Solution

He_Ren

A. 树

- 给定一张 n 个点 m 条边的无向图。
- 对于所有 n-1 <= k <= m，求出选出恰好 k 条边使它连通的方案数。
- 答案对 1e9 + 7 取模。

A. 树 – 做法 1

- 考虑容斥，如果整张图不连通的话那么枚举 1 所在的连通块。

A. 树 – 做法 1

- 考虑容斥，如果整张图不连通的话那么枚举 1 所在的连通块。
- f[S][i] 表示钦定 S 连通，且选了 i 条边的方案数，$1 \in S$。

A. 树 – 做法 1

- 考虑容斥，如果整张图不连通的话那么枚举 1 所在的连通块。
- f[S][i] 表示钦定 S 连通，且选了 i 条边的方案数，$1 \in S$。
- 设 cnt[S] 表示两端都在 S 中的边数。

A. 树 – 做法 1

- 考虑容斥，如果整张图不连通的话那么枚举 1 所在的连通块。
- f[S][i] 表示钦定 S 连通，且选了 i 条边的方案数，$1 \in S$。
- 设 cnt[S] 表示两端都在 S 中的边数。
- 转移:

$$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$

A. 树 – 做法 1

- 考虑容斥，如果整张图不连通的话那么枚举 1 所在的连通块。
- f[S][i] 表示钦定 S 连通，且选了 i 条边的方案数，$1 \in S$。
- 设 cnt[S] 表示两端都在 S 中的边数。
- 转移：

$$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$

- 复杂度 O(3^n m^2)，难以通过。

A. 树 – 做法 1

- $$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$
- 注意到 f[T] 向 f[S] 转移时，会影响转移系数的只有 cnt[S\T]。

A. 树 – 做法 1

- $$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$
- 注意到 f[T] 向 f[S] 转移时，会影响转移系数的只有 cnt[S\T]。
- 将 cnt[S\T] 相等的 T 放在一起转移。

A. 树 – 做法 1

- $$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S,\\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$
- 注意到 f[T] 向 f[S] 转移时，会影响转移系数的只有 cnt[S\T]。
- 将 cnt[S\T] 相等的 T 放在一起转移。
- 复杂度 O(3^n m)，可以通过。

A. 树 – 做法 2

- $$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$
- 将 f[S] 看成多项式，将转移写成多项式的形式。

A. 树 – 做法 2

- $$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$
- 将 f[S] 看成多项式，将转移写成多项式的形式。
$$\mathrm{f}[\mathrm{S}] = (1+x)^{cnt[S]} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} f[T] \cdot (1+x)^{cnt[S \setminus \mathrm{T}]}$$

A. 树 – 做法 2

- $$f[S][i] = \binom{cnt[S]}{i} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} \sum_{j} f[T][j] \binom{cnt[S \setminus T]}{i-j}$$
- 将 f[S] 看成多项式，将转移写成多项式的形式。
$$\mathrm{f[S]} = (1+x)^{cnt[S]} - \sum_{\substack{T \subsetneq S, \\ 1 \in T}} f[T] \cdot (1+x)^{cnt[S \setminus \mathrm{T}]}$$
- 换元，设 y=(1+x)，乘法就变成了平移，可以 O(m) 完成。

A. 树 – 做法 2

- 最后代入即可，总复杂度 O(3^n m)

A. 树 – 做法 3

- 容斥，考虑图不连通的情况。
- 将点划分为若干点集 $S_1, S_2, \ldots, S_k$，钦定两个 S 之间没有连边，每个 S 内部的边任意选择。

A. 树 – 做法 3

- 容斥，考虑图不连通的情况。
- 将点划分为若干点集 $S_1, S_2, \dots, S_k$，钦定两个 S 之间没有连边，每个 S 内部的边任意选择。
- 设容斥系数为 coef，所有 S 内部的总边数为 t，那么给选择 i 条边的答案的贡献为 $coef \cdot \binom{t}{i}$。

A. 树 – 做法 3

- 容斥，考虑图不连通的情况。
- 将点划分为若干点集 $S_1, S_2, \dots, S_k$，钦定两个 S 之间没有连边，每个 S 内部的边任意选择。
- 设容斥系数为 coef，所有 S 内部的总边数为 t，那么给选择 i 条边的答案的贡献为 $coef \cdot \binom{t}{i}$。
- 问题是如何确定 coef。我们希望 coef 只和 k 有关。

A. 树 – 做法 3

- 考虑一个实际有 x 个连通块的方案，它会贡献的总系数为

$$\sum_{i=1}^{x} \left\{ {x \atop i} \right\} coef[i]$$

- 我们希望它等于 $[x = 1]$。

A. 树 – 做法 3

- 考虑一个实际有 x 个连通块的方案，它会贡献的总系数为

$$\sum_{i=1}^{x} \left\{ {x \atop i} \right\} coef[i]$$

- 我们希望它等于 $[x = 1]$。
- 斯特林数有如下性质:

$$\sum_{i=0}^{x} \left\{ {x \atop i} \right\} \left[{i \atop y} \right] (-1)^{i-y} = [x = y]$$

A. 树 – 做法 3

- 斯特林数有如下性质:

$$\sum_{i=0}^{x} \left\{ {x \atop i} \right\} \left[{i \atop y} \right] (-1)^{i-y} = [x = y]$$

- 代入 $y = 1$ 得

$$\sum_{i=0}^{x} \left\{ {x \atop i} \right\} (i-1)!\,(-1)^{i-1} = [x = 1]$$

- 因此 $coef[k] = (k-1)!\,(-1)^{k-1}$。

A. 树 – 做法 3

- dp[S][i][j] 表示划分集合 S，将其划分为 i 个点集，点集内部的总边数是 j 的方案数。

A. 树 – 做法 3

- dp[S][i][j] 表示划分集合 S，将其划分为 i 个点集，点集内部的总边数是 j 的方案数。
- 复杂度 O(3^n n m)

A. 树 – 做法 3

- 只要认为 $S_1, S_2, \dots, S_k$ 这个序列是有序的，就自然地有 k! 的系数。
- 再钦定 $1 \in S_1$ 这个系数就会变成 (k-1)!。
- 这样就不需要把 k 记在状态中了。
- 具体来说，如果认为 S 是无序的，转移时需要钦定 lowbit 在当前的 S 中，但如果认为 S 是有序的，就不需要做这样的钦定。

A. 树 – 做法 3

- 只要认为 $S_1, S_2, \ldots, S_k$ 这个序列是有序的，就自然地有 k! 的系数。
- 再钦定 $1 \in S_1$ 这个系数就会变成 (k-1)!。
- 这样就不需要把 k 记在状态中了。
- 具体来说，如果认为 S 是无序的，转移时需要钦定 lowbit 在当前的 S 中，但如果认为 S 是有序的，就不需要做这样的钦定。
- 复杂度优化至 O(3^n m)。

A. 树 – 做法 3

- 只要认为 $S_1, S_2, \ldots, S_k$ 这个序列是有序的，就自然地有 k! 的系数。
- 再钦定 $1 \in S_1$ 这个系数就会变成 (k-1)!。
- 这样就不需要把 k 记在状态中了。
- 具体来说，如果认为 S 是无序的，转移时需要钦定 lowbit 在当前的 S 中，但如果认为 S 是有序的，就不需要做这样的钦定。
- 复杂度优化至 O(3^n m)。
- 做法 2 和做法 3 本质相同。

A. 树 – 做法 4

- 做法 2、3 可以用子集卷积优化至 O(2^n poly(n) poly(m))。

A. 树 – 做法 4

- 做法 2、3 可以用子集卷积优化至 O(2^n poly(n) poly(m))。
- 本题也可以直接用集合幂级数做。

A. 树 – 做法 4

- 做法 2、3 可以用子集卷积优化至 $O(2^n \text{poly}(n) \text{poly}(m))$。
- 本题也可以直接用集合幂级数做。
- 设将集合 S 连通的信息为 $f[S]$，那么任意选边的信息 $g = \exp(f)$，反过来有 $f = \ln(g)$。

A. 树 – 做法 4

- 做法 2、3 可以用子集卷积优化至 O(2^n poly(n) poly(m))。
- 本题也可以直接用集合幂级数做。
- 设将集合 S 连通的信息为 f[S]，那么任意选边的信息 g = exp(f), 反过来有 f = ln(g)。
- 这个 ln 可以直接用 FWT 算。
- $f = \ln(1+g) \Rightarrow f' = \frac{g'}{1+g} \Rightarrow f' = g' - f'g$
- 复杂度 O(2^n poly(n) poly(m))。
- 写出来应该和做法 2、3 差不多。

B. 差

- 有一个序列，给定每相邻三个数的极差，求任意合法序列。

B. 差

- 只需要考虑原序列的差分。

B. 差

- 只需要考虑原序列的差分。
- 设 $a[i+1] – a[i] = c[i]$，那么 $b[i] = \max(| c[i] |, | c[i+1] |, | c[i] + c[i+1] |)$

B. 差

- 只需要考虑原序列的差分。
- 设 a[i+1] – a[i] = c[i]，那么 b[i] = max(| c[i] |, | c[i+1] |, | c[i] + c[i+1] |)
- 考虑 dp，dp 时可以只记 c[i] 的绝对值。

B. 差

- dp[i][x] 表示只考虑 1 至 i，| c[i] | = x 是否合法。
- 转移时，先只保留 0 <= x <= b[i] 的项，然后:
  - x => b[i] – x
  - b[i] => [0, b[i]]
  - x => b[i]

B. 差

- dp[i][x] 表示只考虑 1 至 i，| c[i] | = x 是否合法。
- 转移时，先只保留 0 <= x <= b[i] 的项，然后:
  - x => b[i] – x
  - b[i] => [0, b[i]]
  - x => b[i]
- 可以观察到 dp[i] 由一段区间和若干单点构成，转移只有删除前后缀，和翻转、平移。
- 区间直接维护，单点用双端队列维护。
- 容易记录方案，总复杂度 O(n)。

B. 差

- dp[i][x] 表示只考虑 1 至 i，| c[i] | = x 是否合法。
- 转移时，先只保留 0 <= x <= b[i] 的项，然后:
  - x => b[i] – x
  - b[i] => [0, b[i]]
  - x => b[i]
- 可以观察到 dp[i] 由一段区间和若干单点构成，转移只有删除前后缀，和翻转、平移。
- 区间直接维护，单点用双端队列维护。
- 容易记录方案，总复杂度 O(n)。

C.贼

- 给定一个长度为 n 的字符串 s。
- Q 次查询，单独把一个区间拿出来求 z 函数，再求和的结果。
- n,Q <= 2e5，字符集为小写字母。

C.贼

- 设 LCP(i,j) 为原串中后缀 i,j 的最长公共前缀。
- 答案等于 $\sum_{i=L}^{R}\min\left(R-i+1,\ LCP(L,i)\right)$
- 将原串倒过来建 SAM，求 fail 树，LCP(i,j) 即为 i,j 对应点 LCA 的 len。

C.贼

- 点分治，考虑分开 L 到 i 路径的分治中心 rt。
- 为了避免混淆，定义 “分治区域” 为点分治时 rt 分开的各个连通块。

C.贼

- 如果 rt 在 LCA 到 i 的路径上，此时:
  - i 和根不在同一分治区域，且 i 和 L 不在同一分治区域。
- LCA 只与 L 有关，设它的 len 为 k。
- 于是 min(r-i+1, k) 将 i 分为了两个区间，分别统计 i 的个数即可。

C.贼

- 如果 rt 在 LCA 到 L 的路径上，此时:
  - i 和根在同一分治区域，且 L 和根不在同一分治区域。
- LCA 只与 i 有关，设它的 len 为 k。
- 计算 min(r-i+1, k) 时：$r - i + 1 \leq k$ 等价于 $r + 1 \leq k + i$。
- 这部分是二维数点。

C.贼

- 总复杂度 $O(n \log^2 n)$