The 14th BUPT Campus Programming Contest

tutorial

BUPT ICPC TEAM

A. Divisible

计算输入数字模 31 的值，判断是否为 0。

```
std::string s;
std::cin >> s;
int r = 0;
for (char c : s)
  ((r *= 10) += c - '0') %= 31;
```

K. Binary Tree

两个人在树上删子树，每次只能删满二叉树，不能删的人输。

K. Binary Tree

两个人在树上删子树，每次只能删满二叉树，不能删的人输。
满二叉树的大小都是奇数，每次有一个人删都会少奇数个点。

K. Binary Tree

两个人在树上删子树，每次只能删满二叉树，不能删的人输。
满二叉树的大小都是奇数，每次有一个人删都会少奇数个点。
如果是总共偶数个点，那么先手输；否则后手输。

L. Strong

简单动态规划，用 $f(i)$ 表示前 i 个数字最多分出多少段，则

$$f(i) = \max\left\{f(i-1), \max\{f(j)+1 \mid l \le a_{j+1} + a_{j+2} + ... + a_i \le r\}\right\}$$

```
for (int i = 1; i <= n; ++i) {
    long long x = 0;
    f[i] = f[i - 1];
    for (int j = i; j >= 1; --j) {
        x += a[j];
        if (l <= x && x <= r) f[i] = std::max(f[i], f[j - 1] + 1);
    }
}
```

L. Strong

题目等价于，选出尽可能多的满足所要求条件的不相交的区间。

这又是一个经典的贪心问题，写出来的代码也很简单。

```
int ans = 0, last = 0;
for (int i = 1, j; i <= n; ++i) {
    for (long long sum = a[j = i];
            j >= 1 && !(l <= sum && sum <= r);
            sum += a[--j]);
    if (j > last) {
        ans++;
        last = i;
    }
}
```

L. Strong

两种做法都可以优化到 $O(n \log n)$。

D. Polygon

能构成多边形的条件是最长的边长度不超过剩下所有边长之和。考虑计算不符合条件的概率。

设线段长度是 1，那么要求的就是最长边长度超过 $\frac{1}{2}$ 的概率。设最长边长度为 x。

D. Polygon

假设最长边最靠左，也就是最靠左的点到左端点的距离为 x，那么这个点有 n 种选择，剩下的点都得在剩下的 $1-x$ 的范围内。这部分的概率为 $n(1-x)^{n-1}$。最长边靠右同理。

那么这类情况的概率为

$$\int_{\frac{1}{2}}^{1} \mathrm{d}x \cdot 2n(1-x)^{n-1} = \frac{1}{2^{n-1}}$$

D. Polygon

假设最长边最靠左，也就是最靠左的点到左端点的距离为 x，那么这个点有 n 种选择，剩下的点都得在剩下的 $1-x$ 的范围内。这部分的概率为 $n(1-x)^{n-1}$。最长边靠右同理。

那么这类情况的概率为

$$\int_{\frac{1}{2}}^{1} \mathrm{d}x \cdot 2n(1-x)^{n-1} = \frac{1}{2^{n-1}}$$

D. Polygon

假设最长边不是靠边的，那么最长边两端的点的编号有 $n(n-1)$ 种选择。设最长边的左端距离区间左端点的距离为 y，那么这部分的概率就是

$$\int_{\frac{1}{2}}^{1} \mathrm{d}x \int_{0}^{1-x} \mathrm{d}y \cdot n(n-1)(1-x)^{n-2} = \frac{n-1}{2^n}$$

D. Polygon

假设最长边不是靠边的，那么最长边两端的点的编号有 $n(n-1)$ 种选择。设最长边的左端距离区间左端点的距离为 y，那么这部分的概率就是

$$\int_{\frac{1}{2}}^{1} \mathrm{d}x \int_{0}^{1-x} \mathrm{d}y \cdot n(n-1)(1-x)^{n-2} = \frac{n-1}{2^n}$$

总和为 $\frac{n+1}{2^n}$，用 1 减一下就是答案。

D. Polygon

假设最长边不是靠边的，那么最长边两端的点的编号有 $n(n-1)$ 种选择。设最长边的左端距离区间左端点的距离为 y，那么这部分的概率就是

$$\int_{\frac{1}{2}}^{1} \mathrm{d}x \int_{0}^{1-x} \mathrm{d}y \cdot n(n-1)(1-x)^{n-2} = \frac{n-1}{2^n}$$

总和为 $\frac{n+1}{2^n}$，用 1 减一下就是答案。

$$[0.75, 0.5, 0.3125, 0.1875, ...]$$

H. Watch Dogs: Legion

等等党永不为奴！

H. Watch Dogs: Legion

等等党永不为奴！

考虑我们会做什么样的决策：对于第 i 天，如果不打折就肯定不买，因为总是可以最后以原价买；如果打折的话，什么情况下买？

H. Watch Dogs: Legion

等等党永不为奴！

考虑我们会做什么样的决策：对于第 i 天，如果不打折就肯定不买，因为总是可以最后以原价买；如果打折的话，什么情况下买？

现在是否买，取决于后面有没有可能会更香。假设后面的所有天期望得到的最香价格是 c，而现在打折的概率是 p，折后的价格为 x，

H. Watch Dogs: Legion

等等党永不为奴！

考虑我们会做什么样的决策：对于第 i 天，如果不打折就肯定不买，因为总是可以最后以原价买；如果打折的话，什么情况下买？

现在是否买，取决于后面有没有可能会更香。假设后面的所有天期望得到的最香价格是 c，而现在打折的概率是 p，折后的价格为 x，如果打折之后没有后面期望的价格香，那就算打折也不买；否则打折就买。

H. Watch Dogs: Legion

等等党永不为奴！
考虑我们会做什么样的决策：对于第 i 天，如果不打折就肯定不买，因为总是可以最后以原价买；如果打折的话，什么情况下买？
现在是否买，取决于后面有没有可能会更香。假设后面的所有天期望得到的最香价格是 c，而现在打折的概率是 p，折后的价格为 x，如果打折之后没有后面期望的价格香，那就算打折也不买；否则打折就买。

$$c = \min\{c, px + (1-p)c\}$$

E. Draw

枚举在哪条边上找圆心，然后以剩下的所有顶点为圆心画半径为 r 的圆，会在枚举的这条边上交出一个区间。这条边上被最多区间覆盖的点就是圆心在这条边时的答案。

对于所有的边的答案取 $\max$。

F. Running

经过的点一定构成一棵树。对于每个点，考虑方案是以这个点为 LCA 的情况。

F. Running

经过的点一定构成一棵树。对于每个点，考虑方案是以这个点为 LCA 的情况。
设 $f[i][j][0/1/2]$ 表示 i 的子树中经过 j 个点，回到 i/有一条链没有回到 i/有两条链没有回到 i 的最小代价，分别对应着路径没有端点、一个端点、两个端点在 i 的子树内。

F. Running

经过的点一定构成一棵树。对于每个点，考虑方案是以这个点为 LCA 的情况。
设 $f[i][j][0/1/2]$ 表示 i 的子树中经过 j 个点，回到 i/有一条链没有回到 i/有两条链没有回到 i 的最小代价，分别对应着路径没有端点、一个端点、两个端点在 i 的子树内。
转移就是一个比较简单的背包合并。

C. Treasure

建立一个并查集，中间包含 $n+m$ 个点，表示每一行和每一列。
一旦 (i_0, j_0) 处有宝藏，就把并查集中表示第 i_0 行的点和表示第 j_0 列的点并起来。
最终，对于每个位置 (i, j)，如果 i 和 j 处在同一个集合中，(i, j) 处就有宝藏。
这样就可以把完整的宝藏图造出来，然后就是一个简单的 dp 了。

B. Concatenation

建立后缀数组。

B. Concatenation

建立后缀数组。
设与后缀 l_0 的 LCP 大于 $r_0 - l_0 + 1$ 的区间为 $[b_0, t_0]$，与后缀 l_1 的 LCP 大于 $r_1 - l_1 + 1$ 的区间为 $[b_1, t_1]$。

B. Concatenation

建立后缀数组。
设与后缀 l_0 的 LCP 大于 $r_0 - l_0 + 1$ 的区间为 $[b_0, t_0]$，与后缀 l_1 的 LCP 大于 $r_1 - l_1 + 1$ 的区间为 $[b_1, t_1]$。
在 $[b_0, t_0]$ 中会有一个子区间 $[b, t]$ 使得这个后缀去掉前 $r_0 - l_0 + 1$ 个字符之后得到的后缀落在区间 $[b_1, t_1]$ 之内，可以二分得到。区间的长度 $t - b + 1$ 就是答案。

I. Subarray Sum

垃圾出题人

将区间分成两半 X_L 和 X_R，分治求解，对于两边求出 (A_L, B_L) 和 (A_R, B_R) 数组。

$$A_{Lk} = \sum_{b \text{ is a subarray of } A_L} [|b| = k] \sum_{i=1}^{|b|} i \times b_i$$

$$B_{Lk} = \sum_{b \text{ is a subarray of } A_L} [|b| = k] \sum_{i=1}^{|b|} b_i$$

I. Subarray Sum

$$A_k = \sum_{i=0}^{n_L} \sum_{j=0}^{n_R} [i+j=k] \left(\binom{n_R}{j} A_i + \binom{n_L}{i} (A_j + i \times B_j) \right)$$

$$B_k = \sum_{i=0}^{n_L} \sum_{j=0}^{n_R} [i+j=k] \left(\binom{n_R}{j} B_i + \binom{n_L}{i} B_j \right)$$

I. Subarray Sum

设数列为 a. 以 $n = 5, i = 3$ 为例, 计算每个选择方案下 a_1 到 a_5 每个数字的系数, 将所有方案下系数之和记为该位置的数字对答案的贡献.

a_1	a_2	a_3	a_4	a_5
1	2	3		
1	2		3	
1	2			3
1		2	3	
1		2		3
1			2	3
	1	2	3	
	1	2		3
	1		2	3
		1	2	3
—	—	—	—	—
6	9	12	15	18

I. Subarray Sum

可以看出这是个等差数列, 下面分析等差数列是如何形成的, 以及如何对每个不同的 n, i 求得该等差数列.

首先看首项, 对于 a_1 来说, 如果 a_1 出现在最后选择的子序列中, 其有且仅有 1 的贡献.

而 a_1 出现的次数, 等价于从剩余 $(n-1)$ 个数字中选择 $(i-1)$ 个数字的方案数, 即 $\binom{n-1}{i-1}$.

I. Subarray Sum

再看公差, 即分析相邻两项 a_i 与 a_{i+1} 的贡献之差, 可以根据这两项是否出现在子序列中分四种情况讨论.

1. 二者都没有出现, 则二者贡献都为 0
2. 二者均出现, 此时 a_{i+1} 的贡献一定比 a_i 大 1
3. a_i 出现,a_{i+1} 没有出现.
4. a_{i+1} 出现,a_i 没有出现.

其中 3,4 两种情况下, 二者最终的贡献是相同的.

I. Subarray Sum

证明:
对于每一个仅有 a_i 出现的子序列 $\{a_{b_1}, a_{b_2}, ..., a_{b_{k-1}\neq i}, a_{b_k=i}, a_{b_{k+1}\neq i+1}...a_{b_m}\}$, 一定存在唯一一个子序列 $\{a_{b_1}, a_{b_2}, ..., a_{b_{k-1}\neq i}, a_{b_k=i+1}, a_{b_{k+1}\neq i+1}...a_{b_m}\}$, 其中 a_i 没有出现, 只有 a_{i+1} 出现, 而二者都在子序列中排第 k 位, 因此二者的贡献是等额的.
综上, 当且仅当 a_i 与 a_{i+1} 同时出现在子序列中时, 后者的贡献比前者大 1. 因此贡献差等价于二者同时出现的次数, 即从剩余 $n-2$ 个数中选择 $i-2$ 个数的方案数, 其值为 $\binom{n-2}{i-2}$.
一旦得到每个等差数列, 可以通过后缀和与后缀和的后缀和来 $O(1)$ 计算答案, 整体复杂度 $O(n)$.

G. Colored Graph

将每条边按照两个顶点编号的最大值排序，然后按照这个顺序构造 Kruscal 重构树。那么，一个点经过编号小于某个限制能经过的点就对应了重构树中的一棵子树。子树对应着 DFS 的区间，问题就变成了序列上的区间待修改颜色数的问题，可以用莫队解决。

J. Summation

$$
\begin{aligned}
&[1 \le x_1 \le n][1 \le x_2 \le n][1 \le y_1 \le m][1 \le y_2 \le m] \\
=&([x_1 < x_2] + [x_2 < x_1] + [x_1 = x_2])([y_1 < y_2] + [y_2 < y_1] + [y_1 = y_2]) \\
=&[x_1 < x_2][y_1 < y_2] + [x_1 < x_2][y_2 < y_1] + [x_2 < x_1][y_1 < y_2] + [x_2 < x_1][y_2 < y_1] \\
+&[x_1 < x_2][y_1 = y_2] + [x_2 < x_1][y_1 = y_2] + [x_1 = x_2][y_1 < y_2] + [x_1 = x_2][y_2 < y_1] \\
+&[x_1 = x_2][y_1 = y_2] \\
=&2[x_1 < x_2][y_1 < y_2] + 2[x_1 > x_2][y_1 < y_2] \\
+&2([x_1 < x_2][y_1 = y_2] + [x_1 = x_2][y_1 < y_2]) + [x_1 = x_2][y_1 = y_2]
\end{aligned}
$$

J. Summation

$$\sum_{T} S_2(\lfloor \frac{n}{T} \rfloor) S_2(\lfloor \frac{m}{T} \rfloor) \sum_{d|T} d^4 \mu(d) (\frac{T}{d})^3 - \frac{nm(n+1)(m+1)}{2}$$

$$+\frac{1}{2} \sum_{T} \left(S_2(\frac{n}{T}) S_2(\frac{m}{T}) \sum_{d|T} d^4 \mu(d) (\frac{T}{d})^3 - (S_1(\frac{n}{T}) S_2(\frac{m}{T}) + S_2(\frac{n}{T}) S_1(\frac{m}{T})) \sum_{d|T} d^3 \mu(d) (\frac{T}{d})^2 + S_1(\frac{n}{T}) S_1(\frac{m}{T}) \sum_{d|T} d^2 \mu(d) (\frac{T}{d}) \right)$$