中间转弯的部分，并且找到一种更优的路径。

因此，激光最多只能在 $2N+2$ 条直线上行进————$N+1$ 条水平直线和 $N+1$ 条竖直直线，与源点和 N 个可以放镜子的点对应。

于是我们可以把这个问题转化为最短路问题。我们想要到达一条经过终点的水平或竖直的直线，并且我们从一条经过源点的水平或竖直的直线出发。我们可以在两条直线间切换，当且仅当这两条直线的交点是给定的 N 个点之一，并且我们想要最小化切换的次数。

2.2 技巧

由于坐标范围很大，我们需要把坐标离散化后再处理。

另外，由于所有的边权都为 1，可以用bfs 来实现最短路，而不用Dijkstra，而且时间复杂度更优，最短路部分为 $O(N)$。排序的常数比Dijkstra 小多了。

2.3 总结

本题来自USACO16DEC Gold T3:Lasers and Mirrors

主要考察了图的建立以及最短路，其中把点转为边的思想比较巧妙。

3 干草堆猜测

3.1 思路

我们可以发现，如果有一种方案满足问题 $1 \ldots M$，那么同一种方案也满足问题 $1 \ldots M-1$。所以我们可以二分答案，并把问题转化为确定一组问题是否能满足的判定性问题。

3.2 所有的 A 互不相同

考虑出现矛盾的充要条件，设当前问题为 Q_l, Q_r, A，如果 $Q_l \ldots Q_r$ 间的所有草堆的答案都大于 A，那么存在矛盾，因为 A 为 $\mathrm{RMQ}(Q_l, Q_r)$ 的上界。而如果满足条件，那么其他位置都可以放置任意值而不造成任何问题。把问题按照 A 降序排序，对于每个问题先判断再染色，用数据结构维护区间染色即可。

3.3 所有的 A 相同

只要所有问题的区间有交就可以满足条件，反之显然草堆的数量重复出现，不符合题意。

3.4 所有情况

对于 A 相同的问题一起处理，先判断区间交是否被完全染色，再染色区间并。用并查集实现区间染色 (当然线段树也可以)，一次判定的时间复杂度为 $O(N + Q \log Q)$。实际上只要开始时排序，判定时取出符合条件的问题即可，总时间复杂度 $O(N \log Q)$。

官方的方法有些不同，时间复杂度是 $O(N \log Q + Q \log^2 Q)$，无法通过数据加强。

3.5 并查集维护区间染色

维护 $N+1$ 个点的并查集 $0 \ldots N$，每个点的父亲表示从这个点出发染色段的左端，初始时 $f[i] = i$。利用这些信息很容易实现查询。

算法 1 可能因为递归的`Root` 而栈溢出，为了避免这个问题，还有另一种算法 2。

很明显，这里不能用按秩合并，只能用路径压缩。因此理论时间复杂度与线段树相同，但是常数小，且实现简单。可以做`codevs1191` 来练习。

算法 1

```
procedure PAINT(f, L, R)                    ▷ f 为并查集，[L, R] 为染色区间
    while ROOT(L − 1) ≠ ROOT(R) do
        f[ROOT(R)] ← ROOT(L − 1)
    end while
end procedure
```

算法 2

```
procedure PAINT(f, L, R)
    while L ≤ R do
        if ROOT(R) = R then
            f[R] ← ROOT(L − 1)
            R ← R − 1
        else
            R ← ROOT(R)
        end if
    end while
end procedure
```

3.6 总结

本题来自`USACO08JAN Gold T1:Haybale Guessing`