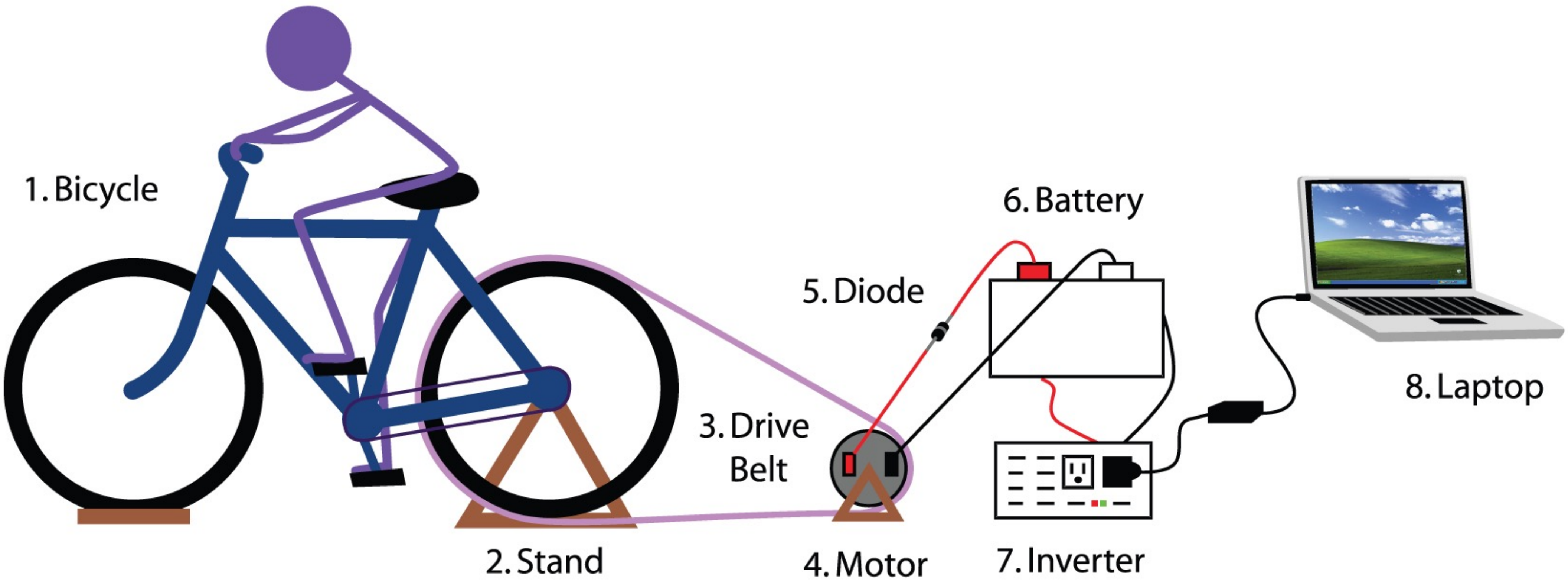
aec-online-{i,ii}-2021 题目讲解



II 题目分布

这一场的难度标记是 easy: JM medium-easy: AFGKL medium: HE medium-hard: BCI hard: D，其中 FJLM 是命题会开完后补的题，分别补简单贪心、模拟、数据结构，和究极签到。

- easy
  - J. Leaking Roof
  - M. Addition
- medium-easy
  - A. Sort
  - F. Leapfrog
  - G. Limit
  - K. Meal
  - L. Euler Function
- medium
  - E. Nearest Point
  - H. Set
- medium-hard
  - B. Mailman
  - C. Range Subsequence
  - I. Discrete Mathematics
- hard
  - D. Linear Algebra

II.A Sort

- 给定一个长为 n 的数组，每次你可以将其划分成不超过 k 个非空连续子段并将其重新排列。这种操作可以进行无限次。
- 判断是否存在一种操作方案，使得的数组从小到大排序，且所有操作划分的段数和不超过 3n。
- $n <= 1000$；$\Sigma n <= 30000$

II.A Sort 题解

- K = 1 检查数组是否排序即可
- K = 2 所有操作本质都是轮转，因此最多只有一种操作，检查所有方案即可。
- K >= 3 考虑插入排序算法。不难发现不论其在什么位置，我总可以通过一次操作，将其插到数组开头有序部分的末尾。段数和 <= 3 段 * n 次 = 3n

II.B Mailman

- 给定一张 n×m 的无向带权网格图，每条边均可以经过任意多次。支持如下操作，共 Q 次：
- 加入一条连接 (x1,y1),(x2,y2) 长度为 v，仅能经过一次的边。
- 每次加入边后，询问如下式子的值：
- 从 (1,1) 出发，经过每条边至少一次，最后回到 (1,1)，最小经过距离。
- n <= 3；m, Q <= 50000

II.B Mailman 题解

- 经过的路径总是一条欧拉回路。问题可以转化为如下方式：
- 给一张网格图，动态修改某个点的度数。
- 你可以在已有的边(x,y)的基础上额外加一条连接(x,y)的边，询问将所有点度数变为偶数的最小花费。(不同询问互不影响)

II.B Mailman 题解

- 直接考虑线段树，每个节点维护一段连续的列区间。节点存储左侧边界节点度数状态为 S，右侧边界节点度数状态为 T 的最小花费解。
- 修改和询问在这个的基础上暴力询问支持即可。
- 状态数为 2^n，因此总复杂度为 O(Q log m * 512)，可以通过本题。

II.C Range Subsequence

- 序列 s 是 t 的子序列当且仅当能通过删除元素操作从 t 得到 s。
  - e.g., $s=(2,6,3)$, $t=(2,4,5,6,3)$
- 序列 s 是一个 range 当且仅当它的第 i 个数字恰好比第 $i-1$个数字大 1。
  - e.g., $s=(2,3,4)$
- 给出一个长度 n 的序列 A，你需要处理 m 组询问。
- 每组询问给出一个区间 $[l,r]$，你需要计算只考虑 A 的第 l 到第 r 个元素的情况下，有多少个**本质不同**的 range 的子序列。
- $n \le 10^5, m \le 10^5, A_i \in [1,n]$

II.C Range Subsequence 题解

- 莫队，问题变成了已知 $[l+1,r]$ 的答案，计算 $[l,r]$ 的答案（其他几个莫队操作与这个对称）。
- 定义 $f(l,r)$ 表示从 l 出发，依次向后走到最近的 $A_l+1, A_l+2, \ldots$，在跨过 r 之前走了多少步。
- 这也等于 $[l,r]$ 中以 A_l 开头的最长 Range 子序列长度。
- 那么 $[l,r]$ 的答案等于 $[l+1,r]$ 的答案加上 $f(l,r)-f(ne_l,r)$，其中 ne_l 表示 l 后面第一个值等于 A_l 的位置。
- $f(l,r)$ 可以做到 $O(n^{1.5})$ 预处理，$O(1)$ 询问：
  - 对每一个位置 i 和 $j \leq n^{0.5}$，预处理 $f(i, j\times n^{0.5})$ 和 $f(i, i+j)$ 即可。
- 时间复杂度 $O(n^{1.5})$。

II.D Linear Algebra

- 题意：给定多项式g(x)，求多项式f(x)使得f(x)f(x+1)=g(x)。
- 做法：分治，假设当前计算指数[l,r)的答案。先递归计算[mid,r)的答案，然后使用卷积计算出这部分之前的f(x+1)，计算出新增加部分对f(x)f(x+1)的贡献，然后处理g(x)系数的[l+n,mid+n)项。
- 时间复杂度$O(n \log^2 n)$

II.E Nearest Point

- 给出 n 个二维平面上的点，定义第 i 号点的最近点为剩下点中横坐标与它最接近的点。（如果有多个取编号最小的）
- 从 $[-\pi,\pi)$ 中随机一个角度 α，并将所有点绕原点逆时针旋转 α 的角度。
- 对每一对 (i,j) $(1 \le i,j \le n)$，询问在旋转之后，点 j 是点 i 最近点的概率。
- $n \le 50$

II.E Nearest Point 题解

- 现在考虑每个点作为点 i 的最近点的概率。
- 显然我们只需要关系所有点横坐标距离 i 的相对顺序，而不关心具体值。
- 对于任意两个点 $\{j,k\}$：
  - 如果 i,j,k 共线，j,k 中必有一个点恒比另一个近。
  - 否则，存在四个临界角度，分别对应 j,k 在 i 左右两侧的四种情况。
- 一共有 $O(n^2)$ 个临界角度，把 α 的取值分成了 $O(n^2)$ 段，每一段内 i 的最近点是不变的。
- 因此可以在每一段里任取一个角度，算出最近点，并把这一段的概率都算到那个点上。暴力的时间复杂度是 $O(n^3\log n)$。
- 对每个 i 都做一遍，时间复杂度为 $O(n^4\log n)$。
- 临界角度可以推公式，也可以二分/三分求解。

II.F Leapfrog

- 题目来源于炉石传说酒馆战棋随从跳蛙骑士。
- 现在有 n 个跳蛙骑士，第 i 个的属性值为 a_i，具有亡语 (b_i, t_i)，表示在该跳蛙死亡后，会随机 t_i 次，每次随机选择一个还活着的跳蛙，使它的属性值增加 b_i 并获得该亡语。
- 在游戏过程中，一个跳蛙会获得很多亡语。它死亡后这些亡语会依次触发。
- 现在游戏进行了 $n-1$ 轮，每一轮都有一个随机的跳蛙骑士死亡。
- 你需要计算最后或者的那个跳蛙骑士最大可能的属性值。
- 对 998244353 取模。
- $n \le 10^5, a_i, b_i \le 10^4, t_i \in \{2,3\}$

II.F Leapfrog 题解

- （贪心）显然最优情况下，第 i 轮跳蛙的亡语应该都加到第 $i+1$ 轮上。
- 因此，相当于要找到一个跳蛙的排列 p 来最大化：
  - $a_{p_n} + \sum_{i=1}^{n-1} b_{p_i} \times t_{p_i}^{n-i}$
- 定义排列 p' 为将所有跳蛙以 t_i 第一关键字降序，b_i 第二关键字降序，a_i 第三关键字升序排序后的结果。
- **引理 1**：p 的前 $n-23$ 个元素一定和 p' 的前 $n-23$ 个元素完全一样。
  - 原因: 亡语 (b_i, t_i) 的 $b_i \le 10^4$ 且 $t_i \in \{2,3\}$
  - 对 $\sum_{i=1}^{n-1} b_{p_i} \times t_{p_i}^{n-i}$ 的贡献：第 $i = n-23$ 项，$10000 \times 2^{23} < 3^{23}$，优先选 $t_i = 3$
  - 因此前 $n-23$ 个元素向 p' 调整一定能变得更优。

II.F Leapfrog 题解

- **引理 1**： p 的前 $n-$ 23 个元素一定和 p' 的前 $n-23$ 个元素完全一样。
- 因此，只需要考虑 $n \le 23$ 的情况。
- **引理 2**：前 $n-1$ 个跳蛙中，所有 t_i 相同的跳蛙按照 b_i 降序排列。
  - 通过调整法易得。
- 所以，可以用动态规划来排列 $t_i = 2$ 和 $t_i = 3$ 跳蛙的相对顺序。
- $dp[i][j][0/1]$ 表示前 i 个位置放了 j 个 $t_i = 2$ 的跳蛙，以及是否有跳蛙被放到了最后。
- 转移的时候，枚举下一个放进去的跳蛙的 t_i 以及是否放到最后。
- 考虑 $n \le 23$ 情况的时间复杂度 $O(n^2)$。

II.G Limit

- 题意：计算 $\lim_{x\to 0} x^{-t} \cdot \sum_{1\le i\le n} a_i \cdot \ln(1+b_i\cdot x)$，其中 $0 \le t \le 5$
- 由 $\ln(1+b_i\cdot x) = \sum_{j\ge 0} \frac{(-b_i)^j x^j}{j+1}$
- 我们可以把原来的式子展开成:

$$x^{-t} \cdot \sum_{i=1}^{n} \sum_{j\ge 0} \frac{a_i \cdot (-b_i)^j x^j}{j+1} = \sum_{j\ge 0} x^{j-t} \cdot \sum_{1\le i\le n} \frac{a_i \cdot (-b_i)^j}{j+1}$$

由极限的定义，我们从小到大枚举 j，找到第一个 j 使得 x^{j-t} 的系数非0即可。

II.H Set

- 只需随机 k 个大小为 $\ell = \lceil 512/r \rceil$ 的集合即可.
- 出错当且仅当存在 $S_{i_1}, S_{i_2}, \dots, S_{i_r}$ 和一个大小为 127 的集合 X 使得：$S_{i_1} \subseteq X, S_{i_2} \subseteq X, \dots, S_{i_r} \subseteq X$
- 由 Union bound, 出错的概率不超过:
- $\binom{k}{r} \cdot \binom{256}{127} \cdot \left(\frac{\binom{127}{\ell}}{\binom{256}{\ell}} \right)^r \leq 5.8 \cdot 10^{-70}$

II.I Discrete Mathematics

- 询问 F_p[x] 上满足次数为 n,首项为1 ,次高项为 k 的不可约多项式个数，答案对 998244353 取模。
- n,p<=2500, p>=5, 0<p<k

II.I Discrete Mathematics

- 数学结论：http://combos.org/Tpoly
- 提交通项，咋写都行。

II.I Discrete Mathematics

- 暴力容斥 DP :
- f[i][j][k] 表示 i 次首一多项式，由若干个低于 j 次的不可约首一多项式乘起来，次高项为 k，有多少种方案。
- G[i][k] 表示 i 次首一不可约多项式，次高项为 k , 有多少种方案。
- 时间复杂度 O(n^2 p^2)

II.I Discrete Mathematics

- 性质1：n 相同时，所有次高项非 0 的时候答案相等。
- 假设 ∀ m < n, ∀ 1 ⩽ k < p，满足 g[i][1] = g[i][k]
- 不妨假设 h[i][j][k] 表示无序的选择了 j 个 i 次的不可约首一多项式相乘，且次高项为 k 的方案数。
- 首先假设 g[i][1] = g[i][0]。此时我们利用 Polya 定理将无序问题转化成有序问题。不妨去假设 j! 个置换中，每个置换环的大小。
- 如果存在任意一个长度不为 p 的倍数的置换环。则在假设条件下，我们总可以通过调整这个置换环上的元素次高项，使得乘积次高项可以均匀分布在 0 ~p-1 上。因此此部分贡献到 h[i][j][0]~h[i][j][p-1] 的贡献相同。
- 否则如果所有置换环长度为 p 的倍数，则不论取什么环上元素，乘积次高项恒为 0。此时仅仅会贡献到 h_{i,j,0} 上。
- 因而 ∀ m < n, 0 ⩽ j, 1 ⩽ k < p, h[m][j][k]=h[m][j][1]

II.I Discrete Mathematics

- 一般的，我们可以先假设 g[i][0],最后加入这些元素。
- 由于若干个次高项为 0 的多项式相乘次高项仍为 0。因而仅考虑次高项为 0 的元素时仅有 h_{m,j,0} 处可能有值。自然满足如上条件。
- 将 h 看成是固定物品体积为 i 时候的背包问题方案数后，f[i][j][k] 即为限制物品体积不超过 i 的背包方案计数。利用背包的合并规则，我们仍然有：\forall m < n,\forall j,\forall 1 \le k < p,f[m][j][k]=f[m][j][1]。
- 利用 g[i][j]=p^(i-1)-f_[i-1][i][j]知道，\forall 1 \le k < p,g[n][k]=g[n][1]
- 由于 n=1 时有 g[i][1] = 1满足原条件，因而利用归纳法，性质 1 成立。

II.I Discrete Mathematics

- 性质1的证明已经给了一个 O(n^2 logn) 的做法。

II.J Leaking Roof

- 题意：n×n 的网格每个格子有一个高度，格子的水会向和它相邻的高度比他低的格子流动，问所有高度0的格子最后有多少水。
- 做法：按高度从高到低排序，按此顺序直接模拟即可。
- 时间复杂度 $O(n^2 \log n)$

II.K Meal

- 有 n 个菜，有 n 个人打菜，每个人打一道菜。
- 第 i 个人对第 j 个菜的喜爱度是 a[i][j]。
- 每个人会随机生成一个喜爱序列，生成方式是如果 j 没有加入序列，则会以正比于 a[i][j] 的概率加入序列末尾。
- 每个人会打在喜爱序列中最靠前的没有被打的菜。
- 问第 i 个人打第 j 个菜的概率，对 998244353 取模。
- n <= 20

II.K Meal 题解

- 假设我们已经知道前 i 个人打的菜的集合，则有性质：
- 如果 k 没有被打过，则第 k 道菜在第 i+1 个人的偏爱序列里，排在所有没打过菜的第一位的概率正比于 a[i+1][k]。
- 证明较为显然，这里略去。
- 因此直接状态压缩 DP 即可，复杂度 $O(n\ 2^n)$

II.L Euler Function

- 给出一个长度为 n 的序列 x，你需要维护 m 次操作：
  - 修改：将一个区间内的所有 x_i 乘上一个给定值 w_i。
  - 询问：询问一个区间内所有 $\varphi(x_i)$ 的和，对 998244353 取模。
- $\varphi(n)$ 是欧拉函数，值等于 $[1,n]$ 内与 n 互质的数的数量。
- $n \le 10^5, w, x_i \le 100$

II.L Euler Function 题解

- 欧拉函数：设 n 的质因数分解是 $\prod p_i^{t_i}$，那么 $\varphi(n) = \prod (p_i - 1) p_i^{t_i - 1}$。
- 因此，如果 w 的所有质因子在 x_i 都出现了，$\varphi(x_i \times w) = \varphi(x_i) \times w$。
- 用线段树来维护区间欧拉函数的和。
- 考虑对区间 $[l, r]$ 的乘 w 操作。
  - 假设这个区间中有 m 个位置不满足 w 中的所有质因子在 x_i 中都出现了。
  - 这些位置把区间划分成了 $O(m)$ 段，每一段内部是区间乘维护区间和。
- 因此，可以把这个操作拆分成 $O(m)$ 个单点修改和区间乘操作。
- 因为每一次单点修改后，对应位置上出现的质因子个数至少增加 1，所以 m 的和不超过$25n$，25为100以内的质数个数。
- 至于找到所有不满足的位置，可以直接用线段树维护，也可以对每个质因子分别用并查集维护。时间复杂度 $O(m \log n)$。

II.M Addition

- Question: Why do computer programmers confuse Christmas and Halloween?
- Answer: Because 25 Dec = 31 Oct

-- http://www.electronicsweekly.com

II.M Addition

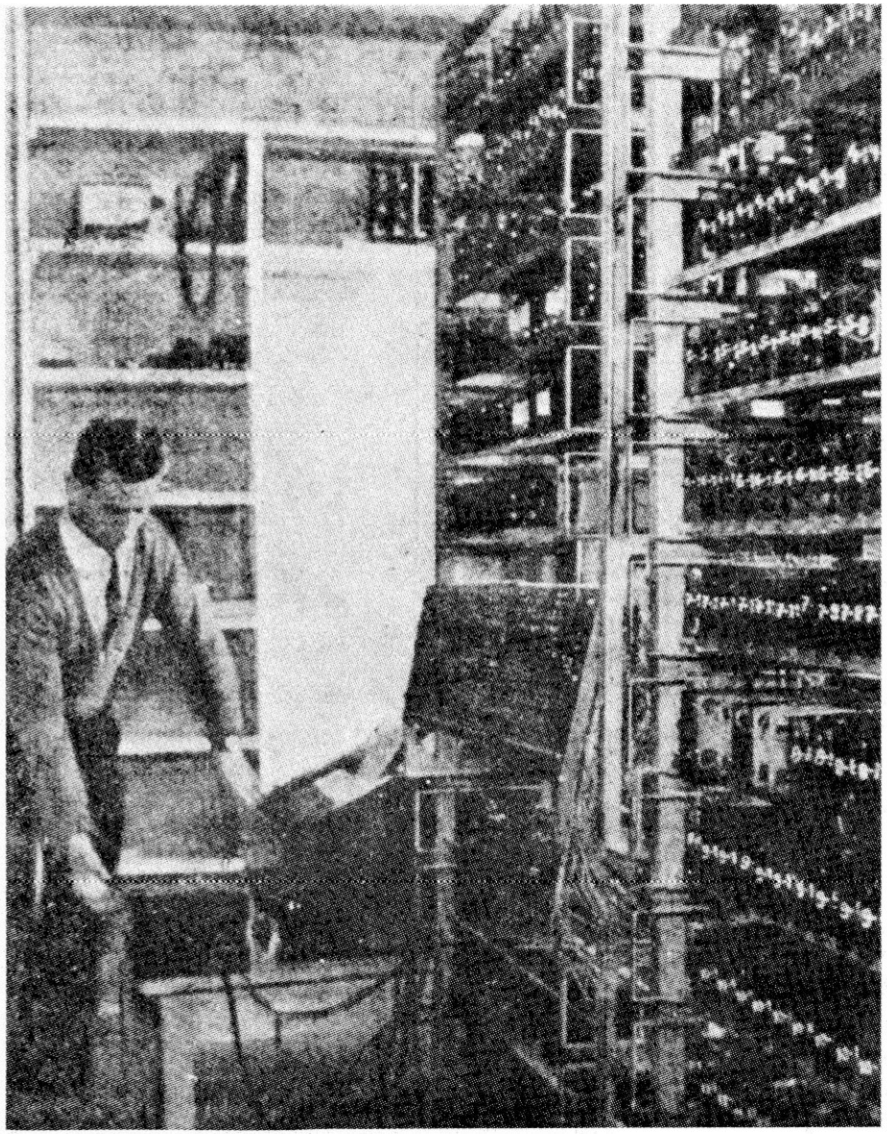
- 假设所有二进制数字表达式为 $\sum_{i=0}^{n-1} v_i \cdot sgn_i \cdot 2^i$
- 给定 a,b， 计算 a+b 的二进制表达式，保证有解。
- 32 <= n <= 60

II.M Addition 题解

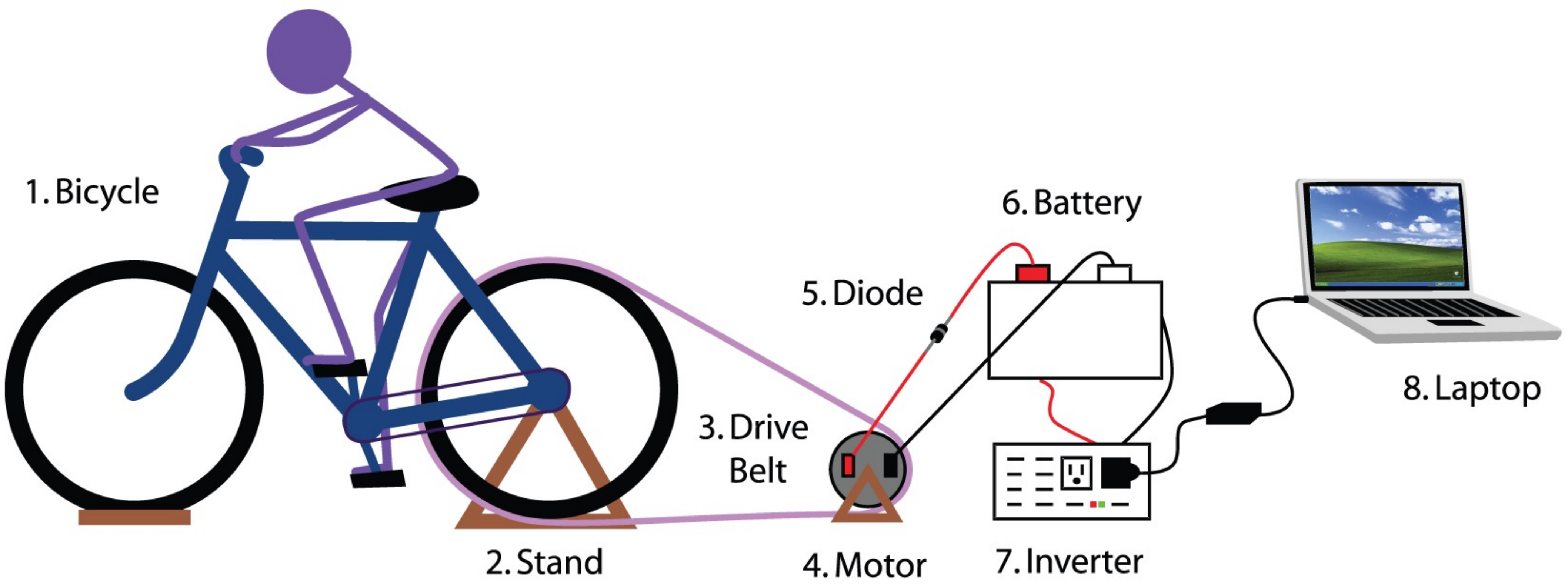
- 从低位到高位按位确定每一位填的值即可。
- 也就是说，答案第 i 位是 1 当且仅当填到第 i 位的时候剩余的数字第 i 位是1。

Negabinary & BINEG

n	negabinary	n	negabinary
1	1	11	11111
2	110	12	11100
3	111	13	11101
4	100	14	10010
5	101	15	10011
6	11010	16	10000
7	11011	17	10001
8	11000	18	10110
9	11001	19	10111
10	11110	20	10100

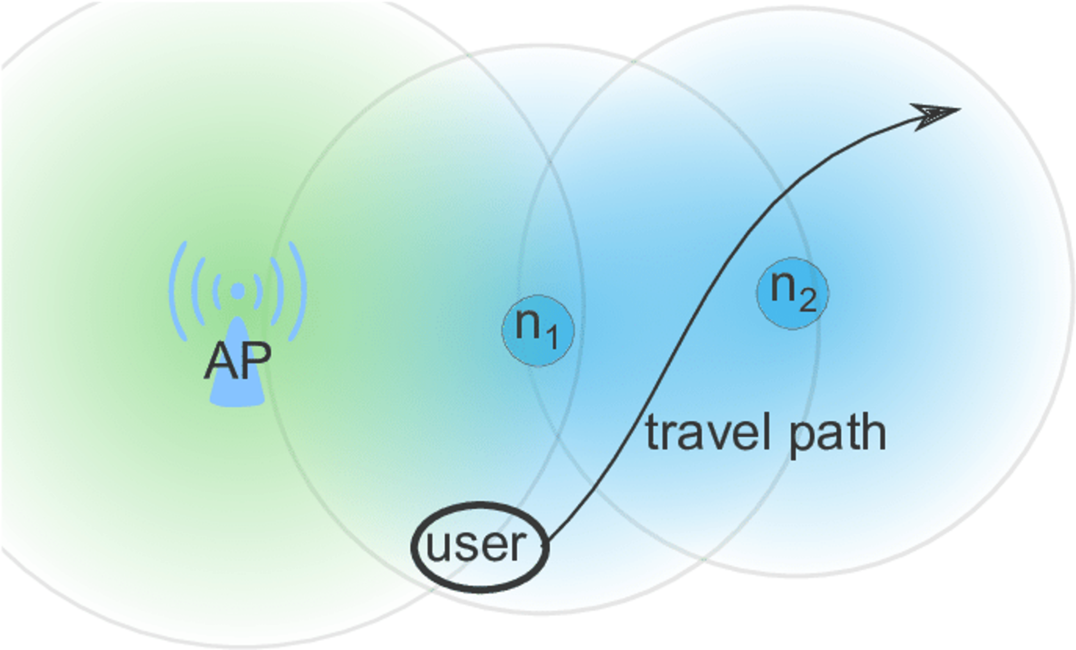


BINEG computer [IEEE AHC' 80]


1. Bicycle
2. Stand
3. Drive Belt
4. Motor
5. Diode
6. Battery
7. Inverter
8. Laptop

aec-online-i-2021

- A. Busiest Computing Nodes
- B. Convex Polygon
- C. Driver Licenses
- D. Edge of Taixuan
- E. Infinite File System
- F. Land Overseer
- G. Longest Prefix Matching
- H. Mesh Analysis
- I. Neighborhood Search
- J. Red-Black Paths
- K. Segment Routing



ICPC Policies and Procedures

- Mission
  - The “ICPC International Collegiate Programming Contest” is an extra-curricular, competitive programming sport of the universities of the world. ICPC competitions provide gifted students opportunities to interact, demonstrate, and improve their teamwork, programming, and problem-solving prowess. The ICPC is a global platform for academia, industry, and community to shine the spotlight on and raise the aspirations of the next generation of computing professionals as they pursue excellence.

ICPC Policies and Procedures

- Goals - Decisions shall be governed by the following goals:
  - Attract as many students as possible.
  - Attract as many colleges and universities as possible.
  - Draw from as many geographical regions of the world as possible.
  - Provide equitable access to the ICPC World Finals.
  - Strive for competitive contests.
  - Involve industry, community, and shine the spotlight on students.
  - Maintain and support the volunteer base.

- Culture
  - Organization
    - Promote regional integrity.
    - Provide global coordination.
  - Policy
    - Keep it simple.
    - Serve, don' t rule.
    - Never have the same problem for the same reason.
  - Principles:
    - Put people first.
    - Practice goodwill.
    - Follow the Golden Rule.
    - As needed, solve the solvable problems; resolve the resolvable issues; avoid the rest.
    - Grow strong by resolving conflicts.

http://icpc.pku.edu.cn

wechat: ICPCNews