选择题解析

第 1 题 A

本题考查解线性方程组。会 CRT（中国剩余定理）可以加速计算，但是直接枚举也很快。

若使用枚举法，只需要分别计算 $k \cdot 17 + 9$ 除以 7 是否余 6 即可。答案是 111。

第 2 题 D

A. `set`：集合，每个元素只能出现一次，无法完成。

B. `multiset`：多重集合，但是 `count()` 函数是 $O(\log |S| +$ 出现次数$)$，复杂度不对。

C. `priority_queue`：优先队列，甚至不支持随机访问，无法完成。

D. `map`：映射，可以规定 `cnt[x]` 为 `x` 的出现次数，可以稳定但此操作 $O(\log |S|)$。

第 3 题 D

本题考查 Linux 操作系统的使用。背诵即可。

第 4 题 D

本题考查欧拉路径的概念。

一张 100 个结点的简单无向图存在欧拉路径，则图上至多有 () 条边。

A. 4950 B. 4900 C. 99 D. 前三个选项都不对

一个无向连通图存在欧拉路径（注意和“是欧拉图”区分），当且仅当度数为奇数的点有 0 或 2 个。

100 个结点的完全图有 4950 条边。每次拆掉一条边，可以减少至多 2 个度数为奇数的点，至少要拆掉 49 条边才能让度数为奇数的点剩下至多 2 个，从而答案是 4901。

第 5 题 A

本题考查 `bitset` 的使用。

可以把 `bitset` 看成一个二进制数，从低到高第 `i` 个二进制位（0-index）就是 `a[i]`。

依次模拟每一步：

`bitset<8> a: 00000000`

`a[4]=1: 00010000`

`a.flip(): 11101111`

`a.flip(2): 11101011`

此时 `a<<3` 是 `01011000`

`a&=a<<3` 是 `01001000`

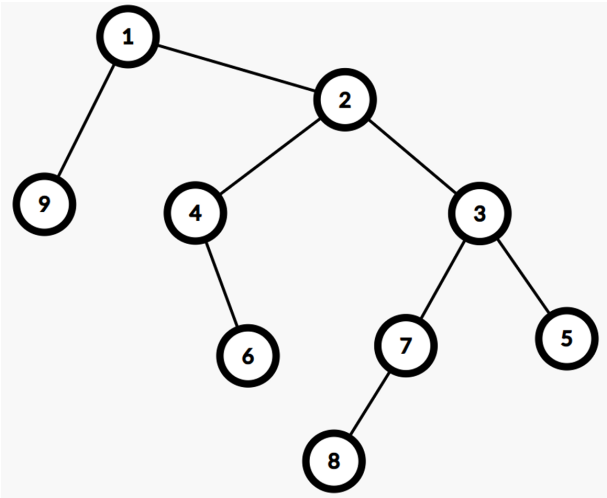
第 6 题 C

- 如果 `B=1`，那么相当于“数 1 的个数”，轻松就卡掉了。
- 如果 `B=6`，那么 B^{64} 自然溢出后是 0，完全没有使用前面字符的信息。
- 如果 $B = 2^{63} + 1$，那么 $B^2 \equiv 1 \pmod{2^{64}}$，所以相当于“分别数奇数和偶数位的 1”。

第 7 题 D

本题考查二叉树的遍历、小根堆的概念。

每次把当前子树中最小的结点作为根，然后以该结点为界，把序列分成左子树部分和右子树部分，递归下去，直到整棵二叉树被建立。二叉树如下面左图。



第 8 题 B

A. 读者不难把这个算法卡到 `O(nm log n)`——毕竟优先队列修改是 `O(log n)` 的。能否卡到指数级我还没仔细研究过，我怀疑可以，类似 SLF 优化的卡法。

B. 不难发现这就是 Dijkstra。

C. 显然，每个点只会更新有限多次，所以算法总是可以结束。

D. 如果我直接让这个图是一个环，那么任意时刻优先队列里面只有 `1` 个元素，从而程序用时为 `Cn`，其中 `C` 为某种常数。因此，$T(n) \le O(n)$，而显然 $T(n) \ge O(n)$（任意算法至少要读入所有的边），所以 $T(n) = O(n)$。

第 9 题 C

本题考查最近公共祖先的概念。

我们发现，无论根是什么，两个点的最近公共祖先，一定两点间路径上某个点。

只有 C 是两点间路径上两个点，选 C。

第 10 题 C

分类讨论即可。

$x \in [\frac{1}{6}, \frac{2}{9})$ 时，`f(x)=f(9x)=0`

$x \in [\frac{2}{9}, \frac{1}{3}]$ 时，`f(x)=f(9x)=1`

$x \in (\frac{1}{3}, \frac{1}{2}]$ 时，`f(x)=f(3x)=0`

因此概率为 $\dfrac{\frac{1}{3}-\frac{2}{9}}{\frac{1}{2}-\frac{1}{6}} = \frac{1}{3}$。

第 11 题 A

本题考查组合数、圆排列。

首先，把 8 个小朋友分进两个圆圈，有 $\binom{8}{4}/2$ 种方法，其中除以 2 表示“交换两个圆圈得到的方案一致”。

对于每个圈圈，有 6 种排法，因此方案数为 $\frac{8\times7\times6\times5}{2\times4!}\times(3!)^2=1260$。

第 12 题 A

本题考查时间复杂度分析，有一点技巧性。

$T(n)$ 有点难以观察，我们尝试估计一下 $f(k)=T(2^k)$。

$f(k)=f(k/2)+k$，继续展开得 $f(k)=k+k/2+k/4+\ldots+1\le 2k$。

因此 $T(n)=f(\log_2 n)=O(\log_2 k)$，选 A。

第 13 题 B

本题考查快速排序的流程。依次把每个选项代入计算即可。

A. $2,3,4,\ldots,n,1,n+1,n+2,\ldots,2n$：第一轮排序选择了 2，是边缘的数值；但是第二轮及以后，选取的数值就都是接近中位数的了，卡复杂度失败。

B. 每一轮选中的数字都是接近于区间最小值的，从而可以让递归次数为 $O(n)$，可以卡掉。

C 就是随机化快排。

第 14 题 B

本题为数论综合，考查欧拉函数、欧拉定理、卡特兰数、威尔逊定理。

$3^{48}\equiv(10^2)^{48}\equiv10^{96}\equiv1\pmod{97}$

$\varphi(119)=\varphi(7\times17)=96$

由递推式 $C_{n+1}=\frac{4n+2}{n+2}C_n$，可得 $C_9=\frac{34}{10}C_8=34\times143\equiv17\times(2\times46)\equiv-85\equiv12\pmod{97}$。

96! 由威尔逊定理，模 97 意义下和 -1 同余，也就是和 96 同余。

第 15 题 B

本题考查新闻。

解析略。

阅读 T1

判断题

1. 显然 `function_2(x, y)` 和 `function_2(y, x)` 的统一位不可能同时为 `1`，因此替换后答案永远为 `0`。

2. 根据逻辑优先级，去掉括号后结果不变。

3. 不对，若某一位是 `1 1`，那么在输出时这一位时 `0`，因此并不一定大于任意输入。

选择题

1. 上述替换后在读入时发生了数据溢出，是通常所说的未定义行为（ub），没有确定输出。

2. 首先排除 C，D 选项，根据前面的分析，该代码相当于返回 `function_2` 中传进去靠前的参数。

阅读 T2

`solve1` 函数中 `l` 数组表示前缀中 10 子序列的数量，`r` 数组表示后缀中 01 子序列的数量，函数功能为计算字符串中 1，101，10101 子序列的数量。

`solve2` 函数使用线性动态规划的方法，计算字符串中 10101.....01 串的数量（长度任意，01 交错，第一个和最后一个字符是 1）。

判断题

1. 输入 `7 1001101` 会输出两个 `17`。

2. 通常情况下，根据两个函数的功能，`solve1` 的返回值不会超过 `solve2`，但是在极限数据下 `solve2` 会超过 `int` 的数据范围，在溢出情况下会出现反例。

3. 当输入为 `10 1100110011` 时，`solve1` 的返回值为 `70`（其他输入也可能大于 `50`）。

选择题

1. 当输入 `7 1010101` 时，`solve2` 计入的 1010101 子序列无法被 `solve1` 计算。

2. 两个函数返回值相同，相当于输入字符串中不出现 1010101 子序列。

3. 当输入 `14 11001100110011` 时，1010101 子序列的数量为 `2^7=128`。

阅读 T3

该程序在判断将能否 n 个数字（$a_1, a_2, ..., a_n$）划分为若干不相交的非空子集，使每个子集中所有元素的异或和的二进制中有 m 位是 1。

判断题

1. 当 ty=1 时，数字不会超过 20，二进制不会超过 5 位，所以无需在意二进制中的高位。

2. $\{0, 1, 2, 3, 4\}$ 没有合法分组。

3. 选出 m 个二的整次幂作为一个集合，其他元素作为另一个集合。

选择题

1. 依次检查每个输入能否分组即可，输入量较小，枚举难度不大。

2.n 为偶数时，x 和 $x \bigotimes 1$ 放在一个集合，异或和为 1 ，$m = 1$ 恰好合法。

3. 枚举二进制，再枚举子集的时间复杂度为 $O(n3^n)$ ，证明可以考虑二项式定理。

填空 T1

题目实际上就是要支持两个操作，一个是区间推平，一个是询问一段区间是否完全相同（顺便判一下区间左右的值是否不同）。区间推平的常用方法除了线段树，odt 也是一种常见方法。如果写过 odt，本题将会特别简单。odt 的大致思路是将一段值相同的区间用一个元素去表示。

`vector list` 均是线性表，其成员函数 `emplace` 需要传入下标作为插入位置。而 `map\<odt_node>` 无法通过编译，因为 `map` 定义需要 `key` 和 `value`，因此 Blank 2 选择 C，无需看懂代码即可通过语法判断。

`split` 意思是切，可以猜测 `split` 函数的效果是将 p 所在的一块切出来。Blank 3 中 `count` 必然不对，因为 `set` 没有成员函数 `count`。如果是 `find`，找不到元素会返回 `.end()` 迭代器，后续将会对最后一块进行操作，显然不对。故 Blank 3 的答案应为 A,B 中的一个。

12 行将 p 放进函数内，但是 `lower_bound` `upper_bound` 接受的参数都是 `odt_node`，而 p 是 `int`，这里调用了 p 的构造函数，将 p 变成了一个 `odt_node`，其 $l = p, r = val = 0$，因此必然使用了左端点进行比较，否则无意义。

`set` 中的元素需要满足有序性，本题 `set` 中保存的类型为 `odt_node`，Blank 1 就是要填写 `odt_node` 的有序性。由 24 行从左到右遍历区间可以看出是按照升序，结合第 12 行可以看出 Blank 1 应该选 A。

回到 Blank 3，如果采用 `upper_bound` 必然不会出现 `it->l==p`，借此可以推断出答案是 A `lower_bound`。如果是 `upper_bound`，则 $t.l = p$，而插入的新元素是 $[t.l, p-1]$，会出现问题。

33 行是把极长的连续段放到 `set` 中，36 行是把到末尾这一段放进去，仿照 33 行不难看出 Blank 5 选 D。

24 行在检查 $l \sim r$ 的元素是否完全相同，25 行是处理不是完全相同的情况，`it` 是第一个不相同的块，因此直到 `it` 前面那一块都相同，所以 Blank 4 选 A。同时注意到 B,D 的效果完全相同，因为后面直接 return 了，所以答案肯定不是 B,D。

第 20 题

可以发现代码大致分为 3 个部分，分别是 2 个参数的 `solve`，3 个参数的 `solve` 和剩余部分。

2 个参数的 `solve` 的代码完全给出，可以看出其实际上就是把序列分成两段，然后求两段的最长公共子序列。同理，3 个参数的 `solve` 实际上就是把序列分成三段，然后求三段的最长公共子序列。所以转移条件是三个值完全相同，Blank 1 选 C。

32,33 行在枚举一个长度为 m 的子串。剩余部分是个状压，34 行枚举非空子集，应为该子串的子集，因此 Blank 3 选 D。

37,38 行就在选出子集，一定要看清楚 t 的类型是字符串！`k>>i&1` 证明第 $l+i$ 个元素被选中，因此 Blank 4 选 B。同时可以看出 `L,R` 记录了选中元素的左右端点。

`solve` 函数解决了 $k = 2, 3$ 的情况，其实也解决了 $k = 4$ 的情况，因为 $k = 4$ 的情况在 $k = 2$ 时已经被计算。猜测剩余部分应该是在解决 $k \geq 5$ 的情况。

当 $k \geq 5$ 时，根据鸽笼原理，一定可以找到一个长度 ≤ 16 的子串，该子串的子集就是 $k \geq 5$ 的循环节。`B` 是用来限制子串的最大长度，因此 $B = 16$，Blank 2 选 C。

`L,R` 记录了选中元素的左右端点。我们枚举的是循环节，因此只需把剩余循环节补全。代码枚举了 $[0, L-1]$ 和 $[R+1, n]$，其中每出现过一整个 `t` 串，`res` 就会统计一次数量（初始数量是 1）。如果匹配，$p+=1$，如果此时 $p = siz(t)$ 就说明循环节补全完了，当且仅当此时 $p \mod siz(t)$ 是零，还要让 `res` 统计一次，Blank 5 选 A。