Solution

ExfJoe

福建省长乐第一中学

March 15, 2017

Outline

斐波那契

斐波那契

- $k = 1$：矩乘即可

斐波那契

- $k=1$：矩乘即可
- $f(1)=f(3)-f(2), f(1)+f(2)=f(3)-f(2)+f(4)-f(3)=f(4)-f(2)$

斐波那契

- $k=1$：矩乘即可
- $f(1)=f(3)-f(2), f(1)+f(2)=f(3)-f(2)+f(4)-f(3)=f(4)-f(2)$
- 注意到 $\sum_{i=1}^{n} f(i)=f(n+2)-f(2)$

斐波那契

- $k = 1$：矩乘即可
- $f(1) = f(3) - f(2), f(1) + f(2) = f(3) - f(2) + f(4) - f(3) = f(4) - f(2)$
- 注意到 $\sum_{i=1}^{n} f(i) = f(n+2) - f(2)$
- 设 $S(k, i)$ 表示第 k 维固定为 i 时 k 维超立方体内所有元素的和

斐波那契

- $k=1$：矩乘即可
- $f(1)=f(3)-f(2), f(1)+f(2)=f(3)-f(2)+f(4)-f(3)=f(4)-f(2)$
- 注意到 $\sum_{i=1}^{n} f(i)=f(n+2)-f(2)$
- 设 $S(k,i)$ 表示第 k 维固定为 i 时 k 维超立方体内所有元素的和
- 当 $i>1$ 时：$S(k,i)=S(k,i-1)+S(k,i-2)$

斐波那契

- $k=1$：矩乘即可
- $f(1)=f(3)-f(2), f(1)+f(2)=f(3)-f(2)+f(4)-f(3)=f(4)-f(2)$
- 注意到 $\sum_{i=1}^{n} f(i)=f(n+2)-f(2)$
- 设 $S(k,i)$ 表示第 k 维固定为 i 时 k 维超立方体内所有元素的和
- 当 $i>1$ 时：$S(k,i)=S(k,i-1)+S(k,i-2)$
- (想象一个 $1\times n$ 变为 $n\times n$ 变为 $n\times n\times n$ 的过程，每一层对应元素仍然满足斐波那契递推式)

斐波那契

- $k=1$：矩乘即可
- $f(1)=f(3)-f(2), f(1)+f(2)=f(3)-f(2)+f(4)-f(3)=f(4)-f(2)$
- 注意到 $\sum_{i=1}^{n} f(i)=f(n+2)-f(2)$
- 设 $S(k,i)$ 表示第 k 维固定为 i 时 k 维超立方体内所有元素的和
- 当 $i>1$ 时：$S(k,i)=S(k,i-1)+S(k,i-2)$
- (想象一个 $1\times n$ 变为 $n\times n$ 变为 $n\times n\times n$ 的过程，每一层对应元素仍然满足斐波那契递推式)
- $S(k,0)=\sum_{i=0}^{n-1} S(k-1,i)=S(k-1,n+1)-S(k-1,1)$

斐波那契

- $k=1$：矩乘即可
- $f(1)=f(3)-f(2), f(1)+f(2)=f(3)-f(2)+f(4)-f(3)=f(4)-f(2)$
- 注意到 $\sum_{i=1}^{n} f(i)=f(n+2)-f(2)$
- 设 $S(k,i)$ 表示第 k 维固定为 i 时 k 维超立方体内所有元素的和
- 当 $i>1$ 时：$S(k,i)=S(k,i-1)+S(k,i-2)$
- (想象一个 $1\times n$ 变为 $n\times n$ 变为 $n\times n\times n$ 的过程，每一层对应元素仍然满足斐波那契递推式)
- $S(k,0)=\sum_{i=0}^{n-1} S(k-1,i)=S(k-1,n+1)-S(k-1,1)$
- $S(k,1)=\sum_{i=1}^{n} S(k-1,i)=S(k-1,n+2)-S(k-1,2)$

斐波那契

斐波那契

- 答案为 $S(k+1,1)$

斐波那契

- 答案为 $S(k+1,1)$
- 考虑用矩乘形式处理：$(S(k,0),S(k,1))$

斐波那契

- 答案为 $S(k+1,1)$
- 考虑用矩乘形式处理：$(S(k,0),S(k,1))$
- 设 A 为斐波那契数列的转移矩阵

斐波那契

- 答案为 $S(k+1,1)$
- 考虑用矩乘形式处理：$(S(k,0),S(k,1))$
- 设 A 为斐波那契数列的转移矩阵
- $(S(k,0),S(k,1)) = (S(k-1,n+1),S(k-1,n+2)) - (S(k-1,1),S(k-1,2)) = (S(k-1,0),S(k-1,1)) \times (A^n - A)$

斐波那契

- 答案为 $S(k+1,1)$
- 考虑用矩乘形式处理：$(S(k,0),S(k,1))$
- 设 A 为斐波那契数列的转移矩阵
- $(S(k,0),S(k,1))=(S(k-1,n+1),S(k-1,n+2))-(S(k-1,1),S(k-1,2))=(S(k-1,0),S(k-1,1))\times(A^n-A)$
- 预处理 $B=A^n-A$，再求 $(f(0),f(1))\times(B^k)$ 即可

斐波那契

- 答案为 $S(k+1,1)$
- 考虑用矩乘形式处理：$(S(k,0),S(k,1))$
- 设 A 为斐波那契数列的转移矩阵
- $(S(k,0),S(k,1)) = (S(k-1,n+1),S(k-1,n+2)) - (S(k-1,1),S(k-1,2)) = (S(k-1,0),S(k-1,1)) \times (A^n - A)$
- 预处理 $B = A^n - A$，再求 $(f(0),f(1)) \times (B^k)$ 即可
- 即把第一维当做斐波那契数列，再把维与维之间转移当做斐波那契数列

斐波那契

- 答案为 $S(k+1,1)$
- 考虑用矩乘形式处理：$(S(k,0),S(k,1))$
- 设 A 为斐波那契数列的转移矩阵
- $(S(k,0),S(k,1)) = (S(k-1,n+1),S(k-1,n+2)) - (S(k-1,1),S(k-1,2)) = (S(k-1,0),S(k-1,1)) \times (A^n - A)$
- 预处理 $B = A^n - A$，再求 $(f(0),f(1)) \times (B^k)$ 即可
- 即把第一维当做斐波那契数列，再把维与维之间转移当做斐波那契数列
- 时间复杂度 $O(T(\log k + \log n))$

Outline

远行

远行

- 30 分：加边直接加，查询时 BFS 一下。时间复杂度 $O(NQ)$

远行

- 30 分：加边直接加，查询时 BFS 一下。时间复杂度 $O(NQ)$
- 离线：考虑先把整棵树建出来，然后再重头处理操作

远行

- 30 分：加边直接加，查询时 BFS 一下。时间复杂度 $O(NQ)$
- 离线：考虑先把整棵树建出来，然后再重头处理操作
- 对于一次查询 u，不妨令 (A,B) 为 u 所在联通块中最远的两个点，那么 u 的最远点必然是 A 或 B，由于我们提前把树建好了，所以距离可以直接求

远行

- 30 分：加边直接加，查询时 BFS 一下。时间复杂度 $O(NQ)$
- 离线：考虑先把整棵树建出来，然后再重头处理操作
- 对于一次查询 u，不妨令 (A, B) 为 u 所在联通块中最远的两个点，那么 u 的最远点必然是 A 或 B，由于我们提前把树建好了，所以距离可以直接求
- 对于加入 (u, v) 这条边，我们相当于要维护出新联通块的最远的两个点，那么显然的，这两个点必然在原来两个联通块的最远点这四个点中选，直接暴力更新即可

远行

- 30 分：加边直接加，查询时 BFS 一下。时间复杂度 $O(NQ)$
- 离线：考虑先把整棵树建出来，然后再重头处理操作
- 对于一次查询 u，不妨令 (A, B) 为 u 所在联通块中最远的两个点，那么 u 的最远点必然是 A 或 B，由于我们提前把树建好了，所以距离可以直接求
- 对于加入 (u, v) 这条边，我们相当于要维护出新联通块的最远的两个点，那么显然的，这两个点必然在原来两个联通块的最远点这四个点中选，直接暴力更新即可
- 时间复杂度 $O((N + Q) \log N)$

远行

远行

- 50 分：当强制在线时，我们没有办法预处理出树的倍增数组

远行

- 50 分：当强制在线时，我们没有办法预处理出树的倍增数组
- 事实上，在合并两个联通块时，我们可以采用启发式合并，把小的合并到大的中去

远行

- 50 分：当强制在线时，我们没有办法预处理出树的倍增数组
- 事实上，在合并两个联通块时，我们可以采用启发式合并，把小的合并到大的中去
- 比如说连接两个点 u, v，u 所在联通块比 v 的大，那么我们就以 u 为 v 的父亲，然后直接递归 v 的原来的联通块，把新的倍增数组求出来。那么每个点总共只会被递归到 $O(logN)$ 次，所以总的复杂度就是 $O(N\log^2 N + Q\log N)$

远行

- 50 分：当强制在线时，我们没有办法预处理出树的倍增数组
- 事实上，在合并两个联通块时，我们可以采用启发式合并，把小的合并到大的中去
- 比如说连接两个点 u, v，u 所在联通块比 v 的大，那么我们就以 u 为 v 的父亲，然后直接递归 v 的原来的联通块，把新的倍增数组求出来。那么每个点总共只会被递归到 $O(logN)$ 次，所以总的复杂度就是 $O(N\log^2 N + Q\log N)$
- 100 分：观察我们需要进行的操作，只是加边，然后查询某两点的距离，这显然可以用 LCT 来做，那么时间复杂度就变成了 $O((N+Q)\log N)$

Outline

珠宝

珠宝

- 20 分：基础的 01 背包，要用滚动数组。时间复杂度为 $O(NK)$

珠宝

- 20 分：基础的 01 背包，要用滚动数组。时间复杂度为 $O(NK)$
- 费用与权值一样：注意到只有 300 种费用，并且假如费用相同，那么权值也是相同的，这就相当于是一个多重背包，有经典的单调队列做法，可以做到 $O(KC)$，其中 C 为不同的费用个数

珠宝

- 20 分：基础的 01 背包，要用滚动数组。时间复杂度为 $O(NK)$
- 费用与权值一样：注意到只有 300 种费用，并且假如费用相同，那么权值也是相同的，这就相当于是一个多重背包，有经典的单调队列做法，可以做到 $O(KC)$，其中 C 为不同的费用个数
- 对于费用 s，假如最终选了 i 个这样的珠宝，那么我们必然是选择权值最大的 i 个

珠宝

- 20 分：基础的 01 背包，要用滚动数组。时间复杂度为 $O(NK)$
- 费用与权值一样：注意到只有 300 种费用，并且假如费用相同，那么权值也是相同的，这就相当于是一个多重背包，有经典的单调队列做法，可以做到 $O(KC)$，其中 C 为不同的费用个数
- 对于费用 s，假如最终选了 i 个这样的珠宝，那么我们必然是选择权值最大的 i 个
- 不妨把费用为 s 的珠宝按权值从大到小排序，令 $V(s,i)$ 为排好序后的前缀和

珠宝

- 20 分：基础的 01 背包，要用滚动数组。时间复杂度为 $O(NK)$
- 费用与权值一样：注意到只有 300 种费用，并且假如费用相同，那么权值也是相同的，这就相当于是一个多重背包，有经典的单调队列做法，可以做到 $O(KC)$，其中 C 为不同的费用个数
- 对于费用 s，假如最终选了 i 个这样的珠宝，那么我们必然是选择权值最大的 i 个
- 不妨把费用为 s 的珠宝按权值从大到小排序，令 $V(s,i)$ 为排好序后的前缀和
- 假如我们要买 i 个费用为 s 的珠宝，所得到的价值就是 $V(s,i)$

珠宝

- 20 分：基础的 01 背包，要用滚动数组。时间复杂度为 $O(NK)$
- 费用与权值一样：注意到只有 300 种费用，并且假如费用相同，那么权值也是相同的，这就相当于是一个多重背包，有经典的单调队列做法，可以做到 $O(KC)$，其中 C 为不同的费用个数
- 对于费用 s，假如最终选了 i 个这样的珠宝，那么我们必然是选择权值最大的 i 个
- 不妨把费用为 s 的珠宝按权值从大到小排序，令 $V(s,i)$ 为排好序后的前缀和
- 假如我们要买 i 个费用为 s 的珠宝，所得到的价值就是 $V(s,i)$
- 可以发现的是，我们总有 $V(s,i)-V(s,i-1) \geq V(s,i+1)-V(s,i)$，所以 $V(s)$ 是一个上凸函数 (斜率单调不升)

珠宝

珠宝

- 令 $F(s,i)$ 表示只考虑费用为 1 到 s 之间的珠宝，用掉了 i 万元，能得到的最大收益，设费用为 s 的物品有 c_s 个，那么有：

$$F(s,i)=\max_{j=0}^{\min\{c_s,\lfloor\frac{i}{s}\rfloor\}}\{F(s-1,i-sj)+V(s,j)\}$$

珠宝

- 令 $F(s,i)$ 表示只考虑费用为 1 到 s 之间的珠宝，用掉了 i 万元，能得到的最大收益，设费用为 s 的物品有 c_s 个，那么有：

$$F(s,i)=\max_{j=0}^{\min\{c_s,\lfloor\frac{i}{s}\rfloor\}}\{F(s-1,i-sj)+V(s,j)\}$$

- 我们把所有的 i 按照 $i \bmod s$ 的值分类，对于每一类分开来做

珠宝

- 令 $F(s,i)$ 表示只考虑费用为 1 到 s 之间的珠宝，用掉了 i 万元，能得到的最大收益，设费用为 s 的物品有 c_s 个，那么有：

$$F(s,i)=\max_{j=0}^{\min\{c_s,\lfloor\frac{i}{s}\rfloor\}}\{F(s-1,i-sj)+V(s,j)\}$$

- 我们把所有的 i 按照 $i \bmod s$ 的值分类，对于每一类分开来做
- 对于一个模数 j，以及一个 i 满足 $i \bmod s=j$，不妨令 $x=\frac{i-j}{s}$，我们把 x 和 $F(s-1,i)$ 变成平面上的一个点

珠宝

- 令 $F(s,i)$ 表示只考虑费用为 1 到 s 之间的珠宝，用掉了 i 万元，能得到的最大收益，设费用为 s 的物品有 c_s 个，那么有：

$$F(s,i)=\max_{j=0}^{\min\{c_s,\lfloor\frac{i}{s}\rfloor\}}\{F(s-1,i-sj)+V(s,j)\}$$

- 我们把所有的 i 按照 $i \bmod s$ 的值分类，对于每一类分开来做
- 对于一个模数 j，以及一个 i 满足 $i \bmod s=j$，不妨令 $x=\frac{i-j}{s}$，我们把 x 和 $F(s-1,i)$ 变成平面上的一个点
- 即 $(x,F(s-1,i))$，把 $F(s-1,i)$ 设为 $Y[x]$，那么对于某个 $k\geq x$，i 对 $F(s,ks+j)$ 的贡献就是 $V(s,k-x)+Y[x]$。

珠宝

珠宝

- 考虑两个点 $G[x], G[x1]$，满足 $x < x1$，我们可以分析对于哪些 $k \geq x1$，会有 $x1$ 的贡献比 x 大，就相当于

$$
\begin{aligned}
V(s, k - x1) + Y[x1] &> V(s, k - x) + Y[x] \\
Y[x1] - Y[x] &> V(s, k - x) - V(s, k - x1) \\
\frac{Y[x1] - Y[x]}{x1 - x} &> \frac{V(s, k - x) - V(s, k - x1)}{x1 - x} \\
&= \frac{V(s, k - x) - V(s, k - x1)}{(k - x) - (k - x1)}
\end{aligned}
$$

珠宝

- 考虑两个点 $G[x], G[x1]$，满足 $x < x1$，我们可以分析对于哪些 $k \geq x1$，会有 $x1$ 的贡献比 x 大，就相当于

$$
\begin{aligned}
V(s, k - x1) + Y[x1] &> V(s, k - x) + Y[x] \\
Y[x1] - Y[x] &> V(s, k - x) - V(s, k - x1) \\
\frac{Y[x1] - Y[x]}{x1 - x} &> \frac{V(s, k - x) - V(s, k - x1)}{x1 - x} \\
&= \frac{V(s, k - x) - V(s, k - x1)}{(k - x) - (k - x1)}
\end{aligned}
$$

- 之前已经证明 $V(s)$ 是一个上凸函数，满足斜率单调不升，因此，必然存在一个数 t，满足 $k > t$ 与对于 k，$x1$ 的贡献比 x 大两者互为充要条件

珠宝

- 考虑两个点 $G[x], G[x1]$，满足 $x < x1$，我们可以分析对于哪些 $k \geq x1$，会有 $x1$ 的贡献比 x 大，就相当于

$$\begin{aligned} V(s, k - x1) + Y[x1] &> V(s, k - x) + Y[x] \\ Y[x1] - Y[x] &> V(s, k - x) - V(s, k - x1) \\ \frac{Y[x1] - Y[x]}{x1 - x} &> \frac{V(s, k - x) - V(s, k - x1)}{x1 - x} \\ &= \frac{V(s, k - x) - V(s, k - x1)}{(k - x) - (k - x1)} \end{aligned}$$

- 之前已经证明 $V(s)$ 是一个上凸函数，满足斜率单调不升，因此，必然存在一个数 t，满足 $k > t$ 与对于 k，$x1$ 的贡献比 x 大两者互为充要条件
- 枚举模数 j，从小到大处理 k，用决策单调性优化 DP 的常用方法优化 DP 即可，时间复杂度 $O(N \log N + CK \log K)$