

# ACM/ICPC Template The Last Dance

## 浙江工商大学

最后一舞

May 18, 2023

## Contents

0	头文	件 · · · · · · · · · · · · · · · · · · ·	1
	0.1	header	. 1
1	字符	串	2
	1.1	hash	. 2
	1.2	KMP	
	1.3	automaton	
	-		
		manacher	
	1.5	Z-function	
	1.6	suffix-array	
	1.7	suffixAutomaton	. 7
<b>2</b>	动态	规划	ę
	2.1	01knapSack	. (
	2.2	simpleBoundedKnapSack	. (
	2.3	binaryBoundedKnapSack	. (
		monotonousQueueMultipleKnapSack	
	2.5	unboundedKnapSack	
	2.6	hybridKnapSack	
	2.7	$groupKnapSack \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	
	2.8	KnapSack2d	. 11
	2.9	treeKnapSack	. 12
	2.10	LIS	. 12
		digitDP	
		bitmaskingDP	
		quadrilateralOptimization	
		baseRingTreeDP	
		segmentTreeDP	
	2.16	LCS	. 17
_	W-10	/+ I P	
3	数据		18
3		BTree	. 18
3			. 18
3	3.1	BTree	. 18
3	3.1 3.2	BTree	. 18 . 20 . 20
3	3.1 3.2 3.3 3.4	BTree	. 18 . 20 . 20 . 21
3	3.1 3.2 3.3 3.4 3.5	BTree	. 18 . 20 . 20 . 21 . 22
3	3.1 3.2 3.3 3.4 3.5 3.6	BTree	. 18 . 20 . 20 . 21 . 22 . 23
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7	BTree	. 18 . 20 . 20 . 21 . 22 . 23
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂	. 18 . 20 . 20 . 21 . 22 . 23 . 24
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数	. 18 . 20 . 20 . 21 . 22 . 23 . 24 . 28
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS	. 18 . 20 . 20 . 21 . 22 . 23 . 24 . 28
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数	. 18 . 20 . 20 . 22 . 22 . 23 . 24 . 26
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11	BTree pbds-tree fenwick . fenwick2d SegmentTree SegmentTree  SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题	. 18 . 20 . 20 . 22 . 23 . 24 . 26 . 28 . 30
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12	BTree pbds-tree fenwick . fenwick2d SegmentTree . SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护连续区间异或值	. 18 . 20 . 20 . 22 . 22 . 22 . 24 . 26 . 28 . 30
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13	BTree pbds-tree fenwick . fenwick2d SegmentTree . SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护连续区间异或值 线段树维护区间异或	. 18 . 20 . 20 . 21 . 22 . 22 . 24 . 26 . 30 . 31
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13	BTree pbds-tree fenwick . fenwick2d SegmentTree . SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护连续区间异或值 线段树维护区间异或 LazySegmentTree	. 18 . 20 . 20 . 21 . 22 . 22 . 24 . 28 . 30 . 31 . 32 . 33
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护连续区间异或值 线段树维护区间异或 LazySegmentTree SparseTable	. 18 . 20 . 20 . 21 . 22 . 25 . 26 . 28 . 30 . 31 . 32 . 33
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree  SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护面队问题 线段树维护百溪区间异或值 线段树维护区间异或 LazySegmentTree SparseTable CartesianTree	. 18 . 20 . 20 . 21 . 22 . 22 . 28 . 30 . 31 . 32 . 33 . 33
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17	BTree pbds-tree fenwick fenwick. fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护连续区间异或值 线段树维护区间异或 LazySegmentTree SparseTable CartesianTree DancingLinks	. 18 . 20 . 20 . 21 . 22 . 22 . 28 . 26 . 30 . 31 . 32 . 33 . 33
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree  SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护面队问题 线段树维护百溪区间异或值 线段树维护区间异或 LazySegmentTree SparseTable CartesianTree	. 18 . 20 . 20 . 21 . 22 . 22 . 28 . 26 . 30 . 31 . 32 . 33 . 33
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18	BTree pbds-tree fenwick fenwick. fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护连续区间异或值 线段树维护区间异或 LazySegmentTree SparseTable CartesianTree DancingLinks	. 18 . 20 . 20 . 21 . 22 . 22 . 25 . 26 . 30 . 31 . 33 . 33 . 33 . 33 . 33
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.19	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护语队问题 线段树维护语队问题 线段树维护已间异或 LazySegmentTree SparseTable CartesianTree DancingLinks ChthollyTree	. 18 . 20 . 20 . 21 . 22 . 25 . 25 . 26 . 30 . 31 . 35 . 35 . 35 . 36 . 37 . 38
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.19 3.20	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护插队问题 线段树维护插队问题 线段树维护面队宫脚穿 LIS 线段树维护区间异或 LazySegmentTree SparseTable CartesianTree DancingLinks ChthollyTree monotonousQueue monotonousStack	. 18 . 20 . 20 . 21 . 22 . 25 . 26 . 26 . 30 . 31 . 35 . 35 . 35 . 36 . 40 . 41
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.19 3.20 3.21	BTree pbds-tree fenwick fenwick2d SegmentTree. SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护通小字典序 LIS 线段树维护通小字典序 LIS 线段树维护语队问题 线段树维护百队问题 线段树维护区间异或值 线段树维护区间异或 LazySegmentTree SparseTable CartesianTree DancingLinks ChthollyTree monotonousQueue monotonousStack difference	. 18 . 20 . 20 . 21 . 22 . 23 . 24 . 26 . 30 . 31 . 33 . 35 . 35 . 37 . 39 . 40 . 41 . 42
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.20 3.21 3.22	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree SegmentTree SegmentTree  \$ ValueSegmentTree \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$	. 18 . 20 . 20 . 21 . 22 . 23 . 24 . 26 . 30 . 31 . 33 . 35 . 35 . 37 . 39 . 40 . 41 . 42 . 43
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.20 3.21 3.22 3.23	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree SegmentTree SegmentTree  \$ Ag My 和 亦开点合并分裂 \$ 我段树维护 LIS 方案数 \$ 我段树维护最小写典序 LIS \$ 我段树维护插队问题 \$ 我段树维护连续区间异或值 \$ 我段树维护区间异或	. 18 . 20 . 20 . 22 . 22 . 22 . 28 . 30 . 31 . 32 . 33 . 35 . 35 . 37 . 40 . 41 . 42 . 43
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.20 3.21 3.22 3.23 3.24	BTree pbds-tree fenwick fenwick. fenwick2d SegmentTree SegmentTree SegmentTree  女良树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护摄小字典序 LIS 线段树维护语队问题 线段树维护语队问题 线段树维护区间异或值 线段树维护区间异或值 CartesianTree SparseTable CartesianTree DancingLinks ChthollyTree monotonousQueue monotonou	. 18 . 20 . 20 . 22 . 22 . 22 . 28 . 30 . 31 . 32 . 33 . 35 . 37 . 37 . 40 . 41 . 42 . 43 . 44
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.20 3.21 3.22 3.23 3.24 3.25	BTree pbds-tree fenwick fenwick2d SegmentTree SegmentTree2d ValueSegmentTree 线段树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护插队问题 线段树维护语域区间异或值 线段树维护连续区间异或值 线段树维护连续区间异或值 CartesianTree DancingLinks ChthollyTree monotonousQueue monotonousQueue monotonousStack difference trie HashTable 2-4 维前缀和 simpleDSU	. 18 . 20 . 20 . 22 . 22 . 22 . 28 . 30 . 33 . 33 . 33 . 35 . 37 . 40 . 41 . 42 . 44 . 44
3	3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.19 3.20 3.21 3.22 3.23 3.24 3.25 3.26	BTree pbds-tree fenwick fenwick. fenwick2d SegmentTree SegmentTree SegmentTree  女良树动态开点合并分裂 线段树维护 LIS 方案数 线段树维护最小字典序 LIS 线段树维护摄小字典序 LIS 线段树维护语队问题 线段树维护语队问题 线段树维护区间异或值 线段树维护区间异或值 CartesianTree SparseTable CartesianTree DancingLinks ChthollyTree monotonousQueue monotonou	. 18 . 20 . 20 . 21 . 22 . 22 . 25 . 26 . 30 . 31 . 32 . 33 . 35 . 37 . 37 . 40 . 41 . 42 . 44 . 44 . 44 . 44

## ACM/ICPC Template The Last Dance, 最后一舞

		varietyDSU												47
		KD 求矩阵权值和 .												48
		KD 求最近点对距离												50
		CDQ 分治												51
	3.32	$\operatorname{cdq}$ 处理逆序数 $\ldots$			 	 	 	 		 	 	 		51
	3.33	cdq 处理二维偏序 .			 	 	 	 		 	 	 		52
	3.34	cdq 套 cdq 处理三维	[偏序		 	 	 	 		 	 	 		53
	3.35	cdq 套树状数组处理	三维偏	序.	 	 	 	 		 	 	 		54
		cdq 维护矩阵内二维												55
		advanced01trie												56
		advancedArray												57
		anvancedDSU												58
		扫描线求面积并												60
		扫描线求周长并。												61
		石油线水周式开 · · · 区间第 k 小 · · · · ·												63
		区间前 k 大												64
		树套树维护三维偏序												65
		线段树套主席树-二维												67
		Scapegoat												69
		Splay												72
	3.48	01trie			 	 	 	 		 	 	 		77
	3.49	${\rm Treap}  \dots  \dots$			 	 	 	 		 	 	 		78
4	博弈													80
	4.1													
	4.2	fibonacci												
	4.3	nim			 	 	 	 		 	 	 		80
	4.4	$ wythoff \dots \dots \dots $			 	 	 	 		 	 	 		80
	4.5	$\operatorname{sgFunction} \ldots$			 	 	 	 		 	 	 		81
5		森林												82
5		$forest \dots \dots \dots$												82
5		$\begin{array}{cccccccccccccccccccccccccccccccccccc$			 	 	 	 		 	 	 		
5	5.1	$forest \dots \dots \dots$			 	 	 	 		 	 	 		82
5	5.1 5.2	$\begin{array}{cccccccccccccccccccccccccccccccccccc$			 	 	 	 		  	 	 		82 82
5	5.1 5.2 5.3	forest dfs-forest		· · ·	 	 	 	 	 	  	 	  	 	82 82 84
5	5.1 5.2 5.3 5.4	forest			 	 	 	 	  	 	   	 	 	82 82 84 85
5	5.1 5.2 5.3 5.4 5.5	forest			 	 	 	 	  		   	 	 	82 82 84 85 87
5	5.1 5.2 5.3 5.4 5.5 5.6	forest			 	 	 	 	· · · · · · · · ·		 	 	 	82 82 84 85 87 89
5	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8	forest			 	 	 	 			 	 	 	82 82 84 85 87 89 90
5	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9	forest	· · · · · · · · · · · · · · · ·		 			 			 	 · · · · · · · · · · · · · · · · · · ·	 	82 82 84 85 87 89 90 93
5	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10	forest	· · · · · · · · · · · · · · · · · · ·		 			 			 	 	 	82 82 84 85 87 89 90 93
5	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11	forest	····· ···· ···· !息 ·		 						 	 		82 82 84 85 87 89 90 93 94
5	$5.1 \\ 5.2 \\ 5.3 \\ 5.4 \\ 5.5 \\ 5.6 \\ 5.7 \\ 5.8 \\ 5.9 \\ 5.10 \\ 5.11 \\ 5.12$	forest	·····································		 						 			822 844 855 877 899 900 933 944 959
5	$5.1 \\ 5.2 \\ 5.3 \\ 5.4 \\ 5.5 \\ 5.6 \\ 5.7 \\ 5.8 \\ 5.9 \\ 5.10 \\ 5.11 \\ 5.12$	forest	·····································		 						 			82 82 84 85 87 89 90 93 94
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13	forest	·····································		 						 			82 82 84 85 87 89 90 93 94 95 96
	5.1       5.2       5.3       5.4       5.5       5.6       5.7       5.8       5.9       5.10       5.11       5.12       5.13	forest	·····································											822 844 855 877 899 900 933 944 959
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13	forest	· · · · · · · · · · · · · · · · · · ·											82 82 84 85 87 89 90 93 94 95 96 98
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数学</b>	forest	· · · · · · · · · · · · · · · · · · ·											82 82 84 85 87 89 90 93 94 95 96 98 <b>100</b> 100
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数</b> 6.1 6.2 6.3	forest												82 82 84 85 87 89 90 93 94 95 96 100 103 105
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 6.1 6.2 6.3 6.4	forest												822 828 844 855 878 899 90 93 94 95 96 98 100 103 105 105
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 6.1 6.2 6.3 6.4 6.5	forest												82 82 84 85 87 89 90 93 94 95 96 98 <b>100</b> 103 105 110
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数</b> : 6.1 6.2 6.3 6.4 6.5 6.6	forest												82 82 84 85 87 89 90 93 94 95 96 98 100 103 105 110 110
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数</b> : 6.2 6.3 6.4 6.5 6.6 6.7	forest												822 824 844 855 877 899 90 93 944 95 96 98 100 103 105 110 1110 1111
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数</b> : 6.2 6.3 6.4 6.5 6.6 6.7 6.8	forest												82 82 84 85 87 89 90 93 94 95 96 98 105 105 110 111 111
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数</b> : 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9	forest												822 824 844 855 877 899 9093 944 955 966 988 1005 1105 1110 1111 1111 1111
	5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 5.10 5.11 5.12 5.13 <b>数</b> 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 6.10	forest												82 82 84 85 87 89 90 93 94 95 96 98 100 103 105 110 111 111 111 112
	5.15.25.35.45.55.65.75.85.95.105.125.136.26.36.46.56.66.76.86.96.106.11	forest												82 82 84 85 87 89 90 93 94 95 96 98 100 103 105 110 111 111 112 112 113
	5.15.25.35.45.55.65.75.85.95.105.125.136.26.36.46.56.66.76.86.96.106.11	forest												82 82 84 85 87 89 90 93 94 95 96 98 100 103 105 110 111 111 111 112

## ACM/ICPC Template The Last Dance, 最后一舞

	常见积性函数	
6.15	Miller-Rabin	116
6.16	quadraticResidue	117
6.17	bags	119
6.18	EX-BSGS	
	CRT	
00	EX-CRT	
	EX-Lucas	
	min25	
	BM	
	duSieve	
	反演相关	
	simpson	
	Bell	
	Catalan	
6.29	Lucas	131
	伯努利数	
6.31	步移-组合数前缀和	
6.32	康托展开	
6.33	模数非质数的组合	137
6.34	k 次最小置换复原	138
	Striling	
	第二类斯特林数-行	
	第二类斯特林数-列	
	第一类斯特林数-行	
	第一类期待林数-列	
0.59	第一 <del>次则行</del> 桥数-列	145
	普通型母函数	
	指数型母函数	
	伯努利数求和	
	$fibonacci BigInteger \ \dots $	
	fibonacciCircular	
	Matrix	
	拉格朗日插值求和	
6.48	SolveLinearSystem	159
6.49	矩阵求逆	160
6.50	eraseLinearBasis	161
	simpleLinearBasis	
	intervalModifyLinearBasis	
	noIntervalModifyLinearBasis	
	FWT	
	cdq 分治 FFT	
	27/2007	
	多项式快速幂	
	多项式除法、取模	
	多项式 ln-Exp-Pow	
	任意模数 MTT-拆系数法	
	任意模数 NTT-三模数法	
	多项式优化常系数齐次线性递推	
	FFT 加速带有通配符字符串匹配	
	FFT 加速朴素字符串匹配	
	x 不连续、暴力插值	
	x 连续、前缀优化	
	多项式 ln-exp-pow 处理边界为 1	
	二次剩余处理边界不为 1	
	二维几何。	
	三维几何	
	一年/ いり・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	
0.12	1 Oiy-2	199

## ACM/ICPC Template The Last Dance, 最后一舞

	6.73	多项式快速幂	4
		Geometry	
7	图论		_
	7.1	graph	
	7.2	isBipartiteGraph	
	7.3	hungry	
	7.4	KM	
	7.5	galeShapley	
	7.6	bellman-ford	4
	7.7	dijkatra	5
	7.8	floyd	5
	7.9	spfa	6
	7.10	Kruskal	6
	7.11	prim	6
	7.12	topSort	7
		scc	
		cycles	
		ternaryRingCount	
		eulerian-path	
		twoSat	
		maxAssignment	
		tarjan	
		biconnected-components	
		differenceConstraints	
		AHU	
		Astar	
		dinic	
		ISAP	
		mcmf	
		conclusion.md	
	1.21	Conclusion.ind	O
8	其他	23	7
	8.1	simpleMO	7
	8.2	modifyMO	
	8.3	rollbackMO	
	8.4	treeMO	
	8.5	fastIO	
	8.6	simulatedAnnealing	
	8.7	Cantor	
	8.8	BigInteger	
	8.9	debug	
		1000000000000000000000000000000000000	
	0.10	1110120	$\sim$

## 0 头文件

#### 0.1 header

```
#include "bits/stdc++.h"
  using namespace std;
3
 4 typedef long long ll;
  typedef long double ld;
5
6 typedef unsigned long long ull;
7 typedef vector<ll> VI;
8 typedef pair<int, int> pii;
9 typedef pair<double, double> pdd;
10 typedef pair<ll, ll> pll;
12 #define endl "\n"
13 #define fi first
14 #define se second
15 #define eb emplace_back
16 #define mem(a, b) memset(a, b, sizeof(a))
17
18 const ll INF = 0x3f3f3f3f;
19 const ll mod = 998244353;
20 const double eps = 1e-6;
21 const double PI = acos(-1);
22 const double R = 0.57721566490153286060651209;
23
24 void solve() {
25
26 }
27
28 signed main() {
        ios_base::sync_with_stdio(false);
29
        // cin.tie(nullptr);
30
       // cout.tie(nullptr);
31
   #ifdef FZT_ACM_LOCAL
32
        freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
33
34
        signed test_index_for_debug = 1;
35
        char acm_local_for_debug = 0;
36
        do {
37
            if (acm_local_for_debug == '$') exit(0);
38
            if (test_index_for_debug > 20)
39
                throw runtime_error("Check the stdin!!!");
40
            auto start_clock_for_debug = clock();
41
            solve();
42
            auto end_clock_for_debug = clock();
43
            cout << "Test " << test_index_for_debug << " successful" << endl;
cerr << "Test " << test_index_for_debug++ << " Run Time: "</pre>
44
45
                 << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<</pre>
46
        "s" << endl;
                       -----" << endl:
            cout << "-
47
        } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
48
   #else
49
        solve();
50
   #endif
51
        return 0;
52
53
```

### 1 字符串

#### 1.1 hash

```
struct Hash {
1
   using ui64 = unsigned long long;
2
3
        static constexpr int P = 1331;
4
        std::vector<ui64> h, p;
5
6
        Hash(std::string s) : h(s.size() + 1), p(s.size() + 1) {
7
            p[0] = 1;
            for (int i = 1; i <= (int) s.size(); i++) {
8
                p[i] = p[i - 1] * P;
h[i] = h[i - 1] * P + s[i - 1] - '0';
9
10
            }
11
12
        }
13
        ui64 rangeSum(int l, int r) {
14
            return h[r] - h[l - 1] * p[r - l + 1];
15
16
17 };
    1.2 KMP
   template <typename T>
   std::vector<int> kmp_table(int n, const T &s) {
3
        std::vector<int> p(n, 0);
        int k = 0;
4
        for (int i = 1; i < n; i++) {
5
            while (k > 0 \& !(s[i] == s[k])) {
6
7
                k = p[k - 1];
8
            if (s[i] == s[k]) {
9
10
                k++;
11
12
            p[i] = k;
13
14
        return p;
15
   }
16
17
   template <typename T>
   std::vector<int> kmp_table(const T &s) {
18
        return kmp_table((int) s.size(), s);
19
  }
20
21
   template <typename T>
22
   std::vector<int> kmp_search(int n, const T &s, int m, const T &w, const std::vector<int
       > &p) {
        assert(n >= 1 && (int) p.size() == n);
24
        std::vector<int> res;
25
26
        int k = 0;
        for (int i = 0; i < m; i++) {
27
28
            while (k > 0 \&\& (k == n || !(w[i] == s[k]))) {
                k = p[k - 1];
29
30
            if (w[i] == s[k]) {
31
32
                k++;
33
            }
```

```
if (k == n) {
34
                res.push_back(i - n + 1);
35
36
        }
37
38
        return res;
        // returns 0-indexed positions of occurrences of s in w
39
   }
40
41
   template <typename T>
42
   std::vector<int> kmp_search(const T &s, const T &w, const std::vector<int> &p) {
        return kmp_search((int) s.size(), s, (int) w.size(), w, p);
44
45
   1.3 automaton
   struct Automaton {
        static constexpr int ALPHABET_SIZE = 26;
2
3
        std::vector<std::vector<int>> tr;
        std::vector<int> e;
4
        std::vector<int> fail;
5
        int tot;
6
7
        Automaton(int n): tr(n, std::vector<int>(ALPHABET_SIZE)), e(n), fail(n), tot(0) {}
8
        Automaton(int m, std::vector<std::string> s) : Automaton(m) {
9
10
            for(int i = 0;i < (int) s.size(); i++) {</pre>
11
                insert(s[i]);
12
13
            build();
14
        }
15
16
        void insert(std::string s) {
            int u = 0;
17
18
            for(int i = 0;i < (int) s.size(); i++) {</pre>
                if(!tr[u][s[i] - 'a']) tr[u][s[i] - 'a'] = ++tot;
19
                u = tr[u][s[i] - 'a'];
20
21
22
            e[u]++;
23
        }
24
25
        void build() {
            std::queue<int> q;
26
            for(int i = 0;i < 26; i++) {</pre>
27
28
                if(tr[0][i]) q.push(tr[0][i]);
29
30
            while(q.size()) {
                int u = q.front();
31
32
                q.pop();
                for(int i = 0;i < 26; i++) {
33
                     if(tr[u][i]) {
34
                         fail[tr[u][i]] = tr[fail[u]][i];
35
36
                         q.push(tr[u][i]);
                     } else {
37
38
                         tr[u][i] = tr[fail[u]][i];
39
40
                }
            }
41
        }
42
43
```

```
int query(std::string t) {
44
            int u = 0, res = 0;
45
            for(int i = 0;i < (int) t.size(); i++) {</pre>
46
                u = tr[u][t[i] - 'a'];
for(int j = u;j && e[j] != -1; j = fail[j]) {
47
48
                     res += e[j], e[j] = -1;
49
50
            }
51
52
            return res;
53
        }
  };
54
   1.4 manacher
   template <typename T>
1
   std::vector<int> manacher(int n, const T &s) {
3
        if (n == 0) {
            return std::vector<int>();
4
5
6
        std::vector<int> res(2 * n - 1, 0);
        int l = -1, r = -1;
7
        for (int z = 0; z < 2 * n - 1; z++) {
8
            int i = (z + 1) >> 1;
9
10
            int j = z >> 1;
            int p = (i >= r ? 0 : std::min(r - i, res[2 * (l + r) - z]));
11
            while (j + p + 1 < n \& i - p - 1 >= 0) {
12
                if (!(s[j + p + 1] == s[i - p - 1])) {
13
                    break;
14
                }
15
16
                p++;
17
            if (j + p > r) {
18
                ĺ = i - p;
19
20
                r = j + p;
21
            res[z] = p;
22
23
        return res;
24
25
   }
26
27 template <typename T>
   std::vector<int> manacher(const T &s) {
28
        return manacher((int) s.size(), s);
29
30 }
        Z-function
1 template <typename T>
   std::vector<int> z_function(int n, const T &s) {
3
        std::vector<int> z(n, n);
        int l = 0, r = 0;
4
        for (int i = 1; i < n; i++) {
5
            z[i] = (i > r ? 0 : std::min(r - i + 1, z[i - l]));
6
            while (i + z[i] < n \& s[z[i]] == s[i + z[i]]) {
7
8
                z[i]++;
9
            if (i + z[i] - 1 > r) {
10
```

```
l = i;
11
                r = i + z[i] - 1;
12
            }
13
14
15
        return z;
16
   }
17
   template <typename T>
18
   std::vector<int> z_function(const T &s) {
19
20
        return z_function((int) s.size(), s);
21
22
   template <typename T>
23
   std::vector<int> z_search(int n, const T &s, int m, const T &w, std::vector<int> &z) {
24
        assert(n >= 1 \&\& (int) z.size() == n);
25
        std::vector<int> p(m);
26
        int l = 0, r = -1;
27
        for (int i = 0; i < m; i++) {
28
            p[i] = (i > r ? 0 : std::min(r - i + 1, z[i - l]));
29
            while (i + p[i] < m \& p[i] < n \& s[p[i]] == w[i + p[i]]) {
30
                p[i]++;
31
32
            if (i + p[i] - 1 > r) {
33
                l = i;
34
                r = i + p[i] - 1;
35
36
            }
37
        }
38
        return p;
   };
39
40
   template <typename T>
41
   std::vector<int> z_search(const T &s, const T &w, std::vector<int> &z) {
42
        return z_search((int) s.size(), s, (int) w.size(), w, z);
43
44 };
   1.6 suffix-array
   template <typename T>
2
   std::vector<int> suffix_array(int n, const T &s, int char_bound) {
3
        std::vector<int> a(n);
4
        if (n == 0) {
            return a;
5
6
        if (char_bound != -1) {
7
            std::vector<int> aux(char_bound, 0);
8
            for (int i = 0; i < n; i++) {
9
10
                aux[s[i]]++;
            }
11
            int sum = 0;
12
            for (int i = 0; i < char_bound; i++) {
13
14
                int add = aux[i];
                aux[i] = sum;
15
                sum += add;
16
17
            }
            for (int i = 0; i < n; i++) {
18
19
                a[aux[s[i]]++] = i;
20
        } else {
21
```

```
iota(a.begin(), a.end(), 0);
22
            sort(a.begin(), a.end(), [&s](int i, int j) { return s[i] < s[j]; });</pre>
23
24
       std::vector<int> sorted_by_second(n);
25
26
       std::vector<int> ptr_group(n);
       std::vector<int> new_group(n);
27
28
       std::vector<int> group(n);
29
       group[a[0]] = 0;
       for (int i = 1; i < n; i++) {
30
            group[a[i]] = group[a[i - 1]] + (!(s[a[i]] == s[a[i - 1]]));
31
32
33
       int cnt = group[a[n - 1]] + 1;
34
       int step = 1;
       while (cnt < n) {</pre>
35
            int at = 0;
36
            for (int i = n - step; i < n; i++) {</pre>
37
38
                sorted_by_second[at++] = i;
39
            for (int i = 0; i < n; i++) {
40
                if (a[i] - step >= 0) {
41
                    sorted_by_second[at++] = a[i] - step;
42
43
44
            for (int i = n - 1; i >= 0; i--) {
45
46
                ptr_group[group[a[i]]] = i;
47
            for (int i = 0; i < n; i++) {
48
                int x = sorted_by_second[i];
49
                a[ptr\_group[group[x]]++] = x;
50
51
            new_aroup[a[0]] = 0;
52
            for (int i = 1; i < n; i++) {
53
                if (group[a[i]] != group[a[i - 1]]) {
54
                    new_group[a[i]] = new_group[a[i - 1]] + 1;
55
                } else {
56
                    int pre = (a[i - 1] + step >= n ? -1 : group[a[i - 1] + step]);
57
                    int cur = (a[i] + step >= n ? -1 : group[a[i] + step]);
58
59
                    new\_group[a[i]] = new\_group[a[i - 1]] + (pre != cur);
                }
60
61
            swap(group, new_group);
62
            cnt = group[a[n - 1]] + 1;
63
            step <<= 1;
64
65
       return a;
66
   }
67
68
69 template <typename T>
   std::vector<int> suffix_array(const T &s, int char_bound) {
70
71
        return suffix_array((int) s.size(), s, char_bound);
72 }
73
74
   template <typename T>
   std::vector<int> build_lcp(int n, const T &s, const std::vector<int> &sa) {
75
       assert((int) sa.size() == n);
76
77
       std::vector<int> pos(n);
78
       for (int i = 0; i < n; i++) {
79
            pos[sa[i]] = i;
80
       }
```

```
std::vector<int> lcp(std::max(n - 1, 0));
81
        int k = 0;
82
        for (int i = 0; i < n; i++) {
83
             k = std::max(k - 1, 0);
84
             if (pos[i] == n - 1) {
85
86
                 k = 0;
             } else {
87
                 int j = sa[pos[i] + 1];
88
                 while (i + k < n \&\& j + k < n \&\& s[i + k] == s[j + k]) {
89
90
                     k++;
91
92
                 lcp[pos[i]] = k;
             }
93
        }
94
95
        return lcp;
96
97
   template <typename T>
98
    std::vector<int> build_lcp(const T &s, const std::vector<int> &sa) {
99
        return build_lcp((int) s.size(), s, sa);
100
101 }
    1.7 suffixAutomaton
    struct SuffixAutomaton {
 2
        static constexpr int ALPHABET_SIZE = 26, N = 1e4;
 3
        struct Node {
             int len;
 4
 5
             int link;
             int next[ALPHABET_SIZE];
 6
             int siz;
 7
             Node() : len(0), link(0), next{} {}
 8
 9
        } t[2 * N];
10
        int cntNodes;
        SuffixAutomaton() {
11
             cntNodes = 1;
12
13
             std::fill(t[0].next, t[0].next + ALPHABET_SIZE, 1);
14
             t[0].len = -1;
15
16
        int extend(int p, int c) {
             if (t[p].next[c]) {
17
18
                 int q = t[p].next[c];
19
                 if (t[q].len == t[p].len + 1)
                     return q;
20
21
                 int r = ++cntNodes;
                 t[r].len = t[p].len + 1;
22
                 t[r].link = t[q].link;
23
                 std::copy(t[q].next, t[q].next + ALPHABET_SIZE, t[r].next);
24
25
                 t[q].link = r;
                 while (t[p].next[c] == q) {
26
27
                     t[p].next[c] = r;
                     p = t[p].link;
28
29
                 }
30
                 return r;
             }
31
32
             int cur = ++cntNodes;
             t[cur].len = t[p].len + 1; t[cur].siz++;
33
             while (!t[p].next[c]) {
34
```

## 2 动态规划

18 19

20

return dp.back();

#### 2.1 01knapSack

```
template <typename T>
   T knapSack(const std::vector<int>& v, const std::vector<T>& w, int V) {
       int n = (int) v.size();
3
       std::vector<T> dp(V + 1);
4
       for (int i = 0; i < n; i++) {
5
            for (int j = V; j >= v[i]; j--) {
6
7
                dp[j] = std::max(dp[j], dp[j - v[i]] + w[i]);
8
9
       return dp.back();
10
11 }
        simpleBoundedKnapSack
   template <typename T>
   T simpleMultipleKnapSack(const std::vector<int>& v, const std::vector<T>& w, const std
       ::vector<int>& cnt, int V) {
       int n = (int) v.size();
3
       std::vector<T> dp(V + 1);
4
       for (int i = 0; i < n; i++) {
5
6
            for (int j = V; j >= 0; j--) {
                for (int k = 1; k <= cnt[i]; k++) {</pre>
7
                    if (j - k * v[i] < 0) break;
8
                    dp[j] = std::max(dp[j], dp[j - k * v[i]] + k * w[i]);
9
10
                }
           }
11
12
       return dp.back();
13
   }
14
   2.3 binaryBoundedKnapSack
   template <typename T>
   T binaryMultipleKnapSack(const std::vector<int>& v, const std::vector<T>& w, const std
       ::vector<int>& cnt, int V) {
3
       int n = (int) v.size();
       std::vector<T> dp(V + 1);
4
       std::vector<std::array<T, 2>> bags;
5
       for (int i = 0; i < n; i++) {
6
            for (int k = 1; k <= cnt[i]; k <<= 1) {</pre>
7
                cnt[i] -= k;
8
                bags.push_back({v[i] * k, w[i] * k});
9
10
           if (cnt[i] > 0) {
11
                bags.push_back({v[i] * cnt[i], w[i] * cnt[i]});
12
           }
13
14
       for (auto& [nv, nw] : bags) {
15
           for (int j = V; j >= nv; j--) {
16
                dp[j] = std::max(dp[j], dp[j - nv] + nw);
17
```

21 }

10

#### 2.4 monotonousQueueMultipleKnapSack

```
1 // no correct
   template <typename T>
  T monotonousQueueMultipleKnapSack(const std::vector<int>& v, const std::vector<T>& w,
       const std::vector<int>& cnt, int V) {
       int n = (int) v.size();
4
       std::vector<T> dp(V + 1);
5
       std::vector<int> que(V);
6
       for (int i = 0; i < n; i++) {
7
            std::vector<T> ndp(V + 1);
8
9
            for (int j = 0; j < v[i]; j++) {
                int head = 1, tail = 0;
10
                for (int k = j; k \leftarrow V; k += v[i]) {
11
                    while (head <= tail && que[head] < k - cnt[i] * v[i]) head++;</pre>
12
13
                    if (head \leftarrow tail) ndp[k] = std::max(dp[k], dp[que[head]] + (k - que[
       head]) / v[i] * w[i]);
                    while (head <= tail && dp[k] >= dp[que[tail]] + (k - que[head]) / v[i]
14
       * w[i]) tail--;
                    que[++tail] = k;
15
16
17
            dp = std::move(ndp);
18
19
20
       return dp.back();
21
  }
        unboundedKnapSack
   template <typename T>
   T entireKnapSack(const std::vector<int>& v, const std::vector<T>& w, int V) {
2
3
       int n = (int) v.size();
4
       std::vector<T> dp(V + 1);
5
       for (int i = 0; i < n; i++) {
            for (int j = v[i]; j <= V; j++) {</pre>
6
                dp[j] = std::max(dp[j], dp[j - v[i]] + w[i]);
7
8
9
10
       return dp.back();
11
  }
   2.6 hybridKnapSack
   template <typename T>
   T hybridKnapSack(const std::vector<int>& v, const std::vector<T>& w, const std::vector<
       int>& cnt, int V) {
3
       int n = (int) v.size();
       std::vector<T> dp(V + 1);
4
       std::vector<std::tuple<int, T, int> > bags;
5
6
       for (int i = 0; i < n; i++) {
7
            if (cnt[i] < 0) {</pre>
8
                bags.push_back(std::make_tuple(v[i], w[i], -1));
9
            } else if (cnt[i] == 0) {
```

bags.push\_back(std::make\_tuple(v[i], w[i], 0));

```
} else {
11
                for (int k = 1; k <= cnt[i]; k <<= 1) {</pre>
12
13
                    cnt[i] -= k;
                    bags.push_back(std::make_tuple(v[i] * k, w[i] * k, -1));
14
15
                if (cnt[i] > 0) {
16
                    bags.push_back(std::make_tuple(v[i] * cnt[i], w[i] * cnt[i], -1));
17
                }
18
            }
19
20
        for (auto& bag : bags) {
21
22
            int v, op; T w;
            std::tie(v, w, op) = bag;
23
            if (op == -1) {
24
                for (int j = V; j >= v; j--) {
25
                    dp[j] = std::max(dp[j], dp[j - v] + w);
26
                }
27
28
            } else {
                for (int j = v; j \leftarrow V; j++) {
29
30
                    dp[j] = std::max(dp[j], dp[j - v] + w);
                }
31
            }
32
33
34
        return dp.back();
35 }
   2.7 groupKnapSack
   template <typename T>
2
   T groupKnapSack(const std::vector<std::array<int, 2>>>& groups, int V) {
3
        std::vector<T> dp(V + 1);
4
        for (auto& group : groups) {
            for (int j = V; j >= 0; j--) {
5
                for (auto& [v, w] : group) {
6
7
                    if (j >= v) {
                        dp[j] = std::max(dp[j], dp[j - v] + w);
8
                    }
9
                    /*
10
11
                     if (j >= v) {
12
                         // from groups[i], ensure dp[i][j-v] has at least one item because
       dp[i][j-v] is not -1
                        if (\sim dp[i][j - v]) dp[i][j] = std::max(dp[i][j], dp[i][j - v] + w);
13
                        if (\sim dp[i - 1][j - v]) dp[i][j] = std::max(dp[i][j], dp[i - 1][j - v])
14
       \vee  + w);
                         // from groups[i - 1], ensure dp[i-1][j-v] has at least one item
15
       because dp[i-1][j-v] is not -1
                         // the order is not swap
16
                     }
*/
17
18
                }
19
            }
20
21
22
        return dp.back();
23
   }
```

#### 2.8 KnapSack2d

```
template <typename T>
   T twoDimensionalKnapSack(const std::vector<int>& v, const std::vector<int>& m, const
       std::vector<T>& w, int V, int M) {
       int n = (int) v.size();
3
       std::vector<std::vector<T>> dp(V + 1, std::vector<T>(M + 1));
4
       for (int i = 0; i < n; i++) {
5
            for (int j = V; j >= v[i]; j--) {
6
                for (int k = M; k >= m[i]; k--) {
7
                    dp[j][k] = std::max(dp[j][k], dp[j - v[i]][k - m[i]] + w[i]);
8
                }
9
10
            }
11
       }
       return dp.back().back();
12
13
   2.9 treeKnapSack
   template <typename T>
   T treeKnapSack(const forest<T>& q) {
       std::vector<int> siz(n);
3
       std::vector<std::vector<T>> dp(n);
4
5
       std::function<void(int, int)> dfs = [&](int u, int fa) {
6
            siz[u] = 1;
7
            for (int id : g.g[u]) {
8
                auto& e = g.edges[id];
9
                int to = e.from ^ e.to ^ u;
10
                if (to == fa) continue;
11
12
                dfs(to, u);
                int now = min(siz[u] + siz[v] + 1, M);
13
                int t[MAX_M]; for (int i = 0; i <= M; i++) t[i] = INF/-INF;//初始化
14
                for (int i = 0; i \le siz[u]; i++)
15
                    for (int j = 0; j \le Siz[v] \&\& i + j \le M; j++) {
16
                        //...转移方程
17
18
19
                for (int i = 0; i <= now; i++) f[u][i] = min/max(f[u][i], t[i]);
                siz[u] = now;
20
21
           }
22
       };
  }
23
   2.10 LIS
   template <typename T>
   int lis(const std::vector<T>& a) {
       std::vector<T> u;
3
4
       for (const T& x : a) {
            auto it = upper_bound(u.begin(), u.end(), x);
5
            if (it == u.end()) {
6
                u.push_back(x);
7
           } else {
8
                *it = x;
9
10
11
12
       return (int) u.size();
13 }
```

#### 2.11 digitDP

```
1 // the numbers of exist 49 in 0~n
   template <typename T>
   T digitDP(T n) {
3
4
       T x = n;
       std::vector<int> digit;
5
       while (x) {
6
           digit.push_back(x % 10);
7
           x /= 10;
8
       }
9
10
       std::vector<std::vector<T>> dp(digit.size(), std::vector<T>(10));
11
       std::function<T(int, int, bool limit)> dfs = [&](int pos, int pre, bool limit) ->
12
       int {
           if (pos == -1) return 1;
13
           if (!limit && dp[pos][pre]) return dp[pos][pre];
14
           int up = limit ? digit[pos] : 9;
15
           T ans = 0;
           for (int i = 0; i <= up; i++) {
17
               if (pre == 4 && i == 9) {
18
19
                    continue;
               }
20
               ans += dfs(pos - 1, i, limit && i == digit[pos]);
21
22
           if (!limit) dp[pos][pre] = ans;
23
24
       };
25
26
       return dfs(digit.size() - 1, 0, 1);
27 }
          bitmaskingDP
   2.12
1 // 杭电1565
2
3 int n;
4 int a[22][22];
5 int dp[22][1 << 18]; // 第一维是行数, 第二位是该行的方案数, 继承了前面所有行数的方案数
   int tot[1 << 18]; // 方案数
7
   int calc(int i, int k)
8
9
   {
10
       int cnt = 1, res = 0;
11
       while(k)
12
       {
           if(k & 1) res += a[i][cnt];
13
           k >>= 1;
14
15
           cnt++;
       }
16
17
       return res;
18
   }
19
20
   void solve()
21
       while(cin >> n) {
22
23
           mem(dp, 0);
24
           int cnt = 0;
25
```

```
for (int i = 0; i <= (1 << n) - 1; i++) { // 预处理
26
               if ((i & (i >> 1)) == 0) // 判断i这个二进制是否满足相邻没有两个1的条件
27
                   tot[++cnt] = i;
28
           }
29
30
31
           for (int i = 1; i <= n; i++)
               for (int j = 1; j <= n; j++)
32
33
                   cin >> a[i][j];
34
           for (int i = 1; i <= n; i++) { // 行遍历
35
               for (int k = 1; k <= cnt; k++) { // 第i行k的二进制排列的数, 与下面的j进行&
36
37
                   int val = calc(i, tot[k]); // 计算k的二进制中1所在a数组里的权值
                   for (int j = 1; j <= cnt; j++) { // 第i-1行j的二进制排列的数, 与上面的k进行&
38
       并进行状态转移
                       if ((tot[j] & tot[k]) == 0)
39
                           dp[i][k] = max(dp[i][k], dp[i - 1][j] + val);
40
41
                   }
               }
42
           }
43
44
           int ans = -1;
45
           for (int j = 1; j <= cnt; j++)</pre>
46
               ans = max(ans, dp[n][j]);
47
48
49
           cout << ans << endl;</pre>
       }
50
  }
51
   2.13 quadrilateralOptimization
1 //四边形优化区间dp(n^3 -> n^2)
2 //a < b < c < d, f[l][r] = min(f[l][k] + f[k + 1][r] + cost(l, r))
3 //1. cost(b, c) <= cost(a, d)
  //2. cost(a, c) + cost(b, d) <= cost(a, d) + cost(b, c), 即交叉小于包含
   template <typename T>
6
   void quadrilateralOptimization() {
7
       for (int len = 2; len <= n; len++) {</pre>
8
           for (int l = 1, r; l + len - 1 <= n; l++) {
9
               r = l + len - 1;
10
               mn[l][r] = 0x3f3f3f3f;
11
               for (int k = m[l][r - 1]; k <= m[l + 1][r]; k++)
12
                   if (mn[l][k] + mn[k + 1][r] + cost(l, r) < mn[l][r]) {
13
                       mn[l][r] = mn[l][k] + mn[k + 1][r] + cost(l, r);
14
15
                       m[l][r] = k;
                   }
16
17
           }
18
       }
   }
19
   2.14 baseRingTreeDP
  int flag, S, E;//flag是否找到环, SE为环上两个点
   void findCircle(int u, int fa) {
3
4
       vis[u] = 1;
       for (int i = head[u], v; i; i = e[i].nxt)
5
```

```
if ((v = e[i].to) != fa) {
6
7
                if (vis[v]) flag = 1, S = u, E = v;
                else findCircle(v, u);
8
            }
9
10
   }
11
   void dp(int u, int fa) {
12
       //dp过程
13
14
       for (int i = head[u], v; i; i = e[i].nxt)
15
16
            if ((v = e[i].to) != fa && v) {
17
                dp(v, u);
18
            }
19
   }
20
21
   ll calc(int u) {
22
23
       flag = 0;
       findCircle(u, 0);
24
       if (flag) {
25
            for (int i = head[S], v; i; i = e[i].nxt)
26
27
                if ((v = e[i].to) == E) {
                    e[i].to = e[i ^ 1].to = 0;//删边操作, 注意e[tot]中tot从2开始
28
29
30
                }
31
            11 \text{ res} = 0;
            dp(S, 0); res = max(res, ...);
32
            dp(E, 0); res = max(res, ...);
33
            return res;
34
35
       else {
36
37
            dp(u, 0);
            return ...;
38
39
       }
40 }
   2.15
          segmentTreeDP
1 #define lc u << 1
  #define rc u << 1 | 1
3 #define mid (t[u].l + t[u].r) / 2
4
5
   const int N = 3e5 + 10;
   const int K = 100 + 10;
6
7
   int n, k;
   int dp[N][K];
8
9
   struct Tree {
10
       int 1, r;
11
12
       int mx;
13
       int tag;
14 }t[N << 2];
15
16
  inline void push_up(int u) {
       t[u].mx = max(t[lc].mx, t[rc].mx);
17
18
19
20 inline void push_down(int u) {
```

```
if(!t[u].tag) return ;
21
22
        t[lc].tag += t[u].tag;
        t[rc].tag += t[u].tag;
23
        t[lc].mx += t[u].tag;
t[rc].mx += t[u].tag;
24
25
        t[u].tag = 0;
26
27
   }
28
29
   void build(int u, int l, int r, int k) {
        t[u].l = l; t[u].r = r;
30
31
        t[u].tag = t[u].mx = 0;
        if(1 == r) {
32
            t[u].mx = dp[l][k];
33
            return ;
34
35
        int m = (1 + r) / 2;
36
        build(lc, l, m, k);
37
38
        build(rc, m + 1, r, k);
39
        push_up(u);
40
   }
41
   void modify(int u, int ql, int qr, int val) {
42
        if(ql <= t[u].l && t[u].r <= qr) {
43
            t[u].mx += val;
44
            t[u].tag += val;
45
            return ;
46
        }
47
        int ans = -INF;
48
        push_down(u);
49
        if(ql <= mid) modify(lc, ql, qr, val);</pre>
50
51
        if(qr > mid) modify(rc, ql, qr, val);
52
        push_up(u);
   }
53
54
   int query(int u, int ql, int qr) {
55
        if(ql <= t[u].l && t[u].r <= qr) return t[u].mx;</pre>
56
57
        push_down(u);
58
        int ans = 0;
        if(ql \le mid) ans = max(ans, query(lc, ql, qr));
59
60
        if(qr > mid) ans = max(ans, query(rc, ql, qr));
61
        return ans;
   }
62
63
64
   void solve() {
        cin >> n >> k;
65
66
        vector<int> a(n + 1), pre(n + 1), from(n + 1);
67
        set<int> s;
        for(int i = 1;i <= n; i++) {</pre>
68
            cin >> a[i];
69
70
            s.insert(a[i]);
71
            from[i] = pre[a[i]]; pre[a[i]] = i;
72
            dp[i][1] = (int)s.size();
73
        for(int i = 2;i <= k; i++) {
74
            build(1, 1, n, i - 1);
75
            for(int j = i; j <= n; j++) {</pre>
76
77
                modify(1, from[j], j - 1, 1);
                dp[j][i] = max(dp[j-1][i-1]+1, query(1, i-1, j-1));
78
            }
79
```

```
80
        }
        cout << dp[n][k] << endl;</pre>
81
82 }
   2.16 LCS
1 #include <bits/stdc++.h>
2 using namespace std;
3
   int main() {
4
      int n; cin >> n;
5
6
      vector<int> p(n + 1), q(n + 1);
      for(int i = 1;i <= n; i++) cin >> p[i];
7
      for(int i = 1; i \leftarrow n; i++) cin \Rightarrow q[i];
8
9
      vector<int> id(n + 1);
10
      for(int i = 1; i <= n; i++) id[q[i]] = i;</pre>
11
      vector<int> lis;
12
      // 将b中与a中的元素相关的位置,从大到小push进lis中
13
      for(int i = 1;i <= n; i++) {</pre>
14
        vector<int> d;
15
        for(int j = p[i]; j \ll n; j += p[i]) {
16
          d.emplace_back(id[j]);
17
18
        }
        sort(d.begin(), d.end(), greater<int>());
19
        for(auto x : d) lis.emplace_back(x);
20
21
22
      vector<int> f;
23
24
      f.push_back(lis[0]);
      for(int i = 1;i < (int)lis.size(); i++) {</pre>
25
26
        if(lis[i] > f.back()) f.push_back(lis[i]);
27
          int pos = lower_bound(f.begin(), f.end(), lis[i]) - f.begin();
28
29
          f[pos] = lis[i];
30
        }
31
      }
32
      cout << f.size() << endl;</pre>
33
34 }
```

## 3 数据结构

#### 3.1 BTree

```
template<class T>
1
2
3
   struct TreeNode {
        T value;
4
        TreeNode *left;
5
        TreeNode *right;
6
   };
7
8
   template<class T>
9
10
   TreeNode<T> *createTree(const T *pre, const T *in, const int len) {
        TreeNode<T> *t = NULL;
11
        if (len > 0) {
12
            t = new TreeNode<T>;
13
            t->value = pre[0];
14
15
            int index;
            for (index = 0; index < len; index++) {</pre>
16
                if (in[index] == pre[0]) {
17
                     break;
18
                }
19
20
            if (index == len) {
21
22
                index = -1;
23
            }
24
            t->left = createTree(pre + 1, in, index);
25
            t->right = createTree(pre + index + 1, in + index + 1, len - index - 1);
26
27
        return t;
28
   }
29
30
   template<class T>
31
   int preOrder(TreeNode<T> *root, queue<T> &out) {
32
        if (root) {
            int count = 1;
33
34
            out.push(root->value);
35
            count += pre0rder(root->left, out);
36
            count += pre0rder(root->right, out);
37
            return count;
        } else {
38
39
            return 0;
40
        }
   }
41
42
   template<class T>
43
   int inOrder(TreeNode<T> *root, queue<T> &out) {
44
        if (root) {
45
            int count = 1;
46
            count += inOrder(root->left, out);
47
48
            out.push(root->value);
            count += inOrder(root->right, out);
49
50
            return count;
51
        } else {
52
            return 0;
53
        }
54
   }
55
```

```
template<class T>
    void postOrder(TreeNode<T> *root, queue<T> &out) {
57
         if (root) {
58
             postOrder(root->left, out);
59
             postOrder(root->right, out);
60
             out.push(root->value);
61
         } else {
62
             return;
63
         }
64
    }
65
66
67
    template<class T>
    T *convertQueueToArray(queue<T> &out, int len) {
68
         T *list = new T[len];
69
         int now = 0;
70
         while (!out.empty() && now < len) {</pre>
71
72
             list[now] = out.front();
             out.pop();
73
74
             now++;
75
         return list;
76
    }
77
78
79
    template<class T>
80
    void destroyTree(TreeNode<T> *root) {
81
         if (root) {
             destroyTree(root->left);
82
             destroyTree(root->right);
83
             delete root;
84
85
         } else return;
    }
86
87
    template<class T>
88
    void insertIntoBSTree(TreeNode<T> *root, const T &value) {
89
         if (!root) {
90
             return;
91
92
93
         if (value < root->value) {
             if (root->left) {
94
                 insertIntoTree(root->left, value);
95
             } else {
96
                 root->left = new TreeNode<T>;
97
                 root->left->value = value;
98
99
                 root->left->left = NULL;
                 root->left->right = NULL;
100
             }
101
         } else if (value > root->value) {
102
             if (root->right) {
103
                 insertIntoTree(root->right, value);
104
105
106
                 root->right = new TreeNode<T>;
107
                 root->right->value = value;
108
                 root->right->left = NULL;
                 root->right->right = NULL;
109
             }
110
111
         }
112
113
114 template<class T>
```

```
TreeNode<T> *createBSTree(T *list, int len) {
        if (len < 1) {
116
           return NULL;
117
118
        TreeNode<T> *root = new TreeNode<char>;
119
        root->value = list[0];
120
        root->left = NULL;
121
        root->right = NULL;
122
        for (int i = 1; i < len; i++) {
123
           insertIntoBSTree(root, list[i]);
124
125
126
        return root;
127 }
    3.2 pbds-tree
 1 // RBTree 红黑树
 2 #include <ext/pb_ds/tree_policy.hpp>
 3 #include <ext/pb_ds/assoc_container.hpp>
 4 // 红黑树
   __gnu_pbds::tree<int, null_type, less<int>, rb_tree_tag,
       tree_order_statistics_node_update> t;
 6 // null_type无映射(低版本q++为null_mapped_type)
 7 // 类似multiset
    __gnu_pbds::tree<int, null_type, less_equal<int>, rb_tree_tag,
       tree_order_statistics_node_update> t;
 9 find_by_order(size_t order);
10 // 结点更新
11 tree_order_statistics_node_update
12 insert(p);
13 erase(it);
14 // 求k在树中是第几小,假设插入当前值判断当前值是第几小,最小为第0小
15 order_of_key(p);
16 // 找到第order小的迭代器
17 find_by_order(order);
18 // 前驱
19 lower_bound(p);
20 // 后驭
21 upper_bound(p);
22 // 合并
23 a.join(b);
24 // 分割 key小于等于v的元素属于a. 其余的属于b
25 a.split(v, b);
27 // 优先队列
28 #include <ext/pb_ds/priority_queue.hpp>
29 #include <ext/pb_ds/assoc_container.hpp>
30 // 操作类似于stl的优先队列
   typedef __qnu_pbds::priority_queue<node, greater<node>, __qnu_pbds::thin_heap_tag> heap
    heap::point_iterator; // 指向元素的指针
    3.3 fenwick
 1 template <typename T>
 2 class fenwick {
 3 public:
```

```
std::vector<T> fenw;
4
5
        int n;
6
        fenwick(int _n) : n(_n) {
7
8
            fenw.resize(n);
9
        }
10
        void modify(int x, T v) {
11
            while (x < n) {
12
13
                fenw[x] += v;
14
                x = (x + 1);
15
            }
        }
16
17
        T get(int x) {
18
            T v{};
19
            while (x >= 0) {
20
21
                v += fenw[x];
22
                x = (x & (x + 1)) - 1;
23
24
            return v;
25
        }
26 };
    3.4 fenwick2d
   template <typename T>
   class Fenwick2d {
2
3
   public:
        std::vector<std::vector<T>> fenw;
4
5
        const int n, m;
6
        Fenwick2d(int _n, int _m) : n(_n), m(_m) {
7
8
            fenw.resize(n);
            for (int i = 0; i < n; i++) {
9
10
                fenw[i].resize(m);
11
            }
12
        }
13
14
        inline void modify(int i, int j, T v) {
15
            int x = i;
            while (x < n) {
16
                int y = j;
17
                while (y < m) {
18
19
                     fenw[x][y] += v;
20
                     y = (y + 1);
21
22
                x = (x + 1);
            }
23
        }
24
25
        inline T get(int i, int j) {
26
27
            T v{};
28
            int x = i;
            while (x >= 0) {
29
                int y = j;
30
                while (y >= 0) {
31
                     v \leftarrow fenw[x][y];
32
```

```
y = (y \& (y + 1)) - 1;
33
34
                x = (x & (x + 1)) - 1;
35
36
37
            return v;
38
       }
39
  };
   3.5 SegmentTree
   struct Info {
2
3
   };
4
   Info operator+(const Info& a, const Info& b) {
5
6
   }
7
8
   template<class Info,
9
       class Merge = std::plus<Info>>
10
   struct SegmentTree {
11
12
       const int n;
       const Merge merge;
13
       std::vector<Info> info;
14
       SegmentTree(int n) : n(n), merge(Merge()), info(4 * n + 10) {}
15
       SegmentTree(std::vector<Info> init) : SegmentTree(init.size()) {
16
            std::function<void(int, int, int)> build = [&](int p, int l, int r) {
17
18
                if (r - l == 1) {
                    info[p] = init[l];
19
20
                    return;
21
                int m = (l + r) / 2;
22
23
                build(2 * p, l, m);
                build(2 * p + 1, m, r);
24
25
                pull(p);
26
27
            build(1, 0, n);
28
29
       void pull(int p) {
30
            info[p] = merge(info[2 * p], info[2 * p + 1]);
31
       void modify(int p, int l, int r, int x, const Info &v) {
32
            if (r - l == 1) {
33
34
                info[p] = v;
35
                return;
36
            int m = (l + r) / 2;
37
            if (x < m) {
38
                modify(2 * p, l, m, x, v);
39
40
            } else {
                modify(2 * p + 1, m, r, x, v);
41
            }
42
           pull(p);
43
44
       void modify(int p, const Info &v) {
45
            modify(1, 0, n, p, v);
46
47
       Info rangeQuery(int p, int l, int r, int x, int y) {
48
```

```
if (1 >= y | 1 r <= x) {
49
                return Info();
50
51
            if (1 >= x && r <= y) {
52
53
                return info[p];
54
            int m = (l + r) / 2;
55
            return merge(rangeQuery(2 * p, l, m, x, y), rangeQuery(2 * p + 1, m, r, x, y));
56
57
        Info rangeQuery(int 1, int r) {
58
59
            return rangeQuery(1, 0, n, l, r);
60
        }
   };
61
   3.6 SegmentTree2d
   #define lc u << 1
   #define rc u << 1 | 1
3
   #define m (l + r) / 2
4
   const int N = 1e3 + 10;
5
6
7
   struct Tree_y {
8
        int mx, mn;
9
10
        Tree_y operator + (const Tree_y &rhs) {
            Tree_y ans;
11
12
            ans.mx = max(mx, rhs.mx);
13
            ans.mn = min(mn, rhs.mn);
14
            return ans;
        }
15
16
   };
17
18
  int leafx[N], leafy[N];
19
   struct Tree_x {
20
       Tree_y ty[N << 2];
21
22
23
        void build(int u, int l, int r) {
24
            ty[u].mx = -INF; ty[u].mn = INF;
25
            if(l == r) {
                leafy[l] = u;
26
27
                return ;
28
29
            build(lc, l, m);
            build(rc, m + 1, r);
30
        }
31
32
        Tree_y query(int u, int l, int r, int ql, int qr) {
33
            if(qr < l || r < ql) return (Tree_y) {-INF, INF};</pre>
34
35
            if(ql <= l && r <= qr) return ty[u];
            return query(lc, l, m, ql, qr) + query(rc, m + 1, r, ql, qr);
36
37
38
   }tx[N << 2];
39
40
   int n;
41
   void build(int u, int l, int r) {
```

```
tx[u].build(1, 1, n);
43
44
       if(l == r) {
           leafx[l] = u;
45
46
           return ;
47
       build(lc, 1, m);
48
       build(rc, m + 1, r);
49
   }
50
51
   // (x,y)单点更新, 首先更新叶子节点, 然后向上合并父亲节点
   void modify(int x, int y, int val) {
53
       int valx = leafx[x];
54
       int valy = leafy[y];
55
       tx[valx].ty[valy].mn = tx[valx].ty[valy].mx = val;
56
       for(int i = valx; i; i >>= 1) {
57
            for(int j = valy; j; j >>= 1) {
58
                if(i == valx && j == valy) continue ;
59
                if(j == valy) {
60
                    // 如果当前更新的列就是需要更新的叶子节点,那么由当前行的两个儿子节点来更新
61
                    tx[i].ty[j] = tx[i << 1].ty[j] + tx[i << 1 | 1].ty[j];
62
               }
63
               else {
64
                    tx[i].ty[j] = tx[i].ty[j << 1] + tx[i].ty[j << 1 | 1];
65
66
               }
67
           }
       }
68
   }
69
70
   Tree_y query(int u, int l, int r, int ql, int qr, int qx, int qy) {
71
       if(qr < l || r < ql) return (Tree_y) {-INF, INF};</pre>
72
       if(ql \leftarrow 1 && r \leftarrow qr) return tx[u].query(1, 1, n, qx, qy);
73
       return query(lc, l, m, ql, qr, qx, qy) + query(rc, m + 1, r, ql, qr, qx, qy);
74
75 }
   3.7
        ValueSegmentTree
   // 权值线段树,相当于一个桶,每个节点用来表示一个区间的数***出现的次数***。
3
   #include <bits/stdc++.h>
4
5
   using namespace std;
6
7
   #define lc u << 1
   #define rc u << 1 | 1
8
   #define m (l + r) / 2
   #define mid (t[u].l + t[u].r) / 2
10
11
12 const int N = 1e5 + 10;
13
  int t[N << 2];</pre>
15
16 int a[N];
17
18
   void push_up(int u) {
19
       t[u] = t[lc] + t[rc];
20
21
22 void build(int u, int l, int r) {
```

```
23
       if(l == r) {
           t[u] = a[l]; // a[l]表示数为l的个数
24
25
           return ;
26
       build(lc, l, m);
27
       build(rc, m + 1, r);
28
29
       push_up(u);
   }
30
31
   void update(int u, int l, int r, int k, int cnt) { // k这个数的个数增加cnt
32
33
       if(l == r) {
34
           t[u] += cnt;
35
           return ;
36
       }
       if(k \ll m)
37
           update(lc, l, m, k, cnt);
38
39
           update(rc, m + 1, r, k, cnt);
40
41
       push_up(u);
42
   }
43
   int query(int u, int l, int r, int k) { // 查询k这个数的个数
44
       if(l == r) {
45
46
           return t[u];
47
       if(k \ll m)
48
           return query(lc, l, m, k);
49
50
       else
           return query(rc, m + 1, r, k);
51
   }
52
53
   int k_max_th(int u, int l, int r, int k) { // 查询第k大的值
54
       if(l == r) {
55
           return 1;
56
57
       if(t[lc] >= k) return k_max_th(lc, l, m, k);
58
59
       else return k_max_th(rc, m + 1, r, k - t[lc]);
60 }
        线段树动态开点合并分裂
   3.8
   #define mid (l+r)/2
   static const int MAX_N = N * 40;
int rt[N], now;
4 int lc[MAX_N], rc[MAX_N];
5 11 sum[MAX_N];
6 int tot, rub[MAX_N];
   int newNode() { return rub[0] ? rub[rub[0]--] : ++tot; }
7
   void remove(int &u) {
8
9
       lc[u] = rc[u] = sum[u] = 0;
       rub[++rub[0]] = u;
10
       u = 0;
11
   }
12
   void push_up(int u) { sum[u] = sum[lc[u]] + sum[rc[u]]; }
13
   void build(int &u, int l, int r) {
14
15
       u = newNode();
       if (1 == r) {
16
17
            sum[u] = cnt[l];
```

```
18
            return;
19
        build(lc[u], l, mid); build(rc[u], mid + 1, r);
20
21
        push_up(u);
22
   }
   void update(int &u, int l, int r, int p, ll k) {
23
24
        if (!u) u = newNode();
25
        if (l == r) {
26
            sum[u] += k;
27
            return;
28
29
        if (p <= mid) update(lc[u], l, mid, p, k);</pre>
        else update(rc[u], mid + 1, r, p, k);
30
        push_up(u);
31
32
   ll querySum(int u, int l, int r, int ql, int qr) {
33
        if (!u) return 0;
34
        if (ql <= l && r <= qr) return sum[u];</pre>
35
36
        11 \text{ res} = 0;
        if (ql <= mid) res += querySum(lc[u], l, mid, ql, qr);</pre>
37
38
        if (qr > mid) res += querySum(rc[u], mid + 1, r, ql, qr);
        return res;
39
   }
40
41
   int queryKth(int u, int l, int r, ll k) {
42
        if (l == r) return l;
        if (k <= sum[lc[u]]) return queryKth(lc[u], l, mid, k);</pre>
43
        else return queryKth(rc[u], mid + 1, r, k - sum[lc[u]]);
44
45
   void merge(int u, int v, int l, int r) {
46
        if (l == r) {
47
            sum[u] += sum[v];
48
            return:
49
50
        if (lc[u] && lc[v]) merge(lc[u], lc[v], l, mid), remove(lc[v]);
51
        else if (lc[v]) lc[u] = lc[v], lc[v] = 0;
52
        if (rc[u] && rc[v]) merge(rc[u], rc[v], mid + 1, r), remove(rc[v]);
53
        else if (rc[v]) rc[u] = rc[v], rc[v] = 0;
54
55
        push_up(u);
56
   }
   void split(int &newp, int &u, int l, int r, int ql, int qr) {//分裂出[ql, qr]间的点
57
        if (!u) return;
58
        if (ql <= l && r <= qr) {
59
            newp = u;
60
            u = 0;
61
            return;
62
63
        if (!newp) newp = newNode();
64
        if (ql <= mid) split(lc[newp], lc[u], l, mid, ql, qr);</pre>
65
        if (qr > mid) split(rc[newp], rc[u], mid + 1, r, ql, qr);
66
67
        push_up(u);
68
        push_up(newp);
69
70
   #undef mid
```

#### 3.9 线段树维护 LIS 方案数

1 // 线段树维护序列总LIS的长度mx.fi和方案数mx.se

```
// 以及对于每个点,可以存在于多少个LIS种
3
   // https://nanti.jisuanke.com/t/39611
4
5
   namespace Tree_LIS {
6
        const int N = 1e6 + 10;
7
8
   #define lc t[u].l
9
   #define rc t[u].r
10
   #define mid (l + r) / 2
11
12
13
        struct Tree {
             int 1, r;
14
             ll len; // 长度
15
            ll sum; // 个数
16
        }t[N << 2];</pre>
17
18
        int root, cnt;
19
        void init() {
20
21
            mem(t, 0);
22
             cnt = root = 0;
23
24
25
        void push_up(int u) {
26
             if(t[lc].len == t[rc].len) {
                 t[u].len = t[lc].len;
27
                 t[u].sum = (t[lc].sum + t[rc].sum) % mod;
28
29
             else if(t[lc].len < t[rc].len) {</pre>
30
                 t[u].len = t[rc].len;
31
32
                 t[u].sum = t[rc].sum;
33
             }
             else {
34
                 t[u].len = t[lc].len;
35
                 t[u].sum = t[lc].sum;
36
             }
37
38
        }
39
        void modify(int &u, int l, int r, int p, int le, int su) {
40
             if(!u) u = ++cnt;
41
             if(l == r) {
42
                 if(t[u].len == le) t[u].sum = (t[u].sum + su) % mod;
43
                 else if(t[u].len < le) {</pre>
44
45
                      t[u].len = le;
                      t[u].sum = su;
46
47
                 }
                 return;
48
49
             if(!lc) lc = ++cnt;
50
51
             if(!rc) rc = ++cnt;
             if(p <= mid) modify(lc, l, mid, p, le, su);</pre>
52
53
             else modify(rc, mid + 1, r, p, le, su);
54
            push_up(u);
        }
55
56
        pll query(int u, int l, int r, int ql, int qr) {
   if(ql <= l && r <= qr) return pll{t[u].len, t[u].sum};</pre>
57
58
             pll lson = \{0, 0\}, rson = \{0, 0\};
59
             if(!lc) lc = ++cnt;
60
```

```
61
             if(!rc) rc = ++cnt;
             if(ql <= mid) lson = query(lc, l, mid, ql, qr);</pre>
62
             if(qr > mid) rson = query(rc, mid + 1, r, ql, qr);
63
             if(lson.fi == rson.fi) return pll{lson.fi, (lson.se + rson.se) % mod};
64
             else if(lson.fi < rson.fi) return rson;</pre>
65
             else return lson;
66
67
         }
    };
68
69
    using namespace Tree_LIS;
70
71
72
    ll quick_pow(ll a, ll b) {
         ll ans = 1;
73
         while(b) {
74
             if(b \& 1) ans = ans * a % mod;
75
             a = a * a % mod;
76
77
             b >>= 1;
78
         return ans % mod;
79
80 }
81
    void solve() {
82
         int n; cin >> n;
83
         int L = 0, R = 1e9 + 7;
84
85
         vector<int> a(n + 1);
         vector<pll> l(n + 1), r(n + 1);
86
         for(int i = 1;i <= n; i++) cin >> a[i];
87
         init();
88
         modify(root, L, R, a[1], 1, 1);
89
90
         l[1] = \{1, 1\};
         for(int i = 2;i <= n; i++) {</pre>
91
             pll temp = query(root, L, R, 0, a[i] - 1);
92
             if(temp.fi == 0) temp = {0, 1};
93
             modify(root, L, R, a[i], temp.fi + 1, temp.se);
94
             l[i] = {temp.fi + 1, temp.se};
95
         }
96
97
         pll mx = query(root, L, R, 0, R);
98
         init();
         modify(root, L, R, R - a[n], 1, 1);
99
         r[n] = \{1, 1\};
100
         for(int i = n - 1; i >= 1; i--) {
101
             pll temp = query(root, L, R, 0, R - a[i] - 1);
102
             if(temp.fi == 0) temp = \{0, 1\};
103
104
             modify(root, L, R, R - a[i], temp.fi + 1, temp.se);
             r[i] = \{temp.fi + 1, temp.se\};
105
106
         for(int i = 1;i <= n; i++) {
107
             if(r[i].fi + l[i].fi - 1 == mx.fi) {
108
                 cout << (r[i].se * l[i].se % mod * quick_pow(mx.se, mod - 2) % mod + mod) %</pre>
109
         mod << " ";
110
111
             else cout << 0 << " ";
112
113
         cout << endl;</pre>
114 }
```

#### 3.10 线段树维护最小字典序 LIS

```
// 线段树维护LIS输出字典序最小的路径
   const int N = 1e5 + 10;
3
   #define lc u << 1
4
5
   #define rc u << 1 | 1
   #define mid (t[u].l + t[u].r) / 2
6
7
   struct Tree {
8
9
        int l, r;
10
        int mx;
        int id;
11
12
   }t[N << 2];</pre>
13
   inline void push_up(int u) {
14
        if (t[lc].mx > t[rc].mx)
15
            t[u].mx = t[lc].mx, t[u].id = t[lc].id;
16
17
        else if (t[lc].mx < t[rc].mx)</pre>
18
            t[u].mx = t[rc].mx, t[u].id = t[rc].id;
19
        else
            t[u].mx = t[lc].mx, t[u].id = min(t[lc].id, t[rc].id);
20
21 }
22
23
   void build(int u, int l, int r) {
24
        t[u].l = l;
        t[u].r = r;
25
        t[u].mx = t[u].id = 0;
26
27
        if (l == r)
            return;
28
        int m = (l + r) >> 1;
29
        build(lc, l, m);
30
        build(rc, m + 1, r);
31
32
        push_up(u);
   }
33
34
   void modify(int u, int ql, int qr, int val, int id) {
35
        if (ql <= t[u].l && t[u].r <= qr) {</pre>
36
37
            if (t[u].mx < val || (t[u].mx == val && t[u].id > id)) {
38
                t[u].mx = val;
                t[u].id = id;
39
40
            }
            return;
41
42
        if (ql <= mid)</pre>
43
            modify(lc, ql, qr, val, id);
44
        if (qr > mid)
45
            modify(rc, ql, qr, val, id);
46
47
        push_up(u);
   }
48
49
   pii query(int u, int ql, int qr) {
51
        if (ql <= t[u].l && t[u].r <= qr)</pre>
52
            return pii{t[u].mx, t[u].id};
53
        pii lson = \{-1, -1\}, rson = \{-1, -1\};
        if (ql <= mid)</pre>
54
            lson = query(lc, ql, qr);
55
        if (qr > mid)
56
57
            rson = query(rc, ql, qr);
58
        if (lson.x > rson.x)
            return lson;
59
```

```
else if (lson.x < rson.x)</pre>
60
61
             return rson;
         else
62
             return {lson.x, min(lson.y, rson.y)};
63
64 }
65
    void solve() {
66
         int n;
67
         cin >> n;
68
         assert(1 \le n \&\& n \le 1e5);
69
         build(1, 1, 1e5);
70
71
         vector<int> a(n + 1), ans(n + 1), fa(n + 1);
         pii res = \{0, 0\};
72
         for (int i = 1; i <= n; i++) {
73
74
             cin >> a[i];
             assert(1 <= a[i] && a[i] <= 1e5);
 75
             if (a[i] == 1) {
76
                  fa[i] = 0;
77
                 ans[i] = 1;
78
                 modify(1, a[i], a[i], 1, i);
79
                 continue;
80
             }
81
             pii temp = query(1, 1, a[i] - 1);
82
83
             ans[i] = temp.x + 1;
84
             fa[i] = temp.y;
             modify(1, a[i], a[i], ans[i], i);
85
             if (res.x < ans[i])</pre>
86
                 res = pii{ans[i], i};
87
         }
88
         vector<int> v;
89
90
         int tt = res.second;
         while (tt) {
91
             v.push_back(tt);
92
             tt = fa[tt];
93
         }
94
         cout << v.size() << endl;</pre>
95
96
         for (int i = v.size() - 1; i >= 0; i--) {
97
             cout << v[i] << (i == 0 ? endl : " ");
98
         }
    }
99
100
101
102
   // 线段树维护LIS方案数
```

#### 3.11 线段树维护插队问题

```
1 // n个人,每个人a_i要顺序坐在pos_i,问最终的序列如何
  // 最后一个人一定坐在自己喜欢坐的位置,去掉该位置,倒数第二个人成为最后一个人,所以就是查找空位置的第
      pos位置
3
4
  const int N = 4e5 + 10;
5
  #define lc u << 1
6
7
  #define rc u << 1 | 1
  #define mid (l + r) / 2
8
  int sum[N << 2], ans[N];</pre>
9
10
11 void push_up(int u) {
```

```
sum[u] = sum[lc] + sum[rc];
12
   }
13
14
   void build(int u, int l, int r) {
15
        if(l == r) {
16
            sum[u] = 1;
17
            ans[1] = 0;
18
            return ;
19
20
        build(lc, l, mid);
21
22
        build(rc, mid + 1, r);
23
        push_up(u);
   }
24
25
   void modify(int u, int l, int r, int k, int val) {
26
        if(l == r) {
27
            ans[l] = val;
28
29
            sum[u] = 0;
30
            return ;
31
        if(sum[lc] >= k) modify(lc, l, mid, k, val);
32
        else modify(rc, mid + 1, r, k - sum[lc], val);
33
        push_up(u);
34
35 }
36
   void solve() {
37
        int n;
38
        while(~scanf("%d", &n)) {
39
            vector<pii> p(n + 1);
40
            for(int i = 1; i <= n; i++) {
41
                scanf("%d%d",&p[i].fi, &p[i].se);
42
43
            build(1, 1, n);
44
            for(int i = n;i >= 1; i--) {
45
                modify(1, 1, n, p[i].fi + 1, p[i].se);
46
47
48
            for(int i = 1;i <= n; i++) {</pre>
49
                printf("%d ", ans[i]);
50
            printf("\n");
51
52
        }
   }
53
```

#### 3.12 线段树维护连续区间异或值

12

```
13
   高pos不管
14
15
   vector<pii> g[N], len;
   int l[N], r[N];
17
18
   void modify(int pos, int l, int r, int L, int R, int val) {
19
       if(L \le 1 \& r \le R) {
20
21
            // 把低pos设置为0
            int ql = (l \land val) \& (((1 << 30) - 1) \land (1 << pos) - 1);
22
23
            int qr = ql + (1 << pos) - 1;
24
            len.push_back(pii{ql, qr});
25
            return ;
26
       }
27
       int mid = (l + r) / 2;
       if(L <= mid) modify(pos - 1, l, mid, L, R, val);</pre>
28
       if(R > mid) modify(pos - 1, mid + 1, r, L, R, val);
29
30 }
31
   void dfs(int u, int fa, int w) {
32
       modify(30, 0, (1 \ll 30) - 1, l[u], r[u], w);
33
       for(auto e : g[u]) {
34
35
            if(e.fi == fa) continue ;
36
            dfs(e.fi, u, e.se ^ w);
37
       }
38 }
         线段树维护区间异或
   3.13
   const int N = 2e5 + 10;
1
   #define lc u << 1
3
4
   #define rc u << 1 | 1
5
6
   struct Tree {
       int sum, tag;
7
   }t[21][N << 2];
9
   int a[N];
10
11
   void push_up(int id, int u) {
       t[id][u].sum = t[id][lc].sum + t[id][rc].sum;
12
   }
13
14
   void push_down(int id, int u, int l, int r) {
15
16
       if(!t[id][u].tag) return ;
       int m = (l + r)^{-}/2;
17
       t[id][lc].sum = (m - l + 1) - t[id][lc].sum;
18
       t[id][rc].sum = (r - m) - t[id][rc].sum;
19
       t[id][lc].tag ^= 1;
20
       t[id][rc].tag ^= 1;
21
22
       t[id][u].tag = 0;
   }
23
24
   void build(int id, int u, int l, int r) {
25
26
       if(l == r) {
27
            t[id][u].sum = (a[l] >> id) & 1;
28
            t[id][u].tag = 0;
29
            return ;
```

```
30
        int m = (l + r) / 2;
31
        build(id, lc, l, m);
build(id, rc, m + 1, r);
32
33
34
        push_up(id, u);
   }
35
36
   void modify(int id, int u, int l, int r, int ql, int qr) {
37
        if(ql \ll l \& r \ll qr) {
38
            t[id][u].sum = (r - l + 1) - t[id][u].sum;
39
40
            t[id][u].tag ^= 1;
41
            return ;
        }
42
        push_down(id, u, l, r);
43
        int m = (l + r) / 2;
44
        if(ql <= m) modify(id, lc, l, m, ql, qr);</pre>
45
        if(qr > m) modify(id, rc, m + 1, r, ql, qr);
46
47
        push_up(id, u);
   }
48
49
   int query(int id, int u, int l, int r, int ql, int qr) {
50
        if(ql <= l && r <= qr) return t[id][u].sum;</pre>
51
52
        push_down(id, u, l, r);
53
        int ans = 0;
54
        int m = (l + r) / 2;
        if(ql \ll m) ans += query(id, lc, l, m, ql, qr);
55
56
        if(qr > m) ans += query(id, rc, m + 1, r, ql, qr);
57
        return ans;
   }
58
59
   void solve() {
60
        int n, m; cin >> n >> m;
61
62
        for(int i = 1;i <= n; i++) cin >> a[i];
        for(int i = 0;i <= 20; i++) {
63
            build(i, 1, 1, n);
64
65
66
        while(m--) {
67
            int opt; cin >> opt;
68
            if(opt == 1) {
                 int l, r; cin >> l >> r;
69
70
                 11 \text{ ans} = 0;
                 for(int i = 0;i <= 20; i++) {
71
                     ans += query(i, 1, 1, n, l, r) * (1ll << i);
72
73
                 cout << ans << endl;</pre>
74
75
            }
            else {
76
                 int 1, r, k; cin >> 1 >> r >> k;
77
                 for(int i = 0;i <= 20; i++) {
78
79
                     if((k >> i) & 1) modify(i, 1, 1, n, l, r);
80
                 }
81
            }
82
        }
83 }
```

## 3.14 LazySegmentTree

1

```
struct Info {
3
   };
4
5
   struct Tag {
6
7
8
   };
9
  Info operator+(const Info& a, const Info& b) {
10
11
12 }
13
14 void apply(Info &a, const Tag &b) {
15
   }
16
17
18 void apply(Tag &a, const Tag &b) {
19
20 }
21
   template<class Info, class Tag,
22
       class Merge = std::plus<Info>>
23
   struct LazySegmentTree {
24
25
       const int n;
26
       const Merge merge;
       std::vector<Info> info;
27
28
       std::vector<Tag> tag;
       LazySegmentTree(int n): n(n), merge(Merge()), info(4 * n + 10), tag(4 * n + 10) {}
29
       LazySegmentTree(std::vector<Info> init) : LazySegmentTree(init.size()) {
30
            std::function<void(int, int, int)> build = [&](int p, int l, int r) {
31
                if (r - l == 1) {
32
                    info[p] = init[l];
33
                    return;
34
                }
35
                int m = (l + r) / 2;
36
                build(2 * p, l, m);
37
                build(2 * p + 1, m, r);
38
39
                pull(p);
40
            build(1, 0, n);
41
42
       void pull(int p) {
43
            info[p] = merge(info[2 * p], info[2 * p + 1]);
44
45
       void apply(int p, const Tag &v) {
46
47
            ::apply(info[p], v);
            ::apply(tag[p], v);
48
49
       void push(int p) {
50
51
            apply(2 * p, tag[p]);
            apply(2 * p + 1, tag[p]);
52
53
            tag[p] = Tag();
54
       void modify(int p, int l, int r, int x, const Info &v) {
55
            if (r - l == 1) {
56
                info[p] = info[p] + v;
57
58
                return;
59
            int m = (l + r) / 2;
60
```

```
push(p);
61
             if (x < m) {
62
                 modify(2 * p, l, m, x, v);
63
64
             } else {
                 modify(2 * p + 1, m, r, x, v);
65
66
            pull(p);
67
68
        void modify(int p, const Info &v) {
69
70
            modify(1, 0, n, p, v);
71
72
        Info rangeQuery(int p, int l, int r, int x, int y) {
             if (1 >= y | | r <= x) {
73
                 return Info();
74
75
             if (1 >= x \& r <= y) {
76
77
                 return info[p];
78
             int m = (l + r) / 2;
79
            push(p);
80
             return merge(rangeQuery(2 * p, l, m, x, y), rangeQuery(2 * p + 1, m, r, x, y));
81
82
        Info rangeQuery(int l, int r) {
83
             return rangeQuery(1, 0, n, l, r);
84
85
        void rangeApply(int p, int l, int r, int x, int y, const Tag &v) {
86
             if (1 >= y | | r <= x) {
87
                 return;
88
89
             if (1 >= x \& r <= y) {
90
                 apply(p, v);
91
                 return;
92
93
             int m = (l + r) / 2;
94
95
            push(p);
             rangeApply(2 * p, l, m, x, y, v);
96
97
             rangeApply(2 * p + 1, m, r, x, y, v);
98
            pull(p);
99
        }
100
        void rangeApply(int l, int r, const Tag &v) {
101
             return rangeApply(1, 0, n, l, r, v);
        }
102
103 };
    3.15
          SparseTable
 1 // 倍增思想加DP优化
 2
 3 #include <bits/stdc++.h>
 4 using namespace std;
 5
   const int N = 1e5 + 10;
 6
 7
 8
    int a[N];
 9
10
11 template <typename T, class F = std::function<T(const T&, const T&)>>
    class SparseTable {
```

```
public:
13
14
        int n;
        std::vector<std::vector<T>> mat;
15
16
        F func;
17
        SparseTable(const std::vector<T>& a, const F& f) : func(f) {
18
            n = static_cast<int>(a.size());
19
            int max_log = 32 - __builtin_clz(n);
20
            mat.resize(max_log);
21
            mat[0] = a;
22
23
            for (int j = 1; j < max_log; j++) {
24
                mat[j].resize(n - (1 << j) + 1);
                for (int i = 0; i <= n - (1 << j); i++) {
25
                     mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
26
                }
27
            }
28
        }
29
30
        T get(int from, int to) const {
31
            assert(0 \le from \& from \le to \& to \le n - 1);
32
            int lg = 32 - __builtin_clz(to - from + 1) - 1;
33
            return func(mat[lg][from], mat[lg][to - (1 << lg) + 1]);</pre>
34
        }
35
36
   };
37
38
   // 二维ST表
   int f[105][105][105];
39
40
   void Two_ST() {
41
42
        int n, m;
43
        cin >> n >> m;
        for(int i = 1;i <= n; i++)</pre>
44
            for(int j = 1; j \le m; j++){
45
                cin >> a[i];
46
                f[i][j][0] = a[i];
47
            }
48
49
50
   // 大矩阵分成四个小矩阵求最值
51
52
        for(int k = 1; k < log2(n); k++) {
            for(int i = 1;i <= n; i++) {
53
                for(int j = 1; j <= m; j++) {</pre>
54
                     if((i + (1 << (k - 1)) <= n) && (j + (1 << (k - 1)) <= m))
55
                         f[i][j][k] = max(max(f[i][j + (1 << (k - 1))][k - 1], f[i + (1 << (k - 1))][k - 1])
56
       k - 1)[j][k - 1], max(f[i][j][k - 1], f[i + (1 << (k - 1))][j + (1 << (k - 1))][
       k - 1]));
            }
57
        }
58
59
        int T;
60
61
        cin >> T;
62
        while(T--) {
63
            int 1, r, k;
            cin >> 1 >> r >> k;
64
            int len = log2(k);
65
            int s = max(max(f[l][r][len], f[l + k - (1 << len)][r + k - (1 << len)][len]),
66
       \max(f[l + k - (1 << len)][r][len], f[l][r + k - (1 << len)][len]));
67
68
```

69 }

## 3.16 CartesianTree

```
1 // 笛卡尔树是一种由数列构造的特殊二叉搜索树,每个节点都有两个键值,first为下标,second为权值
   // 笛卡尔树满足两个性质,在下标递增的情况下就是一个大/小根堆
3
   // 笛卡尔树, 静态建树, 区间最值跳转
4
   struct CartesianTree {
       int rt; // 根节点
pii ch[N]; // 左右儿子
int st[N]; // 单调栈
6
7
8
9
       void build(int n, int p[]) {
10
           rt = 0;
11
           int t = 0;
12
13
           for (int i = 1; i <= n; i++) {
               ch[i] = \{0, 0\};
14
               // 决定了大于还是小于
15
               while (t && p[st[t]] > p[i]) --t;
16
17
               if (t) {
                   // 上一个点的右儿子作为自己的左儿子
18
                   // 成为上一个点的右儿子
19
                   ch[i].first = ch[st[t]].second;
20
21
                   ch[st[t]].second = i;
22
               } else { // 自己作为根节点
                   ch[i].first = rt;
23
24
                   rt = i;
25
26
               st[++t] = i;
27
           }
28
   } dika;
29
   3.17
         DancingLinks
   // Dancing Links
   struct DLX {
3
       int n, m, size;
       int U[MaxNode], D[MaxNode], L[MaxNode], R[MaxNode], Row[MaxNode], Col[MaxNode];
4
       int H[MaxN], S[MaxM];
5
       int ansd, ans[MaxN];
6
7
       void init(int _n, int _m) {
8
9
           n = _n;
           m = _m;
10
           for (int i = 0; i <= m; i++) {
11
12
               S[i] = 0;
               U[i] = D[i] = i;
13
               L[i] = i - 1;
14
               R[i] = i + 1;
15
16
           R[m] = 0;
17
           L[0] = m;
18
19
           size = m;
           for (int i = 0; i <= n; i++) {
20
```

H[i] = -1;

21

```
22
            }
        }
23
24
        void Link(int r, int c) {
25
26
            ++S[Col[++size] = c];
            Row[size] = r;
27
            D[size] = D[c];
28
            U[D[c]] = size;
29
            U[size] = c;
30
            D[c] = size;
31
32
            if (H[r] < 0) {
33
                 H[r] = L[size] = R[size] = size;
34
            } else {
                 R[size] = R[H[r]];
35
                 L[R[H[r]]] = size;
36
                 L[size] = H[r];
37
38
                 R[H[r]] = size;
            }
39
        }
40
41
        void remove(int c) {
42
            L[R[c]] = L[c];
43
            R[L[c]] = R[c];
44
            for (int i = D[c]; i != c; i = D[i]) {
45
46
                 for (int j = R[i]; j != i; j = R[j]) {
                     U[D[j]] = U[j];
47
                     D[U[j]] = D[j];
48
                     --S[Col[j]];
49
                 }
50
            }
51
        };
52
53
        void resume(int c) {
54
            for (int i = U[c]; i != c; i = U[i])
55
                 for (int j = L[i]; j != i; j = L[j])
56
                     ++S[Col[U[D[j]] = D[U[j]] = j]];
57
58
            L[R[c]] = R[L[c]] = c;
59
        }
60
        bool Dance(int d) {
61
62
            if (R[0] == 0) {
                 for (int i = 0; i < d; i++) {
    printf("%d%c", ans[i], " \n"[i == d - 1]);</pre>
63
64
65
                 return true;
66
            }
67
            int c = R[0];
68
            for (int i = R[0]; i != 0; i = R[i]) if (S[i] < S[c]) c = i;
69
            remove(c);
70
71
            for (int i = D[c]; i != c; i = D[i]) {
72
                 ans[d] = Row[i];
73
                 for (int j = R[i]; j != i; j = R[j])remove(Col[j]);
                 if (Dance(d + 1))return true;
74
                 for (int j = L[i]; j != i; j = L[j])resume(Col[j]);
75
76
77
            resume(c);
78
            return false;
79
        }
80 };
```

## 3.18 ChthollyTree

```
// 要先Split右端点(r+1), 在Split左端点(l)
2
3
   #include <bits/stdc++.h>
4
   using namespace std;
5
6
   typedef long long ll;
7
8
9 struct node {
10
        int 1, r;
        mutable ll v;
11
12
        node (int L, int R = -1, ll V = 0) : l(L), r(R), v(V) {}
13
        bool operator < (const node &rhs) const {</pre>
            return 1 < rhs.1;</pre>
14
15
        }
16
   };
17
   set<node> s;
18
19
   auto Split(int pos) {
20
        auto it = s.lower_bound(node(pos));
21
        if(it != s.end() && it -> l == pos) return it;
22
23
        --it;
        int L = it \rightarrow l, R = it \rightarrow r;
24
25
        ll V = it \rightarrow v;
26
        s.erase(it);
27
        s.insert(node(L, pos - 1, V));
        return s.insert(node(pos, R, V)).first;
28
   }
29
30
31
   void assign_val(int l, int r, ll val) { // 推平操作
32
        auto itr = Split(r + 1);
33
        auto itl = Split(l);
        s.erase(itl, itr);
34
        s.insert(node(l, r, val));
35
36 }
37
38
   void add(int l, int r, ll val) { // 区间加
39
        auto itr = Split(r + 1);
40
        auto itl = Split(l);
        for( ;itl != itr; ++itl) {
41
42
            itl -> v += val;
        }
43
44
   }
45
   ll kth(int l, int r, int k) { // 区间第k小
46
47
        vector<pair<ll, int> > v;
        auto itr = Split(r + 1);
48
        auto itl = Split(l);
49
50
        for( ;itl != itr; ++itl) {
            v.push_back(pair<ll, int>{itl -> v, itl -> r - itl -> l + 1});
51
52
        }
53
        sort(v.begin(), v.end());
        for(auto it = v.begin();it != v.end(); ++it) {
54
            k -= it -> second;
55
            if(k <= 0) return it -> first;
56
        }
57
```

```
}
58
59
   ll quick_pow(ll a, ll b, ll p);
60
61
   11 qpow(int 1, int r, int ex, int p) {
62
        auto itr = Split(r + 1);
63
        auto itl = Split(l);
64
        11 \text{ ans} = 0;
65
        for( ;itl != itr; ++itl)
66
            ans = (ans + ll(itl -> r - itl -> l + 1) * quick_pow(itl -> v, ll(ex), ll(p)) %
67
        ll(p)) % ll(p);
68
        return ans % ll(p);
69
   }
   int main() {
70
71
        int n, m;
72
        cin >> n >> m;
        for (int i = 1; i \le n; ++i){
73
            11 x;
74
75
            cin >> x;
            s.insert(node(i,i,x));
76
77
        s.insert(node(n + 1, n + 1, 0));
78
        while(m--) {
79
            int opt, 1, r, x;
80
81
            cin >> opt >> l >> r >> x;
            if(opt == 1) add(l, r, x);
82
            else if(opt == 2) assign_val(l, r, x);
83
            else if(opt == 3) cout << kth(1, r, x) << endl;
84
            else {
85
                int y;
86
87
                cin >> y;
                cout << qpow(1, r, x, y) << endl;
88
89
            }
90
        }
   }
91
          monotonousQueue
   3.19
   template <typename T>
2
   struct monotonousQueue {
3
        std::vector<T> a;
4
        monotonousQueue(const std::vector<T>& init) : a(init) {}
5
        std::vector<T> Max(int k) {
6
7
            int n = (int) a.size();
            int head = 0, tail = -1;
8
            std::queue<int> que(n);
9
            std::vector<T> ans;
10
11
            for (int i = 0; i < n; i++) {
                while (head <= tail && a[que[tail]] <= a[i]) tail--;</pre>
12
13
                que[++tail] = i;
                while (que[head] + k <= i) head++;</pre>
14
                if (i >= k - 1) ans.push_back(a[que[head]]);
15
16
            }
17
            return ans;
18
        }
19
        std::vector<T> Min(int k) {
20
```

```
int n = (int) a.size();
21
            int head = 0, tail = -1;
22
23
            std::queue<int> que(n);
            std::vector<T> ans;
24
            for (int i = 0; i < n; i++) {
25
                while (head <= tail && a[que[tail]] >= a[i]) tail--;
26
                que[++tail] = i;
27
                while (que[head] + k <= i) head++;</pre>
28
29
                if (i >= k - 1) ans.push_back(a[que[head]]);
            }
30
31
            return ans;
32
        }
   };
33
   3.20
          monotonousStack
   #include <bits/stdc++.h>
   using namespace std;
3
   struct Monotone_stack {
4
        static const int N = 1e5 + 10;
5
6
        int a[N];
7
        stack<int> s;
8
        int n;
9
10
        void read() {
11
            cin >> n;
            for(int i = 1;i <= n; i++) cin >> a[i];
12
13
        }
14
15
        void Monotone_min() {
            for(int i = 1;i <= n; i++) {
16
17
                 if(s.empty() | | s.top() >= a[i])
18
                     s.push(a[i]);
19
                else {
                     while(!s.empty() && s.top() < a[i]) {</pre>
20
21
                         cout << s.top() << endl;</pre>
22
                         s.pop();
23
24
                     s.push(a[i]);
25
                }
26
            while(!s.empty()) {
27
                cout << s.top() << endl;</pre>
28
29
                s.pop();
30
            }
        }
31
32
        void Monotone_max() {
33
            for(int i = 1;i <= n; i++) {
34
                 if(s.empty() | | s.top() <= a[i])
35
                     s.push(a[i]);
36
37
                else {
38
                     while(!s.empty() && s.top() > a[i]) {
                         cout << s.top() << endl;</pre>
39
40
                         s.pop();
41
42
                     s.push(a[i]);
```

```
}
43
44
            while(!s.empty()) {
45
46
                cout << s.top() << endl;</pre>
47
                s.pop();
48
            }
49
        }
50
   }Worker;
51
   3.21
          difference
   #include <bits/stdc++.h>
2
   using namespace std;
3
4
   template <typename T>
5
   struct difference {
6
        int n;
7
8
        std::vector<T> d;
9
        difference(int _n) : n(_n), d(_n + 1) {}
10
        difference(std::vector<T>& init) : difference(init.size()) {
11
            d[0] = init[0];
12
            for (int i = 1; i < n; i++) {</pre>
13
14
                d[i] = init[i] - init[i - 1];
15
            }
16
        }
17
        void modify(int l, int r, T v) {
18
19
            assert(0 \le 1 \&\& 1 \le n - 1 \&\& 0 \le r \&\& r \le n - 1);
            d[l] += v;
20
21
            d[r + 1] -= v;
22
        }
23
        void solve() {
24
25
            for (int i = 1; i < n; i++) {
26
                d[i] += d[i - 1];
27
            }
28
        }
29
   };
30
31
   template <typename T>
32
   struct difference2d {
        int n, m;
33
        std::vector<std::vector<T>> d;
34
35
        difference2d(int _n, int _m) : n(_n), m(_m), d(_n + 1, std::vector<T>(_m + 1)) {}
36
        difference2d(std::vector<std::vector<T>>& init) : difference2d(init.size(), init
37
        [0].size()) {
            for (int i = 0; i < n; i++) {
38
                 for (int j = 0; j < m; j++) {
39
                     if (i == 0 | i j == 0) d[i][j] = init[i][j];
40
                     else d[i][j] = init[i][j] - init[i - 1][j] - init[i][j - 1] + init[i - 1][j]
41
       1][j - 1];
42
43
            }
        }
44
```

```
45
        void modify(int x1, int y1, int x2, int y2, T v) {
46
            assert(0 \le x1 \le n - 1 \&\& 0 \le y1 \le n - 1 \&\& 0 \le x2 \le n - 1 \&\& 0 \le y2 \le n
47
         - 1);
            d[x1][y1] += v;
48
            d[x1][y2 + 1] -= v;
49
50
            d[x2 + 1][y1] -= v;
            d[x2 + 1][y2 + 1] += v;
51
        }
52
53
   };
    3.22 trie
   class Trie {
1
   private:
2
        Trie* next[26] = {nullptr};
3
4
        int val;
   public :
5
6
        Trie() {}
7
        void insert(std::string& s) {
8
            Trie* root = this;
9
            for (char &c : s) {
10
                if (root -> next[c] == nullptr) {
11
                     root -> next[c] = new Trie();
12
13
                root = root -> next[c];
14
            }
15
            root -> val ++;
16
17
        }
18
19
        void del(std::string& s) {
            Trie* root = this;
20
            for (char &c : s) {
21
                root = root -> next[c];
22
23
24
            root -> val --;
        }
25
26
        int search(std::string& s) {
27
            int ans = 0;
28
            Trie* root = this;
29
            for (char& c : s) {
30
                if (!root -> next[c]) break ;
31
32
                root = root -> next[c];
                ans += root -> val;
33
            }
34
35
            return ans;
        }
36
37 };
   3.23 HashTable
1 template<typename T>
2 class HashTable{
3
   private:
        const int maxn;
```

```
std::vector<std::vector<T>> key, val;
5
6
7
   public:
        HashTable(int n) : maxn(n), key(n), val(n) {}
8
9
        int hash(int x){
10
            return (((long long)x * (x + 1)) ^ x) % maxn;
11
12
        void insert(int x){
13
            int u = hash(x);
14
            for(int v = 0; v < (int)key[u].size(); ++v)</pre>
15
                if(key[u][v] == x){
16
17
                     ++val[u][v];
18
                     return;
19
            key[u].push_back(x), val[u].push_back(1);
20
21
        T query(int x){
22
            int u = hash(x);
23
24
            for(int v = 0; v < (int)key[u].size(); ++v)</pre>
                if(key[u][v] == x)
25
26
                     return val[u][v];
27
            return 0;
28
        }
29
  };
   3.24 2-4 维前缀和
   // 统计(a,b)到(c,d)这个矩阵中的所有0子矩阵
1
2
3 \quad const int N = 50 + 10;
4
   int sum[N][N];
   int Q[N][N][N][N];
5
6
7
   void solve() {
        int n, m, q; cin >> n >> m;
8
9
        for(int i = 1; i <= n; i++) {
10
            string s; cin >> s;
            for(int j = 1; j \le m; j++) {
11
                sum[i][j]´= (s[j´-´1]´-`'0') + sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j
12
         - 1];
13
14
        }
15
16
        for(int a = 1; a <= n; a++) {
            for(int b = 1;b <= m; b++) {
17
                for(int c = a; c <= n; c++) {</pre>
18
                     for(int d = b;d <= m; d++) {</pre>
19
                         if(sum[c][d] - sum[a - 1][d] - sum[c][b - 1] + sum[a - 1][b - 1] ==
20
        0) {
                             Q[a][b][c][d]++;
21
                         }
22
23
                    }
24
                }
            }
25
        }
26
27
        for(int a = n; a >= 1; a--) {
28
```

```
for(int b = m;b >= 1; b--) {
29
                 for(int c = 1; c <= n; c++) {
30
                      for(int d = 1;d <= m; d++) {</pre>
31
                          Q[a][b][c][d] += Q[a + 1][b][c][d];
32
33
34
                 }
            }
35
        }
36
37
38
        for(int a = n; a >= 1; a--) {
             for(int b = m;b >= 1; b--) {
39
40
                 for(int c = 1; c <= n; c++) {</pre>
                      for(int d = 1;d <= m; d++) {</pre>
41
                          Q[a][b][c][d] += Q[a][b + 1][c][d];
42
43
                 }
44
            }
45
        }
46
47
        for(int a = n; a >= 1; a--) {
48
             for(int b = m;b >= 1; b--) {
49
                 for(int c = 1; c <= n; c++) {</pre>
50
                      for(int d = 1;d <= m; d++) {</pre>
51
                          Q[a][b][c][d] += Q[a][b][c - 1][d];
52
53
                      }
54
                 }
            }
55
        }
56
57
        for(int a = n; a >= 1; a--) {
58
59
             for(int b = m;b >= 1; b--) {
                 for(int c = 1; c <= n; c++) {
60
                      for(int d = 1;d <= m; d++) {</pre>
61
                          Q[a][b][c][d] += Q[a][b][c][d - 1];
62
63
                 }
64
            }
65
66
        }
67
        while(q--) {
68
             int a, b, c, d; cin >> a >> b >> c >> d;
69
             cout << Q[a][b][c][d] << endl;</pre>
70
71
        }
72
   }
    3.25 \quad simpleDSU
1 class DSU {
2
   private:
3
        std::vector<int> f, siz;
        std::vector<int> dep;
   public :
5
        DSU(int n) : f(n), dep(n), siz(n, 1) { iota(f.begin(), f.end(), 0); }
6
7
        int find(int x) {
8
            while(x != f[x]) x = f[x] = f[f[x]];
9
             return x;
10
        bool same(int x, int y) { return find(x) == find(y); }
11
```

```
bool merge(int x, int y) {
12
13
            x = find(x);
            y = find(y);
14
            if(x == y) return false;
15
            if (dep[x] > dep[y]) std::swap(x, y);
16
17
            siz[y] += siz[x];
            f[x] = y;
18
            dep[y] = std::max(dep[y], dep[x] + 1);
19
20
            return true;
21
22
       int size(int x) { return siz[find(x)]; }
23
   };
   3.26 valueDSU
   class DSU {
   private:
       std::vector<int> f, siz, val;
3
4
   public:
5
       DSU(int n) : f(n), val(n), siz(n, 1) { iota(f.begin(), f.end(), 0); }
6
7
       int find(int x) {
8
9
            if (x != f[x]) {
                int fa = f[x];
10
                f[x] = find(f[x]);
11
                val[x] += val[fa];
12
            }
13
            return f[x];
14
       }
15
       bool same(int x, int y) { return find(x) == find(y); }
16
       bool merge(int x, int y, int v) {
17
            int nx = find(x);
18
            int ny = find(y);
19
            if(nx == ny) return false;
20
            siz[nx] += siz[ny];
21
22
            f[ny] = nx;
23
           val[ny] = val[x] + v - val[y];
24
            return true;
25
       int size(int x) { return siz[find(x)]; }
26
27
  };
   3.27
          modifyDSU
   struct node {
1
2
       int x, y, z;
3 };
4
  struct UnionFind {
5
6
   private:
        int rk[N], pre[N], siz[N], totNode;//N为最大点数
7
       stack<node> st;//node记录上次修改的内容
8
9
   public:
       void init(int tot) {
10
11
            totNode = tot;
            for (int i = 1; i <= totNode; i++)</pre>
12
```

```
pre[i] = i, siz[i] = rk[i] = 1;
13
14
       int find(int x) { while (x ^ pre[x]) x = pre[x]; return x; }
15
       void merge(int x, int y) {//按秩合并
16
            x = find(x), y = find(y);
17
            if (x == y) return;
18
            if (rk[x] < rk[y]) swap(x, y);
19
            st.push(node{ y, rk[x], siz[y] });
20
            pre[y] = x, rk[x] += rk[x] == rk[y], siz[x] += siz[y];
21
22
23
       int start() { return st.size(); }
24
       void end(int last) {//撤回merge操作
            while (st.size() > last) {
25
                node tp = st.top();
26
                rk[pre[tp.x]] -= tp.y, siz[pre[tp.x]] -= tp.z;
27
                pre[tp.x] = tp.x;
28
29
                st.pop();
            }
30
31
32
       bool judge() { return siz[find(1)] == totNode; }
33
  };
   3.28 varietyDSU
   class DSU {
1
2
   private:
3
       std::vector<int> f, siz;
4
5
   public:
       DSU(int n) : f(n), siz(n, 1) { iota(f.begin(), f.end(), 0); }
6
7
       int find(int x) {
            while(x != f[x]) x = f[x] = f[f[x]];
8
9
            return x;
10
       bool same(int x, int y) { return find(x) == find(y); }
11
       bool merge(int x, int y) {
12
13
            x = find(x);
            y = find(y);
14
            if(x == y) return false;
15
            siz[x] += siz[y];
16
17
            f[y] = x;
18
            return true;
19
        int size(int x) { return siz[find(x)]; }
20
21
   };
22
   int main() {
23
       int n, q;
24
       cin >> n >> q;
25
       for(int i = 1; i \le 2 * n; i++) f[i] = i;// **
26
27
       while(q--) {
28
            int flag, x, y;
29
30
            cin >> flag >> x >> y;
            if(flag) { // 敌人
31
                merge(x + n, y);
32
33
                merge(y + n, x);
34
            }
```

```
else {
35
36
               merge(x, y); // 同伴
37
38
       int ans = 0;
39
       for(int i = 1;i <= n; i++) {
40
           if(f[i] == i) ans ++;
41
42
43
       cout << ans << endl;</pre>
44
   }
   3.29
          KD 求矩阵权值和
   #include <bits/stdc++.h>
   using namespace std;
2
3
   const int MAX = 200005;
4
   const double alpha = 0.75;
5
   //查包含在x1,y1,x2,y2为左下角和右上角的矩形里面权值之和
6
7
   //K-D Tree 二维划分树
   int n, ans, rt, WD, tot, top, rub[MAX];
8
9
   struct node {
10
11
       int x[2], w;
12 } p[MAX];
13
   struct K_D_tree {
14
15
       int ls, rs, siz, mn[2], mx[2], sum;
       //mn[0], mx[0] -> x的取值范围
16
17
       //mn[1], mx[1] -> y的取值范围
       node tmp;
18
19
   } t[MAX];
20
   int operator < (const node &a, const node &b) { return a.x[WD] < b.x[WD]; }</pre>
21
22
   int newnode() {
23
24
       if (top) return rub[top--];
25
       else return ++tot;
26
   }
27
28
   void push_up(int u) {
       for (int i = 0; i <= 1; i++) {//更新x, y的取值范围
29
30
           t[u].mn[i] = t[u].mx[i] = t[u].tmp.x[i];
31
           if (t[u].ls) {//左子树的最大最小值
32
               t[u].mn[i] = min(t[u].mn[i], t[t[u].ls].mn[i]);
               t[u].mx[i] = max(t[u].mx[i], t[t[u].ls].mx[i]);
33
34
           if (t[u].rs) {//右子树的最大最小值
35
               t[u].mn[i] = min(t[u].mn[i], t[t[u].rs].mn[i]);
36
37
               t[u].mx[i] = max(t[u].mx[i], t[t[u].rs].mx[i]);
           }
38
39
       t[u].sum = t[t[u].ls].sum + t[t[u].rs].sum + t[u].tmp.w;
40
41
       t[u].siz = t[t[u].ls].siz + t[t[u].rs].siz + 1;
42
   }
43
   int build(int 1, int r, int wd) {
44
       if (l > r) return 0;
45
```

```
46
        int u = newnode();
        int m = (l + r) >> 1;
47
        WD = wd; nth_element(p + l, p + m, p + r + 1);
48
        t[u].tmp = p[m];
49
        t[u].ls = build(l, m - 1, wd ^ 1);
50
        t[u].rs = build(m + 1, r, wd ^ 1);
51
52
        push_up(u);
        return u;
53
    }
54
55
    void pia(int u, int num) {//拍扁回炉重做
56
57
        if (t[u].ls) pia(t[u].ls, num);
        p[t[t[u].ls].siz + num + 1] = t[u].tmp, rub[++top] = u;
58
        if (t[u].rs) pia(t[u].rs, t[t[u].ls].siz + num + 1);
59
    }
60
61
    void check(int &u, int wd) {//检查是否平衡, 不平衡则需要重建
62
        if (t[u].siz * alpha < t[t[u].ls].siz | | t[u].siz * alpha < t[t[u].rs].siz) pia(u,
        0), u = build(1, t[u].siz, wd);
    }
64
65
    void insert(int &u, node tp, int wd) {//插入点
66
        if (!u) {
67
            u = newnode();
68
69
            t[u].ls = t[u].rs = 0, t[u].tmp = tp;
70
            push_up(u);
            return;
71
72
        if (tp.x[wd] <= t[u].tmp.x[wd]) insert(t[u].ls, tp, wd ^ 1);</pre>
73
        else insert(t[u].rs, tp, wd ^ 1);
74
75
        push_up(u);
        check(u, wd);
76
    }
77
78
    bool in(int x1, int y1, int x2, int y2, int X1, int Y1, int X2, int Y2) {//完全被包含
79
        return (x1 <= X1 && X2 <= x2 && y1 <= Y1 && Y2 <= y2);
80
81
82
    bool out(int x1, int y1, int x2, int y2, int X1, int Y1, int X2, int Y2) {//完全无交集
83
        return (x1 > X2 || x2 < X1 || y1 > Y2 || y2 < Y1);
84
85
86
    int query(int u, int x1, int y1, int x2, int y2) {
87
88
        if (!u) return 0;
        int res = 0;
89
90
        if (in(x1, y1, x2, y2, t[u].mn[0], t[u].mn[1], t[u].mx[0], t[u].mx[1])) return t[u
        if (out(x1, y1, x2, y2, t[u].mn[0], t[u].mn[1], t[u].mx[0], t[u].mx[1])) return 0;
91
        if (in(x1, y1, x2, y2, t[u].tmp.x[0], t[u].tmp.x[1], t[u].tmp.x[0], t[u].tmp.x[1]))
92
         res += t[u].tmp.w;
93
        res += query(t[u].ls, x1, y1, x2, y2) + query(t[u].rs, x1, y1, x2, y2);
94
        return res;
95 }
96
    void init() {
97
98
        ans = rt = top = tot = 0;
99
100
101
```

```
102 void solve() {
        scanf("%d", &n);
103
        while(1) {
104
             int opt; scanf("%d",&opt);
105
106
             if(opt == 1) {
                 int x, y, w; scanf("%d%d%d",&x,&y,&w);
107
                 insert(rt, node{x \land ans, y \land ans, w \land ans}, 0);
108
             }
109
             else if(opt == 2) {
110
                 int x1, y1, x2, y2; scanf("%d%d%d",&x1,&y1,&x2,&y2);
111
                 ans = query(rt, x1 ^ ans, y1 ^ ans, x2 ^ ans, y2 ^ ans);
112
113
                 printf("%d\n",ans);
             }
114
            else break;
115
        }
116
117
          KD 求最近点对距离
 1 // 二维平面里最近点对距离
 2 #include <bits/stdc++.h>
 3 using namespace std;
 4
 5 typedef long long ll;
    const int N = 1e5 + 10;
 6
    struct node {
 7
 8
        ll x[2];
 9
    }a[N], b[N];
10
11
   int now, n;
    ll ans;
12
13
    map<pair<ll, ll>, int> mp;
14
15 bool cmp(node a, node b) { return a.x[now] < b.x[now]; }</pre>
16
17  ll sqr(int x) { return (ll)x * x; }
   ll dis(node a, node b) { return sqr(a.x[0] - b.x[0]) + sqr(a.x[1] - b.x[1]); }
20
    void build(int l, int r, int d) {
21
        if(l >= r) return ;
        int m = (l + r) >> 1;
22
        now = d;
23
        nth_element(a + 1, a + m, a + r, cmp);
24
        build(l, m, d ^ 1);
25
26
        build(m + 1, r, d \wedge 1);
    }
27
28
    void query(int 1, int r, int d, node p) {
29
30
        if(l >= r) return ;
        int m = (l + r) >> 1;
31
        int ql = l, qr = m;
32
        ll res = dis(a[m], p);
33
        if(ans == 0 || res && ans > res) ans = res;
34
35
        if(p.x[d] > a[m].x[d]) ql = m + 1, qr = r;
        query(ql, qr, d ^ 1, p);
36
37
        if(ans > sqr(a[m].x[d] - p.x[d]))
38
             query(l + m - ql + 1, m + r - qr, d \wedge 1, p);
39 }
```

```
40
   void solve() {
41
       scanf("%d",&n);
42
       ll sum = 5e18;
43
       for(int i = 0;i < n; i++) {
    scanf("%lld %lld",&a[i].x[0],&a[i].x[1]);</pre>
44
45
          if(mp[{a[i].x[0], a[i].x[1]}]) sum = 0;
46
          else mp[{a[i].x[0], a[i].x[1]}]++;
47
          b[i] = a[i];
48
49
       }
       build(0, n, 0);
50
       for(int i = 0; i < n; i++) {
51
52
          ans = 0;
          query(0, n, 0, b[i]);
53
54
          sum = min(ans, sum);
55
       printf("%.4lf\n",sqrt(1.0 * sum));
56
57
   }
         CDQ 分治
   3.31
  // 原问题: "任意两个元素之间的贡献"
  // 降维成的子问题: "左段元素对右段每一个元素的贡献"
5 // 分而治之, 最基础的应用为归并排序
  // 将一个区间分成两个区间[l,mid]和[mid+1,r],然后排序使其有序
7 // 注意事项:
  // 1、离散化
  // 2、如果有多个点的参量完全相同,由于顺着添加,查出来的最后一个答案才是正确的
  // 对于y排序,我们可以直接对两段区间Sort,然而归并排序本身就是分治,我们可以在CDQ的过程中进行归并排
      序,要比Sort少一个log
11
3.32 cdq 处理逆序数
   const int N = 1e5 + 10;
2
   int ans[N], cnt[N];
3
4
   struct star {
5
       int x, y, id;
6
   }a[N], tmp[N];
   bool cmp(star a, star b) {
8
       if(a.y == b.y) return a.x < b.x;
9
       return a.y < b.y;</pre>
10
   }
11
12
   void cdq(int 1, int r) {
13
       if(l == r) return ;
14
15
       int m = (1 + r) / 2;
       cdq(1, m);
16
17
       cdq(m + 1, r);
18
       int p = 1, q = m + 1;
       for(int i = l;i <= r; i++) {</pre>
19
20
          if((p \le m \& a[p].x \le a[q].x) | | q > r) {
```

```
tmp[i] = a[p++];
21
22
            }
23
            else {
                 ans[a[q].id] += p - 1;
24
25
                 tmp[i] = a[q++];
26
27
        for(int i = l;i <= r; i++) a[i] = tmp[i];</pre>
28
   }
29
30
   void solve() {
31
32
        int n; cin >> n;
        for(int i = 1; i <= n; i++) cin >> a[i].x >> a[i].y, a[i].id = i;
33
        sort(a + 1, a + n + 1, cmp);
34
35
        cdq(1, n);
        for(int i = 1;i <= n; i++) cnt[ans[i]]++;</pre>
36
37
        for(int i = 0;i < n; i++) cout << cnt[i] << endl;</pre>
38
   3.33 cdq 处理二维偏序
1
   const int N = 1e5 + 10;
3 int ans[N], cnt[N];
4
   struct star {
5
6
        int x, y, id;
7
   }a[N], tmp[N];
8
   bool cmp(star a, star b) {
9
10
        if(a.x == b.x) return a.y < b.y;
        return a.x < b.x;</pre>
11
12
   }
13
   void cdq(int 1, int r) {
14
        if(l == r) return ;
15
16
        int m = (l + r) / 2;
17
        cdq(1, m);
18
        cdq(m + 1, r);
19
        int p = 1, q = m + 1;
20
        for(int i = l;i <= r; i++) {</pre>
            if((p \le m \&\& a[p].x \le a[q].x) || q > r) {
21
22
                 tmp[i] = a[p++];
            }
23
            else {
24
                 ans[a[q].id] += i - 1;
25
                 tmp[i] = a[q++];
26
            }
27
28
        for(int i = l;i <= r; i++) a[i] = tmp[i];</pre>
29
   }
30
31
32
   void solve() {
33
        int n; cin >> n;
        for(int i = 1;i <= n; i++) cin >> a[i].x >> a[i].y;
34
35
        sort(a + 1, a + n + 1, cmp);
        cdq(1, n);
36
37
        for(int i = 1;i <= n; i++) cnt[ans[i]]++;</pre>
```

```
for(int i = 0;i < n; i++) cout << cnt[i] << endl;</pre>
38
  }
39
   3.34 cdq 套 cdq 处理三维偏序
   // 一维排序、二维cdq、三维树状数组
1
2
   const int N = 1e5 + 10;
3
   struct node {
4
5
        int x, y, z;
6
        int id;
        int tag;
7
8
        bool operator < (const node &a)const{</pre>
9
            if(x != a.x) return x < a.x;
            if(y != a.y) return y < a.y;
10
11
            return z < a.z;</pre>
12
        bool operator == (const node &a)const{
13
            return x == a.x & y == a.y & z == a.z;
14
15
   }a[N], b[N], tmp[N];
16
17
   int ans[N];
18
   int n;
19
20
21
   void cdq2(int l, int r) {
22
        if(l == r) return ;
        int mid = (l + r) / 2;
23
        cdq2(1, mid); cdq2(mid + 1, r);
24
        int p = 1, q = mid + 1, cnt = 0;
25
        for(int i = l;i <= r; i++) {</pre>
26
27
            if(q > r | | (p \le mid \&\& b[p].z \le b[q].z)) {
28
                if(b[p].tag == 0) cnt++;
29
                tmp[i] = b[p++];
30
            }
            else {
31
32
                if(b[q].tag == 1) ans[b[q].id] += cnt;
33
                tmp[i] = b[q++];
34
            }
35
        for(int i = l;i <= r; i++) b[i] = tmp[i];</pre>
36
   }
37
38
   void cdq1(int l, int r) {
39
        if(l == r) return ;
int mid = (l + r) / 2;
40
41
42
        cdq1(l, mid); cdq1(mid + 1, r);
43
        int p = 1, q = mid + 1;
        /* 因为是计算左端元素对右端元素的影响, 所以需要打个标记tag来记录他是左端还是右端元素 */
44
45
        for(int i = l;i <= r; i++) {
            if(q > r \mid | (p \le mid \&\& a[p].y \le a[q].y)) {
46
                a[p].tag = 0;
47
48
                b[i] = a[p++];
            }
49
            else {
50
51
                a[q].tag = 1;
52
                b[i] = a[q++];
53
            }
```

```
54
        for(int i = l;i <= r; i++) a[i] = b[i];
55
        cdq2(1, r);
56
   }
57
58
   void solve() {
59
60
        n = read();
        for(int i = 1;i <= n; i++) {</pre>
61
            a[i].x = read(), a[i].y = read(), a[i].z = read();
62
63
            a[i].id = i;
64
        }
65
        sort(a + 1, a + n + 1);
        for(int i = n - 1; i >= 1; i--) {
66
            if(a[i + 1] == a[i]) ans[a[i].id] = ans[a[i + 1].id] + 1;
67
68
        cdq1(1, n);
69
        for(int i = 1;i <= n; i++) cout << ans[i] << endl;</pre>
70
71
         cdq 套树状数组处理三维偏序
1 // 一维排序、二维cdq、三维树状数组
3
   const int N = 1e5 + 10;
   struct node {
4
        int x, y, z;
5
6
        int id;
7
        bool operator < (const node &a)const{</pre>
8
            if(x != a.x) return x < a.x;
            if(y != a.y) return y < a.y;
9
            return z < a.z;</pre>
10
11
        bool operator == (const node &a)const{
12
            return x == a.x &  y == a.y & z == a.z;
13
14
   }a[N], b[N];
16
   int n;
17
18
19
   int ans[N];
20
   struct BIT {
21
   #define lowbit(x) (x & (-x))
22
        int n;
23
24
        int t[N];
25
        void init(int _n) {
26
27
            mem(t, 0);
            n = _n;
28
29
        }
30
        void update(int x, int val) {
31
32
            while (x \le n) {
                t[x] += val;
33
                x += lowbit(x);
34
35
            }
        }
36
37
```

```
int query(int x) {
38
39
            int ans = 0;
            while (x) {
40
                ans += t[x];
41
42
                x -= lowbit(x);
43
44
            return ans;
45
   }bit;
46
47
   void cdq(int 1, int r) {
48
49
       if(l == r) return ;
       int mid = (l + r) / 2;
50
       cdq(l, mid);
51
       cdq(mid + 1, r);
52
       int p = 1, q = mid + 1;
53
       for(int i = l;i <= r; i++) {
54
            if(q > r | l (p \le mid \&\& a[p].y \le a[q].y)) {
55
                bit.update(a[p].z, 1);
56
                b[i] = a[p++];
57
            }
58
            else {
59
                ans[a[q].id] += bit.query(a[q].z);
60
61
                b[i] = a[q++];
62
            }
63
       for(int i = l; i \leftarrow mid; i++) bit.update(a[i].z, -1);
64
       for(int i = l;i <= r; i++) a[i] = b[i];</pre>
65
   }
66
67
   void solve() {
68
       n = read();
69
70
       int mx = 0;
       for(int i = 1;i <= n; i++) {</pre>
71
            a[i].x = read(), a[i].y = read(), a[i].z = read();
72
            a[i].id = i;
73
74
           mx = max(mx, a[i].z);
75
       bit.init(mx);
76
77
       sort(a + 1, a + n + 1);
       for(int i = n - 1; i >= 1; i--) {
78
            if(a[i] == a[i + 1]) ans[a[i].id] = ans[a[i + 1].id] + 1;
79
80
81
       cdq(1, n);
       for(int i = 1;i <= n; i++) cout << ans[i] << endl;</pre>
82
83
  }
         cdg 维护矩阵内二维数点
   3.36
1 // 求二维平面上(x1,y1)到(x2,y2)的矩阵中数点
  // 利用前缀和思想,把问题划分成[x2,y2] - [x1-1,y2] - [x2,y2-1] + [x1-1,y1-1]
3
4
5
   // 所有要建立4个虚点为查询点, 而原本实点为修改点
6
7
   const int N = 3e6 + 10;
  struct node {
```

```
10
        int x, y, opt, id;
        // opt为操作类型, 1为修改, 0为查询
11
        bool operator < (const node& o) const {</pre>
12
             return x == o.x ? (y == o.y ? opt : y < o.y) : x < o.x;
13
14
15
        // 注意排序顺序, 坐标相同时, 要使opt放前面, 因为要先修改
16
        bool operator == (const node &o) const {
17
             return x == o.x \&\& y == o.y;
18
19
   }a[N], tmp[N];
21
  int ans[N];
22
23
   void cdq(int 1, int r) {
24
        if(l == r) return ;
25
        int mid = (l + r) / 2;
26
27
        cdq(l, mid); cdq(mid + 1, r);
        int p = 1, q = mid + 1, cnt = 0;
28
        for(int i = l;i <= r; i++) {
29
             if(q > r | | (p \le mid \&\& a[p].y \le a[q].y)) {
30
                 cnt += a[p].opt;
31
                 tmp[i] = a[p++];
32
33
             }
34
             else {
                 ans[a[q].id] += cnt;
35
                 tmp[i] = a[q++];
36
37
38
        for(int i = l;i <= r; i++) a[i] = tmp[i];</pre>
39
40
   }
41
   void solve() {
42
        int n = read(); m = read();
43
        for(int i = 1;i <= n; i++) {</pre>
44
             a[i].x = read(), a[i].y = read(), a[i].opt = 1;
45
46
        }
47
        int _n = 0;
        for(int i = 1;i <= m; i++) {</pre>
48
             int x1 = read(), y1 = read(), x2 = read(), y2 = read();
49
            a[++n] = (node)\{x2, y2, 0, ++_n\};
50
            a[++n] = (node)\{x2, y1 - 1, 0, ++_n\};

a[++n] = (node)\{x1 - 1, y2, 0, ++_n\};

a[++n] = (node)\{x1 - 1, y1 - 1, 0, ++_n\};
51
52
53
        }
54
        sort(a + 1, a + n + 1);
55
        cdq(1, n);
56
        for(int i = 1; i + 3 \le n; i += 4) {
57
             cout << ans[i] - ans[i + 1] - ans[i + 2] + ans[i + 3] << endl;
58
59
        }
60 }
    3.37
           advanced01trie
1 #include <bits/stdc++.h>
2
   using namespace std;
4 const int N = 2e7 + 10;
```

```
int t[N][2];
   int cnt, root[N], sz[N][2];
7
8
   int a[N];
9
   void insert(int pre, int &now, int i, int x) {
10
        if(i < 0) return ;</pre>
11
12
        now = ++cnt;
        int d = x >> i \& 1;
13
        t[now][d \land 1] = t[pre][d \land 1];
14
        sz[now][d \land 1] = sz[pre][d \land 1]; sz[now][d] = sz[pre][d] + 1;
15
16
        insert(t[pre][d], t[now][d], i - 1, x);
   }
17
18
   int query(int 1, int r, int i, int x) {
19
        if(i < 0) return 0;</pre>
20
21
        int d = x >> i \& 1;
22
        int tmp = sz[r][d ^ 1] - sz[l][d ^ 1];
        if(tmp > 0) return query(t[l][d ^1], t[r][d ^1], i - 1, x) + (1 << i);
23
        else return query(t[l][d], t[r][d], i - 1, x);
24
25 }
26
27 int main() {
28
        int n, m;
29
        cin >> n >> m;
        for(int i = 1;i <= n; i++) {</pre>
30
            int x;
31
32
            cin >> x;
            insert(root[i - 1], root[i], 30, x);
33
34
        while(m--) {
35
36
            int 1, r, x;
37
            cin >> 1 >> r >> x;
            cout << query(root[l - 1], root[r], 30, x) << endl;</pre>
38
39
        }
40 }
   3.38 advancedArray
   #include <bits/stdc++.h>
3
   using namespace std;
   const int N = 1e5 + 10;
5
6
   struct Node {
7
        int l, r, val;
8
   }hjt[N * 40];
9
  int cnt, root[N];
   int a[N];
12
   inline void build(int &now, int l, int r) {
13
        now = ++cnt;
14
        if(l == r) {
15
            hjt[now].val = a[l];
16
17
            return ;
18
19
        int m = (l + r) >> 1;
```

```
build(hjt[now].1, 1, m);
20
21
        build(hjt[now].r, m + 1, r);
   }
22
23
   inline void modify(int ver, int &now, int l, int r, int pos, int value) {
24
        hjt[now = ++cnt] = hjt[ver];
25
26
        if(l == r) {
            hjt[now].val = value;
27
28
            return ;
        }
29
30
        int m = (l + r) >> 1;
        if(pos <= m) modify(hjt[ver].l, hjt[now].l, l, m, pos, value);</pre>
31
        else modify(hjt[ver].r, hjt[now].r, m + 1, r, pos, value);
32
   }
33
34
   inline int query(int now, int l, int r, int pos) {
35
        if(l == r) return hjt[now].val;
36
        int m = (l + r) >> 1;
37
        if(pos <= m) return query(hjt[now].1, 1, m, pos);</pre>
38
39
        else return query(hjt[now].r, m + 1, r, pos);
40 }
41
   int main() {
42
43
        int n, m;
44
        cin >> n >> m;
        for(int i = 1;i <= n; i++) cin >> a[i];
45
        build(root[0], 1, n);
46
        for(int i = 1;i <= m; i++) {</pre>
47
            int ver, opt;
48
49
            cin >> ver >> opt;
50
            if(opt == 1) {
                int pos, value;
51
52
                cin >> pos >> value;
                modify(root[ver], root[i], 1, n, pos, value);
53
            }
54
            else {
55
56
                int pos;
57
                cin >> pos;
                root[i] = root[ver];
58
59
                cout << query(root[i], 1, n, pos) << endl;</pre>
60
            }
        }
61
62
   }
   3.39
          anvancedDSU
1 #include <bits/stdc++.h>
2 using namespace std;
3
   const int N = 1e5 + 10;
4
5
   struct Node {
6
7
        int l, r, val;
   }hjt[N * 40 * 2];
8
int cnt, rootfa[N], rootdep[N], tot;
11
   int n;
12
```

```
inline void build(int &now, int l, int r) {
        now = ++cnt;
14
        if(l == r) {
15
            hjt[now].val = ++tot;
16
17
            return ;
        }
18
19
        int m = (l + r) >> 1;
        build(hjt[now].l, l, m);
20
        build(hjt[now].r, m + 1, r);
21
22
   }
23
24
   inline void modify(int ver, int &now, int l, int r, int pos, int value) {
        hjt[now = ++cnt] = hjt[ver];
25
26
        if(l == r) {
27
            hjt[now].val = value;
            return ;
28
29
30
        int m = (l + r) >> 1;
        if(pos <= m) modify(hjt[ver].l, hjt[now].l, l, m, pos, value);</pre>
31
        else modify(hjt[ver].r, hjt[now].r, m + 1, r, pos, value);
32
33 }
34
   inline int query(int now, int l, int r, int pos) {
35
        if(l == r) return hjt[now].val;
37
        int m = (l + r) >> 1;
        if(pos <= m) return query(hjt[now].1, 1, m, pos);</pre>
38
39
        else return query(hjt[now].r, m + 1, r, pos);
   }
40
41
   inline int find(int ver, int x) {
42
        int fx = query(rootfa[ver], 1, n, x);
43
        return fx == x ? x : find(ver, fx);
44
   }
45
46
   inline void merge(int ver, int x, int y) {
47
        x = find(ver - 1, x);
48
49
        y = find(ver - 1, y);
50
        if(x == y) {
            rootfa[ver] = rootfa[ver - 1];
51
            rootdep[ver] = rootdep[ver - 1];
52
53
        else {
54
            int depx = query(rootdep[ver - 1], 1, n, x);
55
            int depy = query(rootdep[ver - 1], 1, n, y);
56
            if(depx < depy) {</pre>
57
                modify(rootfa[ver - 1], rootfa[ver], 1, n, x, y);
58
59
                rootdep[ver] = rootdep[ver - 1];
            }
60
            else if(depx > depy) {
61
62
                modify(rootfa[ver - 1], rootfa[ver], 1, n, y, x);
63
                rootdep[ver] = rootdep[ver - 1];
64
            }
            else {
65
                modify(rootfa[ver - 1], rootfa[ver], 1, n, x, y);
66
                modify(rootdep[ver - 1], rootdep[ver], 1, n, y, depy + 1);
67
68
            }
69
        }
70
   }
71
```

```
int main() {
72
73
        int m;
74
        cin >> n >> m;
        build(rootfa[0], 1, n);
75
76
        for(int ver = 1;ver <= m; ver++) {</pre>
77
            int opt, x, y;
            cin >> opt;
78
            if(opt == 1) {
79
80
                cin >> x >> y;
81
                merge(ver, x, y);
            }
82
83
            else if(opt == 2) {
                cin >> x;
84
                rootfa[ver] = rootfa[x];
85
                rootdep[ver] = rootdep[x];
86
            }
87
            else {
88
                cin >> x >> y;
89
                rootfa[ver] = rootfa[ver - 1];
90
                rootdep[ver] = rootdep[ver - 1];
91
                int u = find(ver, x);
92
                int v = find(ver, y);
93
                if(u == v) cout << 1 << endl;
94
95
                else cout << 0 << endl;</pre>
96
            }
        }
97
98 }
   3.40
          扫描线求面积并
1 // 横向扫描
   #include <bits/stdc++.h>
3
   using namespace std;
4
  const int N = 2e5 + 10;
5
6
```

```
7 #define lc u << 1</pre>
   #define rc u \ll 1 | 1
9
   #define mid (t[u].l + t[u].r) \gg 1
10
11 int n, cnt;
12
13
   double v[N];
   struct L {
14
15
        double x, y1, y2;
16
        int state;
        bool operator < (L rhs) {return x < rhs.x; }</pre>
17
   }line[N << 2];
18
19
   struct Node {
20
21
        int 1, r, cover;
        double len;
22
23
   }t[N << 2];
24
   inline void push_up(int u) {
25
        if(t[u].cover) t[u].len = v[t[u].r + 1] - v[t[u].l];
26
        else if(t[u].l == t[u].r) t[u].len = 0;
27
28
        else t[u].len = t[lc].len + t[rc].len;
```

```
29 }
30
   void build(int u, int l, int r) {
31
        t[u].l = l; t[u].r = r;
32
33
        if(l == r) {
             t[l].cover = t[l].len = 0;
34
35
             return;
        }
36
        int m = (l + r) >> 1;
37
        build(lc, l, m);
38
        build(rc, m + 1, r);
39
40
   }
41
    void modify(int u, int ql, int qr, int state) {
42
        if(ql <= t[u].l && t[u].r <= qr) {</pre>
43
             t[u].cover += state;
44
45
             push_up(u);
             return ;
46
47
        if(ql <= mid) modify(lc, ql, qr, state);</pre>
48
        if(qr > mid) modify(rc, ql, qr, state);
49
        push_up(u);
50
  }
51
52
53
   void init() {
54
        cin >> n;
        for(int i = 1;i <= n; i++) {</pre>
55
             double x1, y1, x2, y2;
56
             scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
57
             line[i] = L\{x1, y1, y2, 1\}; v[i] = y1;
58
59
             line[n + i] = L\{x2, y1, y2, -1\}; v[n + i] = y2;
        }
60
        n <<= 1;
61
        sort(line + 1, line + n + 1);
62
        sort(v + 1, v + n + 1);
63
        cnt = unique(v + 1, v + n + 1) - (v + 1);
64
        build(1, 1, cnt);
65
66
   }
67
   void solve() {
68
        double ans = 0;
69
        for(int i = 1;i <= n; i++) {</pre>
70
             int ql = lower_bound(v + 1, v + cnt + 1, line[i].y1) - v;
int qr = lower_bound(v + 1, v + cnt + 1, line[i].y2) - v - 1;
71
72
             modify(1, ql, qr, line[i].state);
73
74
             ans += t[1].len * (line[i + 1].x - line[i].x);
75
        cout << ans << endl;</pre>
76
  }
77
78
79
   int main() {
80
        init();
81
        solve();
82
  }
```

## 3.41 扫描线求周长并

1 // 纵向扫描

```
#include <bits/stdc++.h>
   using namespace std;
3
   const int N = 2e5 + 10;
5
6
   #define INF 0x3fffff
7
   #define lc u << 1</pre>
8
9 #define rc u << 1 | 1
10 #define mid (t[u].l + t[u].r) >> 1
11
12 int n;
13
14 struct L {
        int y, x1, x2;
15
16
        int state;
        bool operator < (L rhs) {return y < rhs.y; }</pre>
17
   }line[N << 2];
18
19
20 struct Node {
21
        int l, r, cover;
22
        bool ls, rs;
23
        int num;
24
        int len;
25 }t[N << 2];
26
   inline void push_up(int u) {
27
28
        if(t[u].cover) {
            t[u].len = t[u].r - t[u].l + 1;
29
            t[u].ls = t[u].rs = 1;
30
            t[u].num = 1;
31
32
        else if(t[u].l == t[u].r) {
33
            t[u].ls = t[u].rs = 0;
34
            t[u].len = t[u].num = 0;
35
36
        }
        else {
37
38
            t[u].len = t[lc].len + t[rc].len;
39
            t[u].ls = t[lc].ls; t[u].rs = t[rc].rs;
            t[u].num = t[lc].num + t[rc].num - (t[lc].rs & t[rc].ls);
40
        }
41
   }
42
43
   void build(int u, int l, int r) {
44
        t[u].l = l; t[u].r = r;
45
        if(l == r) {
46
47
            t[u].len = t[u].cover = t[u].ls = t[u].rs = t[u].num = 0;
48
            return ;
49
        int m = (l + r) >> 1;
50
        build(lc, l, m);
52
        build(rc, m + 1, r);
53 }
54
   void modify(int u, int ql, int qr, int state) {
55
        if(ql <= t[u].l && t[u].r <= qr) {
56
57
            t[u].cover += state;
58
            push_up(u);
59
            return ;
60
        }
```

```
if(ql <= mid) modify(lc, ql, qr, state);</pre>
61
        if(qr > mid) modify(rc, ql, qr, state);
62
        push_up(u);
63
64 }
65
   void init() {
66
67
        cin >> n;
        int mx = -INF, mn = INF;
68
        for (int i = 1; i <= n; i++) {
69
            int x1, x2, y1, y2;
70
            cin >> x1 >> y1 >> x2 >> y2;
71
72
            mx = max(mx, max(x1, x2));
            mn = min(mn, min(x1, x2));
73
            line[i] = L{y1, x1, x2, 1};
74
            line[n + i] = L\{y2, x1, x2, -1\};
75
        }
76
77
        n <<= 1;
        sort(line + 1, line + n + 1);
78
        build(1, mn, mx);
79
80 }
81
   void solve() {
82
        int ans = 0;
83
84
        int last = 0;
85
        for(int i = 1;i <= n; i++) {</pre>
            modify(1, line[i].x1, line[i].x2 - 1, line[i].state);
86
            ans += abs(t[1].len - last); // 横线
87
            ans += (line[i + 1].y - line[i].y) * 2 * t[1].num; // 竖线
88
            last = t[1].len;
89
90
        printf("%d\n",ans);
91
   }
92
93
   int main() {
94
95
        init();
        solve();
96
97
  }
          区间第 k 小
   3.42
   #include <bits/stdc++.h>
2
   using namespace std;
3
   const int N = 1e5 + 10;
5
   vector<int> v;
6
7
   struct Node {
8
        int l, r;
9
        int val;
   }hjt[N * 40];
   int root[N], cnt;
12
int get_id(int x) { return lower_bound(v.begin(), v.end(), x) - v.begin() + 1; }
14
   void insert(int pre, int &now, int l, int r, int p) {
15
16
        hjt[++cnt] = hjt[pre];
17
        now = cnt;
18
        hjt[now].val++;
```

```
if(l == r) return ;
19
       int m = (l + r) >> 1;
20
       if(p <= m) insert(hjt[pre].l, hjt[now].l, l, m, p);</pre>
21
22
       else insert(hjt[pre].r, hjt[now].r, m + 1, r, p);
23 }
24
25
   int query(int L, int R, int l, int r, int k) {
       if(l == r) return l;
26
27
       int m = (l + r) >> 1;
       int tmp = hjt[hjt[R].l].val - hjt[hjt[L].l].val;
28
29
       if(k <= tmp) return query(hjt[L].l, hjt[R].l, l, m, k);</pre>
30
       else return query(hjt[L].r, hjt[R].r, m + 1, r, k - tmp);
   }
31
32
33
   int main() {
       int n, q;
34
       cin >> n >> q;
35
36
       vector < int > a(n + 1);
       for(int i = 1;i <= n; i++) { cin >> a[i]; v.push_back(a[i]); }
37
       sort(v.begin(), v.end());
38
       v.erase(unique(v.begin(), v.end());
39
40
       for(int i = 1;i <= n; i++) {
41
42
            insert(root[i - 1], root[i], 1, n, get_id(a[i]));
43
       }
44
       while(q--) {
45
            int 1, r, k;
46
            cin >> 1 >> r >> k;
47
            cout << v[query(root[l - 1], root[r], 1, n, k) - 1] << endl;</pre>
48
49
       }
50 }
          区间前k大
   3.43
1
   #include <bits/stdc++.h>
   using namespace std;
5 typedef long long ll;
6
   const int N = 1e6 + 10;
7
   int a[N];
   vector<int> v;
   int cnt, root[N];
9
10
   struct Node {
11
12
       int 1, r;
13
       ll sum;
14
       int num;
15
       int val;
   }hjt[N * 40];
16
17
   int getid(int x) { return lower_bound(v.begin(), v.end(), x) - v.begin() + 1; }
18
19
   void insert(int pre, int &now, int l, int r, int p, int val) {
20
21
       now = ++cnt;
       hjt[now] = hjt[pre];
22
       hjt[now].num++; hjt[now].sum += val;
23
```

```
24
        if(l == r) {
            hjt[now].val = val;
25
26
            return ;
        }
27
28
        int m = (l + r) >> 1;
        if(p <= m) insert(hjt[pre].l, hjt[now].l, l, m, p, val);</pre>
29
30
        else insert(hjt[pre].r, hjt[now].r, m + 1, r, p, val);
   }
31
32
   11 query(int L, int R, int l, int r, int k) {
33
34
        if(l == r) return hjt[R].val * k;
35
        int m = (l + r) >> 1;
        int tmp = hjt[hjt[R].r] .num - hjt[hjt[L].r].num;
36
        if(k <= tmp) return query(hjt[L].r, hjt[R].r, m + 1, r, k);</pre>
37
        else return hjt[hjt[R].r].sum - hjt[hjt[L].r].sum + query(hjt[L].l, hjt[R].l, l, m,
38
        k - tmp);
   }
39
40
   void init(int n) {
41
        v.clear();
42
43
        cnt = 0;
        for(int i = 1;i <= n; i++) {</pre>
44
            scanf("%d",&a[i]);
45
            v.push\_back(a[i]); root[i] = 0;
46
47
        }
        sort(v.begin(), v.end());
48
        v.erase(unique(v.begin(), v.end());
49
   }
50
51
   int main() {
52
       int _;
scanf("%d",&_);
53
54
55
        while(_--) {
            int n;
56
            scanf("%d",&n);
57
            init(n);
58
            for(int i = 1; i <= n; i++) {
59
60
                insert(root[i - 1], root[i], 1, n, getid(a[i]), a[i]);
            }
61
            int q;
62
            scanf("%d",&q);
63
            while(q--) {
64
                int l, r, k;
scanf("%d%d%d",&l,&r,&k);
65
66
                int t = r - l + 1;
67
68
                ll ans = query(root[l - 1], root[r], 1, n, k);
69
                printf("%lld\n",1ll * t * (t + 1) * (2 * t + 1) / 6 + ans);
            }
70
        }
71
72 }
          树套树维护三维偏序
1
   const int N = 4e6 + 10;
2
3 int n, k;
4
   struct node {
        int a, b, c;
```

```
int operator < (const node &o) const {</pre>
6
7
            return a != o.a ? (a < o.a) : (b != o.b ? (b < o.b) : (c < o.c));
8
9
10
       int operator == (const node &o) const {
            return a == o.a && b == o.b && c == o.c;
11
12
   }p[N];
13
14
15
   struct Tree1 {
16
       int 1, r;
17
18
   }t1[N << 2];
19
   struct Tree2 {
20
       int 1, r;
21
22
       int num;
23
   }t2[N << 2];
24
25 int root, root2[N];
   int cnt1, cnt2;
26
27
   void vec_insert(int &u, int l, int r, int pos, int val) {
28
29
       if(!u) u = ++cnt2;
30
       t2[u].num += val;
       if(l == r) return
31
       int m = (1 + r) / 2;
32
       if(pos <= m) vec_insert(t2[u].1, 1, m, pos, val);</pre>
33
       else vec_insert(t2[u].r, m + 1, r, pos, val);
34
   }
35
36
   int vec_query(int u, int l, int r, int ql, int qr) {
37
38
       if(!u) return 0;
       if(ql <= l && r <= qr) return t2[u].num;</pre>
39
       int ans = 0;
40
       int mid = (l + r) / 2;
41
42
       if(ql \le mid) ans += vec_query(t2[u].l, l, mid, ql, qr);
43
       if(qr > mid) ans += vec_query(t2[u].r, mid + 1, r, ql, qr);
       return ans;
44
   }
45
46
   // 在第一维权值线段树在[1,k]根据p[x].b插入,第二维权值线段树在[1,k]根据p[x].c插入
47
   void tree_insert(int &u, int l, int r, int x, int val) {
49
       if(!u) u = ++cnt1;
       vec_insert(root2[u], 1, k, p[x].c, val);
50
       if(l == r) return ;
51
52
       int m = (l + r) / 2;
       if(p[x].b \ll m) tree_insert(t1[u].l, l, m, x, val);
53
       else tree_insert(t1[u].r, m + 1, r, x, val);
54
55 }
56
57
   int tree_query(int u, int l, int r, int x) {
58
       if(!u) return 0;
       if(1 \le 1 \& r \le p[x].b) return vec_query(root2[u], 1, k, 1, p[x].c);
59
       int mid = (1 + r) / 2;
60
61
       int ans = 0;
62
       if(1 <= mid) ans += tree_query(t1\lceil u \rceil.1, 1, mid, x);
63
       if(p[x].b > mid) ans += tree_query(t1[u].r, mid + 1, r, x);
64
       return ans;
```

```
65 }
66
   void solve() {
67
68
       cin >> n >> k;
       for(int i = 1;i <= n; i++) cin >> p[i].a >> p[i].b >> p[i].c;
69
       sort(p + 1, p + n + 1);
70
       vector < int > ans(n + 1);
71
       int sum = 1;
72
       for(int i = 1;i <= n; i++) {</pre>
73
            // 因为这些个都一样,如果不这样操作,会使后面的不会对前面的有贡献
74
75
            if(p[i + 1] == p[i]) {
                sum++;
76
                continue;
77
            }
78
            tree_insert(root, 1, k, i, sum);
79
            int res = tree_query(root, 1, k, i);
80
81
           ans[res] += sum;
82
            sum = 1;
83
       for(int i = 1;i <= n; i++) cout << ans[i] << endl;</pre>
84
  }
85
          线段树套主席树-二维区间不同数
   3.45
   const int N = 2e5 + 10;
1
2
3
   int n, m, l, r, a, b, num[N], Last[N], pre[N], Hash[N], ans[N][2];
   int cnt, root[N], sum[N * 20], lc[N * 20], rc[N * 20];
4
5
   struct Query {
       int a, b, 1, id;
6
7
   };
8
9
   struct data {
10
       int a, v;
11
       bool operator<(const data &b) const {</pre>
12
13
            return a < b.a;
14
   } d[N];
15
16
17
   vector<Query> q[N * 4];
18
19
   void addquery(int o, int l, int r, int L, int R, Query qry) {
       if (L \le 1 \&\& R \ge r) 
20
21
            q[o].push_back(qry);
            return;
22
23
       int mid = (l + r) / 2;
24
       if (L <= mid) {</pre>
25
            addquery(o * 2, 1, mid, L, R, qry);
26
27
       if (R > mid) {
28
29
           addquery(o * 2 + 1, mid + 1, r, L, R, qry);
30
       }
   }
31
32
   void build(int y, int &x, int l, int r, int k) {
33
34
       x = ++cnt;
```

```
35
        sum[x] = sum[y] + 1;
        lc[x] = lc[y];
36
        rc[x] = rc[y];
37
        if (l == r) {
38
39
            return;
40
        int mid = (l + r) / 2;
41
        if (k <= mid) {
42
            build(lc[y], lc[x], l, mid, k);
43
        } else {
44
            build(rc[y], rc[x], mid + 1, r, k);
45
46
47
   }
48
   int query(int y, int x, int l, int r, int k) {
49
        if (!x | 1 | 1 == r) {
50
51
            return 0;
52
        int mid = (l + r) / 2;
53
        if (k <= mid) {
54
            return query(lc[y], lc[x], l, mid, k);
55
        } else {
56
            return sum[lc[x]] - sum[lc[y]] + query(rc[y], rc[x], mid + 1, r, k);
57
58
        }
59
   }
60
   void insert(int o, int l, int r) {
61
        if (q[o].size()) {
62
            Hash[0] = 0;
63
            for (int i = 1; i <= r; i++) {
64
                Hash[++Hash[0]] = num[i];
65
66
            sort(Hash + 1, Hash + Hash[0] + 1);
67
            int s = 0;
68
            for (int i = l; i <= r; i++) {
69
                d[++s].a = lower_bound(Hash + 1, Hash + Hash[0] + 1, num[i]) - Hash;
70
71
                d[s].v = pre[i];
72
            }
            sort(d + 1, d + s + 1);
73
            for (int i = 1; i <= s; i++) {
74
                build(root[i - 1], root[i], 0, n, d[i].v);
75
76
            int a, b;
77
            for (int i = 0; i < q[o].size(); i++) {</pre>
78
                a = lower_bound(Hash + 1, Hash + Hash[0] + 1, q[o][i].a) - Hash;
79
                b = upper\_bound(Hash + 1, Hash + Hash[0] + 1, q[o][i].b) - Hash - 1;
80
                ans[q[o][i].id][0] += sum[root[b]] - sum[root[a - 1]];
81
                ans[q[o][i].id][1] += query(root[a - 1], root[b], 0, n, q[o][i].l);
82
83
            memset(root, 0, sizeof(int) * (Hash[0] + 1));
84
            memset(sum, 0, sizeof(int) * (cnt + 1));
85
            memset(lc, 0, sizeof(int) * (cnt + 1));
86
            memset(rc, 0, sizeof(int) * (cnt + 1));
87
            cnt = 0;
88
89
        if (l == r) {
90
91
            return;
92
93
        int mid = (l + r) / 2;
```

```
insert(o * 2, 1, mid);
94
         insert(0 * 2 + 1, mid + 1, r);
95
    }
96
97
    void solve() {
98
         int _;
99
         cin >> _;
100
         while (_--) {
101
             for (int i = 0; i < N * 4; i++) q[i].clear();</pre>
102
103
             for (int i = 0; i < N; i++) ans[i][0] = ans[i][1] = pre[i] = Last[i] = 0;
104
105
             cin >> n >> m;
             for (int i = 1; i <= n; i++) {
106
                 cin >> num[i];
107
                 num[i]++;
108
                 pre[i] = Last[num[i]];
109
110
                 Last[num[i]] = i;
             }
111
112
             for (int i = 1; i <= m; i++) {
                 cin >> 1 >> a >> r >> b;
113
                 a++, b++;
114
                 addquery(1, 1, 1e5 + 1, l, r, (Query) \{a, b, l, i\});
115
116
             insert(1, 1, 1e5 + 1);
117
118
             for (int i = 1; i <= m; i++) {
                 cout << ans[i][1] << endl;</pre>
119
120
             }
         }
121
    }
122
    3.46
           Scapegoat
 1
    // 无旋转平衡,暴力拍扁重构
 2
 3 #include <bits/stdc++.h>
 4 using namespace std;
 5
 6
   namespace Scapegoat_Tree {
 7
    #define MAXN (100000 + 10)
 8
         const double alpha = 0.75;
 9
         struct Node {
         Node * ch[2];
10
         int key, size, cover; // size为有效节点的数量, cover为节点总数量
bool exist; // 是否存在(即是否被删除)
11
12
13
         void PushUp(void) {
             size = ch[0]->size + ch[1]->size + (int)exist;
14
             cover = ch[0] -> cover + ch[1] -> cover + 1;
15
16
         bool isBad(void) { // 判断是否需要重构
17
             return ((ch[0]->cover > cover * alpha + 5) ||
18
                      (ch[1]->cover > cover * alpha + 5));
19
20
21
         };
22
         struct STree {
23
         protected:
             Node mem_poor[MAXN]; //内存池,直接分配好避免动态分配内存占用时间
Node *tail, *root, *null; // 用null表示NULL的指针更方便, tail为内存分配指针, root为根
24
25
             Node *bc[MAXN]; int bc_top; // 储存被删除的节点的内存地址, 分配时可以再利用这些地址
26
```

```
27
            Node * NewNode(int key) {
28
                Node * p = bc_{top}? bc[--bc_{top}] : tail++;
29
                p \rightarrow ch[0] = p \rightarrow ch[1] = null;
30
31
                p->size = p->cover = 1; p->exist = true;
32
                p->key = key;
                return p;
33
34
            }
            void Travel(Node * p, vector<Node *>&v) {
35
                if (p == null) return;
36
                Travel(p->ch[0], v);
37
38
                if (p->exist) v.push_back(p); // 构建序列
39
                else bc[bc_top++] = p; // 回收
                Travel(p->ch[1], v);
40
41
            Node * Divide(vector<Node *>&v, int l, int r) {
42
43
                if (l >= r) return null;
                int mid = (l + r) >> 1;
44
                Node * p = v[mid];
45
                p->ch[0] = Divide(v, l, mid);
46
                p->ch[1] = Divide(v, mid + 1, r);
47
                p->PushUp(); // 自底向上维护, 先维护子树
48
49
                return p;
50
51
            void Rebuild(Node * &p) {
                static vector<Node *>v; v.clear();
52
                Travel(p, v); p = Divide(v, 0, v.size());
53
54
            Node ** Insert(Node *&p, int val) {
55
                if (p == null) {
56
                     p = NewNode(val);
57
                    return &null;
58
59
                else {
60
                     p->size++; p->cover++;
61
62
                    // 返回值储存需要重构的位置,若子树也需要重构,本节点开始也需要重构,以本节点为根重构
63
64
                     Node ** res = Insert(p->ch[val >= p->key], val);
                     if (p->isBad()) res = &p;
65
                     return res;
66
                }
67
68
            void Erase(Node *p, int id) {
69
70
                p->size--;
                int offset = p->ch[0]->size + p->exist;
71
                if (p->exist && id == offset) {
72
                     p->exist = false;
73
                     return;
74
                }
75
76
                else {
77
                     if (id <= offset) Erase(p->ch[0], id);
78
                     else Erase(p->ch[1], id - offset);
79
                }
            }
80
        public:
81
            void Init(void) {
82
83
                tail = mem_poor;
84
                null = tail++;
                null \rightarrow ch[0] = null \rightarrow ch[1] = null;
85
```

```
null->cover = null->size = null->key = 0;
86
                  root = null; bc_top = 0;
87
88
             STree(void) { Init(); }
89
90
             void Insert(int val) {
91
                  Node ** p = Insert(root, val);
92
                  if (*p != null) Rebuild(*p);
93
             }
94
             int Rank(int val) {
95
                  Node * now = root;
96
97
                  int ans = 1;
                  while (now != null) { // 非递归求排名
98
                      if (now->key >= val) now = now->ch[0];
99
100
                      else {
                          ans += now->ch[0]->size + now->exist;
101
102
                          now = now -> ch[1];
                      }
103
104
                  }
105
                  return ans;
106
             int Kth(int k) {
107
                  Node * now = root;
108
                  while (now != null) { // 非递归求第K大
109
110
                      if (now->ch[0]->size + 1 == k && now->exist) return now->key;
                      else if (now->ch[0]->size >= k) now = now->ch[0];
111
                      else k \rightarrow now \rightarrow ch[0] \rightarrow size + now \rightarrow exist, now = now \rightarrow ch[1];
112
                  }
113
114
             void Erase(int k) {
115
                  Erase(root, Rank(k));
116
                  if (root->size < alpha * root->cover) Rebuild(root);
117
118
             void Erase_kth(int k) {
119
                  Erase(root, k);
120
                  if (root->size < alpha * root->cover) Rebuild(root);
121
122
123
         }sTree;
    #undef MAXN
124
125
126
    }
127
    int main() {
128
129
         Scapegoat_Tree::sTree.Init();
         int _; cin >> _;
130
         while(_--) {
131
             int opt, x;
132
             cin >> opt >> x;
133
             if(opt == 1) Scapegoat_Tree::sTree.Insert(x);
134
             else if(opt == 2) Scapegoat_Tree::sTree.Erase(x);
135
136
             else if(opt == 3) cout << Scapegoat_Tree::sTree.Rank(x) << endl;</pre>
137
             else if(opt == 4) cout << Scapegoat_Tree::sTree.Kth(x) << endl;</pre>
138
             else if(opt == 5) cout << Scapegoat_Tree::sTree.Kth(Scapegoat_Tree::sTree.Rank(</pre>
        x) - 1) << endl;
             else if(opt == 6) cout << Scapegoat_Tree::sTree.Kth(Scapegoat_Tree::sTree.Rank(</pre>
139
        x + 1) << endl;
140
141 }
```

#### 3.47 Splay

```
class node {
1
2
   public:
        int id;
3
        node *1;
4
        node *r;
5
        node *p;
6
7
        bool rev;
8
        int sz;
9
        // declare extra variables
10
        node(int _id) {
11
12
             id = _id;
13
             l = r = p = nullptr;
             rev = false;
14
             sz = 1;
15
             // init extra variables
16
        }
17
18
19
        void unsafe_reverse() {
20
             rev ^= 1;
21
             std::swap(l, r);
             pull();
22
23
        }
24
        // apply changes:
25
26
        void unsafe_apply() {
27
28
        void push() {
29
             if (rev) {
30
31
                 if (l != nullptr) {
32
                      1 -> unsafe_reverse();
33
                 if (r != nullptr) {
34
35
                      r -> unsafe_reverse();
                 }
36
                 rev = 0;
37
38
39
             // now push everything else:
        }
40
41
        void pull() {
42
43
             sz = 1;
             // now init from self:
44
45
             if (l != nullptr) {
46
                 l \rightarrow p = this;
47
                 sz += 1 -> sz;
48
                 // now pull from l:
49
50
51
             if (r != nullptr) {
52
53
                 r \rightarrow p = this;
                 SZ += r -> SZ;
54
                 // now pull from r:
55
56
             }
57
```

```
}
 58
 59
     };
 60
     void debug_node(node* v, std::string pref = "") {
 61
62
     #ifdef LOCAL
          if (v != nullptr) {
63
               debug_node(v -> r, pref + " ");
std::cerr << pref << "-" << " " << v -> id << '\n';
64
65
               debug_node(v -> l, pref + " ");
 66
          } else {
 67
               std::cerr << pref << "-" << "NULL" << '\n';
68
 69
     #endif
 70
     }
 71
 72
     namespace splay_tree {
 73
 74
     bool is_bst_root(node* v) {
 75
          if (v == nullptr) {
 76
              return false;
 77
 78
          return (v \rightarrow p == nullptr || (v \rightarrow p \rightarrow l != v \&\& v \rightarrow p \rightarrow r != v));
 79
    }
 80
 81
     void rotate(node* v) {
 82
          node* u = v \rightarrow p;
          assert(u != nullptr);
 83
          u -> push();
 84
          v -> push();
 85
          v -> p = u -> p;
 86
          if (v -> p != nullptr) {
 87
               if (v -> p -> l == u) {
 88
                   v -> p -> l = v;
 89
 90
               if (v -> p -> r == u) {
 91
 92
                   v -> p -> r = v;
 93
 94
 95
          if (v == u -> 1) {
              u \rightarrow 1 = v \rightarrow r;
96
              v \rightarrow r = u;
97
          } else {
98
              u -> r = v -> 1;
99
100
              v -> 1 = u;
101
          u -> pull();
102
103
          v -> pull();
104
    }
105
     void splay(node* v) {
106
107
          if (v == nullptr) {
108
               return ;
109
110
          while (!is_bst_root(v)) {
               node* u = v \rightarrow p;
111
               if (!is_bst_root(u)) {
112
                    if ((u \rightarrow l == v) \land (u \rightarrow p \rightarrow l == u)) {
113
114
                        rotate(v);
115
                   } else {
116
                        rotate(u);
```

```
}
117
             }
118
             rotate(v);
119
        }
120
    }
121
122
    std::pair<node*, int> find(node* v, const std::function<int(node*)> &go_to) {
123
        // go_to returns: 0 -- found; -1 -- go left; 1 -- go right
124
        // find returns the last vertex on the descent and its go_to
125
        if (v == nullptr) {
126
127
             return {nullptr, 0};
128
        }
        splay(v);
129
        int dir;
130
        while (true) {
131
             v -> push();
132
             dir = go_to(v);
133
             if (dir == 0) {
134
135
                 break;
136
             node* u = (dir == -1 ? v -> l : v -> r);
137
             if (u == nullptr) {
138
                 break;
139
140
             }
141
             v = u;
        }
142
        splay(v);
143
        return {v, dir};
144
    }
145
146
    node* get_leftmost(node* v) {
147
        return find(v, [&](node*) { return -1; }).first;
148
149
    }
150
    node* get_rightmost(node* v) {
151
        return find(v, [&](node*) { return 1; }).first;
152
153
154
    node* get_kth(node* v, int k) { // 0-indexed
155
        std::pair<node*, int> p = find(v, [&](node* u) {
156
             if (u -> l != nullptr) {
157
                 if (u -> 1 -> sz > k) {
158
159
                     return -1;
160
                 k -= u -> l -> sz;
161
162
             if (k == 0) {
163
                 return 0;
164
165
166
             k--;
167
             return 1;
168
        });
169
        return (p.second == 0 ? p.first : nullptr);
170 }
171
    int get_position(node* v) { // 0-indexed
172
173
         splay(v);
174
        return (v -> l != nullptr ? v -> l -> sz : 0);
175 }
```

```
176
    node* get_bst_root(node* v) {
177
         splay(v);
178
179
         return v;
    }
180
181
    std::pair<node*, node*> split(node* v, const std::function<bool(node*)> &is_right) {
182
         if (v == nullptr) {
183
              return {nullptr, nullptr};
184
         }
185
186
         std::pair< node*, int> p = find(v, [%](node* u) { return is_right(u) ? -1 : 1; });
187
         v = p.first;
         v -> push();
188
         if (p.second == -1) {
189
              node* u = v \rightarrow 1;
190
              if (u == nullptr) {
191
192
                  return {nullptr, v};
              }
193
             v -> l = nullptr;
194
             u -> p = v -> p;
195
             u = get_rightmost(u);
196
197
             v \rightarrow p = u;
             v -> pull();
198
199
              return {u, v};
200
         } else {
             node* u = v \rightarrow r;
201
              if (u == nullptr) {
202
                  return {v, nullptr};
203
              }
204
             v \rightarrow r = nullptr;
205
206
             v -> pull();
207
              return {v, u};
208
         }
    }
209
210
    std::pair<node*, node*> split_leftmost_k(node* v, int k) {
211
         return split(v, [&](node* u) {
213
              int left_and_me = (u \rightarrow l != nullptr ? u \rightarrow l \rightarrow sz : 0) + 1;
              if (k >= left_and_me) {
214
215
                  k -= left_and_me;
216
                  return false;
              }
217
218
              return true;
219
         });
220
    }
221
    node* merge(node* v, node* u) {
222
223
         if (v == nullptr) {
224
              return u;
225
         if (u == nullptr) {
226
227
              return v;
228
         v = get_rightmost(v);
229
230
         assert(v \rightarrow r == nullptr);
231
         splay(u);
232
         v -> push();
         v -> r = u;
233
234
         v -> pull();
```

```
235
         return v;
236 }
237
    int count_left(node* v, const std::function<bool(node*)> &is_right) {
238
239
         if (v == nullptr) {
              return 0;
240
241
         }
         std::pair< node*, int> p = find(v, [%](node* u) { return is_right(u) ? -1 : 1; });
242
         node* u = p.first;
243
         return (u -> l != nullptr ? u -> l -> sz : 0) + (p.second == 1);
244
245
    }
246
    node* add(node* r, node* v, const std::function<bool(node*)> &go_left) {
247
         std::pair<node*, node*> p = split(r, go_left);
248
249
         return merge(p.first, merge(v, p.second));
    }
250
251
    node* remove(node* v) { // returns the new root
252
253
         splay(v);
254
         v -> push();
         node* x = v \rightarrow 1;
255
         node* y = v \rightarrow r;
256
         v \rightarrow l = v \rightarrow r = nullptr;
257
258
         node* z = merge(x, y);
259
         if (z != nullptr) {
260
              z \rightarrow p = v \rightarrow p;
         }
261
262
         v \rightarrow p = nullptr;
         v -> push();
263
         v -> pull(); // now v might be reusable
264
265
         return z;
266 }
267
    node* next(node* v) {
268
         splay(v);
269
270
         v -> push();
271
         if (v \rightarrow r == nullptr) {
272
              return nullptr;
         }
273
         v = v \rightarrow r;
274
         while (v -> l != nullptr) {
275
276
             v -> push();
              v = v -> 1;
277
278
         }
279
         splay(v);
280
         return v;
281 }
282
283
    node* prev(node* v) {
284
         splay(v);
285
         v -> push();
286
         if (v -> l == nullptr) {
287
              return nullptr;
288
         }
289
         v = v -> 1;
         while (v -> r != nullptr) {
290
291
              v -> push();
              v = v \rightarrow r;
292
         }
293
```

```
294
         splay(v);
295
         return v;
296 }
297
298
    int get_size(node* v) {
299
         splay(v);
300
         return (v != nullptr ? v -> sz : 0);
301
    }
302
    template<typename... T>
303
304
    void apply(node* v, T... args) {
305
         splay(v);
         v -> unsafe_apply(args...);
306
307
    }
308
309 void reverse(node* v) {
310
         splay(v);
311
         v -> unsafe_reverse();
312
313
314 } // namespace splay_tree
    3.48 01trie
    template<typename T>
 2
    class Trie {
 3
    private:
         Trie* next[2] = {nullptr};
 4
 5
         int val;
         const int maxl = 32;
 6
 7
    public:
 8
         Trie() {}
 9
 10
         void insert(T x) {
             Trie* root = this;
 11
             for(int i = maxl; i >= 0; i--) {
 12
 13
                 int u = x >> i & 1;
                 if(root -> next[u] == nullptr) root -> next[u] = new Trie();
 14
 15
                 root = root -> next[u];
 16
                 root -> val ++;
 17
             }
         }
 18
 19
         void del(T x) {
 20
 21
             Trie* root = this;
             for (int i = maxl; i >= 0; i--) {
 22
 23
                 root = root -> next[x >> i & 1];
                 root -> val --;
 24
             }
 25
         }
 26
 27
         T search(T x) {
 28
 29
             T ans = 0;
             Trie* root = this;
 30
             for (int i = maxl; i >= 0; i--) {
 31
                 int u = x >> i \& 1;
 32
                 if (root -> next[!u] && root -> next[!u] -> val) {
 33
 34
                     ans += 1 << i;
```

```
35
                     root = root -> next[!u];
                } else {
36
                     root = root -> next[u];
37
38
39
            }
40
            return ans;
41
        }
42 };
   3.49
          Treap
   struct Tree {
2
        Tree *1;
3
        Tree *r;
        int x;
4
        int siz;
5
        Tree(Tree *v) { *this = *v; }
6
        Tree(int x = 0) : l(nullptr), r(nullptr), x(x), siz(1) {}
7
8
        void pull() {
9
            siz = 1;
            if (l != nullptr) {
10
                siz += l->siz;
11
12
            if (r != nullptr) {
13
                siz += r->siz;
14
15
            }
16
        }
17
   };
18
   int cnt = 0;
19
   constexpr int N = 1e7;
21
   Tree pool[N];
22
  std::mt19937 rnd(std::chrono::steady_clock::now().time_since_epoch().count());
23
24
25 template<class... T>
26
   Tree *newTree(T... x) {
        Tree *t = &pool[cnt++];
27
        *t = Tree(x...);
28
29
        return t;
   }
30
31
   Tree *merge(Tree *a, Tree *b) {
32
        if (a == nullptr) {
33
34
            return b;
35
        if (b == nullptr) {
36
            return a;
37
38
        Tree *t;
39
        if (int(rnd() % (a->siz + b->siz)) < a->siz) {
40
            t = newTree(a);
41
42
            t->r = merge(t->r, b);
43
        } else {
            t = newTree(b);
44
            t->l = merge(a, t->l);
45
46
47
        t->pull();
```

```
return t;
48
49 }
50
   std::pair<Tree *, Tree *> split(Tree *t, int k) {
   if (t == nullptr || k == 0) {
51
52
             return {nullptr, t};
53
54
        if (t->siz == k) {
55
             return {t, nullptr};
56
        }
57
        int szl = t->l == nullptr ? 0 : t->l->siz;
58
        Tree *u = newTree(t);
59
        if (k <= szl) {
60
            auto [a, b] = split(t->l, k);
61
            u->l=b;
62
            u->pull();
63
             return {a, u};
64
        } else {
65
            auto [a, b] = split(t->r, k-1-szl);
66
            u->r = a;
67
            u->pull();
68
69
             return {u, b};
70
        }
71 }
```

## 4 博弈论

#### 4.1 bash

```
1 template <typename T>
   bool bash(const T& a, const T& b) {
3
       return a \% (b + 1);
4 }
   4.2
       fibonacci
   template <typename T>
1
   bool Fibonacci(const T& x) {
       std::unordered_map<int, bool> was;
3
       std::vector<int> fib(51);
4
       fib[1] = 1; fib[2] = 2;
5
6
       for (int i = 3; i \le 50; i++) {
            fib[i] = fib[i - 1] + fib[i - 2];
7
            was[fib[i]] = true;
8
9
       return !was[fib[x]];
10
11 }
   4.3 nim
   template <typename T>
   bool nimGame(const std::vector<T>& stones) {
3
       int res = 0;
       for (int &x : stones) {
4
5
            res ^{\wedge}= x;
6
7
       return res;
8 }
   4.4 wythoff
   template <typename T>
   bool wythoff(T& a, T& b) {
       if (a > b) std::swap(a, b);
3
       T delta = b - a;
T res = delta * (1.0 + sqrt(5.0)) / 2;
4
5
       return !(res == a);
6
   }
7
8
   template <typename T>
9
   bool wythoff_exp(T& a, T& b, T& k) {
10
       k++;
11
       if (a > b) std::swap(a, b);
12
       T delta = (b - a) / k;
13
       T res1 = delta * (2 - k + sqrt(4.0 + k * k)) / 2;
14
       T res2 = delta * (2 + k + sqrt(4.0 + k * k)) / 2;
15
16
       return !(res1 == a && res2 == b);
17 }
```

#### 4.5 sgFunction

```
1 // SG函数
 2 #define N 1001
 3 //f[]:可以取走的石子个数
 4 //sg[]:0~n的SG函数值
 5 int f[N], sg[N], mex[N];
 6
    void getSG(int n) {
 7
 8
         int i, j;
 9
         memset(sg, 0, sizeof(sg));
10
         for (i = 1; i <= n; i++) {
             memset(mex, 0, sizeof(mex));
11
             for (j = 1; f[j] <= i; j++)
    mex[sg[i - f[j]]] = 1;
for (j = 0; j <= n; j++) { //求mes{}中未出现的最小的非负整数
    if (mex[j] == 0) {
12
13
14
15
                       sg[i] = j;
16
                       break;
17
                  }
18
             }
19
         }
20
21 }
```

# 5 树与森林

#### 5.1 forest

```
template <typename T>
   class forest : public graph<T> {
   public:
3
4
        using graph<T>::edges;
5
        using graph<T>::g;
6
        using graph<T>::n;
7
8
        forest(int _n) : graph<T>(_n) {}
9
10
        int add (int from, int to, T cost = 1) {
11
            assert(0 \le from \&\& from < n \&\& 0 \le to \&\& to < n);
12
            int id = (int) edges.size();
            assert(id < n - 1);
13
            g[from].push_back(id);
14
            g[to].push_back(id);
15
            edges.push_back({from, to, cost});
16
17
            return id;
18
        }
   };
19
   5.2 dfs-forest
   template <typename T>
   class dfs_forest : public forest<T> {
3
   public:
        using forest<T>::edges;
4
5
        using forest<T>::q;
6
        using forest<T>::n;
7
8
        std::vector<int> pv;
9
        std::vector<int> pe;
10
        std::vector<int> order;
        std::vector<int> pos;
11
12
        std::vector<int> end;
13
        std::vector<int> sz;
        std::vector<int> root;
14
15
        std::vector<int> depth;
16
        std::vector<T> dist;
17
        dfs_forest(int _n) : forest<T>(_n) {}
18
19
        void init() {
20
            pv = std::vector<int>(n, -1);
21
22
            pe = std::vector<int>(n, -1);
23
            order.clear();
            pos = std::vector<int>(n, -1);
24
            end = std::vector<int>(n, -1);
25
            sz = std::vector<int>(n, 0);
26
            root = std::vector<int>(n, -1);
27
28
            depth = std::vector<int>(n, -1);
29
            dist = std::vector<T>(n);
        }
30
31
        void clear() {
32
```

```
pv.clear();
33
            pe.clear();
34
            order.clear();
35
36
            pos.clear();
37
            end.clear();
            sz.clear();
38
39
            root.clear();
            depth.clear();
40
            dist.clear();
41
        }
42
43
   private:
44
        void do_dfs(int v) {
45
            pos[v] = (int) order.size();
46
47
            order.push_back(v);
            sz[v] = 1;
48
            for (int id : g[v]) {
49
50
                 if (id == pe[v]) {
51
                     continue;
52
                 auto &e = edges[id];
53
                 int to = e.from ^ e.to ^ v;
54
                 depth[to] = depth[v] + 1;
55
                 dist[to] = dist[v] + e.cost;
56
57
                 pv[to] = v;
                 pe[to] = id;
58
                 root[to] = (root[v] != -1 ? root[v] : to);
59
60
                 do_dfs(to);
                 sz[v] += sz[to];
61
62
            end[v] = (int) order.size() - 1;
63
        }
64
65
        void do_dfs_from(int v) {
66
            depth[v] = 0;
67
            dist[v] = T{};
68
69
            root[v] = v;
70
            pv[v] = pe[v] = -1;
            do_dfs(v);
71
72
        }
73
   public:
74
        void dfs(int v, bool clear_order = true) {
75
76
            if (pv.empty()) {
                 init();
77
78
            } else {
79
                 if (clear_order) {
                     order.clear();
80
                 }
81
82
83
            do_dfs_from(v);
84
        }
85
        void dfs_all() {
86
            init();
87
            for (int v = 0; v < n; v++) {
88
                 if (depth[v] == -1) {
89
90
                     do_dfs_from(v);
                 }
91
```

```
92
            assert((int) order.size() == n);
93
94
95 };
    5.3 lca-forest
   template <typename T>
1
   class lca_forest : public dfs_forest<T> {
3
   public:
4
        using dfs_forest<T>::edges;
        using dfs_forest<T>::g;
5
6
        using dfs_forest<T>::n;
7
        using dfs_forest<T>::pv;
        using dfs_forest<T>::pos;
8
9
        using dfs_forest<T>::end;
        using dfs_forest<T>::depth;
10
11
12
        int h;
13
        std::vector<std::vector<int>> pr;
14
        lca_forest(int _n) : dfs_forest<T>(_n) {}
15
16
        inline void build_lca() {
17
            assert(!pv.empty());
18
19
            int max_depth = 0;
20
            for (int i = 0; i < n; i++) {
21
                 max_depth = std::max(max_depth, depth[i]);
22
23
            h = 1;
24
            while ((1 << h) <= max_depth) {</pre>
25
                 h++;
26
            }
27
            pr.resize(n);
28
            for (int i = 0; i < n; i++) {
29
                 pr[i].resize(h);
30
                 pr[i][0] = pv[i];
31
32
            for (int j = 1; j < h; j++) {
33
                 for (int i = 0; i < n; i++) {
                     pr[i][j] = (pr[i][j - 1] == -1 ? -1 : pr[pr[i][j - 1]][j - 1]);
34
35
                 }
36
            }
37
38
        inline bool anc(int x, int y) {
39
            return (pos[x] \leftarrow pos[y] && end[y] \leftarrow end[x]);
40
        }
41
42
        inline int go_up(int x, int up) {
43
44
            assert(!pr.empty());
            up = std::min(up, (1 << h) - 1);
45
            for (int j = h - 1; j >= 0; j--) {
   if (up & (1 << j)) {
46
47
48
                     x = pr[x][j];
49
                     if (x == -1) {
50
                          break;
                     }
51
```

```
}
52
            }
53
54
            return x;
55
56
        inline int lca(int x, int y) {
57
            assert(!pr.empty());
58
            if (anc(x, y)) {
59
                return x;
60
61
            if (anc(y, x)) {
62
63
                return y;
64
            for (int j = h - 1; j >= 0; j--) {
65
                if (pr[x][j] != -1 && !anc(pr[x][j], y)) {
66
67
                    x = pr[x][j];
68
69
            }
70
            return pr[x][0];
        }
71
72
        inline int dist(int x, int y) {
73
            return depth[x] + depth[y] - depth[lca(x, y)] * 2;
74
75
        }
76
   };
   5.4 重链剖分
1
   const int maxn = 4e5 + 10;
2
3
   struct Edge {
4
        int v, next;
5
   e[maxn << 1];
6
   int head[maxn * 2], cnt;
7
8
9
   inline void add(int u, int v) {
10
        e[++cnt].v = v;
11
        e[cnt].next = head[u];
12
        head[u] = cnt;
   }
13
14
   int fa[maxn], dep[maxn], siz[maxn], son[maxn];
15
16
   void dfs1(int u, int par) {
17
        dep[u] = dep[fa[u] = par] + (siz[u] = 1);
18
        for(int i = head[u]; \sim i; i = e[i].next) {
19
            int v = e[i].v;
20
            if(v == par) continue;
21
            dfs1(v, u);
22
            siz[u] += siz[v];
23
            if(!son[u] || siz[v] > siz[son[u]])
24
25
                son[u] = v;
26
        }
   }
27
28
   int dfn[maxn], top[maxn], nodeof[maxn], tim;
29
30
```

```
void dfs2(int u, int topf) {
       nodeof[dfn[u] = ++tim] = u;
32
       top[u] = topf;
33
       if(!son[u]) return ;
34
35
       dfs2(son[u], topf);
       for(int i = head[u]; \sim i; i = e[i].next) {
36
37
            int v = e[i].v;
            if(v == fa[u] || v == son[u]) continue;
38
            dfs2(v, v);
39
       }
40
   }
41
42
   int w[maxn];
43
44
45 #define lc u << 1
   #define rc u << 1 | 1
   #define mid (t[u].l + t[u].r) / 2
   struct Tree {
       int 1, r, sum, tag;
49
50 }t[maxn << 2];
51 inline void push_up(int u);
52 inline void push_down(int u);
53 void build(int u, int l, int r);
54 void modify(int u, int ql, int qr, int v);
   int query(int u, int ql, int qr);
56
   void modify_chain(int x, int y, int val) {
57
       while(top[x] != top[y]) {
58
            if(dep[top[x]] < dep[top[y]]) swap(x, y);</pre>
59
            modify(1, dfn[top[x]], dfn[x], val);
60
61
            x = fa[top[x]];
62
63
       if(dep[x] > dep[y]) swap(x, y);
       modify(1, dfn[x], dfn[y], val);
64
   }
65
66
67
   int query_chain(int x, int y) {
68
       int ans = 0;
       while(top[x] != top[y]) {
69
70
            if(dep[top[x]] < dep[top[y]]) swap(x, y);</pre>
            ans += query(1, dfn[top[x]], dfn[x]);
71
           x = fa[top[x]];
72
73
       if(dep[x] > dep[y]) swap(x, y);
74
       ans += query(1, dfn[x], dfn[y]);
75
76
       return ans;
77
   }
78
79
   signed main() {
80
       memset(head, -1, sizeof(head));
81
       int n; cin >> n;
82
       for(int i = 1;i <= n; i++) cin >> w[i];
83
       for(int i = 1; i <= n - 1; i++) {
            int u, v; cin >> u >> v;
84
            add(u, v);
85
86
            add(v, u);
87
       dfs1(1, 0);
88
       dfs2(1, 1);
89
```

```
build(1, 1, n);
90
91
        int m; cin >> m;
        while(m--) {
92
93
             int opt; cin >> opt;
             if(opt == 1) {
94
95
                 int x, y, val; cin >> x >> y >> val;
                 modify_chain(x, y, val);
96
             }
97
             else if(opt == 2) {
98
                 int x, val; cin >> x >> val;
99
                 modify(1, dfn[x], dfn[x] + siz[x] - 1, val);
100
101
             }
             else if(opt == 3) {
102
                 int x, y; cin >> x >> y;
103
104
                 cout << query_chain(x, y) << endl;</pre>
105
             else if(opt == 4) {
106
107
                 int x; cin >> x;
                 cout << query(1, dfn[x], dfn[x] + siz[x] - 1) << endl;
108
             }
109
        }
110
111 }
    5.5 tree-diameter
    template <typename T>
 2
    std::vector<int> find_tree_diameter(const forest<T>& g, T& diameter) {
 3
        diameter = 0;
 4
        int st = 0, ed = 0;
 5
        std::vector<T> dis(g.n);
 6
        std::vector<int> pre(g.n);
 7
 8
        std::function<void(int, int)> dfs1 = [&](int u, int parent) {
 9
             if (dis[u] > dis[st]) st = u;
             for (int id : g.g[u]) {
10
                 auto& e = g.edges[id];
11
12
                 int to = e.from ^ e.to ^ u;
                 if (to == parent) continue ;
13
14
                 dis[to] = dis[u] + e.cost;
15
                 dfs1(to, u);
16
             }
17
        };
18
        std::function<void(int, int)> dfs2 = [&](int u, int parent) {
19
20
             if (dis[u] > dis[ed]) ed = u;
             for (int id : g.g[u]) {
21
22
                 auto& e = g.edges[id];
                 int to = e.from ^ e.to ^ u;
23
                 if (to == parent) continue ;
24
                 pre[to] = u;
25
                 dis[to] = dis[u] + e.cost;
26
                 dfs2(to, u);
27
28
             }
        };
29
30
31
        dfs1(0, -1);
32
        dis.assign(g.n, 0);
        dfs2(st, -1);
33
```

```
std::vector<int> vertexs{ed};
34
       int now = ed;
35
       do {
36
            vertexs.push_back(pre[now]);
37
            now = pre[now];
38
       } while (now != st);
39
       reverse(vertexs.begin(), vertexs.end());
40
       return vertexs;
41
   }
42
43
   template <typename T>
44
   std::vector<int> find_tree_all_diameters(const forest<T>& g, T& diameter) {
45
       diameter = 0;
46
       std::vector<std::array<T, 2>> dp(g.n);
47
       std::vector<int> down(g.n);
48
       std::vector<T> up(g.n);
49
50
       std::function<void(int, int)> dfs1 = [&](int u, int parent) {
51
            for (int id : g.g[u]) {
52
                auto& e = g.edges[id];
53
                int to = e.from ^ e.to ^ u;
54
                if (to == parent) continue;
55
                dfs1(to, u);
56
                if (dp[to][0] + e.cost > dp[u][0]) {
57
58
                    dp[u][1] = dp[u][0];
                    dp[u][0] = dp[to][0] + e.cost;
59
                    down[u] = to;
60
                } else if (dp[to][0] + e.cost > dp[u][1]) {
61
                    dp[u][1] = dp[to][0] + e.cost;
62
63
                diameter = std::max(diameter, dp[u][1] + dp[u][0]);
64
            }
65
66
       };
67
       std::function<void(int, int)> dfs2 = [&](int u, int parent) {
68
            for (int id : g.g[u]) {
69
70
                auto& e = q.edges[id];
71
                int to = e.from ^ e.to ^ u;
                if (to == parent) continue;
72
73
                up[to] = up[u] + e.cost;
74
                if (down[u] == to) up[to] = std::max(up[to], dp[u][1] + e.cost);
                else up[to] = std::max(up[to], dp[u][0] + e.cost);
75
                dfs2(to, u);
76
77
            }
       };
78
79
80
       dfs1(0, -1);
       dfs2(0, -1);
81
       std::vector<int> vertexs;
82
83
       for (int i = 0; i < q.n; i++) {
84
            std::vector<int> dis{ dp[i][1], dp[i][0], up[i] };
85
            sort(dis.begin(), dis.end());
            if (dis[1] + dis[2] == diameter) vertexs.push_back(i);
86
87
       return vertexs;
88
89
   };
```

#### 5.6 树的重心

```
const int N = 1e5 + 10;
1
2
3
   struct Edge {
        int v, next;
5
   }e[N * 2];
6
   int cnt, head[N * 2];
7
8
   int d[N], R[2], root;
9
10
   int n;
11
12
  int balance;
13
   inline void add(int u, int v) {
14
        e[++cnt].v = v;
15
        e[cnt].next = head[u];
16
17
        head[u] = cnt;
18 }
19
   void DFS(int u, int fa) {
20
        d[u] = 1;
21
22
        int res = 0;
        for(int i = head[u] ; i != -1 ; i = e[i].next) {
23
             int v = e[i].v;
24
25
            if(v == fa) continue;
            DFS(v, u);
26
27
            d[u] += d[v];
28
            res = max(res, d[v]);
        }
29
        res = max(res, n - d[u]);
30
31
        if(res < balance) {</pre>
32
            R[root++] = u;
            balance = res;
33
34
        else if(res == balance) {
35
            R[root++] = u;
36
        }
37
   }
38
39
40
   int main() {
        cin >> n;
41
42
        balance = n / 2;
43
        for(int i = 1; i < n; i++) {
44
            int u, v;
            cin >> u >> v;
45
            add(u, v);
46
            add(v, u);
47
48
        DFS(1, 0);
49
        if(R[0]) cout << R[0] << endl;</pre>
50
        if(R[1]) cout << R[1] << endl;</pre>
51
52 }
```

### 5.7 树的最大匹配

1 // 设状态为f[u][1/0]表示以u为根的子树与儿子连边/不连边的最大匹配

```
const int N = 1e5 + 10;
3
   vector<int> g[N];
   int f[N][2];
5
6
   void dfs(int u, int fa) {
7
       int mn = INF;
8
9
       for(auto v : g[u]) {
           if(v == fa) continue ;
10
           dfs(v, u);
11
12
           f[u][0] += f[v][1]; // u不与儿子连边,即加上所有与儿子连边的v
13
           f[u][1] += f[v][1]; // u与儿子连边,即加上一个不与儿子连边的v和其他所有与儿子连边的v
           mn = min(mx, f[v][1] - f[v][0]);
14
15
       if(mn != INF) f[u][1] = dp[u][1] - mx + 1;
16
17
        树分治-点分治
1 // 题意: n个节点的树, 存在边权, 范围1e18
  // 求任意两点之间点集的子集中两点之间路径异或和为0的个数
  // u<v,u'<v',(u',v') [ path(u,v),求path(u', v')异或和==0
4
   struct Edge {
5
       int to, nxt;
6
       11 w;
7
8
   };
9
   const int N = int(1e5 + 10);
10
   const int M = N \ll 1;
11
12
   struct Grahp {
13
       int head[N];
14
       Edge eg[M];
15
       int tot;
16
       void init(int n) {
17
18
           memset(head, -1, sizeof(int) * ++n);
19
20
       inline void addEdge(int u, int v, ll w) {
21
           eg[tot] = \{v, head[u], w\};
22
           head[u] = tot++;
23
24
   } gh;
25
26
   bool vis[N];
27
   // q队列,fa祖先,sz是子树大小,smx是子树最大
29
   int q[N], fa[N], sz[N], smx[N];
30
   int froot(int s) {
31
32
       int l, r, mn = N, rt = 0;
       q[l = r = 1] = s;
33
34
       while (l \ll r) {
35
           int u = q[l++];
           sz[u] = 1;
36
           smx[u] = 0;
37
           for (int i = gh.head[u]; \sim i; i = gh.eg[i].nxt) {
38
39
               int v = gh.eg[i].to;
```

```
if (v == fa[u] || vis[v]) continue;
40
                fa[v] = u;
41
                q[++r] = v;
42
            }
43
44
       // 反向遍历所有点算size
45
       while (--1) {
46
            int u = q[1];
47
            int mx = max(smx[u], r - sz[u]);
48
            if (mx < mn) mn = mx, rt = u;
49
50
            if (l == 1) break; // 根节点没有fa
51
            sz[fa[u]] += sz[u];
            smx[fa[u]] = max(smx[fa[u]], sz[u]);
52
53
       }
54
       return rt;
   }
55
56
57
  // sons子树方向节点个数,val根到该节点异或和,gc边后继方向的节点个数
58 int sons[N], gc[M];
59 ll val[N];
60 11 ans = 0;
61 int n;
62
63 const int MOD = int(1e9 + 7);
64
   ll nums[N];
65
   int cnt[N];
66
67
   void go(int s, int rt) {
68
       fa[s] = rt;
69
       val[s] = 0;
70
71
       int l, r;
72
       // 不计算s
       q[l = r = 0] = s;
73
       int m = 0;
74
75
       while (l \ll r) {
76
            int u = q[l++];
            nums[m++] = val[u];
77
            for (int i = gh.head[u]; \sim i; i = gh.eg[i].nxt) {
78
79
                int v = gh.eg[i].to;
                if (v == fa[u] || vis[v]) continue;
80
                fa[v] = u;
81
                q[++r] = v;
82
83
                val[v] = val[u] \wedge gh.eg[i].w;
84
                // 这个点方向后面有多少点
85
                sons[v] = gc[i];
86
            }
87
       sort(nums, nums + m);
88
89
       m = unique(nums, nums + m) - nums;
90
       mst(cnt, 0, m);
91
       // 遍历分支
92
       for (int j = gh.head[s]; ~j; j = gh.eg[j].nxt) {
            // 分支的根
93
            int du = gh.eg[j].to;
94
            if (vis[du]) continue;
95
            q[l = r = 1] = du;
96
            while (l <= r) {</pre>
97
                int u = q[l++];
98
```

```
int k = lower_bound(nums, nums + m, val[u]) - nums;
99
                 (ans += 111 * sons[u] * cnt[k] % MOD) %= MOD;
100
                 if (val[u] == 0) {
    (ans += 1ll * sons[u] * (n - gc[j]) % MOD) %= MOD;
101
102
103
                 for (int i = gh.head[u]; \sim i; i = gh.eg[i].nxt) {
104
105
                      int v = gh.eg[i].to;
                      if (v == fa[u] || vis[v]) continue;
106
                      q[++r] = v;
107
                 }
108
109
             }
110
             // 增加这个方向的值
             while (--1) {
111
                 int u = q[1];
112
                 int k = lower_bound(nums, nums + m, val[u]) - nums;
113
                 (cnt[k] += sons[u]) \% = MOD;
114
             }
115
         }
116
117
    }
118
    void work(int u) {
119
         // 换根
120
121
         u = froot(u);
122
         vis[u] = true;
123
         go(u, 0);
         for (int i = gh.head[u]; \sim i; i = gh.eg[i].nxt) {
124
             int v = gh.eg[i].to;
125
             if (vis[v]) continue;
126
             work(v);
127
         }
128
    }
129
130
131
    // 预处理边后继节点个数
    int pdfs(int u, int f) {
132
         int fg_id = -1;
133
134
         int s = 1;
135
         for (int i = gh.head[u]; \sim i; i = gh.eg[i].nxt) {
136
             int v = gh.eg[i].to;
             if (v == f) { // 记录父边ID
137
                 fg_id = i;
138
139
                 continue;
140
             int c = pdfs(v, u);
141
142
             gc[i] = c;
143
             S += C;
144
         }
         // 存在父边
145
         if (\sim fg_id) gc[fg_id] = n - s;
146
         return s;
147
148 }
149
150
    void solve() {
151
         while (cin >> n) {
152
             gh.init(n);
             for (int i = 2; i <= n; i++) {
153
154
                 int u, v;
                 11 w;
155
                 u = i;
156
157
                 cin >> v >> w;
```

```
158
                 gh.addEdge(u, v, w);
                 gh.addEdge(v, u, w);
159
160
             mst(vis, false, n + 1);
161
162
             pdfs(1, 0);
             ans = 0;
163
             work(1);
164
             cout << ans << endl;
165
166
        }
167 }
    5.9 树上 dsu-维护路径信息
    const int N = 1e5 + 10;
 2
    vector<int> g[N];
 3
    int siz[N], dep[N], son[N], dfn[N], nodeof[N], tim;
 5
    void calc(int u, int w) {
        // ....对u这一节点进行单独处理
if(w > 0) // ....计算贡献
 7
 8
        else // ....撤销影响
 9
    }
10
11
    void dfs1(int u, int fa) {
12
13
        dep[u] = dep[fa] + (siz[u] = 1);
14
        nodeof[dfn[u] = ++tim] = u;
        for(auto v : g[u]) {
15
             if(v == fa) continue ;
16
             dfs1(v, u);
17
             siz[u] += siz[v];
18
             if(!son[u] || siz[v] > siz[son[u]]) son[u] = v;
19
20
        }
21
    }
22
23
    void dfs2(int u, int fa, bool keep) {
24
        for(auto v : g[u]) {
25
             if(v == fa || v == son[u]) continue ;
26
             dfs2(v, u, 0);
27
        if(son[u]) {
28
             dfs2(son[u], u, 1);
29
30
         for(auto v : g[u]) {
31
             if(v == fa || v == son[u]) continue;
32
             for(int j = 0; j < siz[v]; j++) {</pre>
33
                 // ....更新答案
34
35
             for(int j = 0; j < siz[v]; j++) {
36
                 calc(nodeof[dfn[v] + j], 1);
37
38
39
        }
40
        calc(u, 1);
        // ....更新答案
41
        if(!keep) {
42
             for(int i = 0; i < siz[u]; i++) calc(nodeof[dfn[u] + i], -1);
43
44
45
    }
```

```
46
   int main() {
47
       int n; cin >> n;
48
49
        for(int i = 1; i < n; i++) {
50
            int u, v;
            g[u].push_back(v);
51
            g[v].push_back(u);
52
53
       dfs1(1, 0);
54
       dfs2(1, 0, 0);
55
  }
56
   5.10 树上 dsu-维护子树信息
   const int N = 2e5 + 10;
2
   vector<int> g[N];
3
   int siz[N], son[N], col[N];
5
   int ans[N], cnt[N];
6
   bool vis[N];
7
   int maxx, sum;
   // maxx为每棵子树里出现最多的颜色, sum为编号和
10
11
12
   void calc(int u, int fa, int val) {
13
14
       针对不同问题, 采取的操作
15
       else if(val > 0 && cnt[col[u]] == maxx) sum += col[u];
16
17
       for(auto v : g[u]) {
18
            if(v != fa && !vis[v]) calc(v, u, w);
19
       }
20 }
21
   void dfs1(int u, int fa) {
22
23
       siz[u] = 1;
24
        for(auto v : g[u]) {
25
            if(v == fa) continue ;
26
            dfs1(v, u);
27
            siz[u] += siz[v];
            if(!son[u] || siz[v] > siz[son[u]]) son[u] = v;
28
29
       }
   }
30
31
   void dfs2(int u, int fa, bool keep) {
32
       for(auto v : g[u]) {
33
            if(v != fa && v != son[u]) {
34
                dfs2(v, u, 0);
35
36
37
       if(son[u]) {
38
            dfs2(son[u], u, 1);
39
           vis[son[u]] = 1;
40
41
       calc(u, fa, 1);
42
43
       ans[u] = sum;
       if(son[u]) vis[son[u]] = 0;
44
```

```
if(!keep) {
45
            calc(u, fa, -1);
46
47
            maxx = sum = 0;
        }
48
   }
49
50
   int main() {
51
        int n; cin >> n;
52
53
        for(int i = 1;i <= n; i++) cin >> col[i];
54
        for(int i = 1;i < n; i++) {
55
            int u, v; cin >> u >> v;
56
            g[u].push_back(v);
57
            g[v].push_back(u);
        }
58
        dfs0(1, 0);
59
        dfs1(1, 0, false);
60
        for(int i = 1;i <= n; i++) cout << ans[i] << endl;</pre>
61
62
   }
          树上 K 祖先
   5.11
1 //倍增KFA,空间大点,但是好写
  vector<int> g[N];
3
   int anc[N][20];
4
   void dfs(int u, int fa) {
5
6
        anc[u][0] = fa;
        for (int i = 1; i \le 19; i++) anc[u][i] = anc[anc[u][i - 1]][i - 1];
7
        for (auto &v: g[u])
8
            if (v != fa) dfs(v, u);
9
10
   }
11
12
   int kthFa(int u, int k) {
13
        int bit = 0;
        while (k) {
14
            if (k \& 1) u = anc[u][bit];
15
16
            k >>= 1;
17
            bit++;
18
        }
19
        return u;
20
   }
21
22
23 //树剖KFA
   int siz[N], son[N], dep[N], fa[N], top[N];
   int id[N], nodeOf[N], cnt;
   void dfs(int u, int par) {
26
        dep[u] = dep[fa[u] = par] + (siz[u] = 1);
27
        for (auto &v: g[u])
28
29
            if (v != par) {
30
                dfs(v, u);
                siz[u] += siz[v];
31
32
                if (!son[u] || siz[v] > siz[son[u]])
33
                    son[u] = v;
            }
34
35
   }
37 void dfs2(int u, int topf) {
```

```
nodeOf[id[u] = ++cnt] = u, top[u] = topf;
38
       if (!son[u]) return;
39
       dfs2(son[u], topf);
40
       for (auto &v: g[u])
41
42
           if (v != fa[u] \&\& v != son[u]) dfs2(v, v);
   }
43
44
   int kthFa(int u, int k) {
45
       while (k >= id[u] - id[top[u]] + 1 && u) {
46
           k = id[u] - id[top[u]] + 1;
47
           u = fa[top[u]];
48
49
       return nodeOf[id[u] - k];
50
  }
51
   5.12 virtualTree
1 //虚树可以处理多次询问,并且每次询问只需要树上的K个关键点
  //建立的虚树能保证点数 < 2 * K
3 //如果对虚树做dp,总体复杂度和∑K有关
4 //考虑dp的时候,需要同时考虑非关键点对答案的影响
5
  int n;
6
7
   struct edge {
8
9
       int nxt, to;
10 } e[N << 1];
  int head[N], tot;
   void add(int u, int v) { e[++tot] = edge\{ head[u], v \}, head[u] = tot; }
  int dep[N], fa[N], topfa[N], siz[N], son[N], dfn[N], cnt;
15
   void dfs(int u, int par) {
16
       dep[u] = dep[fa[u] = par] + (siz[u] = 1);
       int max_son = -1;
17
       for (int i = head[u], v; i; i = e[i].nxt)
18
           if ((v = e[i].to) != par) {
19
20
               dfs(v, u);
21
               siz[u] += siz[v];
22
               if (max_son < siz[v]) son[u] = v, max_son = siz[v];</pre>
23
           }
24
   }
   void dfs2(int u, int topf) {
25
       topfa[u] = topf, dfn[u] = ++cnt;
26
       if (!son[u]) return;
27
28
       dfs2(son[u], topf);
       for (int i = head[u], v; i; i = e[i].nxt)
29
           if ((v = e[i].to) != fa[u] \&\& v != son[u]) dfs2(v, v);
30
31
   int LCA(int x, int y) {
32
       while (topfa[x] != topfa[y]) {
33
34
           if (dep[topfa[x]] < dep[topfa[y]]) swap(x, y);</pre>
           x = fa[topfa[x]];
35
36
37
       return dep[x] < dep[y] ? x : y;</pre>
38
   int getDis(int x, int y) { return dep[x] + dep[y] - 2 * dep[LCA(x, y)]; }
39
40
41 //建立虚树
```

```
42 int tag[N];//tag[u] = 1 <=> 关键点
    vector<int> g[N];//虚树边
44 void add_edge(int u, int v) { g[u].push_back(v); } 45 int st[N], top, rt;//rt为虚树根
    void insert(int u) {
        if (top == 1) {
47
             st[++top] = u;
48
             return;
49
        }
50
        int lca = LCA(u, st[top]);
51
52
        if (lca != st[top]) {
53
             while (top > 1 && dfn[st[top - 1]] >= dfn[lca])
                 add_edge(st[top - 1], st[top]), top--;
54
             if (lca != st[top]) add_edge(lca, st[top]), st[top] = lca;
55
56
        st[++top] = u;
57
58
    bool cmp(const int &x, const int &y) { return dfn[x] < dfn[y]; }</pre>
59
    void build(vector<int> &v) {
60
        st[top = 1] = rt;
61
        sort(v.begin(), v.end(), cmp);
62
        for (auto &i: v) {
63
             tag[i] = 1;
64
65
             if (i != rt) insert(i);
66
        while (top > 1) add_edge(st[top - 1], st[top]), top--;
67
68 }
69
70
    void dp(int u) {
71
72
        //...
73
    }
    void clear(int u) {//清空虚树边和标记,也可以和dp合并
74
        for (auto &v: g[u]) clear(v);
75
        g[u].clear(); tag[u] = 0;
76
    }
77
78
    void solve() {
79
        dp(rt); clear(rt);
80
81
        //...
    }
82
83
    int main() {
84
        scanf("%d", &n);
85
        for (int i = 1; i < n; i++) {
86
             int u, v; scanf("%d%d", &u, &v);
87
88
             add(u, v); add(v, u);
89
        //此处距离为1, 所以用dep替代dis, dis[fa[rt] = 0] = -1
90
91
        dep[0] = -1, rt = 1;
92
        dfs(rt, 0); dfs2(rt, rt);
93
94
        int Q; scanf("%d", &Q);
95
        while (Q--) {
96
             int K; scanf("%d", &K);//读取关键点
97
             for (int i = 1; i <= K; i++) scanf("%d", &a[i]);</pre>
98
99
             //构建虚树
             build(a);
100
```

```
101
            solve();
102
103
104
        return 0;
105
   }
    5.13 LCT
   int ch[N][2], fa[N], rev[N], siz[N];//基本内容
 2 int sum[N], val[N], tag[N];//另外要维护的
 3 #define lc ch[u][0]
 4 #define rc ch[u][1]
 5 #define identify(u) (ch[fa[u]][1] == u)
   #define isRoot(u) (u != ch[fa[u]][0] && u != ch[fa[u]][1])
    void flip(int u) { swap(lc, rc); rev[u] ^= 1; }
 7
    void push_up(int u) {
        siz[u] = siz[lc] + siz[rc] + 1;
 9
10
    }
11
    void push_down(int u) {
12
        if (rev[u]) {
13
            if (lc) flip(lc);
14
            if (rc) flip(rc);
15
            rev[u] = 0;
16
        }
17
18
        //...
19
    }
20
    void update(int u) {//当前点之上的所有点都push_down
21
        if (!isRoot(u)) update(fa[u]);
        push_down(u);
22
23
    void rotate(int u) {
24
        int f = fa[u], fc = identify(u);
25
26
        int g = fa[f], gc = identify(f);
        int uc = fc \wedge 1, c = ch[u][uc];
27
28
        if (!isRoot(f))
29
            ch[g][gc] = u; fa[u] = g;
30
        ch[f][fc] = c, fa[c] = f;
31
        ch[u][uc] = f, fa[f] = u;
32
        push_up(f); push_up(u);
33
    }
    void splay(int u) {//将u变为u所在的Splay的根
34
35
        update(u);
        for (int f; f = fa[u], !isRoot(u); rotate(u))
36
            if (!isRoot(f)) rotate(identify(f) ^ identify(u) ? u : f);
37
38
    int access(int u) {//将(rt, u)之间的路径变为实链
39
        int pre = 0;
40
        for (; u; u = fa[pre = u])
41
42
            splay(u), rc = pre, push_up(u);
43
        return pre;
    }
44
    void makeRoot(int u) {//将u变为整棵树的根(注意:不一定是当前splay的根)
45
46
        u = access(u);
        flip(u);
47
48
    int findRoot(int u) {
49
        access(u), splay(u);
50
```

```
while (lc) push_down(u), u = lc;
51
52
       splay(u);
       return u;
53
54
   }
   void link(int u, int v) {
55
       makeRoot(u); splay(u);
56
       if (findRoot(v) != u) fa[u] = v;
57
   }
58
   void split(int u, int v) {
59
60
       makeRoot(u);
61
       access(v); splay(v);//加了这个就将v变为splay的根
   }
62
   void cut(int u, int v) {
63
       makeRoot(u); splay(u);
64
       if (findRoot(v) == u \&\& fa[v] == u \&\& !ch[v][0]) {
65
            fa[v] = ch[u][1] = 0;
66
            push_up(u);
67
       }
68
   }
69
70
   void fix(int u, int k) {
71
       splay(u); val[u] = k;
72 }
```

## 6 数学

#### 6.1 Mint

```
template <typename T>
   T inverse(T a, T m) {
       T u = 0, v = 1;
3
       while (a != 0) {
4
            T t = m / a;
5
            m -= t * a; std::swap(a, m);
6
            u -= t * v; std::swap(u, v);
7
8
9
       assert(m == 1);
10
       return u;
11 }
12
   template <typename T>
13
   class Modular {
14
   public:
15
       using Type = typename std::decay<decltype(T::value)>::type;
16
17
       constexpr Modular() : value() {}
18
       template <typename U>
19
       Modular(const U& x) {
20
            value = normalize(x);
21
       }
22
23
24
       template <typename U>
       static Type normalize(const U& x) {
25
26
            Type v;
            if (-mod() \le x \& x < mod()) v = static_cast < Type > (x);
27
28
            else v = static_cast<Type>(x % mod());
29
            if (v < 0) v += mod();
30
            return v;
31
       }
32
       const Type& operator()() const { return value; }
33
34
       template <typename U>
35
       explicit operator U() const { return static_cast<U>(value); }
36
       constexpr static Type mod() { return T::value; }
37
       template <typename U> Modular& operator*=(const U& other) { return *this *= Modular
38
       (other); }
       template <typename U> Modular& operator/=(const U& other) { return *this /= Modular
39
       (other); }
       Modular& operator+=(const Modular& other) { if ((value += other.value) >= mod())
40
       value -= mod(); return *this; }
       Modular& operator-=(const Modular& other) { if ((value -= other.value) < 0) value
41
       += mod(); return *this; }
       template <typename U> Modular& operator+=(const U& other) { return *this += Modular
42
       (other); }
       template <typename U> Modular& operator-=(const U& other) { return *this -= Modular
43
       (other); }
       Modular& operator++() { return *this += 1; }
44
       Modular& operator--() { return *this -= 1; }
45
       Modular operator++(int) { Modular result(*this); *this += 1; return result; }
46
       Modular operator--(int) { Modular result(*this); *this -= 1; return result; }
47
       Modular operator-() const { return Modular(-value); }
48
49
```

```
template <typename U = T>
50
       typename std::enable_if<std::is_same<typename Modular<U>::Type, int>::value,
51
       Modular>::type& operator*=(const Modular& rhs) {
   #ifdef _WIN32
52
       uint64_t x = static_cast<int64_t>(value) * static_cast<int64_t>(rhs.value);
53
       uint32_t xh = static_cast < uint32_t > (x >> 32), xl = static_cast < uint32_t > (x), d, m;
54
       55
56
            : "=a" (d), "=d" (m)
: "d" (xh), "a" (xl), "r" (mod())
57
58
       );
59
       value = m;
60
61
   #else
       value = normalize(static_cast<int64_t>(value) * static_cast<int64_t>(rhs.value));
62
63
   #endif
       return *this;
64
65
       template <typename U = T>
66
       typename std::enable_if<std::is_same<typename Modular<U>::Type, long long>::value,
67
       Modular>::type& operator*=(const Modular& rhs) {
            long long q = static_cast<long long>(static_cast<long double>(value) * rhs.
68
       value / mod());
           value = normalize(value * rhs.value - q * mod());
69
            return *this:
70
71
       template <typename U = T>
72
       typename std::enable_if<!std::is_integral<typename Modular<U>::Type>::value,
73
       Modular>::type& operator*=(const Modular& rhs) {
74
            value = normalize(value * rhs.value);
            return *this;
75
       }
76
77
       Modular& operator/=(const Modular& other) { return *this *= Modular(inverse(other.
78
       value, mod())); }
79
       friend const Type& abs(const Modular& x) { return x.value; }
80
81
82
       template <typename U>
       friend bool operator==(const Modular<U>& lhs, const Modular<U>& rhs);
83
84
       template <typename U>
85
       friend bool operator<(const Modular<U>& lhs, const Modular<U>& rhs);
86
87
       template <typename V, typename U>
88
       friend V& operator>>(V& stream, Modular<U>& number);
89
90
   private:
91
       Type value;
92
93
   };
94
95
   template <typename T> bool operator==(const Modular<T>& lhs, const Modular<T>& rhs) {
       return lhs.value == rhs.value: }
   template <typename T, typename U> bool operator==(const Modular<T>& lhs, U rhs) {
96
       return lhs == Modular<T>(rhs); }
   template <typename T, typename U> bool operator==(U lhs, const Modular<T>& rhs) {
97
       return Modular<T>(lhs) == rhs; }
98
   template <typename T> bool operator!=(const Modular<T>& lhs, const Modular<T>& rhs) {
99
       return !(lhs == rhs); }
```

```
100 template <typename T, typename U> bool operator!=(const Modular<T>& lhs, U rhs) {
        return !(lhs == rhs); }
    template <typename T, typename U> bool operator!=(U lhs, const Modular<T>& rhs) {
101
        return !(lhs == rhs); }
102
    template <typename T> bool operator<(const Modular<T>& lhs, const Modular<T>& rhs) {
103
        return lhs.value < rhs.value; }</pre>
104
    template <typename T> Modular<T> operator+(const Modular<T>& lhs, const Modular<T>& rhs
105
        ) { return Modular<T>(lhs) += rhs; }
    template <typename T, typename U> Modular<T> operator+(const Modular<T>& lhs, U rhs) {
        return Modular<T>(lhs) += rhs; }
    template <typename T, typename U> Modular<T> operator+(U lhs, const Modular<T>& rhs) {
107
        return Modular<T>(lhs) += rhs; }
108
    template <typename T> Modular<T> operator-(const Modular<T>& lhs, const Modular<T>& rhs
109
        ) { return Modular<T>(lhs) -= rhs; }
    template <typename T, typename U> Modular<T> operator-(const Modular<T>& lhs, U rhs) {
        return Modular<T>(lhs) -= rhs; }
    template <typename T, typename U> Modular<T> operator-(U lhs, const Modular<T>& rhs) {
111
        return Modular<T>(lhs) -= rhs; }
112
    template <typename T> Modular<T> operator*(const Modular<T>& lhs, const Modular<T>& rhs
113
        ) { return Modular<T>(lhs) *= rhs; }
    template <typename T, typename U> Modular<T> operator*(const Modular<T>& lhs, U rhs) {
        return Modular<T>(lhs) *= rhs; }
    template <typename T, typename U> Modular<T> operator*(U lhs, const Modular<T>& rhs) {
        return Modular<T>(lhs) *= rhs; }
116
    template <typename T> Modular<T> operator/(const Modular<T>& lhs, const Modular<T>& rhs
117
        ) { return Modular<T>(lhs) /= rhs; }
    template <typename T, typename U> Modular<T> operator/(const Modular<T>& lhs, U rhs) {
118
        return Modular<T>(lhs) /= rhs; }
    template <typename T, typename U> Modular<T> operator/(U lhs, const Modular<T>& rhs) {
119
        return Modular<T>(lhs) /= rhs; }
120
    template<typename T, typename U>
121
122
    Modular<T> power(const Modular<T>& a, const U& b) {
        assert(b >= 0);
123
        Modular<T> x = a, res = 1;
124
125
        Up = b;
126
        while (p > 0) {
            if (p \& 1) res *= x;
127
128
            x *= x;
129
            p >>= 1;
130
        return res;
131
    }
132
133
    template <typename T>
135
    bool IsZero(const Modular<T>& number) {
136
        return number() == 0;
137
    }
138
    template <typename T>
139
    std::string to_string(const Modular<T>& number) {
140
141
        return to_string(number());
142
    }
143
```

```
144 // U == std::ostream? but done this way because of fastoutput
    template <typename U, typename T>
    U& operator<<(U& stream, const Modular<T>& number) {
146
        return stream << number();</pre>
147
148 }
149
150 // U == std::istream? but done this way because of fastinput
    template <typename U, typename T>
    U& operator>>(U& stream, Modular<T>& number) {
152
        typename std::common_type<typename Modular<T>::Type, long long>::type x;
153
154
        stream >> x;
155
        number.value = Modular<T>::normalize(x);
156
        return stream;
    }
157
158
159 /*
160 using ModType = int;
162 struct VarMod { static ModType value; };
163 ModType VarMod::value;
164 ModType& md = VarMod::value;
165 using Mint = Modular<VarMod>;
166
167
168 constexpr int md = 998244353;
    using Mint = Modular<std::integral_constant<std::decay<decltype(md)>::type, md>>;
169
170
    std::vector<Mint> fact(1, 1);
171
    std::vector<Mint> inv_fact(1, 1);
172
173
    /*Mint C(int n, int k) {
174
175
        if (k < 0 | | k > n) {
            return 0;
176
177
        while ((int) fact.size() < n + 1) {
178
            fact.push_back(fact.back() * (int) fact.size());
179
180
            inv_fact.push_back(1 / fact.back());
181
        return fact[n] * inv_fact[k] * inv_fact[n - k];
182
183
    }*/
    6.2 Z
   constexpr int P = 998244353;
    using i64 = long long;
    // assume -P \ll x \ll 2P
 3
    int norm(int x) {
 4
        if (x < 0) {
 5
 6
            x += P;
 7
        if (x >= P) {
 8
 9
            x -= P;
10
        }
11
        return x;
12
    }
    template<class T>
    T power(T a, i64 b) {
15
        T res = 1;
```

```
for (; b; b /= 2, a *= a) {
16
            if (b % 2) {
17
                res *= a;
18
19
20
21
        return res;
22
   }
   struct Z {
23
24
        int x;
        Z(int x = 0) : x(norm(x)) \{\}
25
26
        Z(i64 x) : x(norm(x % P)) {}
27
        int val() const {
28
            return x;
        }
29
        Z operator-() const {
30
            return Z(norm(P - x));
31
32
33
        Z inv() const {
            assert(x != 0);
34
            return power(*this, P - 2);
35
36
        Z &operator*=(const Z &rhs) {
37
            x = i64(x) * rhs.x % P;
38
39
            return *this;
40
        Z &operator+=(const Z &rhs) {
41
            x = norm(x + rhs.x);
42
            return *this;
43
44
        Z &operator-=(const Z &rhs) {
45
            x = norm(x - rhs.x);
46
            return *this;
47
48
        Z &operator/=(const Z &rhs) {
49
            return *this *= rhs.inv();
50
51
52
        friend Z operator*(const Z &lhs, const Z &rhs) {
53
            Z res = lhs;
            res *= rhs;
54
            return res;
55
56
        friend Z operator+(const Z &lhs, const Z &rhs) {
57
58
            Z res = lhs;
            res += rhs;
59
            return res;
60
61
        friend Z operator-(const Z &lhs, const Z &rhs) {
62
            Z res = lhs;
63
            res -= rhs;
64
            return res;
65
66
67
        friend Z operator/(const Z &lhs, const Z &rhs) {
68
            Z res = lhs;
            res /= rhs;
69
            return res;
70
71
72
        friend std::istream &operator>>(std::istream &is, Z &a) {
            i64 v;
73
74
            is >> v;
```

```
a = Z(v);
75
76
            return is;
77
        friend std::ostream &operator<<(std::ostream &os, const Z &a) {</pre>
78
79
            return os << a.val();</pre>
80
81 };
   6.3 exgcd
   template <typename T>
1
   std::array<T, 3> exgcd(T a, T b) {
2
        if (b == 0) {
3
            return {a, 1, 0};
4
5
        auto [g, x, y] = exgcd(b, a % b);
6
7
        return \{g, y, x - a / b * y\};
8
        auto [g, x, y] = exgcd < long long > (a, b);
9
        assert(1LL * a * x + 1LL * b * y == g);
10
        if (c % g != 0) {
11
            std::cout << -1 << std::endl;
12
13
            continue ;
14
15
16 }
   6.4 factorizer
1
   namespace factorizer {
2
3 template <typename T>
4 struct FactorizerVarMod { static T value; };
5 template <typename T>
  T FactorizerVarMod<T>::value;
7
8
   template <typename T>
9
   bool IsPrime(T n, const std::vector<T>& bases) {
10
        if (n < 2) {
11
            return false;
12
        std::vector<T> small_primes = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
13
        for (const T& x : small_primes) {
14
15
            if (n \% x == 0) {
16
                return n == x;
            }
17
18
        if (n < 31 * 31) {
19
            return true;
20
        }
21
        int s = 0;
22
23
        T d = n - 1;
        while ((d & 1) == 0) {
24
            d >>= 1;
25
26
            S++;
27
28
        FactorizerVarMod<T>::value = n;
```

```
for (const T& a : bases) {
29
            if (a \% n == 0) {
30
31
                 continue;
32
33
            Modular<FactorizerVarMod<T>> cur = a;
34
            cur = power(cur, d);
            if (cur == 1) {
35
                 continue;
36
            }
37
38
            bool witness = true;
            for (int r = 0; r < s; r++) {
39
40
                if (cur == n - 1) {
41
                     witness = false;
42
                     break;
43
                cur *= cur;
44
45
            if (witness) {
46
47
                return false;
48
            }
49
        return true;
50
51 }
52
53
   bool IsPrime(int64_t n) {
        return IsPrime(n, {2, 325, 9375, 28178, 450775, 9780504, 1795265022});
54
   }
55
56
   bool IsPrime(int32_t n) {
57
        return IsPrime(n, {2, 7, 61});
58
   }
59
60
   // but if you really need uint64_t version...
61
   /*
62
   bool IsPrime(uint64_t n) {
63
        if (n < 2) {
64
65
            return false;
66
        std::vector<uint32_t> small_primes = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
67
        for (uint32_t x : small_primes) {
68
            if (n == x) {
69
70
                 return true;
71
72
            if (n \% x == 0) {
                return false;
73
            }
74
75
        if (n < 31 * 31) {
76
            return true;
77
78
79
        uint32_t s = __builtin_ctzll(n - 1);
        uint64_t d = (n - 1) >> s;
80
        function<br/>
d_t = [an, as, ad](uint64_t) witness = [an, as, ad](uint64_t)
81
            uint64_t cur = 1, p = d;
82
            while (p > 0) {
83
                if (p & 1) {
84
85
                     cur = (\_uint128\_t) cur * a % n;
86
87
                a = (\_uint128\_t) a * a % n;
```

```
p >>= 1;
 88
 89
             if (cur == 1) {
 90
 91
                  return false;
 92
             for (uint32_t r = 0; r < s; r++) {
93
                  if (cur == n - 1) {
 94
                      return false;
 95
 96
 97
                  cur = (\_uint128\_t) cur * cur % n;
             }
 98
99
             return true;
         };
100
         std::vector<uint64_t> bases_64bit = {2, 325, 9375, 28178, 450775, 9780504,
101
         1795265022};
         for (uint64_t a : bases_64bit) {
102
             if (a \% n == 0) {
103
104
                  return true;
105
             if (witness(a)) {
106
                  return false;
107
108
109
         return true;
110
111
    */
112
113
    std::vector<int> least = {0, 1};
114
    std::vector<int> primes;
115
    int precalculated = 1;
116
117
    void RunLinearSieve(int n) {
118
         n = std::max(n, 1);
119
         least.assign(n + 1, 0);
120
         primes.clear();
121
         for (int i = 2; i <= n; i++) {
122
123
              if (least[i] == 0) {
124
                  least[i] = i;
                  primes.push_back(i);
125
126
             for (int x : primes) {
127
                  if (x > least[i] || i * x > n) {
128
129
                      break;
130
                  least[i * x] = x;
131
132
             }
133
         precalculated = n;
134
    }
135
136
137
    void RunSlowSieve(int n) {
138
         n = std::max(n, 1);
139
         least.assign(n + 1, 0);
         for (int i = 2; i * i <= n; i++) {
140
             if (least[i] == 0) {
    for (int j = i * i; j <= n; j += i) {</pre>
141
142
                      if (least[j] == 0) {
143
144
                           least[j] = i;
                      }
145
```

```
}
146
             }
147
148
         primes.clear();
149
         for (int i = 2; i <= n; i++) {
150
             if (least[i] == 0) {
151
                 least[i] = i;
152
                 primes.push_back(i);
153
             }
154
155
         }
156
         precalculated = n;
157
    }
158
    void RunSieve(int n) {
159
         RunLinearSieve(n);
160
161
162
163
    template <typename T>
    std::vector<std::pair<T, int>> MergeFactors(const std::vector<std::pair<T, int>>& a,
        const std::vector<std::pair<T, int>>& b) {
         std::vector<std::pair<T, int>> c;
165
         int i = 0;
166
         int j = 0;
167
         while (i < (int) a.size() || j < (int) b.size()) {</pre>
168
             if (i < (int) a.size() && j < (int) b.size() && a[i].first == b[j].first) {</pre>
169
                 c.emplace_back(a[i].first, a[i].second + b[j].second);
170
                 ++i;
171
172
                 ++j;
                 continue;
173
174
             if (j == (int) b.size() || (i < (int) a.size() && a[i].first < b[j].first)) {</pre>
175
                 c.push_back(a[i++]);
176
             } else {
177
                 c.push_back(b[j++]);
178
179
180
181
         return c;
182
    }
183
    template <typename T>
184
    std::vector<std::pair<T, int>> RhoC(const T& n, const T& c) {
185
         if (n <= 1) {
186
             return {};
187
188
         if ((n & 1) == 0) {
189
             return MergeFactors({{2, 1}}, RhoC(n / 2, c));
190
191
         if (IsPrime(n)) {
192
             return {{n, 1}};
193
194
195
         FactorizerVarMod<T>::value = n;
196
         Modular<FactorizerVarMod<T>> x = 2;
         Modular<FactorizerVarMod<T>> saved = 2;
197
         T power = 1;
198
         T lam = 1;
199
         while (true) {
200
201
             X = X * X + C;
202
             T g = \underline{gcd((x - saved)(), n)};
             if (g != 1) {
203
```

```
return MergeFactors(RhoC(g, c + 1), RhoC(n / g, c + 1));
204
205
             if (power == lam) {
206
207
                  saved = x;
208
                 power <<= 1;
                 lam = 0;
209
210
211
             lam++;
212
         }
213
         return {};
214
    }
215
    template <typename T>
216
    std::vector<std::pair<T, int>> Rho(const T& n) {
217
         return RhoC(n, static_cast<T>(1));
218
219
220
221
    template <typename T>
    std::vector<std::pair<T, int>> Factorize(T x) {
222
         if (x <= 1) {
223
             return {};
224
225
         if (x <= precalculated) {</pre>
226
227
             std::vector<std::pair<T, int>> ret;
228
             while (x > 1) {
                 if (!ret.empty() && ret.back().first == least[x]) {
229
                      ret.back().second++;
230
231
                 } else {
                      ret.emplace_back(least[x], 1);
232
233
234
                 x \neq least[x];
235
             }
236
             return ret;
237
         if (x <= static_cast<int64_t>(precalculated) * precalculated) {
238
             std::vector<std::pair<T, int>> ret;
239
240
             if (!IsPrime(x)) {
241
                  for (T i : primes) {
                      T t = x / i;
242
                      if (i > t) {
243
                          break;
244
245
                      if(x == t * i) {
246
                          int cnt = 0;
247
                          while (x \% i == 0) \{
248
                              x /= i;
249
250
                               cnt++;
                          }
251
                          ret.emplace_back(i, cnt);
252
253
                          if (IsPrime(x)) {
254
                               break;
255
                          }
256
                      }
                 }
257
258
             if (x > 1) {
259
260
                 ret.emplace_back(x, 1);
261
262
             return ret;
```

```
263
        return Rho(x);
264
265
266
267
    template <typename T>
    std::vector<T> BuildDivisorsFromFactors(const std::vector<std::pair<T, int>>& factors)
        std::vector<T> divisors = {1};
269
270
        for (auto& p : factors) {
             int sz = (int) divisors.size();
271
272
             for (int i = 0; i < sz; i++) {
273
                 T cur = divisors[i];
                 for (int j = 0; j < p.second; j++) {
274
                     cur *= p.first;
275
276
                     divisors.push_back(cur);
                 }
277
             }
278
279
        }
        sort(divisors.begin(), divisors.end());
280
        return divisors;
281
282 }
283
284 } // namespace factorizer
    6.5 comb
 1 class Comb {
    public :
 2
 3
         const int n;;
        std::vector<Z> fac, inv, ifac;
 4
        Comb(int n) : n(n), fac(n), inv(n), ifac(n) {
 5
 6
             fac[0] = fac[1] = inv[0] = inv[1] = ifac[0] = ifac[1] = 1;
             for(int i = 2;i < n; i++) {</pre>
 7
                 fac[i] = fac[i - 1] * i;
 8
                 inv[i] = (P - P / i) * inv[P % i];
ifac[i] = ifac[i - 1] * inv[i];
 9
 10
             }
11
12
        Z C(int n, int m) {
13
             if(m < 0 || n < 0 || m > n) return 0;
14
             return fac[n] * ifac[m] * ifac[n - m];
15
16
        Z Lucas(long long m, long long n) { return n ? Lucas(m / P, n / P) * C(m % P, n % P
17
        ):1;}
18 };
    6.6 算术基本定理
    ll get_Count(ll n) {
        ll ans = 1;
        for(int i = 2;i * i <= n; i++) {
 3
             if(n % i == 0) {
 4
 5
                 int a = 0;
                 while(n % i == 0) {
 6
 7
                     a++;
 8
                     n /= i;
 9
                 }
```

```
ans *= (a + 1);
10
             }
11
12
        if(n > 1) ans *= 2;
13
14
        return ans;
15
   }
16
   ll get_Sum(ll n) {
17
        ll ans = 1;
18
        for(int i = 2;i * i <= n; i++) {
19
20
             if(n \% i == 0) {
21
                 11 a = 1;
                 while(n % i == 0) {
22
                      n \neq i;
23
                      a *= i;
24
25
                 ans = ans * (a * i - 1) / (i - 1);
26
             }
27
28
29
        if(n > 1) ans *= (n + 1);
30
        return ans;
31
   }
   6.7
         筛 phi
   int is_prime[N], prime[N], cnt, phi[N];
   void makePhi() {
2
3
        phi[1] = 1, cnt = 0;
        for (int i = 2; i < N; i++) {
4
5
             if (!is_prime[i]) prime[++cnt] = i, phi[i] = i - 1;
             for (int j = 1; j <= cnt && i * prime[j] < N; j++) {
   is_prime[i * prime[j]] = 1;</pre>
6
7
                 if (i % prime[j] == 0) {
8
                      phi[i * prime[j]] = phi[i] * prime[j];
9
10
                      break:
11
                 else phi[i * prime[j]] = phi[i] * phi[prime[j]];
12
13
             }
14
        }
  }
15
         筛 mobius
   const int N = 1e5 + 10;
1
   bool is_prime[N];
  int prime[N], mu[N], cnt;
3
4
   void makeMobius() {
5
        mu[1] = 1; is_prime[0] = is_prime[1] = true;
6
7
        for(int i = 2; i < N; i++) {
             if (!is_prime[i]) {
8
9
                 mu[i] = -1;
                 prime[++cnt] = i;
10
11
             for (int j = 1; j <= cnt && i * prime[j] < N; j++) {
   is_prime[i * prime[j]] = true;</pre>
12
13
                 if (i % prime[j] == 0) {
14
```

```
mu[i * prime[j]] = 0;
15
                    break;
16
17
                mu[i * prime[j]] = -mu[i];
18
            }
19
20
       }
21 }
   6.9
         筛积性函数
   //只需要计算f(p ^ k)即可
   //其余的都可以通过积性函数的性质来计算
3
  int vis[N], prime[N], num;
4
   int f[N], low[N];
5
6
   void makeF(int siz) {//f为积性函数
7
       num = 0, low[1] = f[1] = 1;
8
        for (int i = 2; i \le siz; i++) {
9
            if (!vis[i]) prime[++num] = i, low[i] = i, f[i] = ...;//这里是f(p)的答案
10
            for (int j = 1; j <= num && i * prime[j] <= siz; j++) {</pre>
11
                vis[i * prime[j]] = 1;
12
                if (i % prime[j] == 0) {
13
                    low[i * prime[j]] = low[i] * prime[j];
14
                    if(low[i] == i) {//i = prime[j] ^ k
15
16
                        //只需要这里算一下
                        //考虑 p ^ 1 , p ^ 2, p ^ 3...
17
18
                    else f[i * prime[j]] = 1ll * f[i / low[i]] * f[prime[j] * low[i]] % mod
19
20
                    break:
21
                low[i * prime[j]] = prime[j];
f[i * prime[j]] = 1ll * f[i] * f[prime[j]] % mod;
22
23
            }
24
       }
25
   }
26
   6.10
          欧拉函数
   // 求解单个正整数的欧拉函数
   int Get_phi(int n) {
       int ans = n;
for(int i = 2;i * i <= n; i++) {</pre>
3
4
            if(n % i == 0) {
5
                ans = ans - ans / i;
6
                while(n % i == 0)
7
8
                    n /= i;
            }
9
10
       if(n > 1)
11
12
           ans = ans - ans / n;
13
       return ans;
14
   }
15
  // 埃拉托斯特尼筛求欧拉函数
  int phi[10005];
```

```
18
   void Euler_sieve(int n) {
19
        phi[1] = 1;
20
        for(int i = 2;i <= n; i++) {
21
22
            if(!phi[i]) {
                for(int j = i; j \leftarrow n; j \leftarrow i)
23
                                                    {
24
                     if(!phi[j])
25
                         phi[j] = j;
26
                     phi[j] = phi[j] / i * (i - 1);
27
                }
28
            }
29
        }
   }
30
31
32
   // 欧拉筛求欧拉函数
33
34 \quad const int N = 5e6 + 10;
   bool is_prime[N];
   int prime[N], phi[N], tot;
36
37
   void Euler() {
38
        phi[1] = 1; is_prime[1] = true;
39
        for(int i = 2;i < N; i++){
40
41
            if(!is_prime[i]) {
42
                phi[i] = i - 1;
                prime[++tot] = i;
43
44
            for(int j = 1; j \le tot \&\& i * prime[j] < N; j++){
45
                is_prime[i * prime[j]] = true;
46
                if(i % prime[j]) {
47
                     phi[i * prime[j]] = phi[i] * (prime[j] - 1);
48
                }
49
50
                else{
                     phi[i * prime[j]] = phi[i] * prime[j];
51
52
                     break;
                }
53
54
            }
55
        }
   }
56
   6.11
          原根
   typedef long long 11;
3
   vector<ll> YG;
   ll p, n; // p是模数, n是p的欧拉函数值
4
5
6
   ll gcd(ll a, ll b) {
        return b ? gcd(b, a % b) : a;
7
8
9
   ll quick_pow(ll a, ll b, ll p);
10
11
   ll phi(ll n) {
12
13
        11 \text{ ans} = n;
        for(int i = 2;i * i <= n; i++) {
14
            if(n \% i == 0) {
15
16
                ans = ans - ans / i;
```

```
while(n % i == 0) {
17
                    n \neq i;
18
19
            }
20
21
22
       if(n > 1)
23
            ans = ans - ans / n;
24
       return ans;
   }
25
26
27
   vector<ll> PrimeFac(ll n) { // n的素因子
28
       vector<ll> fac;
       fac.clear();
29
       for(ll i = 2;i * i <= n; i++) {</pre>
30
            if(n % i == 0) {
31
                fac.push_back(i);
32
                while(n % i == 0)
33
34
                    n \neq i;
            }
35
36
       if(n > 1)
37
            fac.push_back(n);
38
        return fac;
39
40 }
41
   bool is_Protogen(ll p) { // 原根p = 2、4、p^k、2*p^k(p为非2的质数, k为任意数)
42
       if(p == 2 || p == 4) return true;
43
       if(p <= 1 || p % 4 == 0) return false;</pre>
44
       11 num = 0;
45
       while(p % 2 == 0) // 2的倍数先筛掉
46
47
       for(int i = 3; i * i \leftarrow p; i++) { // p只能是一个非2的素数的倍数构成,否则没有原根
48
49
            if(p \% i == 0) {
50
                num++;
                while(p % i == 0)
51
                    p /= i;
52
53
            }
54
       if(p > 1) num++;
55
       if(num == 1) return true;
56
       return false;
57
   }
58
59
   ll Protogen(ll p) {
60
       if(!is_Protogen(p)) // 先判断是否存在原根
61
62
            return -1;
63
       n = phi(p);
       if(p == 2) return 1;
64
       if(p == 3) return 2;
65
66
       if(p == 4) return 3;
       vector<ll> fac = PrimeFac(n); // f(p)的素因子
67
68
       for(int i = 2; i \le p - 1; i++) {
            if(gcd(i, p)!= 1) // n是模p的欧拉函数值, i要和n互质
69
70
                continue;
            bool flag = true;
71
            for(ll j = 0; j < fac.size(); j++) {
72
                if(quick_pow(i, n / fac[j] , p) == 1)
73
                    flag = 0;
74
            }
75
```

```
if(flag) // i就是原根
76
77
                  return i;
78
         return -1;
79
    }
80
81
    void Sum_Protogen(ll k) { // 找出n的所有原根
82
         YG.push_back(k);
83
         for(int i = 2; i < n; i++) {
84
             if(gcd(i, n) == 1) // i要与f(n)互质
85
86
                 YG.push_back(quick_pow(k, i, p));
87
         }
    }
88
89
    int main() {
90
         cin >> p;
91
         ll k = Protogen(p); // p的原根
92
93
         cout << k << endl;</pre>
         Sum_Protogen(k);
94
         for(int i = 0; i < YG.size(); i++) {</pre>
95
             cout << YG[i] << " ";
96
97
         cout << endl;</pre>
98
99
         return 0;
100
   }
```

## 6.12 原根表

```
1
   mod
                                                   原根
2
   r*2^k+1
                               k
                                               g
                2
3
   3
        1
            2
                2
4
   5
        1
                3
5
   17
        1
            4
            5
                5
6
   97
        3
                5
   193 3
            6
7
            8
                3
8
   257 1
            15
                9
9
   7681
                     17
            3
                12
10
   12289
                    11
            5
11
   40961
                13
                    3
12
   65537
            1
                16
                     3
                    10
13
   786433
            3
                18
   5767169 11
                     3
14
                19
   7340033 7
                     3
15
                20
                     21
   23068673
16
                11
                         3
                     22
17
   104857601
                25
                         3
                     25
   167772161
                5
18
                7
                     26
                         3
19
   469762049
                         3
   998244353
                119 23
20
                             这个数常用
                479 21
                         3
  1004535809
21
                             加起来不会爆int
                15
                    27
                         31
  2013265921
22
                    27
                         3
23
  2281701377
                17
                              这个数平方刚好不会爆11
  3221225473
                     30
                         5
24
                         3
25
   75161927681 35
                     31
                         7
26
   77309411329 9
                     33
   206158430209
                     3
                         36
                             22
27
                             7
   2061584302081
                     15
                         37
28
                             3
   2748779069441
                     5
                         39
29
                             5
   6597069766657
                     3
                         41
```

```
31 39582418599937 9
                        42
                            5
                        43
                            5
   79164837199873 9
                        44
                            7
33 263882790666241 15
                            45
                        35
                                3
34 1231453023109121
                                3
   1337006139375617
                        19
                            46
36 3799912185593857
                        27
                            47
                                5
                            48
                                19
37 4222124650659841
                        15
38 7881299347898369
                        7
                            50
                                6
39 31525197391593473
                        7
                            52
                                3
40 180143985094819841
                        5
                            55
                                6
41 1945555039024054273 27
                                5
42 4179340454199820289 29
   6.13
          阶乘逆元
   const int N = 5e6 + 10;
   const ll mod = 1e9 + 7;
3
   ll F[N], invn[N], invF[N];
4
5
   void Init() {
6
       F[0] = F[1] = invn[0] = invn[1] = invF[0] = invF[1] = 1;
7
       for(int i = 2; i < N; i++){
8
           F[i] = F[i - 1] * i % mod;
9
           invn[i] = (mod - mod / i) * invn[mod % i] % mod;
10
           invF[i] = invF[i - 1] * invn[i] % mod;
11
       }
12
13
   }
          常见积性函数
   6.14
   //phi[i * j] = phi[i] * phi[j] * gcd(i, j) / phi[gcd(i, j)]
3
5 // d[i * j] = \sum_{x \in \{x | i\}} * \sum_{y \in \{y | j\}} * [gcd(x, y) = 1]
   6.15 Miller-Rabin
   // 二次探测定理: 对素数p,满足x^2≡1(modp)的小于p的正整数解x只有1或p−1.
2
   #include <bits/stdc++.h>
3
   using namespace std;
   typedef long long 11;
5
   const int N = 1e5 + 7;
6
   const int times = 10;
7
8
   ll ksc(ll a, ll b, ll mod) {
9
10
       11 \text{ ans} = 0;
       while(b > 0) {
11
12
           if(b & 1) {
                ans = (ans + a) \% mod;
13
14
15
           a = (a << 1) \% mod;
16
           b >>= 1;
       }
17
```

```
18
       return ans;
   }
19
20
   ll quick_pow(ll a, ll b, ll mod) {
21
22
       ll ans = 1, base = a;
       while(b != 0) {
23
24
            if(b & 1) {
                ans = ans * base % mod;
25
26
            base = base * base % mod;
27
28
            b >>= 1;
29
       }
30
       return ans;
   }
31
32
   bool Miller_Pabin(ll n)//Miller测试的主体结构
33
34
   {
       if(n < 2) return false;</pre>
35
       if(n == 2) return true;
36
       if(n & 1 == 0) return false;//对于偶数的优化
37
       ll k = 0, u = n - 1; //p为Miller测试的k, u为Miller测试的m
38
39
       while(u & 1 == 0){ // 把x拆成u*2^k
40
41
            u >>= 1;
42
            k++;
43
       srand(time(NULL));
44
45
       ll x, pre; // pre为上次探测的x的值
46
47
       for(int i = 1;i <= times; i++) {</pre>
48
            x = rand() % (n - 1) + 1;
49
            x = quick_pow(x, u, n); // 先求出x^u(mod n)
50
            pre = x;
51
            for(int j = 1; j <= k; j++) {
52
                x = ksc(x, x, n);
53
54
                if(x == 1 \&\& pre != 1 \&\& pre != n - 1)
55
                    return false;
56
                pre = x;
57
            if(x != -1)
58
            return false;
59
60
       return true;
61
   }
62
63
   int main() {
64
65
       ll n; cin >> n;
       cout << (Miller_Pabin(n) ? "Prime" : "Not a Prime") << endl;</pre>
66
67 }
   6.16 quadraticResidue
1 typedef long long ll;
2
3 typedef struct{
       11 x, y; // 把求出来的w作为虚部, 则为a + bw
5 }num;
```

```
ll quick_pow(ll a, ll b, ll p) {
7
       ll ans = 1;
8
       while(b) {
9
10
           if(b \& 1) ans = ans * a % p;
           a = a * a % p;
11
12
           b >>= 1;
13
       return ans % p;
14
15
   }
16
17
   num num_mul(num a, num b, ll w, ll p) {// 复数乘法
18
       num ans = \{0, 0\};
19
       ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
20
       ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
21
22
       return ans;
   }
23
24
25
   ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
       num ans = \{1, 0\};
26
27
       while(b) {
28
           if(b & 1)
29
               ans = num_mul(ans, a, w, p);
30
           a = num_mul(a, a, w, p);
31
           b >>= 1;
32
33
       return ans.x % p;
   }
34
35
   ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
36
37
       return quick_pow(a, (p - 1) \gg 1, p);
   }
38
39
   ll Cipolla(ll n, ll p) {// 输入a和p, 是否存在x使得x^2 = a (mod p), 存在二次剩余返回x, 存在二次
40
       非剩余返回-1
                      注意: p是奇质数
41
       n \% = p;
       if(n == 0)
42
           return 0;
43
       if(p == 2)
44
           return 1;
45
       if(legendre(n, p) + 1 == p) // 二次非剩余
46
47
           return -1;
48
       ll a, w;
49
50
51
       while(true) {// 找出a, 求出w, 随机成功的概率是50%, 所以数学期望是2
           a = rand() \% p;
52
           W = ((a * a - n) \% p + p) \% p;
53
54
           if(legendre(w, p) + 1 == p) // 找到w, 非二次剩余条件
55
               break:
56
       }
57
       num x = \{a, 1\};
       return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x, 一个解是x, 另一个解是p-x, 这里的w其实
58
       要开方,但是由拉格朗日定理可知虚部为0,所以最终答案就是对x的实部用快速幂求解
59
   }
60
61
  int main()
62 {
```

```
ll n, p;
63
64
        cin >> n >> p;
        srand((unsigned)time(NULL));
65
        cout << Cipolla(n, p) << endl;</pre>
66
67
        return 0;
68 }
   6.17 bags
   // 求解a^x = b (mod c), 要求gcd(a, c) = 1, 不要求p为素数, x的范围是0 <= x <= p-1
2
   template <typename T>
3
   struct Hash {
4
        int n;
5
6
        int cnt;
        std::vector<int> head, next, hash, id;
7
8
        Hash(int _n): n(_n), head(_n, -1), next(_n), id(_n), hash(_n), cnt(0) {}
9
10
        void insert(T x, T y) {
            T k = x \% n;
11
            hash[cnt] = x;
12
            id[cnt] = y;
13
            next[cnt] = head[k];
14
            head[k] = cnt++;
15
        }
16
17
        T query(T x) {
18
            for(int i = head[x % n]; i != -1; i = next[i]){
19
                if(hash[i] == x)
20
21
                    return id[i];
22
23
            return -1;
24
        }
25 };
26
   template <typename T>
27
   T bsgs(T& a, T& b, T& c) {
        a %= c; b %= c;
29
30
        int cnt = 1;
        if (b == 1) return 0;
31
        Hash<long long> hs(100005);
32
33
        T m = std::ceil(sqrt((double) c));
34
        T x = 1, p = 1;
35
        for (T j = 0; j < m; j++, p = p * a % c) {
            hs.insert(p * b % c, j);
36
37
        for (T i = 1, j; i \le m; i++) {
38
            x = x * p % c;
39
            if ((j = hs.query(x)) != -1) {
40
41
                return i * m - j;
42
            }
43
        return -1;
44
45
   }
```

## 6.18 EX-BSGS

```
1 // a和c不互质
   #include <bits/stdc++.h>
3
   using namespace std;
   typedef long long 11;
5
6
   ll gcd(ll a, ll b) {
7
       return b ? gcd(b, a % b) : a;
8
9
10
   template <typename T>
11
12
   T ex_bsgs(T& a, T& b, T& c) {
       a %= c; b %= c;
13
       if (b == 1) return 0;
14
       T k = 0, tmp = 1, d;
15
       while (true) {
16
            d = \__gcd(a, c);
17
            if (d == 1) {
18
                break;
19
20
            if (b % d) {
21
22
                return -1;
23
            }
24
            b /= d; c /= d;
25
            tmp = tmp * (a / d) % c;
26
            k++;
            if (tmp == b) {
27
28
                return k;
            }
29
       }
30
       std::unordered_map<T, T> mp;
31
32
       T m = std::ceil(sqrt((double) c));
       T x = 1, p = 1;
33
       for(T j = 0; j < m; j++, p = p * a % c) {
34
           mp[p * b % c] = j;
35
       }
36
37
       x = tmp \% c;
38
       for(T i = 1;i <= m; i++) { // 枚举a^im
            x = x * p % c;
39
            if(mp[x]) {
40
                return k + i * m - mp[x];
41
42
43
44
       return -1;
   }
45
   6.19
          CRT
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long ll;
5 ll m[10005], a[10005], n; //a是余数, b是模数
6
   void exgcd(ll a, ll b, ll &x, ll &y) {
7
       if(b == 0) {
8
9
           x = 1;
10
            y = 0;
```

```
11
            return;
12
13
        exgcd(b, a % b , y , x);
        y -= a / b * x;
14
   }
15
16
   ll INV(ll a, ll mod) {
17
18
        11 x, y;
19
        exgcd(a, mod, x, y);
        x = (x \% mod + mod) \% mod;
20
21
        return x;
22
   }
23
   11 CRT() {
24
        ll ans = 0, M = 1;
25
        for(ll i = 1;i <= n; i++) {</pre>
26
            M *= m[i]; // M是所有除数的乘积
27
28
        for(ll i = 1;i <= n; i++) {</pre>
29
            ll mm = M / m[i];
30
            ll ret = INV(mm , m[i]); // 先求逆元
31
            ans = (ans + a[i] * mm % M * ret % M) % M;
32
33 /*
34
   ans = (ans + quick_mul(quick_mul(m , ret , M) , b[i] , M)) % M;
35
   利用快速乘防止爆longlong
36
37
        return (ans + M) % M;
38
   }
39
40
   int main() {
41
42
        11 \text{ ans} = 0;
        scanf("%lld",&n);
43
        for(ll i = 1;i <= n; i++) {</pre>
44
            scanf("%lld%lld",&m[i],&a[i]);
45
            a[i] = (a[i] % m[i] + m[i]) % m[i];// 防止b[i]为负
46
47
        }
48
        ans = CRT(); // 精髓
        printf("%lld",ans);
49
50
        return 0;
51
   }
          EX-CRT
   6.20
1 #include <iostream>
3 using namespace std;
 4
   typedef long long ll;
5
   ll c[100005], m[100005], n;
7
8
9
   ll ksc(ll a, ll b, ll mod) {
10
        11 \text{ ans} = 0;
        while(b > 0) {
11
            if(b & 1) {
12
                ans = (ans + a) \% mod;
13
14
            }
```

```
15
            a = (a << 1) \% mod;
16
            b >>= 1;
17
18
        return ans;
19 }
20
   ll gcd(ll a, ll b) {
21
22
        return b ? gcd(b, a % b) : a;
23
24
25
   ll ex_gcd(ll a, ll b, ll &x, ll &y) {
        ll res, t;
26
27
        if(!b) {
28
            x = 1;
29
            y = 0;
30
            return a;
        }
31
32
        res = ex_gcd(b, a \% b, x, y);
33
        t = x;
34
        x = y;
        y = t - (a / b) * y;
35
36
        return res;
37 }
38
39
   ll INV(ll a, ll mod) {
40
        11 x, y;
        ll d = ex_gcd(a, mod, x, y);
41
        return d ? (x % mod + mod) % mod : -1;
42
   }
43
44
   11 EX_CRT() {
45
46
        11 x, y;
        ll ans = c[1];
47
        ll M = m[1];
48
        for(int i = 2; i <= n; i++) {
49
            ll C = ((c[i] - ans) \% m[i] + m[i]) \% m[i];
50
            ll T = ex_gcd(M, m[i], x, y);
51
52
            if((c[i] - ans) % T)
                return -1;
53
            x = ksc(x, C / T, m[i] / T);
54
            ans += M * x;
55
            M *= (m[i] / T);
56
            ans = (ans \% M + M) \% M;
57
58
59
        return ans;
60 }
61
62 /*
63 ll EX_CRT() // 便于理解
64
   {
65
        for(int i = 2; i <= n; i++)
66
67
            M1 = m[i - 1], M2 = m[i], C1 = c[i - 1], C2 = c[i];
            ll T = gcd(M1, M2); // gcd(M1, M2)
68
            if((C2 - C1) % T) // 无解
69
70
                return -1;
            m[i] = (M1 * M2) / T; // 合并后新同余方程的模
71
            c[i] = INV(M1 / T, M2 / T) * (C2 - C1) / T % (M2 / T) * M1 + C1; // 可快速乘优化
72
            c[i] = (c[i] % m[i] + m[i]) % m[i]; // 合并后新同余方程的余
73
```

```
74
        return c[n];
75
   }
*/
76
77
78
  int main()
79
80
  {
81
        cin >> n;
        for(int i = 1;i <= n; i++)</pre>
82
83
        cin >> c[i] >> m[i];
84
        cout << EX_CRT() << endl;</pre>
85 }
   6.21 EX-Lucas
   // p不为质数,利用中国剩余定理结合求解
   #include <bits/stdc++.h>
3
   using namespace std;
4
5 typedef long long ll;
6
   const int N = 1e5 + 10;
7
8
   ll quick_pow(ll a, ll b, ll P) {
9
10
        ll\ ans = 1;
       while(b) {
11
12
            if(b & 1)
13
                ans = ans * a \% P;
            a = a * a % P;
14
15
            b >>= 1;
16
        return ans % P;
17
18 }
19
20
  ll ex_gcd(ll a, ll b, ll &x, ll &y) {
21
        ll res, t;
22
        if(!b) {
            x = 1;
23
24
            y = 0;
25
            return a;
26
        }
27
        res = ex_gcd(b, a \% b, x, y);
28
        t = x;
29
        x = y;
30
        y = t - (a / b) * y;
        return res;
31
   }
32
33
   ll INV(ll a, ll mod) {
34
35
        11 x, y;
36
        ll d = ex\_gcd(a, mod, x, y);
        return d ? (x % mod + mod) % mod : -1;
37
38
   }
39
   ll fac(ll n, ll P, ll pk) {// 阶乘除去质因子后模质数幂 (n / p^a) % pk
40
41
        if(!n) return 1;
42
        ll ans = 1;
43
        for(int i = 1;i < pk; i++) {// 第三部分: n!与p互质的乘积
```

```
if(i % P)
44
                ans = ans * i % pk;
45
46
        ans = quick_pow(ans, n / pk, pk) % pk; // 第三部分: n!与p互质的乘积,ans循环的次数为n/pk
47
48
        for(int i = 1;i <= n % pk; i++) {// 第四部分: 循环过后n!剩下的部分
            if(i % P) ans = ans * i % pk;
49
50
        return ans * fac(n / P, P, pk) % pk; // 第一部分, p的幂, 个数为n/p;
51
                                                                              第二部分: (n/p)!
   }
52
53
   ll C(ll m, ll n, ll P, ll pk) {// 组合数模质数幂
54
55
        if(n < 0 \mid l \mid m < 0 \mid l \mid n > m) return 0;
        ll f1 = fac(m, P, pk), f2 = fac(n, P, pk), f3 = fac(m - n, P, pk), tmp = 0; // tmp
56
       = pk1 - pk2 - pk3
        for(ll i = m; i ; i /= P)
                                       tmp += i / P;
57
        for(ll i = n; i ; i /= P)
                                       tmp -= i / P;
58
        for(ll i = m - n; i ; i /= P) tmp -= i / P;
59
        return f1 * INV(f2, pk) % pk * INV(f3, pk) * quick_pow(P, tmp, pk) % pk;
60
61 }
62
63 ll p[N], a[N];
  int cnt;
64
65
66
   11 CRT() {
67
        ll M = 1, ans = 0;
        for(int i = 1;i <= cnt; i++) M *= p[i];</pre>
68
        for(int i = 1;i <= cnt; i++) {</pre>
69
            ll m = M / p[i];
70
            ans = (ans + a[i] * m % M * INV(m, p[i]) % M) % M;
71
72
73
        return (ans % M + M) % M;
   }
74
75
   ll EX_Lucas(ll m, ll n, ll P) {
76
        for(int i = 2;i * i <= P; i++) {
77
            if(P % i == 0) {
78
79
                ll tmp = 1;
80
                while(P % i == 0) {
                    tmp *= i;
81
82
                    P /= i;
83
                }
                p[++cnt] = tmp;
84
                a[cnt] = C(m, n, i, tmp);
85
86
            }
87
88
        if(P > 1) {
89
            p[++cnt] = P;
            a[cnt] = C(m, n, P, P);
90
91
92
        return CRT();
93
   }
94
   int main() {
95
        ll m, n, P;
96
        cin >> m >> n >> P;
97
        cnt = 0;
        cout << EX_Lucas(m, n, P) << endl;</pre>
98
99 }
```

## $6.22 \quad min25$

```
1 typedef long long ll;
2
3
   const int N = 1e5 + 10;
5
   namespace Min25 {
6
        int prime[N], id1[N], id2[N], flag[N], ncnt, m;
7
8
        ll g[N], sum[N], a[N], T, n;
9
10
        inline int ID(ll x) {
11
            return x \ll T? id1[x]: id2[n / x];
12
13
        }
14
15
        inline ll calc(ll x) {
            return x * (x + 1) / 2 - 1;
16
17
18
        inline ll f(ll x) {
19
20
            return x;
        }
21
22
        inline void init() {
23
            ncnt = 0, m = 0;
24
            T = sqrt(n + 0.5);
25
            for (int i = 2; i <= T; i++) {
26
27
                if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
28
                for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {</pre>
                     flag[i * prime[j]] = 1;
29
                     if (i % prime[j] == 0) break;
30
                }
31
32
            for (ll l = 1; l \ll n; l = n / (n / l) + 1) {
33
                a[++m] = n / 1;
34
                if (a[m] \leftarrow T) id1[a[m]] = m; else id2[n / a[m]] = m;
35
                q[m] = calc(a[m]);
36
37
            for (int i = 1; i <= ncnt; i++)</pre>
38
                for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)</pre>
39
40
                     g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] - sum[i - 1]);
41
        }
42
        inline ll Solve(ll x) {
43
44
            if (x \le 1) return x;
45
            return n = x, init(), g[ID(n)];
        }
46
47
48 }
   6.23 BM
1 typedef long long ll;
2 const ll mod = 1e9 + 7;
4 typedef vector<ll> VI;
5
```

```
ll quick_pow(ll a, ll b);
   namespace linear_seq {
8
        const ll N = 1e5 + 10;
9
        11 res[N], base[N], _c[N], _md[N];
10
11
        vector<ll> Md;
12
        void mul(ll *a, ll *b, ll k) {
13
            for (ll i = 0; i < 2 * k; i++)
14
                 _c[i] = 0;
15
             for (l\bar{l}\ \bar{i} = 0;\ i < k;\ i++) {
16
17
                 if (a[i]) {
18
                     for (int j = 0; j < k; j++) {
                         _{c[i + j] = (_{c[i + j] + a[i]} * b[j]) \% mod;}
19
                     }
20
                 }
21
22
            for (ll i = 2 * k - 1; i >= k; i--) {
23
24
                 if (_c[i]) {
25
                     for (ll j = 0; j < Md.size(); j++) {</pre>
                          _{c[i - k + Md[j]]} = (_{c[i - k + Md[j]]} - _{c[i]} * _{md[Md[j]]}) % mod;
26
27
                 }
28
29
30
            for (ll i = 0; i < k; i++)
31
                 a[i] = _c[i];
        }
32
33
        ll solve(ll n, VI a, VI b) {
34
            // a 系数 b 初值 b[n + 1] = a[0] * b[n] + ...
35
36
            // cout << b.size() << endl;
            ll ans = 0, pnt = 0;
37
            ll k = a.size();
38
            assert(a.size() == b.size());
39
             for (ll i = 0; i < k; i++)
40
                 _{md[k - i - 1] = -a[i];}
41
42
             _{md}[k] = 1;
            Md.clear();
43
            for (ll i = 0; i < k; i++) {
44
                 if (_md[i] != 0)
45
                     Md.push_back(i);
46
47
            for (ll i = 0; i < k; i++)
48
49
                 res[i] = base[i] = 0;
            res[0] = 1;
50
51
            while ((1ll << pnt) <= n)</pre>
52
                 pnt++;
            for (ll p = pnt; p >= 0; p--) {
53
54
                 mul(res, res, k);
                 if ((n >> p) & 1) {
55
56
                     for (ll i = k - 1; i >= 0; i--)
57
                          res[i + 1] = res[i];
58
                     res[0] = 0;
                     for (ll i = 0; i < Md.size(); i++)</pre>
59
                          res[Md[i]] = (res[Md[i]] - res[k] * _md[Md[i]]) % mod;
60
61
                 }
62
63
            for (ll i = 0; i < k; i++)
                 ans = (ans + res[i] * b[i]) % mod;
64
```

```
65
              return ans;
         }
66
67
         VI BM(VI s) {
68
             VI C(1, 1), B(1, 1);
69
70
              11 L = 0, m = 1, b = 1;
              for (ll n = 0; n < s.size(); n++) {</pre>
71
                  11 d = 0;
72
                  for (ll i = 0; i < L + 1; i++)
73
                       d = (d + (ll)C[i] * s[n - i]) % mod;
74
75
                  if (d == 0)
76
                       m++;
                  else if (2 * L <= n) {
77
                       VI T = C;
78
                       ll c = mod - d * quick_pow(b, mod - 2) % mod;
while (C.size() < B.size() + m)</pre>
79
80
81
                           C.push_back(0);
82
                       for (int i = 0; i < B.size(); i++)</pre>
                           C[i + m] = (C[i + m] + c * B[i]) \% mod;
83
                       L = n + 1 - L;
84
                       B = T;
85
                       b = d;
86
                       m = 1;
87
                  }
88
89
                  else {
                       ll\ c = mod - d * quick_pow(b, mod - 2) % mod;
90
                       while (C.size() < B.size() + m)</pre>
91
                           C.push_back(0);
92
                       for (ll i = 0; i < B.size(); i++)</pre>
93
                           C[i + m] = (C[i + m] + c * B[i]) \% mod;
94
95
                  }
96
97
              }
              return C;
98
         }
99
100
101
         ll gao(VI a, ll n) {
102
              VI c = BM(a);
              c.erase(c.begin());
103
104
              for (ll i = 0; i < c.size(); i++)</pre>
                  c[i] = (mod - c[i]) \% mod;
105
              return solve(n, c, VI(a.begin(), a.begin() + c.size()));
106
         }
107
108
    }
109
110 void solve() {
111
         int n;
         while (~scanf("%d", &n)) {
112
             VI \ V = VI\{1,2,4,7,13,24\};
113
114
                  printf("%d\n", linear_seq::gao(v, n - 1));
115
         }
116 }
    6.24 duSieve
    #include <bits/stdc++.h>
 3 using namespace std;
```

```
4
   typedef long long ll;
   const int N = 1e6 + 10;
6
 7
   unordered_map<int, ll> smu, sphi;
8
   bool isPrime[N];
9
10
  int prime[N], num;
  ll mu[N], phi[N];
11
12
   void makeMobiusAndEuler(int siz) {
13
        mu[1] = phi[1] = 1;
14
15
        for (int i = 2; i <= siz; i++) {
            if (!isPrime[i]) prime[++num] = i, mu[i] = -1, phi[i] = i - 1;
16
            for (int j = 1; j <= num && i * prime[j] <= siz; j++) {
17
                isPrime[i * prime[j]] = 1;
18
                if (i % prime[j] == 0) {
19
                    mu[i * prime[j]] = 0;
20
                    phi[i * prime[j]] = phi[i] * prime[j];
21
22
                    break;
                }
23
                else {
24
                    phi[i * prime[j]] = phi[prime[j]] * phi[i];
25
                    mu[i * prime[j]] = -mu[i];
26
27
                }
28
            }
29
        for (int i = 1; i <= siz; i++) mu[i] += mu[i - 1], phi[i] += phi[i - 1];
30
   }
31
32
   11 getSmu(int n) {
33
34
        if (n < N) return mu[n];</pre>
35
        if (smu[n]) return smu[n];
        ll res = 1;
36
        for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
37
            r = n / (n / 1);
38
            res -= 1ll * (r - l + 1) * getSmu(n / l);
39
40
41
        return smu[n] = res;
   }
42
43
   11 getSphi(int n) {
44
        if (n < N) return phi[n];</pre>
45
        if (sphi[n]) return sphi[n];
46
       ll res = 111 * n * (n + 1) / 2;
47
        for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
48
49
            r = n / (n / 1);
            res -= 1ll * (r - l + 1) * getSphi(n / l);
50
51
        return sphi[n] = res;
52
53 }
   6.25
          反演相关
1 /*
2 莫比乌斯反演
3 g[n] = \sum_{d \mid n} f[d]
   f[d] = \sum_{d \in A} f[d] * mu[n / d]
   二项式反演
```

```
g[n] = \sum_{i=1}^{n} C(n, i) * f[i]
  f[n] = \sum_{i=1}^{n} C(n, i) * g[i] * (-1)^{n - i}
   子集反演
  f(S) = \sum_{T \leq S} g(T)
  g(S) = \sum_{T \in T} f(T) * (-1) ^ {|S| - |T|}
11
   6.26 simpson
  // 求一个函数在一个区间上的数值积分
2
3
   double f(double x) { // 题目中要求的辛普森积分函数,这里简单写一下f(x)=x*x
       return x * x;
4
   }
5
6
   double Simpson(double a, double b) {
7
       double mid = (a + b) / 2.0;
8
       return (b - a) *(f(a) + f(b) + 4.0 * f(mid)) / 6.0;
9
10
   }
11
   double DFS(double a, double b, double eps)
12
13
       double mid = (a + b) / 2.0;
14
       double SA = Simpson(a, mid), SM = Simpson(a, b), SB = Simpson(mid, b);
15
       if(fabs(SA + SB - SM) \le 15.0 * eps)
16
           return SA + SB + (SA + SB - SM) / 15.0;
17
       return DFS(a, mid, eps / 2.0) + DFS(mid, b, eps / 2.0);
18
   }
19
20
  // 求一个函数在0~无穷的上的数值积分,若收敛输出答案,若发散输出orz
   6.27 Bell
1
2
   #include <bits/stdc++.h>
3
   using namespace std;
4
5 typedef long long ll;
6
7
   const int N = 20;
8
   11 S2[N][N];
9
   11 B[N];
10
11
   void Stirling2() {
12
       S2[0][0] = 1;
13
14
       for(int i = 1; i < N; i++) {
15
           for(int j = 1; j <= i; j++) {
16
               S2[i][j] = S2[i - 1][j - 1] + j * S2[i - 1][j];
17
           }
18
19
       }
20
21 }
22
23 // 根据第二类斯特林数
24
```

```
void Bell1() {
25
26
        for(int i = 0; i < N; i++) {
27
            for(int j = 0; j <= i; j++) {
28
                B[i] += S2[i][j];
29
30
        }
31
   }
32
33
   // Bell三角形递推
34
35
36
   ll b[N][N];
37
   void Bell2() {
38
        b[1][1] = 1;
39
        for(int i = 2;i < N; i++) {</pre>
40
            b[i][1] = b[i - 1][i - 1];
41
42
            for(int j = 2; j < N; j++) {
43
                b[i][j] = b[i][j - 1] + b[i - 1][j - 1];
44
            }
45
        }
46
  }
47
48
49 // 自身递推
50
51 ll fac[N];
52
   ll C(ll m, ll n) {
53
        return fac[m] / (fac[n] * fac[m - n]);
54
55
56
   void Bell3() {
57
58
        fac[1] = 1;
59
        for(int i = 2; i < N; i++)
60
61
            fac[i] = fac[i - 1] * i;
62
        B[0] = 1;
63
64
        for(int i = 1; i < N; i++) {
65
            for(int k = 0;k <= i; k++) {</pre>
66
                B[i] += C(i, k) * B[k];
67
68
            }
69
        }
70
  }
   6.28 Catalan
2 #include <bits/stdc++.h>
   using namespace std;
3
4
   typedef long long ll;
5
6
   const int N = 1e5 + 10;
7
9 int C[N];
```

```
10
   // 线性递推
11
12
   void Calc1() {
13
14
        C[0] = 1;
        for(int i = 1;i < N; i++) {</pre>
15
            C[i] = C[i - 1] * (4 * i - 2) / (i + 1);
16
17
   }
18
19
20
  // 组合数求解
21
22 int f[N];
23
   void fac() {
24
        f[0] = 1;
25
        for(int i = 1;i < N; i++) {</pre>
26
            f[i] = f[i - 1] * i;
27
28
        }
29
   }
30
   void Calc2(int n) {
31
        C[n] = f[2 * n] / f[n + 1];
32
33 }
34
35 // 多项式求解
36
   void Calc3(int n) {
37
        if(n == 1)
38
            C[n] = 1;
39
40
        for(int i = 1;i <= n; i++) {</pre>
41
            C[n] += C[n - i] * C[i - 1];
42
43
44 }
   6.29 Lucas
   // mod—定为质数
2
   namespace Comb {
3
        11 mod;
4
        const int N = 1e6 + 10;
5
        ll F[N], invF[N], inv[N];
6
7
        void init() {
8
            F[0] = F[1] = invF[0] = invF[1] = inv[0] = inv[1] = 1;
9
            for (int i = 2; i < N; i++) {
10
                F[i] = F[i - 1] * i % mod;
11
                inv[i] = (mod - mod / i) * inv[mod % i] % mod;
12
                invF[i] = invF[i - 1] * inv[i] % mod;
13
            }
14
        }
15
16
        ll C(ll m, ll n) {
17
            if (m < 0 \mid l \mid n < 0 \mid l \mid n > m) return 0;
18
19
            ll ans = F[m];
20
            ans = ans * invF[n] % mod;
```

```
ans = ans * invF[m - n] % mod;
21
22
             return ans;
        }
23
24
        ll Lucas(ll m, ll n) {
25
             return n ? Lucas(m / mod, n / mod) * C(m % mod, n % mod) % mod : 1;
26
27
28
   }
29
30
31 // Comb::Lucas(m, n)
   6.30
           伯努利数
   namespace BNL {
        const int N = 1e7 + 10, M = 1e6 + 10;
2
3
        struct Complex {
             double x, y;
4
             Complex(double a = 0, double b = 0): x(a), y(b) {}
5
            Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y);
6
         }
            Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y);
7
         }
            Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y,
8
         x * rhs.y + y * rhs.x; }
9
            Complex conj() { return Complex(x, -y); }
10
        } w[N];
11
12
        int tr[N];
        11 F[N], G[N];
13
14
        ll quick_pow(ll a, ll b, ll p) {
15
             ll ans = 1;
16
             while(b) {
17
                 if(b \& 1) ans = ans * a % p;
18
                 a = a * a % p;
19
20
                 b >>= 1;
21
22
             return ans % p;
23
24
        void FFT(Complex *A, int len) {
25
             for (int i = 0; i < len; i++) if(i < tr[i]) swap(A[i], A[tr[i]]); for (int i = 2, lyc = len >> 1; i <= len; i <<= 1, lyc >>= 1)
26
27
                 for (int j = 0; j < len; j += i) {
   Complex *l = A + j, *r = A + j + (i >> 1), *p = w;
28
29
                      for (int k = 0; k < i >> 1; k++) {
30
                          Complex tmp = *r * *p;
31
                          *r = *l - tmp, *l = *l + tmp;
32
                          ++1, ++r, p += lyc;
33
                     }
34
                 }
35
36
        }
37
        inline void MTT(ll *x, ll *y, ll *z, int n) {
38
             int len = 1; while (len <= n) len <<= 1;</pre>
39
             for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
40
        0);
```

```
for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
41
       (2 * PI * i / len));
42
            for (int i = 0; i < len; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
43
            static Complex a[N], b[N];
44
            static Complex dfta[N], dftb[N], dftc[N], dftd[N];
45
46
            for (int i = 0; i < len; i++) a[i] = Complex(x[i] & 32767, x[i] >> 15);
47
            for (int i = 0; i < len; i++) b[i] = Complex(y[i] & 32767, y[i] >> 15);
48
            FFT(a, len), FFT(b, len);
49
            for (int i = 0; i < len; i++) {
50
51
                int j = (len - i) & (len - 1);
                static Complex da, db, dc, dd;
52
                da = (a[i] + a[j].conj()) * Complex(0.5, 0);
53
                db = (a[i] - a[j].conj()) * Complex(0, -0.5);
54
                dc = (b[i] + b[j].conj()) * Complex(0.5, 0);
55
                dd = (b[i] - b[j].conj()) * Complex(0, -0.5);
56
                dfta[j] = da * dc;
57
                dftb[j] = da * dd;
58
                dftc[j] = db * dc;
59
                dftd[j] = db * dd;
60
61
            for (int i = 0; i < len; i++) a[i] = dfta[i] + dftb[i] * Complex(0, 1);
62
63
            for (int i = 0; i < len; i++) b[i] = dftc[i] + dftd[i] * Complex(0, 1);
64
            FFT(a, len), FFT(b, len);
            for (int i = 0; i < len; i++) {
65
                int da = (11)(a[i].x / len + 0.5) \% mod;
66
                int db = (ll)(a[i].y / len + 0.5) \% mod;
67
                int dc = (ll)(b[i].x / len + 0.5) \% mod;
68
                int dd = (ll)(b[i].y / len + 0.5) \% mod;
69
70
                z[i] = (da + ((ll)(db + dc) << 15) + ((ll)(dd << 30)) % mod;
           }
71
72
       }
73
       int getLen(int n) {
74
            int len = 1; while (len < (n << 1)) len <<= 1;
75
            for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
76
            for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
77
       (2 * PI * i / len));
78
            return len;
       }
79
80
       void Get_Inv(ll *f, ll *g, int n) {
81
            if(n == 1) { g[0] = quick_pow(f[0], mod - 2, mod); return ; }
82
            Get_Inv(f, g, (n + 1) >> 1);
83
            int len = getLen(n);
84
            static ll c[N];
85
            for(int i = 0; i < len; i++) c[i] = i < n ? f[i] : 0;
86
87
           MTT(c, g, c, len); MTT(c, g, c, len);
            for(int i = 0; i < n; i++) g[i] = (211 * g[i] - c[i] + mod) % mod;
88
89
            for(int i = n; i < len; i++) g[i] = 0;
90
            for(int i = 0; i < len; i++) c[i] = 0;
       }
91
92
       11 ff[N], invff[N], inv[N];
93
94
       11 B[N];
95
       ll C(ll m, ll n) {
96
```

```
if(m < 0 \mid l \mid n < 0 \mid l \mid n > m)
97
98
                 return 0;
             ll ans = ff[m];
99
             ans = ans * invff[n] % mod;
100
             ans = ans * invff[m - n] % mod;
101
102
             return ans;
        }
103
104
        void init(int m) {
105
             ff[0] = ff[1] = inv[0] = inv[1] = invff[0] = invff[1] = 1;
106
             for(int i = 2; i < M; i++)
107
108
             {
                 ff[i] = ff[i - 1] * i % mod;
109
                 inv[i] = mod - (mod / i) * inv[mod % i] % mod;
110
                 invff[i] = invff[i - 1] * inv[i] % mod;
111
             }
112
113
             for(int i = 0; i \le m + 10; i++) F[i] = invff[i + 1];
114
             Get_Inv(F, G, m + 10);
115
             for(int i = 0; i \le m + 10; i++) B[i] = G[i] * ff[i] % mod;
116
        }
117
118
        ll solve(ll n, int k) {
119
             init(k);
120
121
             ll ans = 0, prod = n \% \mod;
             for(int i = k; ~i ; i--) {
122
                 ans = (ans + prod * B[i] % mod * C(k + 1, i) % mod) % mod;
123
                 prod = prod * n % mod;
124
125
             ans = ans * quick_pow(k + 1, mod - 2, mod) % mod;
126
127
             return ans;
128
        }
129
    }
130
131 void solve() {
        ll n; int k; cin >> n >> k;
132
133
        cout << BNL::solve(n + 1, k) << endl;</pre>
134 }
    6.31 步移-组合数前缀和
    // S(n, m) = \sum_{i=0}^{mC(n, i)}
 2
 3 int x, y;
 4 ll s;
    11 S(int n, int m) {
 5
        while(y < m) (s = s + C(x, ++y)) %= mod;
        while(y > m) (s = s - C(x, y--)) %= mod;
 7
        while(x < n) (s = s * 2 - C(x++, y)) %= mod;
 8
        while(x > n) (s = (s + C(--x, y)) * inv2) %= mod;
 9
10
        return s;
11 }
    6.32
           康托展开
 1 #include <iostream>
 2 #include <vector>
```

```
#include <algorithm>
 4
   using namespace std;
5
   typedef long long ll;
7
   const int mod = 1e9 + 7;
8
9 \quad const int N = 1e5 + 10;
10
11 ll fac[N];
12 int a[N]; // 排列, 康托展开求解
13 int n;
14 ll x; // 逆康托展开求解
15
   void Get_F() {
16
        fac[0] = 1;
17
        for(int i = 1; i < N; i++)
18
            fac[i] = fac[i - 1] * i % mod;
19
  }
20
21
   11 CanTor() {
22
        11 \text{ ans} = 0;
23
24
        for(int i = 1;i <= n; i++) {</pre>
25
            ll\ smaller = 0;
26
            for(int j = i + 1; j <= n; j++) {
27
                if(a[j] < a[i])
                    smaller++;
28
29
            ans = (ans + fac[n - i] * smaller % mod) % mod;
30
31
32
        return ans + 1;
   }
33
34
   void DeCantor() {
35
        vector<int> v; // 存放当前可选数
36
37
        vector<int> a; // 所求的排列组合序
        for(int i = 1;i <= n; i++) {</pre>
38
39
            v.push_back(i);
40
        for(int i = n;i >= 1; i--) {
41
            int r = x % fac[i - 1];
42
            int t = x / fac[i - 1];
43
            x = r;
44
            sort(v.begin(), v.end());
45
46
            a.push_back(v[t]);
            v.erase(v.begin() + t);
47
48
49
        for(int i = 0;i < a.size(); i++)</pre>
            cout << a[i] << " ";
50
        cout << endl;</pre>
51
52 }
53
54 // 线段树优化
55
   const int N = 1000010;
56
57
58 ll fac[N];
   int a[N]; // 排列, 康托展开求解
60
  int n;
61
```

```
struct SegmentTree {
         int ls, rs;
63
         int sum;
64
65
    }t[N << 2];
66
67
    int cnt, root;
68
    void push_up(int u) {
69
         t[u].sum = (t[lc].sum + t[rc].sum) % mod;
70
71
72
73
    void build(int &u, int l, int r) {
74
         if(!u) u = ++cnt;
75
         if(l == r) {
             t[u].sum = 1;
76
77
             return ;
78
         build(lc, l, m);
79
80
         build(rc, m + 1, r);
81
         push_up(u);
    }
82
83
    void update(int &u, int 1, int r, int k) {
84
85
         if(!u) u = ++cnt;
86
         if(l == r) {
             t[u].sum = 0;
87
             return;
88
89
         if(k <= m) update(lc, l, m, k);</pre>
90
         else update(rc, m + 1, r, k);
91
92
         push_up(u);
    }
93
94
    ll query(int u, int l, int r, int ql, int qr) {
95
96
         if(ql > qr) return 0;
         if(ql == 1 \&\& qr == r) {
97
98
             return t[u].sum;
99
         if(qr <= m) return query(lc, l, m, ql, qr) % mod;</pre>
100
         else if(ql > m) return query(rc, m + 1, r, ql, qr) % mod;
101
         else return (query(lc, l, m, ql, m) + query(rc, m + 1, r, m + 1, qr)) % mod;
102
103
104
105
    void Get_F() {
         fac[0] = 1;
106
         for(int i = 1;i < N; i++)</pre>
107
108
             fac[i] = fac[i - 1] * i % mod;
109 }
110
111 void solve()
112
    {
113
         Get_F();
114
         cin >> n;
         build(root, 1, n);
115
         11 \text{ ans} = 0;
116
117
         for(int i = 1; i <= n; i++) {
             cin >> a[i];
118
119
             update(root, 1, n, a[i]);
             ans = (ans + query(root, 1, n, 1, a[i] - 1) * fac[n - i]) % mod;
120
```

```
121 }
122 cout << (ans + 1) % mod << endl;
123 }
```

## 6.33 模数非质数的组合

```
1 // 模数非质数情况下的组合问题
^{2} // one way, use CRT merge ans
  // https://ac.nowcoder.com/discuss/655940?type=101&order=0&pos=2&page=1&channel=-1&
       source_id=discuss_tag_nctrack
   // another way
   // https://ac.nowcoder.com/acm/contest/view-submission?submissionId=47754622
6
7
   #include <bits/stdc++.h>
8
9
   using namespace std;
   typedef long long 11;
10
   const int N = 1e6 + 10;
11
12
   ll qpow(ll a, ll b, ll mod) {
13
       ll res = 1;
14
       while (b) {
15
           if (b & 1) res = res * a % mod;
16
           a = a * a % mod;
17
           b >>= 1;
18
19
       }
20
       return res;
21
   }
22
   ll exgcd(ll a, ll b, ll &x, ll &y) {
23
24
       if (!b) {
           x = 1, y = 0;
25
26
           return a;
27
       ll res = exgcd(b, a \% b, x, y);
28
29
       11 t = y;
30
       y = x - a / b * y;
       x = t;
31
32
       return res;
33
   }
34
   ll inv(ll a, ll b) {
35
36
       11 x = 0, y = 0;
37
       exgcd(a, b, x, y);
38
       return x = (x \% b + b) \% b;
  }
39
40
  //r□为余数,m为模数,其中模数互质
42 //M = pi(mi), Mi = M / mi, invMi = Mi % mi
  //ni满足是除了mi之外的倍数,且模mi为ri
  //利用逆元性质, 即ri * Mi * invMi = ri (mod mi)
   //res = (sigma(ri * Mi * invMi)) % M
45
46
47
   ll china(ll r[], ll m[], int n) {
       ll M = 1, res = 0;
48
       for (int i = 1; i <= n; i++) M *= m[i];</pre>
49
       for (int i = 1; i <= n; i++) {
50
           ll Mi = M / m[i], invMi = inv(Mi, m[i]);
51
```

```
res = (res + r[i] * Mi % M * invMi % M) % M;
52
            //res = (res + mul(mul(r[i], Mi, M), invMi, M)) % M;按位乘
53
54
55
        return (res % M + M) % M;
56
    }
57
    int f[N], g[N], F[N], G[N], invF[N];
58
59
    int calc(int n, int p, int k) {
60
        11 \mod = \text{qpow}(p, k, LONG\_LONG\_MAX);
61
62
        F[0] = 1, G[0] = 0;
        for (int i = 1; i <= n; i++) {
63
             g[i] = 0, f[i] = i;
64
             while (f[i] \% p == 0) f[i] /= p, g[i]++;
65
             F[i] = 111 * F[i - 1] * f[i] % mod;
66
            G[i] = G[i - 1] + g[i];
67
68
        invF[n] = inv(F[n], mod);
69
        for (int i = n; i >= 1; i--) invF[i - 1] = 1ll * invF[i] * f[i] % mod;
70
        int ans = 0;
71
        for (int i = 0; i \le n / 2; i++) {
72
             int t = 1|| * F[n] * invF[n - 2 * i] % mod * invF[i] % mod * invF[i] % mod *
73
                     qpow(p, G[n] - G[n - 2 * i] - 2 * G[i], LONG_LONG_MAX) % mod;
74
75
            ans = (ans + 1ll * t) % mod;
76
77
        return ans;
    }
78
79
    ll r[20], m[20];
80
81
82
    int main() {
    #ifdef ACM_LOCAL
83
        freopen("input.in", "r", stdin);
84
        freopen("output.out", "w", stdout);
85
    #endif
86
87
        int n, p;
88
        scanf("%d%d", &n, &p);
89
        int num = 0;
        for (int i = 2; i * i <= p; i++)
90
             if (p \% i == 0) {
91
                 int k = 0;
92
                 m[++num] = 1;
93
                 while (p % i == 0) p /= i, k++, m[num] *= i;
94
                 r[num] = calc(n, i, k);
95
            }
96
        if (p > 1) {
97
            m[++num] = p;
98
             r[num] = calc(n, p, 1);
99
100
101
        printf("%lld\n", china(r, m, num));
102
103
        return 0;
104 }
    6.34 k 次最小置换复原
```

2 void solve() {

```
int n; cin >> n;
3
        vector<int> a(n + 1), vis(n + 1);
for(int i = 1;i <= n; i++) cin >> a[i];
4
5
        ll\ ans = 1;
6
        for(int i = 1;i <= n; i++) {</pre>
7
            if(!vis[i]) {
8
9
                 int cnt = 0;
                 int x = i;
10
                 while(!vis[x]) {
11
12
                     vis[x] = 1;
13
                     cnt++;
14
                     x = a[x];
                 }
15
                 ans = lcm(ans, cnt);
16
            }
17
18
19
        cout << ans << endl;</pre>
20
   6.35
           Striling
   typedef long long ll;
   const int N = 20;
3
   // 第一类斯特林数
4
   ll S1[N][N];
5
   void Stirling1() {
6
        S1[0][0] = 1;
7
        for(int i = 1; i < N; i++) {
8
            for(int j = 1; j <= i; j++) {</pre>
9
                 S1[i][j] = S1[i - 1][j - 1] + (i - 1) * S1[i - 1][j];
10
11
        }
12
   }
13
14
   // 第二类斯特林数
15
   ll S2[N][N];
16
   void Stirling2() {
17
        S2[0][0] = 1;
18
        for(int i = 1;i < N; i++) {</pre>
19
            for(int j = 1; j <= i; j++) {</pre>
20
                 S2[i][j] = S2[i - 1][j - 1] + j * S2[i - 1][j];
21
            }
22
23
        }
24 }
           第二类斯特林数-行
   6.36
1 // {n,m}->n个不同元素划分成m个相同的集合中(不能有空集)的方案数。
3 // \{n,m\} = \{n-1,m-1\} + k\{n-1,m\}
4
   // \{n,m\}=\sum_{i=0}^n \frac{i}{i!} * {(-1)^{m-i}}{(m-i)!}
5
6
   const int N = 1e6 + 10;
   const ll mod = 167772161;
8
```

```
11
   void init();
12
13 ll qpow(ll a, ll b, ll mod);
14
15 const ll G = 3;
16 const ll invG = qpow(G, mod - 2, mod);
17 int tr[N];
18
   void NTT(ll *A, int len, int type);
   void mul(ll *a, ll *b, int n);
21
22 ll a[N], b[N];
23
   void solve() {
24
25
        init();
26
        int n; cin >> n;
27
        for(int i = 0;i <= n; i++) {
            a[i] = qpow(i, n, mod) * invF[i] % mod;
28
            if(i \& 1) b[i] = mod - invF[i];
29
            else b[i] = invF[i];
30
31
        mul(a, b, 2 * n);
32
33
        for(int i = 0; i <= n; i++) cout << a[i] << (i == n ? endl : " ");
34 }
   6.37 第二类斯特林数-列
   // 把n个不同元素划分成m个相同的集合(不能有空集)的方案数。
 3
   // k!\sum_{i=0}\frac{i_{k}x^i}{i!}=(e^x-1)^k
 4
 5 \text{ const int N} = 6e5 + 10;
   const ll mod = 167772161;
 6
 8 ll quick_pow(ll a, ll b);
 9
10 const ll G = 3;
   const ll invG = quick_pow(G, mod - 2);
11
12
13 int tr[N];
14 bool flag;
15
void NTT(ll *A, int len, int type);
void mul(ll *a, ll *b, int len);
void Get_Der(ll *f, ll *g, int len);
void Get_Tn+(ll *f | ll *g, int len);
19 void Get_Int(ll *f, ll *g, int len);
20 void Get_Inv(ll *f, ll *g, int n);
   void Get_Ln(ll *f, ll *g, int n);
void Get_Exp(ll *f, ll *g, int n)
23 void Get_Pow(ll *f, ll *g, int n, ll k1, ll k2);
24
25 ll a[N], ans[N];
26
27 ll F[N], invF[N], inv[N];
28
   void init();
29
30
```

```
void solve() {
31
       init();
32
       int n; ll k; cin >> n >> k; n++;
33
       if(k >= mod) flag = 1;
34
35
       for(int i = 1;i < n; i++) a[i] = invF[i];</pre>
       Get_Pow(a, ans, n, k % mod, k % (mod - 1));
36
       for(int i = 0;i < n; i++) {</pre>
37
            cout << ans[i] * invF[k] % mod * F[i] % mod << (i == n - 1 ? endl : " ");
38
39
40 }
          第一类斯特林数-行
   6.38
   #include <algorithm>
   #include <cstdio>
3
   #include <cstring>
   typedef long long LL;
5
   const int N = 550050;
6
7
   const int mod = 167772161;
8
  LL pow_mod(LL a, LL b) {
9
     LL ans = 1;
10
      for (; b; b >>= 1, a = a * a % mod)
11
12
       if (b \& 1) ans = ans * a % mod;
13
     return ans;
   }
14
15
  int L, rev[N];
   LL w[N], inv[N], fac[N], ifac[N];
18
  void Init(int n) {
19
20
     L = 1;
21
     while (L <= n) L <<= 1;
      for (int i = 1; i < L; ++i)
23
       rev[i] = (rev[i >> 1] >> 1) | ((i & 1) * L / 2);
24
     LL wn = pow_mod(3, (mod - 1) / L);
     W[L \gg 1] = 1;
     for (int i = L \gg 1; i < L; ++i) w[i + 1] = w[i] * wn % mod;
27
      for (int i = (L \gg 1) - 1; i; --i) w[i] = w[i << 1];
28
   }
29
   void DFT(LL *A, int len) {
30
     int k = __builtin_ctz(L) - __builtin_ctz(len);
31
32
      for (int i = 1; i < len; ++i) {
       int j = rev[i] \gg k;
33
34
       if (j > i) std::swap(A[i], A[j]);
35
      for (int h = 1; h < len; h <<= 1)
36
       for (int i = 0; i < len; i += (h << 1))
37
38
          for (int j = 0; j < h; ++j) {
            LL t = A[i + j + h] * w[j + h] % mod;
39
40
            A[i + j + h] = A[i + j] - t;
41
            A[i + j] += t;
42
43
      for (int i = 0; i < len; ++i) A[i] %= mod;
44
45
```

```
46 void IDFT(LL *A, int len) {
      std::reverse(A + 1, A + len);
47
      DFT(A, len);
48
      int v = mod - (mod - 1) / len;
49
      for (int i = 0; i < len; ++i) A[i] = A[i] * v % mod;
50
51
52
    void offset(const LL *f, int n, LL c, LL *g) {
53
      // g(x) = f(x + c)
54
      // g[i] = 1/i! sum_{j=i}^n j!f[j] c^{(j-i)}(j-i)!
55
      static LL tA[N], tB[N];
56
57
      int l = 1; while (l <= n + n) l <<= 1;
      for (int i = 0; i < n; ++i) tA[n - i - 1] = f[i] * fac[i] % mod;
58
      LL pc = 1;
59
      for (int i = 0; i < n; ++i, pc = pc * c % mod) tB[i] = pc * ifac[i] % mod;</pre>
60
      for (int i = n; i < l; ++i) tA[i] = tB[i] = 0;
61
      DFT(tA, 1); DFT(tB, 1);
62
      for (int i = 0; i < l; ++i) tA[i] = tA[i] * tB[i] % mod;</pre>
63
      IDFT(tA, 1);
64
      for (int i = 0; i < n; ++i)
65
        g[i] = tA[n - i - 1] * ifac[i] % mod;
66
   }
67
68
    void Solve(int n, LL *f) {
70
      if (n == 0) return void(f[0] = 1);
      static LL tA[N], tB[N];
71
72
      int m = n / 2;
      Solve(m, f);
73
      int l = 1; while (l <= n) l <<= 1;
74
      offset(f, m + 1, m, tA);
75
      for (int i = 0; i <= m; ++i) tB[i] = f[i];</pre>
 76
      for (int i = m + 1; i < l; ++i) tA[i] = tB[i] = 0;
77
78
      DFT(tA, 1); DFT(tB, 1);
      for (int i = 0; i < l; ++i) tA[i] = tA[i] * tB[i] % mod;</pre>
79
80
      IDFT(tA, 1);
      if (n & 1)
81
82
        for (int i = 0; i <= n; ++i)
83
           f[i] = ((i ? tA[i - 1] : 0) + (n - 1) * tA[i]) % mod;
84
        for (int i = 0; i <= n; ++i)
85
           f[i] = tA[i];
86
    }
87
88
89 LL f[N];
90
    int main() {
91
92
      int n;
      scanf("%d", &n);
93
      Init(n * 2);
94
      inv[1] = 1;
96
      for (int i = 2; i <= n; ++i) inv[i] = -(mod / i) * inv[mod % i] % mod;
97
      fac[0] = ifac[0] = 1;
98
      for (int i = 1; i <= n; ++i) {
        fac[i] = fac[i - 1] * i % mod;
99
        ifac[i] = ifac[i - 1] * inv[i] % mod;
100
101
      Solve(n, f);
102
103
      for (int i = 0; i <= n; ++i)
104
        printf("%lld ", (f[i] + mod) % mod);
```

```
105 return 0;
106 }
```

## 6.39 第一类斯特林数-列

```
#include <bits/stdc++.h>
1
2
   using namespace std;
3
4 #define Int register int
5 #define mod 167772161
6 #define MAXN 531072
7
   #define Gi 3
   int quick_pow (int a,int b,int c)
9
10
   {
11
        int res = 1;
12
        while (b){
            if (b & 1) res = 1ll * res * a % c;
13
            a = 111 * a * a % c;
14
            b >>= 1;
15
16
17
        return res;
   }
18
19
20 int limit = 1,l,r[MAXN];
21
22
   void NTT (int *a,int type)
23
        for (Int i = 0; i < limit; ++ i) if (i < r[i]) swap (a[i], a[r[i]]);
24
25
        for (Int mid = 1;mid < limit;mid <<= 1){</pre>
26
            int Wn = quick_pow (Gi,(mod - 1) / (mid << 1),mod);</pre>
27
            if (type == -1) Wn = quick_pow (Wn,mod - 2,mod);
28
            for (Int R = mid << 1, j = 0; j < limit; j += R){
                for (Int k = 0, w = 1; k < mid; ++ k, w = 111 * w * Wn % mod)
29
30
                     int x = a[j + k], y = 111 * w * a[j + k + mid] % mod;
31
32
                     a[j + k] = (x + y) \% \mod a[j + k + mid] = (x + mod - y) \% \mod a[j + k]
33
                }
34
            }
35
        if (type == 1) return ;
36
        int Inv = quick_pow (limit, mod - 2, mod);
37
        for (Int i = 0; i < limit; ++ i) a[i] = 1ll * a[i] * Inv % mod;
38
39
   }
40
   int c[MAXN];
41
42
   void Solve (int len,int *a,int *b)
43
44
   {
        if (len == 1) return b[0] = quick_pow (a[0],mod - 2,mod),void ();
45
        Solve ((len + 1) >> 1,a,b);
46
        limit = 1, l = 0;
47
        while (limit < (len << 1)) limit <<= 1,l ++;</pre>
48
        for (Int i = 0; i < limit; ++ i) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (l - 1));
49
50
        for (Int i = 0; i < len; ++ i) c[i] = a[i];
        for (Int i = len; i < limit; ++ i) c[i] = 0;
51
        NTT (c,1); NTT (b,1);
52
```

```
for (Int i = 0;i < limit;++ i) b[i] = 111 * b[i] * (2 + mod - 111 * c[i] * b[i] %
53
        mod) % mod;
        NTT (b,-1);
54
        for (Int i = len; i < limit; ++ i) b[i] = 0;
55
56
57
58
    void deravitive (int *a,int n){
         for (Int i = 1; i \le n; ++ i) a[i - 1] = 111 * a[i] * i % mod;
59
60
        a[n] = 0;
61
    }
62
63
    void inter (int *a,int n){
        for (Int i = n; i >= 1; -- i) a[i] = 1ll * a[i - 1] * quick_pow (i, mod - 2, mod) % mod
64
        a[0] = 0;
65
66
67
68 int b[MAXN];
69
70 void Ln (int *a,int n){
        memset (b,0,sizeof (b));
71
        Solve (n,a,b); deravitive (a,n);
72
        while (limit <= n) limit <<= 1,l ++;</pre>
73
74
        for (Int i = 0; i < limit; ++ i) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (l - 1));
75
        NTT (a,1), NTT (b,1);
        for (Int i = 0; i < limit; ++ i) a[i] = 1ll * a[i] * b[i] % mod;
76
        NTT (a,-1);
77
        inter (a,n);
78
        for (Int i = n + 1; i < limit; ++ i) a[i] = 0;
79
    }
80
81
    int F0[MAXN];
82
83
    void Exp (int *a,int *B,int n){
84
        if (n == 1) return B[0] = 1, void ();
85
        Exp (a,B,(n + 1) >> 1);
86
87
        for (Int i = 0; i < limit; ++ i) FO[i] = B[i];
88
        Ln (F0,n);
        F0[0] = (a[0] + 1 + mod - F0[0]) \% mod;
89
        for (Int i = 1; i < n; ++ i) F0[i] = (a[i] + mod - F0[i]) % mod;
90
        NTT (F0,1); NTT (B,1);
91
        for (Int i = 0;i < limit;++ i) B[i] = 1ll * F0[i] * B[i] % mod;</pre>
92
        NTT (B,-1);
93
94
        for (Int i = n; i < limit; ++ i) B[i] = 0;
95 }
96
97 int read ()
98
        int x = 0; char c = getchar(); int f = 1;
99
        while (c < '0' | | c > '9'){if (c == '-') f = -f;c = getchar();}
100
        while (c \ge 0') & c \le 9') x = (int)((int)(x << 3)  mod + (int)(x << 1)  mod +
101
        c - '0') % mod;c = getchar();}
102
        return x * f;
103 }
104
105
    void write (int x)
106
    {
107
        if (x < 0)\{x = -x; putchar ('-');\}
        if (x > 9) write (x / 10);
108
```

```
putchar (x \% 10 + '0');
109
110 }
111
112
    int n,k;
    int fac[MAXN],A[MAXN],B[MAXN];
113
114
115 signed main()
116 {
        n = read(), k = read();
117
        for (Int i = 0; i < n; ++ i) A[i] = quick_pow (i + 1, mod - 2, mod);
118
119
        Ln (A,n);
        for (Int i = 0; i < n; ++ i) A[i] = 111 * A[i] * k % mod;
120
        Exp (A,B,n); fac [0] = 1;
121
        for (Int i = 1; i \le max(n,k); ++ i) fac[i] = 1ll * fac[i - 1] * i % mod;
122
        for (Int i = n; i >= k; -- i) B[i] = B[i - k];
123
        for (Int i = 0; i < k; ++ i) B[i] = 0; int Inv = quick_pow (fac[k], mod - 2, mod);
124
        for (Int i = 0; i \le n; ++ i) write (111 * B[i] * fac[i] % mod * Inv % mod), putchar (
125
        putchar ('\n');
126
        return 0;
127
128 }
           整数拆分多项式求逆
    6.40
   // NTT求法,任意模数复杂度较高
 3
   #include <bits/stdc++.h>
   using namespace std;
 4
   typedef long long ll;
    const double PI = acos(-1);
    const int N = 1e5 + 10;
 7
 8
 9
    struct Complex {
10
        double x, y;
        Complex(double a = 0, double b = 0): x(a), y(b) {}
11
        Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
12
        Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
13
        Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
14
         rhs.y + y * rhs.x); }
15
        Complex conj() { return Complex(x, -y); }
16
    } w[N];
17
18
    ll mod, g;
    int tr[N];
19
20
   ll F[N], G[N];
21
22 ll quick_pow(ll a, ll b);
23
   int getLen(int n);
24
25
   void NTT(ll *a, int len, int opt);
27
28
   void mul(ll *a, ll *b, ll *z, int n);
29
30 void Get_Inv(ll *f, ll *g, int n);
31
    void ChaiFen(ll *f, ll *a, int n) {
32
```

int len = getLen(n);

33

```
for(int i = 0;i < len; i++) {</pre>
34
           ll a = 1ll * i * (3 * i - 1) / 2; ll b = 1ll * i * (3 * i + 1) / 2;
35
           if(a > len && b > len) break;
36
37
           ll tmp;
           if(i \& 1) tmp = mod - 1;
38
           else tmp = 1;
39
           if(a < len) f[a] = tmp;
40
           if(b < len) f[b] = tmp;</pre>
41
       }
42
43
       Get_Inv(f, g, n);
44
   }
45
46
   int main() {
47
48
       int n;
       cin >> n;
49
50
       ChaiFen(F, G, n);
       for(int i = 1;i <= n; i++) cout << G[i] << endl;</pre>
51
52 }
         普通型母函数
   6.41
1 // 普通型母函数: (1+x^1+x^2+...) (1+x^2+x^4)(1+x^3+x^6..)(...)(...)... 类似整数拆分
2
3 // a_n=1,1,1,1... = \frac{1}{1-x}
4 // a_n=1,0,1,0... = \frac{1}{1-x^2}
5 // a_n=1,2,3,4... = \frac{1}{(1-x)^2}
  // a_n=C(m,n)
                 = (1+x)^m
6
  // a_n=C(m+n,n) = \frac{1}{(1-x)^{m+1}}
7
8
9 #include <bits/stdc++.h>
10 using namespace std;
11
12 typedef long long ll;
13
14 // 求解硬币等普通问题
16 const int N = 1e5 + 10;
17
18 int a[N]; // 权重为i的组合数, a[P]为答案
  int b[N]; // 辅助数组
19
  int P; // 需要被分解的数
20
   int k; // 物品个数
   int v[N]; // 每个物品的权重
  int n1[N]; // 对于每种物品起始的因子(所需要的每个物品最小个数),最小为0
24 int n2[N]; // 对于每种物品最终的因子(所需要的每个物品最大个数),最大为INF
25
26 // 模板一(标准)
27
   void Calc1() {
28
       memset(a, 0, sizeof(a));
29
       a[0] = 1;
30
31
32
       for(int i = 1;i <= k; i++) { // 枚举每个物品因子
           memset(b, 0, sizeof(b));
33
           for(int j = n1[i]; j <= n2[i] && j * v[i] <= P; j++) { // 每个物品从最小因子到最大因
34
      子循环,如果n2是无穷的,则j<=n2[i]可以删去
              for(int m = 0;m + j * v[i] <= P; m++) { // 循环a的每个项
35
```

```
b[m + j * v[i]] += a[m]; // 把结果加到对应项里, 有点dp的味道
36
              }
37
           }
38
          memcpy(a, b, sizeof(b));
39
40
       }
   }
41
42
   // 模板二(数据量大的时候可以用,快速)
43
44
   void Calc2() {
45
       memset(a, 0, sizeof(a));
46
47
       a[0] = 1;
       int last = 0:
48
       for(int i = 1; i <= k; i++) {
49
           int last2 = min(last + n2[i] * v[i], P);//计算下一次的last
50
          memset(b, 0, sizeof(int) * (last2 + 1));//只清空b[0..last2]
51
           for(int j = n1[i]; j \leftarrow n2[i] & j * v[i] \leftarrow last2; j++) //last2
52
              for(int m = 0; m <= last && m + j * v[i] <= last2; m++) //一个是last, 一个是
53
      last2
                  b[m + j * v[i]] += a[m];
54
          memcpy(a, b, sizeof(int) * (last2 + 1));//b赋值给a, 只赋值0..last2
55
           last = last2;//更新last
56
       }
57
58 }
   6.42
         指数型母函数
1
   // 需要借助e^x的泰勒展开,一般求解多重排列数,即有 种物品,已知每种物品的数量为 k1,k2,...,kn 个,求
      从中选出m件物品的排列数。
3
   // 对n个元素全排列,方案数为n!/(n1!n2!...nk!),对n个中的r个元素进行全排列,这里就用到了指数型母函
4
      数. 即G(x)=(1+x/1!+x^2/2!+...+x^k1/k1!)(1+x/1!+x^2/2!+...+x^k2/k2!)...(1+x/1!+x
      ^{2/2!+...+x^{kn/kn!}}
5
   // 化简得G(x)=a0 + a1*x+a2*x^2/2!+...+ap*x^p/p! (p = k1+k2+k3+...) ai为选出i个物品的排列
6
      方案数
7
   // 若题目有规定条件,比如需要物品i出现非0的偶数次,即原式为(x^2/2!+x^4/4!+...+x^ki/ki!)
8
9
10
   #include <bits/stdc++.h>
11
   using namespace std;
12
   typedef long double ld;
13
14
   double num[15]; // 每种物品的数量, 第i个物品有num[i]个
15
16
  double a[15], b[15];
17
18
  double f[120]; // 阶乘
19
20
  void fac()
21
22
23
       f[0] = 1;
       for(int i = 1;i <= 105; i++)
24
           f[i] = f[i - 1] * i;
25
   }
26
27
```

```
void Calc() {
28
29
        int n, m;
        cin >> n >> m;
30
        for(int i = 1; i <= n; i++)
31
32
            cin >> num[i];
33
        memset(a, 0, sizeof(a));
34
        memset(b, 0, sizeof(b));
35
36
        for(int i = 0;i <= num[1]; i++) {</pre>
37
38
            a[i] = 1.0 / f[i];
39
40
        for(int i = 2;i <= n; i++) {</pre>
41
            for(int j = 0; j <= m; j++) {
42
                for(int k = 0;k <= num[i] && j + k <= m; k++) {</pre>
43
                     b[j + k] += a[j] / f[k];
44
45
            }
46
47
            for(int j = 0; j <= m; j++) {
48
49
                a[j] = b[j];
                b[j] = 0;
50
51
            }
52
        }
53
        cout << a[m] * f[m] << endl;</pre>
54
   }
55
   6.43
          伯努利数求和
   namespace BNL {
1
2
        const int N = 1e7 + 10, M = 1e6 + 10;
3
        struct Complex {
            double x, y;
4
            Complex(double a = 0, double b = 0): x(a), y(b) {}
5
            Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y);
6
        }
7
            Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y);
        }
            Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y,
8
        x * rhs.y + y * rhs.x; }
9
            Complex conj() { return Complex(x, -y); }
10
        } w[N];
11
        int tr[N];
12
        11 F[N], G[N];
13
14
        ll quick_pow(ll a, ll b, ll p) {
15
            ll ans = 1;
16
            while(b) {
17
                if(b \& 1) ans = ans * a % p;
18
                a = a * a % p;
19
20
                b >>= 1;
            }
21
22
            return ans % p;
        }
23
24
```

```
void FFT(Complex *A, int len) {
25
            for (int i = 0; i < len; i++) if(i < tr[i]) swap(A[i], A[tr[i]]);
26
            for (int i = 2, lyc = len >> 1; i <= len; i <<= 1, lyc >>= 1)
27
                for (int j = 0; j < len; j += i) {
   Complex *l = A + j, *r = A + j + (i >> 1), *p = w;
28
29
                    for (int k = 0; k < i >> 1; k++) {
30
                         Complex tmp = *r * *p;
31
                         *r = *l - tmp, *l = *l + tmp;
32
33
                        ++1, ++r, p += lyc;
34
                    }
                }
35
36
        }
37
        inline void MTT(ll *x, ll *y, ll *z, int n) {
38
            int len = 1; while (len <= n) len <<= 1;</pre>
39
            for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
40
       0);
            for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
41
       (2 * PI * i / len));
42
            for (int i = 0; i < len; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
43
            static Complex a[N], b[N];
44
            static Complex dfta[N], dftb[N], dftc[N], dftd[N];
45
46
47
            for (int i = 0; i < len; i++) a[i] = Complex(x[i] & 32767, x[i] >> 15);
            for (int i = 0; i < len; i++) b[i] = Complex(y[i] & 32767, y[i] >> 15);
48
            FFT(a, len), FFT(b, len);
49
            for (int i = 0; i < len; i++) {</pre>
50
                int j = (len - i) & (len - 1);
51
                static Complex da, db, dc, dd;
52
                da = (a[i] + a[j].conj()) * Complex(0.5, 0);
53
                db = (a[i] - a[j].conj()) * Complex(0, -0.5);
54
                dc = (b[i] + b[j].conj()) * Complex(0.5, 0);
55
                dd = (b[i] - b[j].conj()) * Complex(0, -0.5);
56
                dfta[j] = da * dc;
57
                dftb[j] = da * dd;
58
                dftc[j] = db * dc;
59
60
                dftd[j] = db * dd;
61
            for (int i = 0; i < len; i++) a[i] = dfta[i] + dftb[i] * Complex(0, 1);
62
            for (int i = 0; i < len; i++) b[i] = dftc[i] + dftd[i] * Complex(0, 1);
63
            FFT(a, len), FFT(b, len);
64
            for (int i = 0; i < len; i++) {
65
                int da = (11)(a[i].x / len + 0.5) \% mod;
66
                int db = (11)(a[i].y / len + 0.5) \% mod;
67
                int dc = (11)(b[i].x / len + 0.5) \% mod;
68
69
                int dd = (11)(b[i].y / len + 0.5) \% mod;
                z[i] = (da + ((ll)(db + dc) << 15) + ((ll)(dd << 30)) % mod;
70
            }
71
72
        }
73
74
        int getLen(int n) {
75
            int len = 1; while (len < (n << 1)) len <<= 1;
            for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 :
76
            for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), sin
77
       (2 * PI * i / len));
78
            return len;
79
        }
```

```
80
         void Get_Inv(ll *f, ll *g, int n) {
81
             if(n == 1) { g[0] = quick_pow(f[0], mod - 2, mod); return ; }
82
             Get_Inv(f, g, (n + 1) >> 1);
83
             int len = getLen(n);
84
             static ll c[N];
85
             for(int i = 0; i < len; i++) c[i] = i < n ? f[i] : 0;
86
             MTT(c, g, c, len); MTT(c, g, c, len);
87
             for(int i = 0; i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
88
             for(int i = n; i < len; i++) g[i] = 0;
89
             for(int i = 0; i < len; i++) c[i] = 0;
90
91
         }
92
         11 ff[N], invff[N], inv[N];
93
         11 B[N];
94
95
         ll C(ll m, ll n) {
96
             if(m < 0 \mid l \mid n < 0 \mid l \mid n > m)
97
                 return 0;
98
             ll\ ans = ff[m];
99
             ans = ans * invff[n] % mod;
100
             ans = ans * invff[m - n] % mod;
101
             return ans;
102
         }
103
104
         void init(int m) {
105
             ff[0] = ff[1] = inv[0] = inv[1] = invff[0] = invff[1] = 1;
106
             for(int i = 2;i < M; i++)</pre>
107
108
                 ff[i] = ff[i - 1] * i % mod;
109
                 inv[i] = mod - (mod / i) * inv[mod % i] % mod;
110
                 invff[i] = invff[i - 1] * inv[i] % mod;
111
             }
112
113
             for(int i = 0; i \leftarrow m + 10; i++) F[i] = invff[i + 1];
114
             Get_Inv(F, G, m + 10);
115
116
             for(int i = 0; i <= m + 10; i++) B[i] = G[i] * ff[i] % mod;
117
         }
118
         ll solve(ll n, int k) {
119
             init(k);
120
             ll ans = 0, prod = n \% \mod;
121
             for(int i = k; ~i ; i--) {
122
123
                 ans = (ans + prod * B[i] % mod * C(k + 1, i) % mod) % mod;
                 prod = prod * n % mod;
124
125
126
             ans = ans * quick_pow(k + 1, mod - 2, mod) % mod;
127
             return ans;
         }
128
129 }
130
131
    void solve() {
132
         ll n; int k; cin >> n >> k;
133
         cout << BNL::solve(n + 1, k) << endl;</pre>
134
   }
```

## 6.44 fibonacciBigInteger

```
class Fib {
2
        int n;
        std::vector<std::vector<int>> f;
3
        Fib(int _n) : n(_n), f(_n, std::vector<int>(_n)) {
    f[1][1] = 1;
4
5
            f[2][1] = 1;
6
            int s, add = 0;
7
            for (int i = 3; i < _n; i++) {
8
                for (int j = 1; j < _n; j++) {
9
                    s = f[i - 2][j] + f[i - 1][j] + add;
10
                    f[i][j] = s % 10;
11
12
                    add = s / 10;
                }
13
            }
14
        }
15
16
        std::string get(const int m) {
17
            int k = n - 1;
18
            while (!f[m][k]) k--;
19
            std::string s = "";
20
            s.push_back(f[n][k] + '0');
21
22
            return s;
23
        }
24 };
          fibonacciCircular
1 // 斐波那契循环节
2
3 typedef long long ll;
4 typedef long double ld;
5 typedef pair<int, int> pdd;
6
7 #define INF 0x7f7f7f
8 #define mem(a,b) memset(a , b , sizeof(a))
9 #define FOR(i, x, n) for(int i = x;i <= n; i++)</pre>
10
11 const ll mod = 1e9 + 7;
12 const int maxn = 5e6 + 10;
13
14 bool is_prime[maxn];
```

15 ll prime[maxn];

p = 0;

void sieve() // 素数筛

mem(is\_prime, true);

if(is\_prime[i])

{

}

is\_prime[0] = is\_prime[1] = false;

for(int i = 2; i < maxn; i++)

prime[++p] = i;

for(int j = i + i; j < maxn; j += i)

is\_prime[j] = false;

16 int p;

17 18

19 {

 $\begin{array}{c} 20 \\ 21 \end{array}$ 

22

 $\frac{23}{24}$ 

 $25 \\ 26 \\ 27$ 

28 29

30

31

```
32
            }
        }
33
34 }
35
   ll gcd(ll a, ll b)
36
37
   {
        return b ? gcd(b, a % b) : a;
38
   }
39
40
   ll quick_pow(ll a, ll b)
41
42
43
        a \% = mod;
        ll\ ans = 1;
44
        11 \text{ base} = a;
45
        while(b)
46
47
            if(b&1)
48
49
            {
                ans = ans * base % mod;
50
51
            base = base * base % mod;
52
53
            b >>= 1;
54
55
        return ans % mod;
56 }
57
   ll num[maxn]; // 所有质数的循环节
58
   ll f[maxn]; // 斐波那契数列
59
60
   void Fib_Cyclic_node()
61
62
   {
63
        num[1] = 3;
        for(int i = 2;i <= p; i++) // 找每个素数的循环节num[1~p]
64
65
            f[1] = 1;
66
            f[2] = 2;
67
            int x = 3;
68
69
            while(true)
70
                f[x] = f[x - 1] + f[x - 2];
71
                f[x] %= prime[i];
72
                if(f[x] == 1 \&\& f[x - 1] == 0) // f[x] \% prime[i] == f[1]
73
74
75
                X++;
76
77
            num[i] = x;
78
        }
79
       11 n;
80
81
        cin >> n; // 如果是质数, 那循环节就是num[n]; 如果是合数, 那循环节就是n的素因子的最小公倍数
82
        ll\ ans = 1;
83
        11 x;
        for(int i = 1;i <= p; i++)</pre>
84
85
            if(n % prime[i] == 0)
86
87
88
                while(n % prime[i] == 0)
89
90
```

```
n /= prime[i];
91
92
                     X++;
                 }
93
94
             ll k = num[i] * quick_pow(prime[i], x - 1);
95
            ans = ans * k / gcd(ans, k); // 最小公倍数
96
97
        cout << ans << endl; // 最小循环节
98
99
    }
100
101 // 广义斐波那契循环节
102
103 // fib(n) = a * fib(n - 1) + b * fib(n - 2)
104 // fib(1) = c fib(2) = d
105 // 求f(n) mod p的循环节
   //c = a * a - 4b是模p的二次剩余时枚举p-1的因子,否则枚举(p+1)(p-1)的因子
106
107
108 typedef long long ll;
109 typedef long double ld;
110 typedef pair<int, int> pdd;
111
112 #define INF 0x7f7f7f
#define mem(a,b) memset(a , b , sizeof(a))
#define FOR(i, x, n) for(int i = x; i <= n; i++)
115
116 ll fac[2][505];
117 ll cnt, ct;
118
    ll pri[6] = {2, 3, 7, 109, 167, 500000003};
ll num[6] = {4, 2, 1, 2, 1, 1};
119
120
121
122 \quad const \ ll \ mod = 1e9 + 7;
123 const int maxn = 5e6 + 10;
124
125 struct Matrix{
        ll m[2][2];
126
127 };
128
129 Matrix A;
130
131 Matrix I = {1, 0, 0 , 1}; // 单位矩阵
132
133 Matrix multi(Matrix a, Matrix b) // 矩阵乘法
134
    {
        Matrix C;
135
        for(int i = 0; i < 2; i++)
136
137
             for(int j = 0; j < 2; j++)
138
139
                 C.m[i][j] = 0;
140
141
                 for(int k = 0; k < 2; k++)
142
                     C.m[i][j] = (C.m[i][j] + a.m[i][k] * b.m[k][j] % mod) % mod;
143
144
                 C.m[i][j] \% = mod;
145
146
             }
147
148
        return C;
149 }
```

```
150
             Matrix quick_Matrix(Matrix a, ll b) // 矩阵快速幂
151
152
                          Matrix ans = I, base = a;
153
154
                          while(b)
155
                          {
                                      if(b & 1)
156
                                      {
157
                                                   ans = multi(a, base);
158
159
160
                                      base = multi(base, base);
161
                                     b >>= 1;
162
163
                          return ans;
164
165
             ll quick_pow(ll a, ll b);
166
167
          ll legendre(ll a, ll p) // 勒让德符号 = {1, -1, 0}
168
169
             {
                          ll k = quick_pow(a, (p - 1) >> 1);
170
                          if(k == 1)
171
                                      return 1;
172
173
                          else
174
                                      return -1;
             }
175
176
             void DFS(int dept,ll product = 1)
177
178
                          if(dept == cnt)
179
180
                                       fac[1][ct++] = product;
181
                                      return;
182
183
                          for(int i=0; i<=num[dept]; i++)</pre>
184
185
186
                                      DFS(dept+1,product);
187
                                      product *= pri[dept];
188
                          }
             }
189
190
             bool Fib_node(Matrix a, ll n) // n是否为循环节
191
192
                          Matrix ans = quick_Matrix(a, n);
193
194
                          return (ans.m[0][0] == 1 \&\& ans.m[0][1] == 0 \&\& ans.m[1][0] == 0 \&\& ans.m[1][1] == 0 \&\& ans.m[1][1] == 0 \&\& ans.m[1][0] == 0 \&\& ans.m[1][1] == 0 \&\& ans.m[1][0] == 0
                         1); // 是否为单位矩阵I
195
             }
196
197 ll Fib_Cyclic_node(ll a, ll b, ll c, ll d) // 广义斐波那契循环节斐波那契循环节
198
             {
199
                          fac[0][0] = 1;
200
                          fac[0][1] = 2;
                          fac[0][3] = 500000003;
201
202
                          fac[0][3] = 1000000006;
                          11 c = a * a - 4 * b;
203
                          A.m[0][0] = a;
204
205
                          A.m[0][1] = b;
206
                          A.m[1][0] = 1;
                          A.m[1][1] = 0;
207
```

```
if(legendre(c, mod) == 1) // c是否为1e9+7的二次剩余
208
209
             for(int i = 0; i < 4; i++)
210
211
             {
                 if(Fib_node(A, fac[0][i]))
212
                     return fac[0][i];
213
             }
214
        }
215
        else
216
217
         {
             ct = 0;
218
219
             cnt = 6;
             DFS(0, 1);
220
             sort(fac[1], fac[1] + ct);
221
             for(int i = 0;i < ct; i++)</pre>
222
223
                 if(Fib_node(A, fac[1][i]))
224
225
                     return fac[1][i];
226
             }
227
        }
228
229 }
230
231 int main()
232  {
        ll a, b, c, d;
233
234
        cin >> a >> b >> c >> d;
235
        ll n = Fib_Cyclic_node(a, b, c, d); // 广义斐波那契循环节循环节长度
        cout << n << endl;</pre>
236
237 }
    6.46 Matrix
    template <typename T>
 1
 2
    struct Matrix {
 3
        const int n;
 4
        std::vector<std::vector<T>> mat;
 5
 6
        Matrix(int n = 0) : n(n), mat(n, std::vector<T>(n)) {}
        Matrix(const std::vector<std::vector<T>> &rhs) : n(n) {
 7
 8
            mat = std::move(rhs);
 9
 10
        Matrix(const Matrix<T> &rhs) : n(n) {
 11
            mat = std::move(rhs.mat);
12
        void identify() {
13
             for (int i = 0; i < n; i++) {
14
                 for (int j = 0; j < n; j++) {
15
                     mat[i][j] = (i == j);
16
                 }
17
             }
18
19
20
        T getVal(const int& i, const int& j) { return mat[i][j]; }
21
        T size() { return n; }
        const Matrix operator+(const Matrix& rhs) {
22
23
             Matrix ret(n);
             for (int i = 0; i < n; i++) {
24
                 for (int j = 0; j < n; j++) {
25
```

```
ret.mat[i][j] = mat[i][j] + rhs.mat[i][j];
26
27
            }
28
29
            return ret;
30
        const Matrix operator-(const Matrix& rhs) {
31
            Matrix ret(n);
32
            for (int i = 0; i < n; i++) {
33
                for (int j = 0; j < n; j++) {
34
                     ret[i][j] = mat[i][j] - rhs[i][j];
35
36
37
            }
38
            return ret;
        }
39
        const Matrix operator*(const Matrix &rhs) {
40
            Matrix ret(n);
41
            for (int i = 0; i < n; i++) {
42
                for (int j = 0; j < n; j++) {
43
                     for (int k = 0; k < n; k++) {
44
                         ret.mat[i][j] += mat[i][k] * rhs.mat[k][j];
45
                     }
46
                }
47
            }
48
            return ret;
49
50
        const Matrix operator+() { return *this; }
51
        const Matrix operator-() {
52
            Matrix ret(n);
53
            for (int i = 0; i < n; i++) {
54
55
                for (int j = 0; j < n; j++) {
                     ret.mat[i][j] = -mat[i][j];
56
57
58
            }
            return ret;
59
60
        Matrix & operator += (const Matrix & rhs) {
61
62
            for (int i = 0; i < n; i++) {
63
                for (int j = 0; j < n; j++) {
                    mat[i][j] += rhs.mat[i][j];
64
65
            }
66
            return *this;
67
68
69
        Matrix & operator -= (const Matrix & rhs) {
            for (int i = 0; i < n; i++) {
70
                for (int j = 0; j < n; j++) {
71
72
                    mat[i][j] -= rhs.mat[i][j];
73
            }
74
75
            return *this;
76
77
        const Matrix operator*=(const Matrix &rhs) {
78
            Matrix ret(n);
            for (int i = 0; i < n; i++) {
79
                for (int j = 0; j < n; j++) {
80
                     for (int k = 0; k < n; k++) {
81
                         ret.mat[i][j] += mat[i][k] * rhs.mat[k][j];
82
83
84
                }
```

```
85
             return ret;
86
87
88
    };
89
    template <typename T>
90
    Martix power(Matrix<T> a, T b) {
91
        Matrix<T> res(2);
92
        res.mat[0][0] = res.mat[1][1] = 1;
93
        for (; b /= 2; a *= a) {
94
             if (b % 2) {
95
96
                 res *= a;
97
        }
98
99
        return res;
100
    6.47
           拉格朗日插值求和
 1 namespace polysum {
    #define rep(i, a, n) for (int i=a;i<n;i++)</pre>
    #define per(i, a, n) for (int i=n-1; i>=a; i--)
        const int D = 1010000;///可能需要用到的最高次
 4
        ll a[D], f[D], g[D], p[D], p1[D], p2[D], b[D], h[D][2], C[D], num[D];
 5
 6
 7
        ll powmod(ll a, ll b) {
 8
            ll res = 1;
            a \% = mod;
 9
 10
            assert(b >= 0);
             for (; b; b >>= 1) {
11
12
                 if (b & 1)res = res * a % mod;
                 a = a * a % mod;
13
14
15
             return res;
        }
16
17
18
        ///函数用途: 给出数列的 (d+1) 项, 其中d为最高次方项
        ///求出数列的第n项,数组下标从0开始
19
20
        ll calcn(int d, ll *a, ll n) { /// a[0].. a[d] a[n]
21
             if (n <= d) return a[n];</pre>
            p1[0] = p2[0] = 1;
22
             rep(i, 0, d + 1) {
23
                 ll t = (n - i + mod) \% mod;
24
                 p1[i + 1] = p1[i] * t % mod;
25
26
             rep(i, 0, d + 1) {
27
                 11 t = (n - d + i + mod) \% mod;
28
                 p2[i + 1] = p2[i] * t % mod;
29
30
             11 \text{ ans} = 0;
31
32
             rep(i, 0, d + 1) {
                 ll t = g[i] * g[d - i] % mod * p1[i] % mod * p2[d - i] % mod * a[i] % mod;
33
34
                 if ((d - i) \& 1) ans = (ans - t + mod) \% mod;
35
                 else ans = (ans + t) \% mod;
36
37
             return ans;
        }
38
39
```

```
void init(int M) {///用到的最高次
40
            f[0] = f[1] = g[0] = g[1] = 1;
41
           rep(i, 2, M + 5) f[i] = f[i - 1] * i % mod;
42
           g[M + 4] = powmod(f[M + 4], mod - 2);
43
           per(i, 1, M + 4) g[i] = g[i + 1] * (i + 1) % mod; /// 费马小定理筛逆元
44
       }
45
46
       ///函数用途: 给出数列的 (m+1) 项, 其中m为最高次方
47
       ///求出数列的前 (n-1) 项的和 (从第0项开始)
48
       ll polysum(ll m, ll *a, ll n) { /// a[0].. a[m] \sum_{i=0}^{n-1} a[i]
49
           for (int i = 0; i \le m; i++) b[i] = a[i];
50
51
52
           ///前n项和, 其最高次幂加1
           b[m + 1] = calcn(m, b, m + 1);
53
           rep(i, 1, m + 2) b[i] = (b[i - 1] + b[i]) \% mod;
54
           return calcn(m + 1, b, n - 1);
55
       }
56
57
       ll qpolysum(ll R, ll n, ll *a, ll m) { /// a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
58
           if (R == 1) return polysum(n, a, m);
59
           a[m + 1] = calcn(m, a, m + 1);
60
           11 r = powmod(R, mod - 2), p3 = 0, p4 = 0, c, ans;
61
           h[0][0] = 0;
62
63
           h[0][1] = 1;
64
           rep(i, 1, m + 2) {
               h[i][0] = (h[i - 1][0] + a[i - 1]) * r % mod;
65
               h[i][1] = h[i - 1][1] * r % mod;
66
67
           rep(i, 0, m + 2) {
68
               ll t = g[i] * g[m + 1 - i] % mod;
69
               if (i & 1) p3 = ((p3 - h[i][0] * t) % mod + mod) % mod, p4 = ((p4 - h[i][1]
70
        * t) % mod + mod) % mod;
               else p3 = (p3 + h[i][0] * t) % mod, p4 = (p4 + h[i][1] * t) % mod;
71
72
           c = powmod(p4, mod - 2) * (mod - p3) % mod;
73
           rep(i, 0, m + 2) h[i][0] = (h[i][0] + h[i][1] * c) % mod;
74
           rep(i, 0, m + 2) C[i] = h[i][0];
75
           ans = (calcn(m, C, n) * powmod(R, n) - c) % mod;
76
           if (ans < 0) ans += mod;
77
78
           return ans;
79
       }
80
       ll solve(ll n, int k) {
81
82
           init(k + 10);
           for (int i = 0; i \le k + 1; i++) num[i] = powmod((i1) i + 1, k);
83
           ll ans = polysum(k + 1, num, n) \% mod;
84
           return ans;
85
       }
86
  }
87
89
   void solve() {
90
       11 n;
91
       int k;
92
       cin >> n >> k;
       cout << polysum::solve(n, k) << endl;</pre>
93
94 }
```

## 6.48 SolveLinearSystem

```
bool IsZero(double v) {
1
2
     return abs(v) < 1e-9;
3
4
   enum GAUSS_MODE {
5
     DEGREE, ABS
6
   };
7
8
   template <typename T>
9
   void GaussianElimination(std::vector<std::vector<T>>& a, int limit, GAUSS_MODE mode =
10
       ABS) {
        if (a.empty() || a[0].empty()) {
11
12
            return;
13
        int h = static_cast<int>(a.size());
14
        int w = static_cast<int>(a[0].size());
15
        for (int i = 0; i < h; i++) {
16
            assert(w == static_cast<int>(a[i].size()));
17
        }
18
        assert(limit <= w);</pre>
19
        std::vector<int> deg(h);
20
        for (int i = 0; i < h; i++) {
21
22
            for (int j = 0; j < w; j++) {
23
                deg[i] += !IsZero(a[i][j]);
24
            }
25
        }
26
        int r = 0;
27
        for (int c = 0; c < limit; c++) {
28
            int id = -1;
29
            for (int i = r; i < h; i++) {
                if (!IsZero(a[i][c]) && (id == -1 \mid \mid (mode == DEGREE && deg[i] < deg[id])
30
        || (mode == ABS \&\& abs(a[id][c]) < abs(a[i][c])))) 
                    id = i;
31
                }
32
33
            if (id == -1) {
34
35
            continue;
36
            }
            if (id > r) {
37
38
                std::swap(a[r], a[id]);
                std::swap(deg[r], deg[id]);
39
40
                for (int j = c; j < w; j++) {
                     a[id][j] = -a[id][j];
41
42
            }
43
            std::vector<int> nonzero;
44
            for (int j = c; j < w; j++) {
45
46
                if (!IsZero(a[r][j])) {
47
                     nonzero.push_back(j);
                }
48
49
            T inv_a = 1 / a[r][c];
50
51
            for (int i = r + 1; i < h; i++) {
                if (IsZero(a[i][c])) {
52
53
                     continue;
54
                T coeff = -a[i][c] * inv_a;
55
```

```
for (int j : nonzero) {
56
                      if (!IsZero(a[i][j])) deg[i]--;
57
                      a[i][j] += coeff * a[r][j];
58
                      if (!IsZero(a[i][j])) deg[i]++;
59
60
             }
61
62
             ++r;
63
        for (r = h - 1; r >= 0; r--) {
64
             for (int c = 0; c < limit; c++) {</pre>
65
                 if (!IsZero(a[r][c])) {
66
67
                      T inv_a = 1 / a[r][c];
                      for (int i = r - 1; i >= 0; i--) {
68
                          if (IsZero(a[i][c])) {
69
                              continue;
70
71
                          T = -a[i][c] * inv_a;
 72
73
                          for (int j = c; j < w; j++) {
                              a[i][j] += coeff * a[r][j];
74
75
76
                      break;
77
                 }
78
79
             }
80
        }
    }
81
82
    template <typename T>
83
    std::vector<T> SolveLinearSystem(std::vector<std::vector<T>> a, const std::vector<T>& b
        , int w) {
        int h = static_cast<int>(a.size());
85
86
        assert(h == static_cast<int>(b.size()));
87
        if (h > 0) {
             assert(w == static_cast<int>(a[0].size()));
88
89
         for (int i = 0; i < h; i++) {
90
             a[i].push_back(b[i]);
91
92
        GaussianElimination(a, w);
93
        std::vector<T> x(w, 0);
94
        for (int i = 0; i < h; i++) {
95
             for (int j = 0; j < w; j++) {
    // when IsZero(a[i][j]) is no solution</pre>
96
97
                 if (!IsZero(a[i][j])) {
98
                     x[j] = a[i][w] / a[i][j];
99
100
                      break;
101
                 }
102
             }
103
104
        return x;
105 }
    6.49
           矩阵求逆
 1 //原始矩阵A[0, n - 1][0, n - 1]
   //右边一个单位阵I, 在a[0, n - 1][n, (n << 1) - 1]
    //将左边A变成单位阵时,右边的I变为A^-1
 4
```

```
11 a[MAX][MAX];
   bool Gauss(int n) {
6
        for (int i = 0, r; i < n; i++) {
7
            r = i;
8
9
            for (int j = i + 1; j < n; j++)
10
                 if (a[j][i] > a[r][i]) r = j;
            if (r != i) swap(a[i], a[r]);
11
            if (!a[i][i]) return false;//无解
12
13
            ll inv = qpow(a[i][i], mod - 2);
14
             for (int k = 0; k < n; k++) {
15
16
                 if (k == i) continue;
                 ll t = a[k][i] * inv % mod;
17
                 for (int j = i; j < (n << 1); j++)
18
                     a[k][j] = (a[k][j] - t * a[i][j] % mod + mod) % mod;
19
20
            for (int j = 0; j < (n << 1); j++) a[i][j] = a[i][j] * inv % mod;
21
22
23
        return true;
   }
24
25
   int main() {
26
        scanf("%d", &n);
27
28
        for (int i = 0; i < n; i++) {
29
            a[i][i + n] = 1;
            for (int j = 0; j < n; j++)
30
                 scanf("%lld", &a[i][j]);
31
32
        if(Gauss(n)) {
33
            for(int i = 0;i < n; i++) {
   for(int j = n;j < n * 2; j++) {
      cout << a[i][j] << " ";</pre>
34
35
36
37
                 cout << endl;</pre>
38
            }
39
40
        else cout << "No Solution" << endl;</pre>
41
42
   }
   6.50 eraseLinearBasis
   // 离线删除操作,维护线性基中每个元素的最晚删除时间。
2
   #include <bits/stdc++.h>
3
4
   using namespace std;
5
   typedef long long ll;
6
   const int maxl = 60;
7
8
   struct LinearBasis {
9
        ll a[maxl + 10], tim[maxl + 10];
10
        int n, size; // 每个相同异或值有2^{n-size}个
11
12
        vector<ll> v;
13
        LinearBasis() {
14
            memset(a, 0, sizeof(a));
15
16
            size = n = 0;
            v.clear();
17
```

```
}
18
19
        void insert(ll x, ll t) {
20
21
            n++;
            for(int i = maxl;i >= 0; i--) {
22
                if(!(x >> i & 1)) continue;
23
                if(a[i]) {
24
                     if(t > tim[i]) swap(t, tim[i]), swap(x, a[i]);
25
26
                    x ^= a[i];
27
                }
28
                else {
                    ++size;
29
                     a[i] = x; tim[i] = t;
30
                     return;
31
                }
32
            }
33
        }
34
35
        void erase(ll t) {
36
            for(int i = maxl;i >= 0; i--) {
37
                if(tim[i] == t) {
38
                     a[i] = tim[i] = 0; --size;
39
40
                     return ;
41
                }
42
            }
        }
43
   };
44
45
   int main() {
46
47
        LinearBasis lb;
48
        int n, m; cin >> n >> m;
        vector<ll> opt(n + 10), a(n + 10), del(n + 10), pre(n + 10);
49
        for(int i = 1;i <= m; i++) {</pre>
50
            cin >> opt[i] >> a[i];
51
            if(opt[i] == 1) pre[a[i]] = i, del[i] = m + 1;
52
            else del[pre[a[i]]] = i;
53
54
55
        11 \text{ ans} = 0;
        for(int i = 1;i <= m; i++) {</pre>
56
            if(opt[i] == 1) lb.insert(a[i], del[i]);
57
            else lb.erase(i);
58
            ans ^= 111 << (lb.n - lb.size);
59
60
        cout << ans << endl;</pre>
61
62
   }
   6.51 simpleLinearBasis
   template<class Info>
   struct LinearBasis {
2
3
        const int n;
        int size;
4
        long long num;
5
6
        // 每个异或值都相同的个数都为2^n-r,所以不同的异或值有2^r个.
        const int maxl = 61;
7
        std::vector<long long> a, v;
8
        LinearBasis(int n) : n(n), size(0), a(maxl) {}
9
10
        LinearBasis(std::vector<Info> init) : LinearBasis(init.size()) {
```

```
auto insert = [&](long long t) {
11
                for (int i = maxl - 1; i >= 0; --i) {
12
                     if (!(t >> i & 1)) continue;
13
                     if (a[i]) t ^= a[i];
14
                     else {
15
16
                         ++size;
                         // Rebuild
17
                         for (int j = i - 1; j >= 0; --j) if (t >> j & 1) t ^= a[j];
18
                         for (int j = i + 1; j < maxl; ++j) if (a[j] >> i & 1) a[j] ^= t;
19
20
21
                         a[i] = t;
22
                         return;
                     }
23
                }
24
25
            };
            for(int i = 0;i < n; i++) insert(init[i]);</pre>
26
            auto basis = [&]() {
27
                for (int i = 0; i < maxl; ++i) if (a[i]) v.push_back(i);
28
29
            };
30
            basis();
            num = 1LL << size;</pre>
31
32
        }
33
34
        // 查询能否xor出x这个数
        bool find(long long x) {
35
            for(int i = maxl - 1; i >= 0; i--) {
36
                if(x >> i & 1) {
37
                     if(!a[i]) return false;
38
                     x ^= a[i];
39
40
41
            }
42
            return true;
        }
43
44
        // 查询异或最大值
45
        long long askmax() {
46
47
            long long ans = 0;
48
            for(int i = maxl - 1; i >= 0; i--) ans = max(ans, ans ^ a[i]);
49
            return ans;
        }
50
51
        // 查询异或最小值
52
        long long askmin() {
53
54
            if((int) v.size() < n) return 0;</pre>
            for(int i = 0;i < maxl; i++) if(a[i]) return a[i];</pre>
55
            return 0;
56
        }
57
58
        // 查询异或第k小
59
        long long askmink(long long x) {
60
61
            if(v.size() != n) x--;
62
            if(!x) return 0;
            if(x >= (1LL << v.size())) return -1;
63
            long long ans = 0;
64
            for(int i = 0;i < (int) v.size(); i++) {</pre>
65
66
                if(x >> i \& 1) ans ^= a[v[i]];
67
68
            return ans;
69
        }
```

```
70
        long long rank(long long x) {
71
72
            long long ret = 0;
            for (int i = 0; i < (int) v.size(); ++i) if (x >> v[i] & 1) ret += 1LL << i;
73
74
            return ret;
75
        }
  };
76
          intervalModifyLinearBasis
   #include<bits/stdc++.h>
2 #define maxn 200005
3 using namespace std;
4
   struct Base{
        int a[31],cnt;
5
        void clear(){memset(a,0,sizeof a),cnt=0;}
6
        void ins(int x){
7
            if(cnt==30) return;
8
            for(int i=29;i>=0&&x;i--) if(x>>i&1){
9
10
                if(a[i]) x^=a[i];
                else {a[i]=x,cnt++;return;}
11
            }
12
13
        void merge(const Base &B){
14
            for(int i=29;i>=0;i--) if(B.a[i]) ins(B.a[i]);
15
16
        }
17
   }t[maxn<<2];
   int n,m,a[maxn],b[maxn];
18
  int arr[maxn];
20 void upd(int i,int v){for(;i<=n;i+=i\&-i) arr[i]^=v;}
  int qxor(int i){int s=0;for(;i;i-=i&-i) s^=arr[i];return s;}
   void upd(int i){t[i]=t[i<<1],t[i].merge(t[i<<1|1]);}</pre>
   void build(int i,int l,int r){
24
        if(l==r) return t[i].ins(b[l]);
        int mid=(l+r)>>1;
25
26
        build(i<<1,1,mid),build(i<<1|1,mid+1,r);
27
        upd(i);
   }
   void mdf(int i,int l,int r,int x){
        if(l==r) {t[i].clear(),t[i].ins(b[x]);return;}
30
        int mid=(l+r)>>1;
31
        x \le mid?mdf(i << 1, 1, mid, x) : mdf(i << 1|1, mid+1, r, x);
32
        upd(i);
33
34
35
   void qry(int i,int l,int r,int x,int y){
        if(x<=l&&r<=y) return t[0].merge(t[i]);</pre>
36
        int mid=(l+r)>>1;
37
        if(x<=mid) qry(i<<1,1,mid,x,y);</pre>
38
        if(y>mid) qry(i<<1|1,mid+1,r,x,y);
39
40
   }
   int main()
41
42
   {
        scanf("%d%d",&n,&m);
43
        for(int i=1;i<=n;i++) scanf("%d",&a[i]),b[i]=a[i]^a[i-1],upd(i,b[i]);</pre>
44
45
        build(1,1,n);
46
        for(int op,1,r,x;m--;){
            scanf("%d%d%d",&op,&l,&r);
47
            if(op==1)
48
```

```
scanf("%d",&x);
49
                upd(l,x),upd(r+1,x),b[l]^=x,b[r+1]^=x;
50
                mdf(1,1,n,1); if(r< n) mdf(1,1,n,r+1);
51
            }
52
            else{
53
                t[0].clear(),t[0].ins(qxor(l)); if(l<r) qry(1,1,n,l+1,r);
54
                printf("%d\n",1<<t[0].cnt);</pre>
55
            }
56
57
        }
   }
58
          noIntervalModifyLinearBasis
   // 扫描r,维护线性基中每个元素的最大左端点l。与删除操作类似。
   // 这个可以强制在线, 把每个r的线性基存下来即可。
   #include <bits/stdc++.h>
   using namespace std;
5
   const int N = 2e5 + 10;
6
7
   struct node {
8
        int l, r, id;
9
        bool operator < (const node &p) const {</pre>
            return r < p.r;</pre>
10
11
   }q[N];
12
13
   const int maxl = 60;
14
15
16
   struct LinearBasis {
        ll a[maxl + 10], pos[maxl + 10];
int n, size; // 每个相同异或值有2^{n-size}个
17
18
19
        vector<ll> v;
20
        LinearBasis() {
21
22
            memset(a, 0, sizeof(a));
            size = n = 0;
23
24
            v.clear();
25
26
27
        void insert(ll t, ll id) {
28
            n++;
            for(int i = maxl;i >= 0; i--) {
29
                 if(!(t >> i & 1)) continue ;
30
31
                 if(a[i]) {
                     if(id > pos[i]) swap(id, pos[i]), swap(t, a[i]);
32
                     t ^= a[i];
33
                }
34
                else {
35
                     a[i] = t;
36
                     pos[i] = id;
37
38
                     return ;
                }
39
40
            }
        }
41
42
        int askmax(ll x) {
43
44
            11 \text{ ans} = 0;
            for(int i = maxl;i >= 0; i--) {
45
```

```
//
                  if(pos[i] >= x \&\& !(ans >> i \& 1)) ans ^= a[i];
46
47
                if(pos[i] >= x) ans = max(ans, ans ^ a[i]);
48
49
            return ans;
       }
50
   };
51
52
   // 给你n个数,每次查询 [公式] 这个区间,问着个区间的最大异或值。
53
54
   void solve() {
55
       LinearBasis lb;
56
       int n; cin >> n;
57
58
       VI a(n + 1);
       for(int i = 1;i <= n; i++) cin >> a[i];
59
       int m; cin >> m;
60
       VI ans(m + 1);
61
       for(int i = 1; i \ll m; i++) cin >> q[i].l >> q[i].r, q[i].id = i;
62
       sort(q + 1, q + m + 1);
63
       for(int i = 1, j = 1; i <= n; i++) {
64
            lb.insert(a[i], i);
65
            for(; j \le m \& q[j].r \le i; j++) ans[q[j].id] = lb.askmax(q[j].l);
66
67
       for(int i = 1;i <= m; i++) cout << ans[i] << endl;</pre>
68
69
  }
   6.54 \quad FWT
1 #include <bits/stdc++.h>
  using namespace std;
   typedef long long ll;
   const 11 mod = 998244353;
5
6
  const int N = 1e5 + 10;
7
   int a[N], b[N];
8
   inline void FWT_OR(int *f, int n, int opt) {
9
10
       for (int 0 = 2, k = 1; 0 \le n; 0 \le 1, k \le 1)
11
            for (int i = 0; i < n; i += 0)
12
                for (int j = 0; j < k; j++)
13
                    f[i + j + k] = (f[i + j + k] + 1] * f[i + j] * opt % mod + mod) % mod;
14
   }
15
   inline void FWT_AND(int *f, int n, int opt) {
        for (int o = 2, k = 1; o \ll n; o \ll 1, k \ll 1)
17
18
            for (int i = 0; i < n; i += o)
19
                for (int j = 0; j < k; j++)
                    f[i + j] = (f[i + j] + 1]l * f[i + j + k] * opt % mod + mod) % mod;
20
   }
21
22
   inline void FWT_XOR(int *f, int n, int opt) {
24
        for (int 0 = 2, k = 1; 0 \le n; 0 \le 1, k \le 1)
            for (int i = 0; i < n; i += o)
25
26
                for (int j = 0; j < k; j++) {
27
                    11 \ a0 = f[i + j], \ a1 = f[i + j + k];
                    f[i + j] = (a0 + a1) \% mod * opt \% mod;
28
                    f[i + j + k] = (a0 - a1 + mod) \% mod * opt \% mod;
29
                }
30
31 }
```

```
32
   inline void mul_OR(int *a, int *b, int n) {
33
       FWT_OR(a, n, 1); FWT_OR(b, n, 1);
34
       for(int i = 0; i < n; i++) a[i] = a[i] * b[i] % mod;
35
36
       FWT_OR(a, n, -1);
   }
37
38
   inline void mul_AND(int *a, int *b, int n) {
39
       FWT_AND(a, n, 1); FWT_AND(b, n, 1);
40
       for(int i = 0; i < n; i++) a[i] = a[i] * b[i] % mod;
41
       FWT_AND(a, n, -1);
42
43
   }
44
   ll quick_pow(ll a, ll b);
45
46
   inline void mul_XOR(int *a, int *b, int n) {
47
       11 inv2 = quick_pow(mod, mod - 2);
48
       FWT_XOR(a, n, 1); FWT_XOR(b, n, 1);
49
       for(int i = 0; i < n; i++) a[i] = a[i] * b[i] % mod;
50
       FWT_XOR(a, n, inv2);
51
52 }
53
   int main() {
54
55
       int n;
56
       cin >> n;
57
       n = 1 << n;
       for(int i = 0;i < n; i++) cin >> a[i];
58
       for(int i = 0;i < n; i++) cin >> b[i];
59
60
       mul_OR(a, b, n);
61
62
       mul_AND(a, b, n);
63
       mul_XOR(a, b, n);
64
65
66 }
         cdg 分治 FFT
   6.55
1 // hdu 7054
  // 求解(1+x^{a1})*(1+x^{a2})*...*(1+x^{an})
   // \sum_{i=1}^n a_i \le 1e6.
   // 可以f[i][j],前i个数的和为j的方案数,可以用生成函数转换,并用多项式求解,同时分治FFT优化。
7
   const int N = 1e5 + 10;
   int tr[N];
9 int getLen(int n);
10
  void FFT(Complex *A, int len);
11
12
  inline void MTT(ll *x, ll *y, ll *z, int len);
13
14
   struct Poly {
15
       11 *p;
16
       int len;
17
       void init(int len) {
18
19
           p = a + cnt;
20
           this -> len = len;
```

```
for(int i = 0; i \leftarrow len; i++) p[i] = read();
21
22
            cnt += len + 1;
23
24
       void mul(const Poly b) {
25
            static ll x[N], y[N];
26
27
            int LEN = getLen(len + b.len);
            for(int i = 0; i \le len; i++) x[i] = p[i];
28
29
            for(int i = 0;i <= b.len; i++) y[i] = b.p[i];</pre>
            for(int i = len + 1; i <= LEN; i++) x[i] = 0;
30
            for(int i = b.len + 1; i <= LEN; i++) y[i] = 0;
31
32
           MTT(x, y, p, LEN);
33
            this -> len += b.len;
34
            // 不知道为啥要两倍,可能会有不为0的情况,管他呢
            for(int i = len + 1; i \le 2 * LEN; i++) p[i] = 0;
35
            for(int i = 0; i \le LEN; i++) x[i] = y[i] = 0;
36
       }
37
   };
38
39
   Poly cdq(int 1, int r) {
40
       Poly res;
41
42
       if(l == r) res.init(len); // 长度
       else {
43
            int mid = (1 + r) / 2;
44
45
            res = cdq(1, mid);
            res.mul(cdq(mid + 1, r));
46
47
48
       return res;
   }
49
50
   void solve() {
51
       mem(a, 0);
52
53
       int n = read();
       cnt = 0;
54
       ll ans = 1;
55
       Poly res = cdq(1, n);
56
       for(int i = 0; i < n; i++) cout << res.p[i] << " ";</pre>
57
58
  }
   6.56 求逆分治 FFT
1 // f[i] = \sum_{j=1}^i f[i-j] * g[j]
  // g相同,可以用多项式求逆
  #include <bits/stdc++.h>
   using namespace std;
   typedef long long ll;
5
  const double PI = acos(-1);
   const int N = 1e5 + 10;
7
8
   struct Complex {
9
10
       double x, y;
       Complex(double a = 0, double b = 0): x(a), y(b) {}
11
12
       Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
       Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
13
       Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
14
        rhs.y + y * rhs.x); }
       Complex conj() { return Complex(x, -y); }
16 } w[N];
```

```
17
   11 mod;
18
   int tr[N];
19
   ll F[N], G[N];
20
21
  ll quick_pow(ll a, ll b);
22
23
  int getLen(int n);
24
25
  void FFT(Complex *A, int len);
27
28
  inline void MTT(ll *x, ll *y, ll *z, int len);
29
30 void Get_Inv(ll *f, ll *g, int n);
31
   void fenzhiFFT(ll *f, ll *g, int n) {
32
33
       static ll a[N];
       for(int i = 1; i < n; i++) a[i] = (mod - f[i]) % mod;
34
35
       a[0] = 1;
       Get_Inv(a, g, n);
36
37
       for(int i = 0;i < n; i++) {</pre>
38
            a[i] = 0;
39
40
       }
41
   }
42
   int main() {
43
44
       int n;
45
       cin >> n;
        for(int i = 1;i < n; i++) cin >> G[i];
46
47
       fenzhiFFT(G, F, n);
48
       for(int i = 0;i < n; i++) cout << F[i] << " ";</pre>
49
  }
50
   6.57 多项式求逆
   #include <bits/stdc++.h>
   using namespace std;
3
   typedef long long ll;
   const double PI = acos(-1);
   const int N = 1e5 + 10;
5
6
   struct Complex {
7
       double x, y;
8
       Complex(double a = 0, double b = 0): x(a), y(b) {}
9
       Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
10
       Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
11
       Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
12
        rhs.y + y * rhs.x); }
       Complex conj() { return Complex(x, -y); }
   } w[N];
14
15
16 ll mod;
   int tr[N];
17
18
   ll F[N], G[N];
  ll quick_pow(ll a, ll b);
```

```
21
   int getLen(int n) {
22
       int len = 1; while (len < (n << 1)) len <<= 1;
23
       for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
24
       for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), <math>sin(2)*
25
       PI * i / len));
       return len;
26
  }
27
28
29 void FFT(Complex *A, int len);
30
31 inline void MTT(ll *x, ll *y, ll *z, int len);
32
   void Get_Inv(ll *f, ll *g, int n) {
33
       if(n == 1) { g[0] = quick_pow(f[0], mod - 2); return ; }
34
       Get_Inv(f, g, (n + 1) >> 1);
35
36
       int len = getLen(n);
37
       static ll c[N];
38
       for(int i = 0; i < len; i++) c[i] = i < n ? f[i] : 0;
39
       MTT(c, g, c, len); MTT(c, g, c, len);
40
       for(int i = 0; i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
41
       for(int i = n; i < len; i++) g[i] = 0;
42
43
       for(int i = 0; i < len; i++) c[i] = 0;
44 }
45
   int main() {
46
       int n;
47
       cin >> n;
48
       for(int i = 0;i < n; i++) cin >> F[i];
49
50
       Get_Inv(F, G, n);
       for(int i = 0;i < n; i++) cout << G[i] << " ";</pre>
51
52 }
   6.58 多项式快速幂
1 // f(x)^k, k较小时, 可取, 每次FFT之后长度*2
   #define maxfft 1048576+5
3
4
   struct cp {
5
       double a, b;
       cp operator+(const cp &o) const { return (cp) {a + o.a, b + o.b}; }
6
       cp operator-(const cp &o) const { return (cp) {a - o.a, b - o.b}; }
7
       cp operator*(const cp &o) const { return (cp) {a * o.a - b * o.b, b * o.a + a * o.b
8
       }; }
       cp operator*(const double &o) const { return (cp) {a * o, b * o}; }
9
       cp operator!() const { return (cp) {a, -b}; }
10
  } w[maxfft];
11
12
13 int pos[maxfft];
14
   void fft_init(int len) {
15
       int j = 0;
16
       while ((1 << j) < len)j++;
17
18
       j--;
       for (int i = 0; i < len; i++)
19
           pos[i] = pos[i >> 1] >> 1 | ((i & 1) << j);
20
21 }
```

```
22
   void fft(cp *x, int len, int sta) {
23
        for (int i = 0; i < len; i++)
24
            if (i < pos[i])swap(x[i], x[pos[i]]);</pre>
25
        w[0] = (cp) \{1, 0\};
26
        for (unsigned i = 2; i \leftarrow len; i \leftarrow 1) {
27
            cp g = (cp) \{cos(2 * PI / i), sin(2 * PI / i) * sta\};
28
            for (int j = i >> 1; j >= 0; j -= 2)w[j] = w[j >> 1];
29
            for (int j = 1; j < i >> 1; j += 2)w[j] = w[j - 1] * g;
30
            for (int j = 0; j < len; <math>j += i) {
31
                cp *a = x + j, *b = a + (i >> 1);
32
                for (int l = 0; l < i >> 1; l++) {
33
                     cp \ o = b[1] * w[1];
34
                     b[1] = a[1] - o;
35
                    a[1] = a[1] + o;
36
                }
37
            }
38
39
        if (sta == -1) for (int i = 0; i < len; i++)x[i].a /= len, x[i].b /= len;
40
41
   }
42
   cp x[maxfft], y[maxfft], z[maxfft];
43
44
   void FFT(int *a, int *b, int n, int m, int *c) {
45
46
        int len = 1;
        while (len < (n + m) >> 1)len <<= 1;
47
        fft_init(len);
48
        for (int i = n / 2; i < len; i++)x[i].a = x[i].b = 0;
49
        for (int i = m / 2; i < len; i++)y[i].a = y[i].b = 0;
50
        for (int i = 0; i < n; i++)(i \& 1 ? \bar{x}[i >> 1].b : x[i >> 1].a) = a[i];
51
        for (int i = 0; i < m; i++)(i \& 1 ? y[i >> 1].b : y[i >> 1].a) = b[i];
52
        fft(x, len, 1), fft(y, len, 1);
53
        for (int i = 0; i < len / 2; i++) {
54
            int j = len - 1 \& len - i;
55
            z[i] = x[i] * y[i] - (x[i] - !x[j]) * (y[i] - !y[j]) * (w[i] + (cp) {1, 0}) *
56
       0.25;
        }
57
58
        for (int i = len / 2; i < len; i++) {
            int j = len - 1 & len - i;
59
            z[i] = x[i] * y[i] - (x[i] - !x[j]) * (y[i] - !y[j]) * ((cp) {1, 0} - w[i \land len ])
60
        >> 1]) * 0.25;
61
        fft(z, len, -1);
62
        for (int i = 0; i < n + m; i++)
63
            if (i & 1)c[i] = (int) (z[i >> 1].b + 0.5) ? 1 : 0;
64
            else c[i] = (int) (z[i >> 1].a + 0.5) ? 1 : 0;
65
   }
66
67
   int n, k, f[maxfft], g[maxfft];
68
70
   void Pow(int *f, int len, int k, int *g) {
71
        g[0] = 1;
72
        while (k) {
            if (k & 1)FFT(g, f, len, len, g);
73
            FFT(f, f, len, len, f);
74
75
            k >>= 1;
76
            len <<= 1;
77
        }
78 }
```

## 6.59 多项式除法、取模

```
1
   #include <bits/stdc++.h>
2
3
   using namespace std;
4 typedef long long ll;
5 const double PI = acos(-1);
  const int N = 3e5 + 10;
7 11 mod;
8
9 struct Complex {
10
       double x, y;
       Complex(double a = 0, double b = 0): x(a), y(b) {}
11
12
       Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
13
       Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
       Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
14
        rhs.y + y * rhs.x); }
       Complex conj() { return Complex(x, -y); }
15
   } w[N];
16
17
  int tr[N];
18
  11 F[N], G[N], D[N], R[N];
19
20
  ll quick_pow(ll a, ll b);
21
22
23 int getLen(int n);
24
25 void FFT(Complex *A, int len);
26
27
  inline void MTT(ll *x, ll *y, ll *z, int len);
28
29
  void Get_Inv(ll *f, ll *g, int n);
30
31
   void rever(ll *f, int n) { for(int i = 0, j = n - 1; i < j; i++, j--) swap(f[i], f[j]);
32
   void Get_Div(ll *f, ll *g, ll *d, ll *r, int n, int m) {
33
       static ll a[N], b[N], invb[N];
34
35
       for(int i = 0; i < n; i++) a[i] = f[i];
36
       for(int i = 0; i < m; i++) b[i] = g[i];
37
       rever(a, n); rever(b, m);
38
       //for(int i = 0; i < n - m + 1; i++) b[i] = i < m ? b[i] : 0;
       Get_Inv(b, invb, n - m + 1);
39
40
       int len = getLen(n);
41
42
       MTT(a, invb, a, len);
       rever(a, n - m + 1);
43
       for(int i = 0; i < len; i++) d[i] = i < n - m + 1 ? a[i] : 0;
44
       MTT(g, d, b, len);
45
       for(int i = 0; i < m; i++) { r[i] = (f[i] - b[i] + mod) % mod; }
46
47
       for(int i = m; i < len; i++) r[i] = 0;
48
       for(int i = 0; i < len; i++) a[i] = b[i] = invb[i] = 0;
49
   }
50
51
52
   int main() {
53
       int n, m;
54
       cin >> n >> m;
       for(int i = 0; i < n; i++) { cin >> F[i]; }
55
```

```
for(int i = 0;i < m; i++) { cin >> G[i]; }
56
        Get_Div(F, G, D, R, n, m);
57
58
        for(int i = 0; i < n - m + 1; i++) cout << D[i] << " ";
59
60
        cout << endl;
        for(int i = 0;i < m - 1; i++) cout << R[i] << " ";</pre>
61
        cout << endl;</pre>
62
63 }
   6.60 多项式 ln-Exp-Pow
   #include <bits/stdc++.h>
   using namespace std;
   typedef long long 11;
   const double PI = acos(-1);
 4
   const int N = 1e6 + 10;
7
   ll quick_pow(ll a, ll b) {
8
9
        ll\ ans = 1;
10
        while(b) {
            if(b \& 1) ans = ans * a % mod;
11
            a = a * a % mod;
12
13
            b >>= 1;
14
        return ans % mod;
15
16
   }
17
18
   const 11 G = 3;
   const ll invG = quick_pow(G, mod - 2);
19
20
21
   int tr[N];
22
   bool flag;
23
   void NTT(ll *A, int len, int type) {
24
25
        for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
26
        for (int i = 2; i <= len; i <<= 1) {
27
            int mid = i / 2;
28
            ll Wn = quick_pow(type == 1 ? G : invG, (mod - 1) / i);
29
            for (int k = 0; k < len; k += i) {
30
                11 w = 1;
                for (int l = k; l < k + mid; l++) {
31
                    ll t = w * A[l + mid] % mod;
32
                    A[l + mid] = (A[l] - t + mod) \% mod;
33
                    A[1] = (A[1] + t) \% mod;
34
                    w = w * Wn % mod;
35
36
                }
            }
37
38
        if (type == -1) {
39
            11 invn = quick_pow(len, mod - 2);
40
            for (int i = 0; i < len; i++)
41
                A[i] = A[i] * invn % mod;
42
43
        }
44
   }
45
   void mul(ll *a, ll *b, int len) {
46
        for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
47
```

```
NTT(a, len, 1), NTT(b, len, 1);
48
        for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
49
        NTT(a, len, -1); NTT(b, len, -1);
50
51 }
52
53
    int getLen(int n) {
         int len = 1; while (len <= (n << 1)) len <<= 1;</pre>
54
         return len;
55
    }
56
57
    void Get_Der(ll *f, ll *g, int len) { for(int i = 1; i < len; i++) g[i - 1] = f[i] * i %
58
         mod; g[len - 1] = 0; }
59
    void Get_Int(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i] = f[i - 1] *</pre>
60
        quick_pow(i, mod - 2) % mod; g[0] = 0; }
61
    void Get_Inv(ll *f, ll *g, int n) {
62
63
        if(n == 1) { g[0] = quick_pow(f[0], mod - 2); return ; }
        Get_Inv(f, g, (n + 1) >> 1);
64
65
        int len = getLen(n);
66
        static ll c[N];
67
        for(int i = 0; i < len; i++) c[i] = i < n ? f[i] : 0;
68
        mul(c, g, len);
70
        mul(c, g, len);
        for(int i = 0; i < n; i++) g[i] = (2ll * g[i] - c[i] + mod) % mod;
71
72
        for(int i = n; i < len; i++) g[i] = 0;
        for(int i = 0; i < len; i++) c[i] = 0;
73
74 }
75
    void Get_Ln(ll *f, ll *g, int n) {
76
        static ll a[N], b[N];
77
78
        Get_Der(f, a, n);
        Get_Inv(f, b, n);
79
        int len = getLen(n);
80
        mul(a, b, len);
81
82
        Get_Int(a, g, len);
        for(int i = n; i < len; i++) g[i] = 0;
83
        for(int i = 0; i < len; i++) a[i] = b[i] = 0;
84
    }
85
86
    void Get_Exp(ll *f, ll *g, int n) {
87
         if(n == 1) return (void)(g[0] = 1);
88
89
        Get_Exp(f, g, (n + 1) >> 1);
90
        static ll a[N];
91
        Get_Ln(g, a, n);
92
        a[0] = (f[0] + 1 - a[0] + mod) \% mod;
93
        for(int i = 1; i < n; i++) a[i] = (f[i] - a[i] + mod) % mod;
94
        int len = getLen(n);
96
        mul(g, a, len);
        for(int i = n; i < len; i++) g[i] = 0;
97
        for(int i = 0; i < len; i++) a[i] = 0;
98
99 }
100
    void Get_Pow(ll *f, ll *g, int n, ll k1, ll k2) {
101
        static ll a[N], b[N], c[N];
102
103
         ll deg = 0; for(int i = 0; i < n && f[i] == 0; i++) ++ deg;
104
        if(deg * k1 > n || (flag && deg)) return ;
```

```
ll f0 = f[deg], f0k = quick_pow(f0, k2), inv0 = quick_pow(f0, mod - 2);
105
        for(int i = deg; i < n; i++) a[i - deg] = f[i] * inv0 % mod;
106
        Get_Ln(a, b, n);
107
        for(int i = 0; i < n - deg * k1; i++) b[i] = b[i] * k1 % mod;
108
        Get_Exp(b, c, n);
109
        deg *= k1;
110
        for(int i = deg; i < n; i++) g[i] = (c[i - deg] * f0k % mod + mod) % mod;
111
        for(int i = 0; i < deg; i++) g[i] = 0;
112
        int len = getLen(n);
113
114
        for(int i = n; i < len; i++) g[i] = 0;
        for(int i = 0; i < len; i++) a[i] = b[i] = c[i] = 0;
115
116
    }
117
118
    11 a[N], ans[N];
119
120
    void solve() {
121
122
        int n; string s; cin >> n >> s;
        11 k1 = 0, k2 = 0;
123
        for(int i = 0; i < s.length(); i++) {</pre>
124
            k1 = (k1 * 10 + s[i] - '0');
125
            flag l= (k1 >= mod);
126
            k1 \% = mod;
127
128
            k2 = (k2 * 10 + s[i] - '0') \% \pmod{-1};
129
        for(int i = 0; i < n; i++) cin >> a[i];
130
        Get_Pow(a, ans, n, k1, k2); // k1是底 % mod, k2是指数 % mod-1
131
        for(int i = 0; i < n; i++) cout << ans[i] << (i == n - 1? endl : " ");
132
133 }
    6.61 任意模数 MTT-拆系数法
 1 //将多项式拆成(a1 * mod + a2) * (b1 * mod + b2)的形式
 2 //=>a1 * b1 * mod ^ 2 + (a2 * b1 + a1 * b2) * mod + a2 * b2
 3 //在利用DFT合并、IDFT合并,最终只需要4次DFT即可
 4 //精度10^14
 5 //4倍空间
 6
 7 #include <bits/stdc++.h>
 8 using namespace std;
 9 typedef long long ll;
10 const double PI = acos(-1);
   const int N = 1e5 + 10;
11
12
13 struct Complex {
14
        double x, y;
        Complex(double a = 0, double b = 0): x(a), y(b) {}
15
        Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
16
        Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
17
        Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
18
         rhs.y + y * rhs.x); }
        Complex conj() { return Complex(x, -y); }
19
    } w[N];
20
21
22 int tr[N];
23
    ll a[N], b[N], ans[N];
24
25 int getLen(int n) {
```

```
int len = 1; while (len <= n) len <<= 1;</pre>
26
        for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
27
        for (int i = 0; i < len; i++) w[i] = w[i] = Complex(cos(2 * PI * i / len), <math>sin(2)*
28
       PI * i / len));
29
        return len;
30 }
31
   void FFT(Complex *A, int len) {
32
        for (int i = 0; i < len; i++) if(i < tr[i]) swap(A[i], A[tr[i]]);</pre>
33
        for (int i = 2, lyc = len >> 1; i <= len; i <<= 1, lyc >>= 1)
34
            for (int j = 0; j < len; j += i) {
   Complex *l = A + j, *r = A + j + (i >> 1), *p = w;
35
36
                 for (int k = 0; k < i >> 1; k++) {
37
                     Complex tmp = *r * *p;
*r = *l - tmp, *l = *l + tmp;
38
39
                     ++1, ++r, p += 1yc;
40
                }
41
            }
42
43
   }
44
   inline void MTT(ll *x, ll *y, ll *z, int len) {
45
        for (int i = 0; i < len; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
46
        static Complex a[N], b[N];
47
48
        static Complex dfta[N], dftb[N], dftc[N], dftd[N];
49
        for (int i = 0; i < len; i++) a[i] = Complex(x[i] & 32767, x[i] >> 15);
50
        for (int i = 0; i < len; i++) b[i] = Complex(y[i] & 32767, y[i] >> 15);
51
        FFT(a, len), FFT(b, len);
52
        for (int i = 0; i < len; i++) {
53
            int j = (len - i) & (len - 1);
54
            static Complex da, db, dc, dd;
55
            da = (a[i] + a[j].conj()) * Complex(0.5, 0);
56
            db = (a[i] - a[j].conj()) * Complex(0, -0.5);
57
            dc = (b[i] + b[j].conj()) * Complex(0.5, 0);
58
            dd = (b[i] - b[j].conj()) * Complex(0, -0.5);
59
            dfta[j] = da * dc;
60
            dftb[i] = da * dd;
61
            dftc[j] = db * dc;
62
            dftd[i] = db * dd;
63
64
        for (int i = 0; i < len; i++) a[i] = dfta[i] + dftb[i] * Complex(0, 1);
65
        for (int i = 0; i < len; i++) b[i] = dftc[i] + dftd[i] * Complex(0, 1);
66
        FFT(a, len), FFT(b, len);
67
        for (int i = 0; i < len; i++) {
68
            ll da = (ll)(a[i].x / len + 0.5) % mod;
69
            ll db = (ll)(a[i].y / len + 0.5) \% mod;
70
            ll dc = (ll)(b[i].x / len + 0.5) \% mod;
71
            ll dd = (ll)(b[i].y / len + 0.5) \% mod;
72
            z[i] = (da + ((ll)(db + dc) << 15) + ((ll)(dd << 30)) % mod;
73
74
        }
75
   }
76
77
   int main() {
78
79
        int n, m;
        scanf("%d%d%lld", &n, &m, &mod);
80
        for (int i = 0; i <= n; i++) scanf("%d", &a[i]);</pre>
81
        for (int i = 0; i <= m; i++) scanf("%d", &b[i]);</pre>
82
83
```

```
MTT(a, b, ans, n + m);
84
       for (int i = 0; i <= n + m; i++)
85
           printf("%s%d", i == 0 ? "" : " ", (ans[i] + mod) % mod);
86
87
88
       return 0;
89 }
   6.62
         任意模数 NTT-三模数法
1 //要求选取的三个模数mod1 * mod2 * mod3 >= p^2*n
2 //优点是精度高,可达10^26
3 //缺点是常数大(9次NTT), 并且还使用了龟速乘
4 //4倍空间
5
6 #include <bits/stdc++.h>
7
   using namespace std;
   typedef long long ll;
   const int MAX = 4e5 + 10;
10
11 ll qmul(ll a, ll b, ll mod) {
12
       11 \text{ res} = 0;
       while (b) {
13
           if (b & 1)
14
               res = (res + a) \% mod;
15
           a = (a << 1) \% mod;
16
17
           b >>= 1;
18
       }
19
       return res;
20
   }
21
   ll qpow(ll a, ll b, ll mod) {
22
       23
       while (b) {
24
25
           if (b & 1) res = qmul(res, a, mod);
26
           a = qmul(a, a, mod);
27
           b >>= 1;
28
29
       return res;
30 }
31
32 \quad const \ ll \ mod1 = 998244353, \ mod2 = 1004535809, \ mod3 = 469762049, \ mod4 = mod1 * mod2;
   const ll G = 3;
   ll a[3][MAX], b[3][MAX], ans[MAX], p;
   int tr[MAX];
35
36
   void NTT(ll *A, int len, int type, ll mod) {
37
       for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
38
       for (int i = 2; i <= len; i <<= 1) {
39
           int mid = i / 2;
40
           ll Wn = qpow(type == 1 ? G : qpow(G, mod - 2, mod), (mod - 1) / i, mod);
41
           for (int k = 0; k < len; k += i) {
42
               11 w = 1;
43
               for (int l = k; l < k + mid; l++) {
44
                    ll t = w * A[l + mid] % mod;
45
                    A[l + mid] = (A[l] - t + mod) \% mod;
46
                    A[l] = (A[l] + t) \% mod;
47
                   W = W * Wn % mod;
48
49
               }
```

```
}
50
51
         if (type != 1) {
52
             ll invn = qpow(len, mod - 2, mod);
53
             for (int i = 0; i < len; i++) A[i] = A[i] * invn % mod;
54
55
         }
    }
56
57
    void mul(int i, int len, ll mod) {
58
         NTT(a[i], len, 1, mod), NTT(b[i], len, 1, mod);
59
         for (int j = 0; j < len; j++) a[i][j] = a[i][j] * b[i][j] % mod;
60
         NTT(a[i], len, -1, mod);
61
    }
62
63
    void CRT(int len) {
64
         ll inv1 = qpow(mod2, mod1 - 2, mod1);
65
         ll inv2 = qpow(mod1, mod2 - 2, mod2);
ll inv3 = qpow(mod4 % mod3, mod3 - 2, mod3);
66
67
         for (int i = 0; i < len; i++) {
68
             11 t = 0;
69
             t = (t + qmul(a[0][i] * mod2 % mod4, inv1, mod4)) % mod4;
70
             t = (t + qmul(a[1][i] * mod1 % mod4, inv2, mod4)) % mod4;
71
             a[1][i] = t;
72
73
             t = (a[2][i] - a[1][i] \% mod3 + mod3) \% mod3 * inv3 % mod3;
74
             ans[i] = (mod4 \% p * t \% p + a[1][i] \% p) \% p;
         }
75
    }
76
77
    void doNTT(int n) {
78
         int len = 1; while (len <= n) len <<= 1;</pre>
79
         for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
80
         mul(0, len, mod1), mul(1, len, mod2), mul(2, len, mod3);
81
82
         CRT(len);
    }
83
84
    int main() {
85
86
87
         int n, m;
         scanf("%d%d%lld", &n, &m, &p);
88
         for (int i = 0; i <= n; i++) {
89
             ll x; scanf("%lld", &x);
90
             a[0][i] = a[1][i] = a[2][i] = x \% p;
91
92
         for (int i = 0; i \le m; i++) {
93
             11 x; scanf("%11d", &x);
94
             b[0][i] = b[1][i] = b[2][i] = x \% p;
95
96
         doNTT(n + m);
97
         for (int i = 0; i \le n + m; i++) printf("%lld ", ans[i]);
98
99
100
         return 0;
101 }
```

## 6.63 多项式优化常系数齐次线性递推

```
#include <bits/stdc++.h>
using namespace std;
```

```
typedef long long ll;
   const double PI = acos(-1);
   const int N = 3e5 + 10;
6
   struct Complex {
8
9
        double x, y;
10
        Complex(double a = 0, double b = 0): x(a), y(b) {}
        Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
11
12
        Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
        Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
13
        rhs.y + y * rhs.x); }
        Complex conj() { return Complex(x, -y); }
14
15
   } w[N];
16
17
   ll mod;
   int n, k, len, tr[N];
   ll a[N], h[N], ans[N], s[N], invG[N], G[N];
   ll quick_pow(ll a, ll b);
21
22
23 int getLen(int n);
24
25 void rever(ll *f, int n);
26
27
   void FFT(Complex *A, int len);
28
   inline void MTT(ll *x, ll *y, ll *z, int len);
29
30
   void Get_Inv(ll *f, ll *g, int n);
31
32
33
   void Mod(ll *f,ll *g) {
34
        static ll tmp[N];
35
        rever(f, k + k - 1);
36
        for(int i = 0;i < k; i++) tmp[i] = f[i];</pre>
37
       MTT(tmp, invG, tmp, len);
38
39
        for(int i = k - 1; i < len; i++) tmp[i] = 0;
40
        rever(f, k + k - 1); rever(tmp, k - 1);
        MTT(tmp, G, tmp, len);
41
42
        for(int i = 0; i < k; i++) g[i] = (f[i] + mod - tmp[i]) % mod;
        for(int i = k; i < len; i++) g[i] = 0;
43
        for(int i = 0; i < len; i++) tmp[i] = 0;
44
   }
45
46
   void fpow(int b) {
        s[1] = 1; ans[0] = 1;
47
        while(b) {
48
49
            if(b & 1) { MTT(ans, s, ans, len); Mod(ans, ans); }
            MTT(s, s, s, len); Mod(s, s);
50
            b >>= 1;
51
52
        }
53
   }
54
   ll DITI(ll *a, ll *h, ll *ans, int n, int k) {
55
        G[k] = 1; for(int i = 1;i <= k; i++) G[k - i] = (mod - a[i]) \% mod;
56
        rever(G, k + 1);
57
        len = getLen(k + 1);
58
59
        Get_Inv(G, invG, k + 1);
        for(int i = k + 1; i < len; i++) invG[i] = 0;
60
61
        rever(G, k + 1);
```

```
fpow(n);
62
63
                   11 \text{ Ans} = 0;
                   for(int i = 0; i < k; i++) Ans = (Ans + 1ll * h[i] * ans[i] % mod) % mod;
64
65
                   return Ans;
        }
66
67
        int main() {
68
                   int n, k;
69
70
                   cin >> n >> k;
                   for(int i = 1; i \le k; i++){ cin >> a[i]; a[i] = a[i] < 0 ? a[i] + mod : a[i]; }
71
72
                   for(int i = 0; i < k; i++) { cin >> h[i]; h[i] = h[i] < 0 ? h[i] + mod : h[i]; }
73
                   ll Ans = DITI(a, h, ans, n, k);
74
                   cout << Ans << endl;</pre>
75
76 }
         6.64 FFT 加速带有通配符字符串匹配
 1 #include <bits/stdc++.h>
 2 using namespace std;
 3 typedef long long ll;
 4
        // p[x] = \sum_{i=0}^{m-1} A[i]^3 * B[x-m+i+1] + \sum_{i=0}^{m-1} A[i] * B[x-m+i+1]^3 - \sum_{i=0}^{m-1} A[i] * B[x-m+i+1] + \sum_{i=0}^{m-1} A[i] * B[x-m+i+1]^3 - \sum_{i=0}^{m-1} A[i] * B[x-m+i+1] + \sum_{i=0}^{m-1} A[
                     2 * \sum_{i=0}^{m-1} A[i]^2 * B[x-m+i+1]^2
 6
        const int N = 1e6 + 1e5;
 7
 8
        ll qpow(ll a, ll b, ll mod) {
 9
                   ll\ ans = 1;
10
11
                   while(b) {
12
                              if(b \& 1) ans = ans * a % mod;
13
                             a = a * a % mod;
14
                             b >>= 1;
15
16
                   return ans % mod;
17 }
18
19 const ll G = 3;
       const ll invG = qpow(G, mod - 2, mod);
21
        int tr[N];
22
        void NTT(ll *A, int len, int type) {
23
                    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);</pre>
24
                   for (int i = 2; i <= len; i <<= 1) {
25
26
                              int mid = i / 2;
                              ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
27
                              for (int k = 0; k < len; k += i) {
28
29
                                        11 w = 1;
                                        for (int l = k; l < k + mid; l++) {
30
                                                   ll t = w * A[l + mid] % mod;
31
                                                   A[l + mid] = (A[l] - t + mod) \% mod;
32
                                                   A[1] = (A[1] + t) \% mod;
33
34
                                                  w = w * Wn % mod;
35
                                        }
                             }
36
37
                   if (type == -1) {
38
                              ll invn = qpow(len, mod - 2, mod);
39
```

```
for (int i = 0; i < len; i++)
40
                                         A[i] = A[i] * invn % mod;
41
                    }
42
        }
43
44
        void mul(ll *a, ll *b, int n) {
45
                    int len = 1; while (len <= n) len <<= 1;</pre>
46
                    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
47
                    NTT(a, len, 1), NTT(b, len, 1);
48
                    for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
49
                    NTT(a, len, -1);
50
51
        }
52
        ll a1[N], a2[N], a3[N], b1[N], b2[N], b3[N];
53
54
        void solve() {
55
56
                    int m, n; cin >> m >> n;
                    string s, t; cin >> t >> s;
57
                    for(int i = 0;i < m; i++) {</pre>
58
                              if(t[i] == '*') continue;
59
                              int temp = t[i] - 'a' + 1;
60
                              a1[i] = temp;
61
                              a2[i] = temp * temp;
62
                              a3[i] = temp * temp * temp;
63
64
                    for(int i = 0;i < n; i++) {</pre>
65
                              if(s[i] == '*') continue;
66
                              int temp = s[i] - 'a' + 1;
67
                              b1[i] = temp;
68
                              b2[i] = temp * temp;
69
                              b3[i] = temp * temp * temp;
70
71
72
                    reverse(a1, a1 + m);
                    reverse(a2, a2 + m);
73
                    reverse(a3, a3 + m);
74
                    mul(a1, b3, n + m);
75
                    mul(a2, b2, n + m);
76
77
                    mul(a3, b1, n + m);
                    vector<int> ans;
78
                    for(int x = m - 1; x < n; x++) {
79
                              ll res = a1[x] + a3[x] - a2[x] * 2;
80
                              if(!res) ans.push_back(x - m + 2);
81
82
83
                    cout << ans.size() << endl;</pre>
                    for(int i = 0; i < ans.size(); i++) cout << ans[i] << (i == ans.size() - 1 ? endl :
84
                    " ");
85 }
                          FFT 加速朴素字符串匹配
 1 #include <bits/stdc++.h>
  2 using namespace std:
 3 const int N = 4e5 + 10;
        // P[x] = \sum_{i=0}^{m-1} A[i] + \sum_{i=0}^{m-1} B[x - m + i + 1] - 2 * \sum_{i=0}^{m} A[i] + \sum_{i=0}^{m-1} A[i] + \sum_{i
                   -1A[i] * B[x - m + i + 1]
  7 // reverse(a)
```

```
9
   // 当串中的字符集较少时,可以针对每个字符进行FFT,计算每个字符对整个串的贡献
10
   ll qpow(ll a, ll b, ll mod);
11
12
13 const ll mod = 998244353;
14 const ll G = 3;
15 const ll invG = qpow(G, mod - 2, mod);
  int tr[N];
17
   void NTT(ll *A, int len, int type) {
18
19
        for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
       for (int i = 2; i <= len; i <<= 1) {
20
            int mid = i / 2;
21
            ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
22
            for (int k = 0; k < len; k += i) {
23
                ll w = 1;
24
                for (int l = k; l < k + mid; l++) {
25
                    ll t = w * A[l + mid] % mod;
26
                    A[l + mid] = (A[l] - t + mod) \% mod;
27
                    A[1] = (A[1] + t) \% mod;
28
29
                    W = W * Wn \% mod;
                }
30
31
            }
32
        if (type == -1) {
33
            ll invn = qpow(len, mod - 2, mod);
34
            for (int i = 0; i < len; i++)
35
                A[i] = A[i] * invn % mod;
36
       }
37
   }
38
39
   void mul(ll *a, ll *b, int n) {
40
       int len = 1; while (len <= n) len <<= 1;</pre>
41
       for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
42
       NTT(a, len, 1), NTT(b, len, 1);
43
       for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
44
45
       NTT(a, len, -1);
   }
46
47
   ll a[N], b[N];
48
49
   void solve() {
50
       string s, t; cin >> s >> t;
51
       int n = s.length(), m = t.length();
52
       for(int i = 0; i < n; i++) a[i] = s[i] - 'a' + 1;
53
       for(int i = 0; i < m; i++) b[i] = t[i] - 'a' + 1;
54
       reverse(b, b + m);
55
       mul(a, b, n + m - 2);
56
57
       double P = 0;
       for(int i = 0;i < m; i++) {
   P += (t[i] - 'a' + 1) * (t[i] - 'a' + 1);
58
59
60
       vector<int> f(n + 1);
61
       for(int i = 1; i < n; i++) {
62
            f[i] = f[i - 1] + (s[i] - 'a' + 1) * (s[i] - 'a' + 1);
63
64
65
        for(int x = m - 1; x < n; x++) {
            double res;
66
```

```
if(x == m - 1) res = P + f[x] - a[x] * 2;
67
            else res = P + f[x] - f[x - m] - a[x] * 2;
if(!res) cout << x - m + 2 << endl;
68
69
        }
70
71 }
   6.66 x 不连续、暴力插值
1
   #include <bits/stdc++.h>
   using namespace std;
3
4 typedef long long ll;
5 const double PI = acos(-1);
  const int N = 3e5 + 10;
7
  ll mod;
8
9
   ll X[N], Y[N];
10
   ll quick_pow(ll a, ll b);
11
12
   ll Lagrange(ll *x, ll *y, int n, int k) {
13
        11 \text{ ans} = 0;
14
        for(int i = 0;i < n; i++) {</pre>
15
            11 s1 = 1, s2 = 1;
16
            for(int j = 0; j < n; j++) {
17
                if(i == j) continue;
18
                s1 = s1 * (k - x[j] + mod) % mod;
19
                s2 = s2 * (x[i] - x[j] + mod) % mod;
20
21
            ans = (ans + 111 * y[i] * s1 % mod * quick_pow(s2, mod - 2) % mod) % mod;
22
23
24
        return ans;
   }
25
26
27
   int main() {
        int n, k;
28
        cin >> n >> k;
29
30
        for(int i = 0;i < n; i++) cin >> X[i] >> Y[i];
31
        cout << Lagrange(X, Y, n, k) << endl;</pre>
32 }
   6.67 x 连续、前缀优化
1
2 #include <bits/stdc++.h>
3 using namespace std;
 4 typedef long long ll;
5 \quad const int N = 1e5 + 10;
6
   ll mod;
7
   11 F[N];
   ll pre[N], suf[N];
9
   ll fac[N], invf[N];
10
11
12
   ll quick_pow(ll a, ll b);
13
14
```

```
15 void init() {
16
        fac[0] = 1;
        for(int i = 1; i < N; i++) fac[i] = fac[i - 1] * i % mod;
17
        invf[N - 1] = quick_pow(fac[N - 1], mod - 2);
18
        for(int i = N - 1; i >= 1; i--) invf[i - 1] = invf[i] * i % mod;
19
   }
20
21
   ll Lagrange(ll *f, int k, int n) {
22
        if(k <= n) return f[k];</pre>
23
        pre[0] = suf[n] = 1;
24
25
        for(int i = 1; i <= n; i++) pre[i] = pre[i - 1] * (k - i + 1) % mod;
        for(int i = n; i >= 1; i--) suf[i - 1] = suf[i] * (k - i) % mod;
26
27
        11 \text{ ans} = 0;
        for(int i = 0; i <= n; i++) {</pre>
28
            int opt = (n - i) & 1 ? -1 : 1;
29
            ans = (ans + 1ll * opt * pre[i] % mod * suf[i] % mod * invf[i] % mod * invf[n -
30
         i] % mod * f[i] % mod + mod) % mod;
31
32
        return f[k] = ans;
33 }
34
   int main() {
35
        init();
36
37
        int n, k;
38
        cin >> n >> k;
        for(int i = 0;i <= n; i++) cin >> F[i];
39
        cout << Lagrange(F, k, n) << endl;</pre>
40
41
42 }
          多项式 ln-exp-pow 处理边界为 1
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 const double PI = acos(-1);
5 const int N = 1e5 + 10;
7
   struct Complex {
8
        double x, y;
        Complex(double a = 0, double b = 0): x(a), y(b) {}
9
        Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
10
        Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); } Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
11
12
         rhs.y + y * rhs.x); }
        Complex conj() { return Complex(x, -y); }
13
14
   } w[N];
15
16 ll mod, inv2;
   int tr[N];
  ll F[N], G[N];
19
20 ll quick_pow(ll a, ll b);
21
22 int getLen(int n);
23
24 void FFT(Complex *A, int len);
25
```

```
inline void MTT(ll *x, ll *y, ll *z, int len);
27
       void Get_Inv(ll *f, ll *g, int n);
28
29
       void Get_Der(ll *f, ll *g, int len) { for(int i = 1; i < len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len
30
                   mod; g[len - 1] = 0; }
31
       void Get_Int(ll *f, ll *g, int len) { for(int i = 1; i < len; i++) g[i] = f[i - 1] *
32
                quick_pow(i, mod - 2) % mod; g[0] = 0; }
33
       void Get_Ln(ll *f, ll *g, int n);
34
35
36 void Get_Exp(ll *f, ll *g, int n);
37
      void Get_Pow(ll *f, ll *g, int n, ll k);
38
39
       void Get_Sqrt(ll *f, ll *g, int n) {
40
                 static ll a[N];
41
42
                 Get_Ln(f, a, n);
                 for(int i = 0; i < n; i++) a[i] = a[i] * inv2 % mod;
43
                 Get_Exp(a, g, n);
44
                 int len = getLen(n);
45
                 for(int i = n; i < len; i++) g[i] = 0;
46
47
                 for(int i = 0; i < len; i++) a[i] = 0;
48 }
                       二次剩余处理边界不为 1
        6.69
 1
       #include <bits/stdc++.h>
 2
       using namespace std;
 3
 4 typedef long long ll;
 5 const double PI = acos(-1);
 6 const int N = 1e5 + 10;
 7
 8
 9
       struct Complex {
                 double x, y;
10
11
                 Complex(double a = 0, double b = 0): x(a), y(b) {}
12
                 Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
                 Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
13
                 Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
14
                  rhs.y + y * rhs.x); }
                 Complex conj() { return Complex(x, -y); }
15
16 } w[N];
17
18 ll mod, inv2;
      int tr[N];
19
20 ll F[N], G[N];
21
22 ll quick_pow(ll a, ll b);
23
24
       typedef struct{
                 ll x, y; // 把求出来的w作为虚部, 则为a + bw
25
26
27
       num num_mul(num a, num b, ll w, ll p) {// 复数乘法
28
                 num ans = \{0, 0\};
29
```

```
ans.x = (a.x * b.x \% p + a.y * b.y \% p * w \% p + p) \% p;
30
       ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
31
       return ans;
32
33 }
34
   ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
35
       num ans = \{1, 0\};
36
       while(b) {
37
           if(b & 1)
38
               ans = num_mul(ans, a, w, p);
39
           a = num_mul(a, a, w, p);
40
41
           b >>= 1;
42
43
       return ans.x % p;
   }
44
45
   ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
46
       return quick_pow(a, (p - 1) >> 1);
47
48
   }
49
   ll Cipolla(ll n, ll p) \{// 输入a和p, 是否存在x使得x^2 = a \pmod{p}, 存在二次剩余返回x, 存在二次
50
       非剩余返回-1
                      注意: p是奇质数
51
       n \% p;
52
       if(n == 0)
53
           return 0;
54
       if(p == 2)
           return 1;
55
56
       ll a, w;
57
       while(true) {// 找出a, 求出w, 随机成功的概率是50%, 所以数学期望是2
58
59
           a = rand() \% p;
           w = ((a * a - n) \% p + p) \% p;
60
           if(legendre(w, p) + 1 == p) // 找到w, 非二次剩余条件
61
               break;
62
       }
63
       num x = \{a, 1\};
64
       return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x, 一个解是x, 另一个解是p-x, 这里的w其实
65
       要开方,但是由拉格朗日定理可知虚部为0,所以最终答案就是对x的实部用快速幂求解
   }
66
67
68 int getLen(int n);
69
70 void FFT(Complex *A, int len);
71
72
   inline void MTT(ll *x, ll *y, ll *z, int len);
73
74 void Get_Inv(ll *f, ll *g, int n);
75
   void Get_Sqrt(ll *f, ll *g, int n) {
76
77
       if(n == 1) { ll t = Cipolla(f[0], mod); g[0] = min(mod - t, t); return ; }
78
       Get_Sqrt(f, g, (n + 1) >> 1);
79
80
       int len = getLen(n);
81
       static ll c[N], invg[N];
       for(int i = 0; i < len; i++) c[i] = i < n ? f[i] : 0;
82
       Get_Inv(g, invg, n);
83
84
       MTT(c, invg, c, len);
85
       for(int i = 0; i < n; i++) g[i] = inv2 * (c[i] + g[i]) % mod;
86
       for(int i = n; i < len; i++) g[i] = 0;
```

```
for(int i = 0; i < len; i++) c[i] = invg[i] = 0;
87
   }
88
89
   int main() {
90
        inv2 = quick_pow(2, mod - 2);
91
92
        int n;
        cin >> n;
93
        for(int i = 0;i < n; i++) cin >> F[i];
94
        Get_Sqrt(F, G, n);
95
        for(int i = 0;i < n; i++) cout << G[i] << " ";</pre>
96
97
  }
         二维几何
   6.70
   #include <iostream>
   #include <cmath>
3
   using namespace std;
5
6
   const double eps = 1e-6;
   const double pi = acos(-1);
7
8
   #define zero(x) (((x) > 0 ? (x) : -(x)) < eps)
9
10
   int sgn(double d) {
11
12
        if(fabs(d) < eps)</pre>
13
            return 0;
14
        if(d > 0)
15
            return 1;
16
        else
17
            return -1;
18
   }
19
20
   int dcmp(double x, double y) {
21
        if(fabs(x - y) < eps)
22
            return 0;
23
        if(x > y)
24
            return 1;
25
        else
26
            return -1;
   }
27
28
   struct Point{ // 点
29
        double x, y;
30
31
        Point(double x = 0, double y = 0) : x(x), y(y) {}
   };
32
33
34
   struct line{
        Point a, b;
35
36
   };
37
  typedef Point Vector; // 向量
38
39
40 // 运算(向量之间)
41
42 Vector operator + (Vector A, Vector B) { // AB
        return Vector(A.x + B.x, A.y + B.y);
43
44
   }
```

```
45
    Vector operator - (Point A, Point B) { // BA
46
        return Vector(A.x - B.x, A.y - B.y);
47
    }
48
49
    Vector operator * (Vector A, double p) { // A * p
50
        return Vector(A.x * p, A.y * p);
51
    }
52
53
    Vector operator / (Vector A, double p) { // A / p
        return Vector(A.x / p, A.y / p);
55
56
57
    bool operator < (const Point& a, const Point& b) { // 将点升序排列
58
        if(a.x == b.x)
59
            return a.y < b.y;</pre>
60
        return a.x < b.x;</pre>
61
    }
62
63
    bool operator == (const Point& a, const Point& b) { // 判断是否为同一点
64
        if(dcmp(a.x, b.x) == 0 \&\& dcmp(a.y, b.y) == 0)
65
66
            return true;
        else
67
            return false;
68
69 }
70
   /*
71
        */
72
   // 向量
73
    double Dot(Vector A, Vector B) { // 内积
74
        return A.x * B.x + A.y * B.y;
75
76
77
    double Cross(Vector A, Vector B) { // 外积
78
79
        return A.x * B.y - A.y * B.x;
80 }
81
    double Length(Vector A) { // 向量取模
82
        return sqrt(Dot(A, A));
83
84
85
    double Angle(Vector A, Vector B) { // 向量夹角
86
        return acos(Dot(A, B) / Length(A) / Length(B));
87
88 }
89
    double Area(Point A, Point B, Point C) { // 计算两向量构成的平行四边形有向面积
90
        return Cross(B - A, C - A);
91
92 }
93
94
    Vector Rotate(Vector A, double rad) { // 计算向量逆时针旋转后的向量
        return Vector(A.x * cos(rad) - A.y * sin(rad), A.x * sin(rad) + A.y * cos(rad));
95
    }
96
97
    Vector Normal(Vector A) { // 计算向量逆时针转90度后的单位法向量
98
        double L = Length(A);
99
100
        return Vector(-A.y / L, A.x / L);
101 }
```

```
102
    bool ToLeftTest(Point a, Point b, Point c) { // 判断bc是不是向ab的逆时针方向转向
103
        return Cross(b - a, c - b) > 0;
104
    }
105
106
107
        */
   // 直线与线段
108
109
    double Pow(double x) {
110
111
        return x * x;
112
113
    double distance (Point p1, Point p2) {// 两点距离
114
        return sqrt(Pow(p1.x - p2.x) + Pow(p1.y - p2.y));
115
116
117
int dots_inline(Point p1, Point p2, Point p3) { // 判断三点共线
        return Cross(p2 - p1, p3 - p1);
119
120
    }
121
    int dot_online_in(Point p, line l) { // 判断点在线段上(包含端点)
122
123
        return zero(Cross(l.b - p, l.a - p) && ((l.a.x - p.x) * (l.b.x - p.x) < eps) && ((l
        (a.y - p.y) * (1.b.y - p.y) < eps);
124
    }
125
    int dot_online_ex(Point p, line l) { // 判断点在线段上(不包含端点)
126
        return dot_online_in(p, 1) && (!zero(p.x - l.a.x) || !zero(p.y - l.a.y)) && (!zero(
127
       p.x - l.b.x) || !zero(p.y - l.b.y));
128
    }
129
    int same_side(Point p1, Point p2, line l) { // 判断两点在线段同侧,点在线段上返回0
130
        return Cross(l.a - l.b, p1 - l.b) * Cross(l.a - l.b, p2 - l.b) > eps;
131
132
   }
133
134
    int opposite_side(Point p1, Point p2, line l) { // 判断两点在线段异侧, 点在线段上返回0
135
        return Cross(l.a - l.b, p1 - l.b) * Cross(l.a - l.b, p2 - l.b) < -eps;
136
137
    int parallel(line u, line v) { // 判断两直线平行
138
        return zero((u.a.x - u.b.x) * (v.a.y - v.b.y) - (u.a.y - u.b.y) * (v.a.x - v.b.x));
139
140
141
    int perpendicular(line u, line v) { // 判断两直线垂直
142
143
        return zero((u.a.x - u.b.x) * (v.a.x - v.b.x) + (u.a.y - u.b.y) * (v.a.y - v.b.y));
144
145
    int intersect_in(line u, line v) {// 判断两线段相交,包括端点和部分重合
146
        if(!dots_inline(u.a, u.b, v.a) || !dots_inline(u.a, u.b, v.b)) {
147
148
            return !same_side(u.a, u.b, v) && !same_side(v.a, v.b, u);
149
        return dot_online_in(u.a, v) || dot_online_in(u.b, v) || dot_online_in(v.a, u) ||
150
       dot_online_in(v.b, u);
    }
151
152
    int intersect_ex(line u, line v) {// 判断两线段相交, 不包括端点和部分重合
153
154
        return opposite_side(u.a, u.b, v) && opposite_side(v.a, v.b, u);
155 }
```

```
156
157
    // 计算两直线交点,注意事先判断直线是否相交
    // 计算两线段交点, 注意事先判断线段相交和平行
158
    Point intersection(line u, line v) {
159
160
        Point ret = u.a;
        double t = ((u.a.x - v.a.x) * (v.a.y - v.b.y) - (u.a.y - v.a.y) * (v.a.x - v.b.x))
161
        /((u.a.x - u.b.x) * (v.a.y - v.b.y) - (u.a.y - u.b.y) * (v.a.x - v.b.x));
        ret.x += (u.b.x - u.a.x) * t;
162
        ret.y += (u.b.y - u.a.y) * t;
163
164
        return ret;
    }
165
166
    Point ptoline(Point p, line l) { // 点到直线最近点
167
        Point t = p;
168
        t.x += l.a.y - l.b.y;
t.y += l.b.x - l.a.x;
169
170
171
        line u = \{p, t\};
172
        return intersection(u, 1);
173
    }
174
    double disptoline(Point p, line l) { // 点到直线距离
175
        return fabs(Cross(p - l.b, l.a - l.b) / distance(l.a, l.b));
176
    }
177
178
    Point ptoseg(Point p, line l) { // 点到线段最近点
179
180
        Point t = p;
        t.x += l.a.y - l.b.y;
181
        t.y += l.b.x - l.a.x;
182
        if(Cross(l.a - p, t - p) * Cross(l.b - p, t - p) > eps)
183
            return distance(p, l.a) < distance(p, l.b) ? l.a : l.b;</pre>
184
185
        line u = \{p, t\};
        return intersection(u, 1);
186
187
   }
188
    double disptoseg(Point p, line l) { // 点到线段距离
189
        Point t = p;
190
191
        t.x += l.a.y - l.b.y;
192
        t.y += 1.b.x - 1.a.x;
        if(Cross(l.a - p, t - p) * Cross(l.b - p, t - p) > eps) {
193
            double dis1 = distance(p, l.a);
194
195
            double dis2 = distance(p, 1.b);
            return dis1 < dis2 ? dis1 : dis2;</pre>
196
197
198
        return fabs(Cross(p - l.b, l.a - l.b) / distance(l.a, l.b));
199 }
200
201 /*
202 // 面积
203
    double area_triangle(Point p1, Point p2, Point p3) { // 三角形面积 (输入三顶点)
204
205
        return fabs(Cross(p1 - p3, p2 - p3)) / 2;
206
    }
207
    double area_triangle(double a, double b, double c) { // 三角形面积 (输入三边长)
208
209
        double s = (a + b + c) / 2;
        return sqrt(s * (s - a) * (s - b) * (s - c));
210
211 }
```

```
212
    double area_polygon(int n, Point *p) { // 计算多边形面积, 顶点按顺时针或逆时针输入
213
        double s1 = 0, s2 = 0;
214
        for(int i = 0;i < n; i++) {</pre>
215
            s1 += p[(i + 1) \% n].y * p[i].x;
216
            s2 += p[(i + 1) \% n].y * p[(i + 2) \% n].x;
217
218
        return fabs(s1 - s2) / 2;
219
220 }
221
222 /*
223 // 球面
224
225
    //计算圆心角 lat 表示纬度,-90<=w<=90,lng 表示经度
226
    //返回两点所在大圆劣弧对应圆心角,0 < = angle < = pi
227
    double angle(double lng1, double lat1, double lng2, double lat2) {
228
        double dlng = fabs(lng1 - lng2) * pi / 180;
229
        while(dlng >= pi + pi) {
230
            dlng -= pi + pi;
231
232
233
        if(dlng > pi)
234
        dlng = pi + pi - dlng;
        lat1 *= pi / 180;
235
        lat2 *= pi / 180;
236
        return acos(cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2));
237
    }
238
239
240 // 计算两点距离
241
    double line_dist(double r, double lng1, double lat1, double lng2, double lat2) {
242
243
        double dlng = fabs(lng1 - lng2) * pi / 180;
244
        while(dlng >= pi + pi) {
            dlng -= pi + pi;
245
246
        }
247
        if(dlng > pi)
        dlng = pi + pi - dlng;
248
249
        lat1 *= pi / 180;
        lat2 *= pi / 180;
250
        return r * sqrt(2 - 2 * (cos(lat1) * cos(lat2) * cos(dlng) + sin(lat1) * sin(lat2))
251
        );
252
    }
253
254
   // 计算球面距离
255
256
    inline double sphere_dist(double r, double lng1, double lat1, double lng2, double lat2)
257
        return r * angle(lng1, lat1, lng2, lat2);
258 }
259
260
        */
261 // 三角形
262
263 // 外心
264
```

```
Point circumcenter(Point a, Point b, Point c) {
266
        line u, v;
267
        u.a.x = (a.x + b.x) / 2;
        u.a.y = (a.y + b.y) / 2;
268
269
        u.b.x = u.a.x - a.y + b.y;
        u.b.y = u.a.y + a.x - b.x;
270
271
        v.a.x = (a.x + c.x) / 2;
272
        v.a.y = (a.y + c.y) / 2;
        v.b.x = v.a.x - a.y + c.y;
273
        v.b.y = v.a.y + a.x - c.y;
274
275
        return intersection(u, v);
276 }
277
278 // 内心
279
    Point incenter(Point a, Point b, Point c) {
280
281
        line u, v;
282
        double m, n;
283
        u.a = a;
        m = atan2(b.y - a.y, b.x - a.x);
284
        n = atan2(c.y - a.y, c.x - a.x);
285
        u.b.x = u.a.x + cos((m + n) / 2);
286
        u.b.y = u.a.y + sin((m + n) / 2);
287
288
        v.a = b;
289
        m=atan2(a.y - b.y, a.x - b.x);
        n=atan2(c.y - b.y, c.x - b.x);
290
        v.b.x=v.a.x + cos((m + n) / 2);
291
        v.b.y=v.a.y + sin((m + n) / 2);
292
293
        return intersection(u, v);
294 }
295
296 // 垂心
297
    Point perpencenter(Point a, Point b, Point c) {
298
299
        line u, v;
300
        u.a = c;
301
        u.b.x = u.a.x - a.y + b.y;
302
        u.b.y = u.a.y + a.x - b.x;
303
        v.a = b;
304
        v.b.x = v.a.x - a.y + c.y;
305
        v.b.y = v.a.y + a.x - c.x;
306
   return intersection(u, v);
307
    }
308
309 // 重心
310 //到三角形三顶点距离的平方和最小的点
311 //三角形内到三边距离之积最大的点
312
313
    Point barycenter(Point a, Point b, Point c) {
314
        line u,v;
315
        u.a.x = (a.x + b.x) / 2;
        u.a.y = (a.y + b.y) / 2;
316
317
        u.b = c;
318
        v.a.x = (a.x + c.x) / 2;
319
        v.a.y = (a.y + c.y) / 2;
320
        v.b = b;
321
    return intersection(u, v);
322
    }
323
```

```
324 //费马点
325
    //到三角形三顶点距离之和最小的点
    Point fermentpoint(Point a, Point b, Point c) {
326
327
        Point u,v;
        double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) + fabs(c.y)
328
329
        int i, j, k;
        u.x = (a.x + b.x + c.x) / 3;
330
        u.y = (a.y + b.y + c.y) / 3;
331
332
        while(step > 1e-10)
333
        for(k = 0; k < 10; step /= 2, k++)
334
            for (i = -1; i <= 1; i++)
                for (j = -1; j \ll 1; j++)
335
336
                v.x = u.x + step * i;
                v.y = u.y + step * j
337
                if(distance(u,a) + distance(u,b) + distance(u,c) > distance(v,a) + distance(u,b)
338
        (v,b) + distance(v,c)
339
                    u = v;
340
        return u;
341
342 }
    6.71
           三维几何
 1 #include <math.h>
 2 #define eps 1e-8
 3 #define zero(x) (((x)>0?(x):-(x))<eps)
 4 struct point3{double x,y,z;};
 5 struct line3{point3 a,b;};
 6 struct plane3{point3 a,b,c;};
   //计算 cross product U x V
 8
    point3 Cross(point3 u,point3 v){
 9
        point3 ret;
10
        ret.x=u.y*v.z-v.y*u.z;
        ret.y=u.z*v.x-u.x*v.z;
11
        ret.z=u.x*v.y-u.y*v.x;
12
13
        return ret;
14
    }
   //计算 dot product U . V
15
    double Dot(point3 u,point3 v){
16
17
        return u.x*v.x+u.y*v.y+u.z*v.z;
18
    }
19
   //矢量差 U - V
20
    point3 subt(point3 u,point3 v){
        point3 ret;
21
22
        ret.x=u.x-v.x;
23
        ret.y=u.y-v.y;
24
        ret.z=u.z-v.z;
25
        return ret;
26
   }
27
   //取平面法向量
    point3 pvec(plane3 s){
28
29
        return Cross(subt(s.a,s.b),subt(s.b,s.c));
30
31
    point3 pvec(point3 s1,point3 s2,point3 s3){
32
        return Cross(subt(s1,s2),subt(s2,s3));
33
    //两点距离,单参数取向量大小
```

```
double distance(point3 p1,point3 p2){
       return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)+(p1.z-p2.z)*(p1.z-p2.z)
36
       );
   }
37
   //向量大小
38
   double vlen(point3 p){
39
       return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
40
   }
41
   //判三点共线
42
   int dots_inline(point3 p1,point3 p2,point3 p3){
43
       return vlen(Cross(subt(p1,p2),subt(p2,p3)))<eps;</pre>
44
45
   }
   //判四点共面
46
   int dots_onplane(point3 a,point3 b,point3 c,point3 d){
47
       return zero(Dot(pvec(a,b,c),subt(d,a)));
48
   }
49
   //判点是否在线段上,包括端点和共线
50
   int dot_online_in(point3 p,line3 l){
51
       return zero(vlen(Cross(subt(p,l.a),subt(p,l.b))))&(l.a.x-p.x)*(l.b.x-p.x)<eps&
52
              (l.a.y-p.y)*(l.b.y-p.y) < eps&&(l.a.z-p.z)*(l.b.z-p.z) < eps;
53
   }
54
   int dot_online_in(point3 p,point3 l1,point3 l2){
55
       return zero(vlen(Cross(subt(p,l1),subt(p,l2))))&&(l1.x-p.x)*(l2.x-p.x)<eps&&
56
              (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
57
58
   }
   //判点是否在线段上,不包括端点
59
   int dot_online_ex(point3 p,line3 l){
60
       return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)||!zero(p.z-l.a.z))&&
61
               (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
62
63
   int dot_online_ex(point3 p,point3 l1,point3 l2){
64
       return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))
65
       &&
              (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
66
   }
67
   //判点是否在空间三角形上,包括边界,三点共线无意义
68
   int dot_inplane_in(point3 p,plane3 s){
70
       return zero(vlen(Cross(subt(s.a,s.b),subt(s.a,s.c)))-vlen(Cross(subt(p,s.a),subt(p,
       s.b)))-
71
                   vlen(Cross(subt(p,s.b),subt(p,s.c)))-vlen(Cross(subt(p,s.c),subt(p,s.a)
       )));
   }
72
   int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
73
74
       return zero(vlen(Cross(subt(s1,s2),subt(s1,s3)))-vlen(Cross(subt(p,s1),subt(p,s2)))
                   vlen(Cross(subt(p,s2),subt(p,s3)))-vlen(Cross(subt(p,s3),subt(p,s1))));
75
76
   }
   //判点是否在空间三角形上,不包括边界,三点共线无意义
   int dot_inplane_ex(point3 p,plane3 s){
78
79
       return dot_inplane_in(p,s)&&vlen(Cross(subt(p,s.a),subt(p,s.b)))>eps&&
80
              vlen(Cross(subt(p,s.b),subt(p,s.c)))>eps&&vlen(Cross(subt(p,s.c),subt(p,s.a)
       ))>eps;
81
   int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
82
       return dot_inplane_in(p,s1,s2,s3)&&vlen(Cross(subt(p,s1),subt(p,s2)))>eps&&
83
              vlen(Cross(subt(p,s2),subt(p,s3)))>eps&&vlen(Cross(subt(p,s3),subt(p,s1)))>
84
       eps;
85
   }
   //判两点在线段同侧,点在线段上返回 0,不共面无意义
```

```
int same_side(point3 p1,point3 p2,line3 l){
        return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))>eps
88
89
    int same_side(point3 p1,point3 p2,point3 l1,point3 l2){
90
91
        return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))>eps;
92
    //判两点在线段异侧,点在线段上返回 0,不共面无意义
93
    int opposite_side(point3 p1,point3 p2,line3 l){
94
        return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))<-</pre>
95
96
    }
    int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2){
97
        return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))<-eps;</pre>
98
99
    //判两点在平面同侧,点在平面上返回 0
100
    int same_side(point3 p1,point3 p2,plane3 s){
101
        return Dot(pvec(s), subt(p1, s.a))*Dot(pvec(s), subt(p2, s.a))>eps;
102
103
    int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
104
        return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))>eps;
105
106
    //判两点在平面异侧,点在平面上返回 0
107
    int opposite_side(point3 p1,point3 p2,plane3 s){
108
109
        return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))<-eps;</pre>
110
    int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
111
        return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))<-eps;</pre>
112
113
    //判两直线平行
114
    int parallel(line3 u,line3 v){
115
        return vlen(Cross(subt(u.a,u.b),subt(v.a,v.b)))<eps;</pre>
116
117
    int parallel(point3 u1,point3 u2,point3 v1,point3 v2){
118
        return vlen(Cross(subt(u1,u2),subt(v1,v2)))<eps;</pre>
119
120 }
121
    //判两平面平行
122
    int parallel(plane3 u,plane3 v){
        return vlen(Cross(pvec(u),pvec(v)))<eps;</pre>
123
124
125
    int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
        return vlen(Cross(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;</pre>
126
127
    //判直线与平面平行
    int parallel(line3 l,plane3 s){
129
        return zero(Dot(subt(l.a,l.b),pvec(s)));
130
131
    int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
132
        return zero(Dot(subt(l1,l2),pvec(s1,s2,s3)));
133
134 }
135
    //判两直线垂直
136
    int perpendicular(line3 u,line3 v){
137
        return zero(Dot(subt(u.a,u.b),subt(v.a,v.b)));
138
    }
    int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2){
139
140
        return zero(Dot(subt(u1,u2),subt(v1,v2)));
141
142
    //判两平面垂首
   int perpendicular(plane3 u,plane3 v){
```

```
return zero(Dot(pvec(u),pvec(v)));
144
145
    int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
146
147
        return zero(Dot(pvec(u1,u2,u3),pvec(v1,v2,v3)));
148
    //判直线与平面平行
149
    int perpendicular(line3 l,plane3 s){
150
        return vlen(Cross(subt(l.a,l.b),pvec(s)))<eps;</pre>
151
152
    int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
153
154
        return vlen(Cross(subt(l1,l2),pvec(s1,s2,s3)))<eps;</pre>
155
    }
156
    //判两线段相交,包括端点和部分重合
    int intersect_in(line3 u,line3 v){
157
        if (!dots_onplane(u.a,u.b,v.a,v.b))
158
            return 0;
159
        if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
160
            return !same_side(u.a,u.b,v)&!same_side(v.a,v.b,u);
161
        return dot_online_in(u.a,v)||dot_online_in(u.b,v)||dot_online_in(v.a,u)||
162
        dot_online_in(v.b,u);
163
    }
    int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2){
164
        if (!dots_onplane(u1,u2,v1,v2))
165
            return 0;
166
167
        if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
            return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
168
169
        return
                dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||dot_online_in(v1,u1,u2)||
170
        dot_online_in(v2,u1,u2);
171
172
    //判两线段相交,不包括端点和部分重合
    int intersect_ex(line3 u,line3 v){
173
174
        return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&opposite_side(v.a,v
        .b,u);
175
    int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2){
176
177
        return dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,
        u1,u2);
178
    //判线段与空间三角形相交,包括交于边界和(部分)包含
179
    int intersect_in(line3 l,plane3 s){
180
        return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
181
               !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
182
183
    int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
184
        return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
185
               !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
186
187
    //判线段与空间三角形相交,不包括交于边界和(部分)包含
188
    int intersect_ex(line3 l,plane3 s){
190
        return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
191
               opposite_side(s.b,s.c,l.a,l.b,s.a)&opposite_side(s.c,s.a,l.a,l.b,s.b);
192
    int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
193
        return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2,s3)&&
194
               opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2);
195
196
197
    //计算两直线交点,注意事先判断直线是否共面和平行!
    //线段交点请另外判线段相交(同时还是要判断是否平行!)
```

```
point3 intersection(line3 u,line3 v){
199
200
        point3 ret=u.a;
        double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
201
202
                 /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
203
        ret.x+=(u.b.x-u.a.x)*t;
204
        ret.y+=(u.b.y-u.a.y)*t;
205
        ret.z+=(u.b.z-u.a.z)*t;
        return ret;
206
207
    point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2){
208
        point3 ret=u1;
209
210
        double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
211
                 /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
        ret.x+=(u2.x-u1.x)*t;
212
        ret.y+=(u2.y-u1.y)*t;
213
        ret.z+=(u2.z-u1.z)*t;
214
215
        return ret;
216
    }
217
    //计算直线与平面交点,注意事先判断是否平行,并保证三点不共线!
   //线段和空间三角形交点请另外判断
218
    point3 intersection(line3 l,plane3 s){
219
220
        point3 ret=pvec(s);
        double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-l.a.z))/
221
222
                 (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
223
        ret.x=l.a.x+(l.b.x-l.a.x)*t;
        ret.y=l.a.y+(l.b.y-l.a.y)*t;
224
225
        ret.z=l.a.z+(l.b.z-l.a.z)*t;
226
        return ret;
227
    point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
228
229
        point3 ret=pvec(s1,s2,s3);
        double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z))/
230
231
                 (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
232
        ret.x=11.x+(12.x-11.x)*t;
233
        ret.y=l1.y+(l2.y-l1.y)*t;
234
        ret.z=l1.z+(l2.z-l1.z)*t;
235
        return ret;
236
    }
237
    //计算两平面交线,注意事先判断是否平行,并保证三点不共线!
238
    line3 intersection(plane3 u,plane3 v){
239
        line3 ret;
        ret.a=parallel(v.a,v.b,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(
240
        v.a, v.b, u.a, u.b, u.
241
                c);
        ret.b=parallel(v.c,v.a,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(
242
        v.c,v.a,u.a,u.b,u.
243
                c);
        return ret;
244
245
246
    line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
247
248
        ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,
        u2,u3);
249
        ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,
        u2,u3);
250
        return ret;
251
252
    //点到直线距离
    double ptoline(point3 p,line3 l){
253
```

```
return vlen(Cross(subt(p,1.a), subt(l.b,l.a)))/distance(l.a,l.b);
254
255
        double ptoline(point3 p,point3 l1,point3 l2){
256
257
                return vlen(Cross(subt(p,l1),subt(l2,l1)))/distance(l1,l2);
258
259
        //点到平面距离
        double ptoplane(point3 p,plane3 s){
260
261
                return fabs(Dot(pvec(s),subt(p,s.a)))/vlen(pvec(s));
262
        double ptoplane(point3 p,point3 s1,point3 s2,point3 s3){
263
                return fabs(Dot(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
264
265
        }
266
        //直线到直线距离
        double linetoline(line3 u,line3 v){
267
                point3 n=Cross(subt(u.a,u.b),subt(v.a,v.b));
268
                return fabs(Dot(subt(u.a,v.a),n))/vlen(n);
269
270
271
        double linetoline(point3 u1,point3 u2,point3 v1,point3 v2){
                point3 n=Cross(subt(u1,u2),subt(v1,v2));
272
                return fabs(Dot(subt(u1,v1),n))/vlen(n);
273
274
275 //两直线夹角 cos 值
        double angle_cos(line3 u,line3 v){
276
277
                return Dot(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
278
        double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2){
279
                return Dot(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(subt(v1,v2));
280
281
        //两平面夹角 cos 值
282
        double angle_cos(plane3 u,plane3 v){
283
284
                return Dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));
285
        double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
286
                return Dot(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3))/vlen(pvec(v1,v2,v3))
287
288
        //直线平面夹角 sin 值
289
290
        double angle_sin(line3 l,plane3 s){
                return Dot(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(pvec(s));
291
292
293
        double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
                return Dot(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen(pvec(s1,s2,s3));
294
295
        }
296
297
       // 球体相交
        double vol_ints(double x1, double y1, double z1, double r1, double x2, double y2,
               double z2, double r2) {
                double sum = 4.00 / 3.00 * PI * r1 * r1 * r1 + 4.00 / 3.00 * PI * r2 * r2 * r2;
299
                double ans = 0;
300
                double dis = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 - z2) 
301
                 z2)); //球心距离
302
                if (dis >= r1 + r2) //没有交到的地方
303
304
                        ans = 0;
                } else if (dis + r1 <= r2)//重合
305
306
307
                        ans = (4.00 / 3.00) * PI * r1 * r1 * r1;
                } else if (dis + r2 <= r1) {
308
                        ans = (4.00 / 3.00) * PI * r2 * r2 * r2;
309
```

```
} else //相交
310
311
             double cal = (r1 * r1 + dis * dis - r2 * r2) / (2.00 * dis * r1);
312
             double h = r1 * (1 - cal);
ans += (1.00 / 3.00) * PI * (3.00 * r1 - h) * h * h;
313
314
             cal = (r2 * r2 + dis * dis - r1 * r1) / (2.00 * dis * r2);
315
             h = r2^* (1.00 - cal);
316
             ans += (1.00 / 3.00) * PI * (3.00 * r2 - h) * h * h;
317
318
319
         return ans;
320 }
    6.72 Poly-Z
    constexpr int P = 998244353;
    using i64 = long long;
    // assume -P \ll x \ll 2P
 3
    int norm(int x) {
         if (x < 0) {
 5
             x += P;
 6
 7
         if (x >= P) {
 8
 9
             x -= P;
10
         }
11
         return x;
12
    }
13
    template<class T>
    T power(T a, int b) {
14
15
         T res = 1;
         for (; b; b /= 2, a *= a) {
    if (b % 2) {
16
17
18
                  res *= a;
19
20
21
         return res;
22 }
23
    struct Z {
24
         int x;
25
         Z(int x = 0) : x(norm(x)) {}
26
         int val() const {
27
             return x;
28
         Z operator-() const {
29
             return Z(norm(P - x));
30
31
         Z inv() const {
32
             assert(x != 0);
33
             return power(*this, P - 2);
34
35
         Z &operator*=(const Z &rhs) {
36
             x = i64(x) * rhs.x % P;
37
             return *this;
38
39
         Z &operator+=(const Z &rhs) {
40
             x = norm(x + rhs.x);
41
             return *this;
42
43
         Z & operator -= (const Z & rhs) {
44
```

```
45
             x = norm(x - rhs.x);
             return *this;
46
47
         Z &operator/=(const Z &rhs) {
48
             return *this *= rhs.inv();
49
50
         friend Z operator*(const Z &lhs, const Z &rhs) {
51
             Z res = lhs;
52
             res *= rhs;
53
54
             return res;
55
         friend Z operator+(const Z &lhs, const Z &rhs) {
56
             Z res = lhs;
57
             res += rhs;
58
             return res;
59
60
         friend Z operator-(const Z &lhs, const Z &rhs) {
61
             Z res = lhs;
62
             res -= rhs;
63
             return res;
64
65
         friend Z operator/(const Z &lhs, const Z &rhs) {
66
             Z res = lhs;
67
             res /= rhs;
68
69
             return res;
70
         }
    };
71
72
    std::vector<int> rev;
73
    std::vector<Z> roots{0, 1};
74
    void dft(std::vector<Z> &a) {
75
         int n = a.size();
76
77
         if (int(rev.size()) != n) {
78
             int k = __builtin_ctz(n) - 1;
79
80
             rev.resize(n);
             for (int i = 0; i < n; i++) {
81
82
                 rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
83
         }
84
85
         for (int i = 0; i < n; i++) {
86
87
             if (rev[i] < i) {</pre>
88
                 std::swap(a[i], a[rev[i]]);
89
90
         if (int(roots.size()) < n) {</pre>
91
             int k = __builtin_ctz(roots.size());
92
             roots.resize(n);
93
             while ((1 << k) < n) {
94
                 Z = power(Z(3), (P - 1) >> (k + 1));
95
96
                 for (int i = 1 \ll (k - 1); i < (1 \ll k); i++) {
                      roots[2 * i] = roots[i];
97
                      roots[2 * i + 1] = roots[i] * e;
98
99
100
                 k++;
101
             }
102
         for (int k = 1; k < n; k *= 2) {
103
```

```
for (int i = 0; i < n; i += 2 * k) {
104
                 for (int j = 0; j < k; j++) {
105
                      \tilde{Z} u = a[i + j];
106
                      Z_v = a[i + j + k] * roots[k + j];
107
                      a[i + j] = u + v;
108
109
                      a[i + j + k] = u - v;
                 }
110
             }
111
         }
112
113 }
    void idft(std::vector<Z> &a) {
114
         int n = a.size();
115
         std::reverse(a.begin() + 1, a.end());
116
117
         dft(a);
         Z inv = (1 - P) / n;
118
         for (int i = 0; i < n; i++) {
119
             a[i] *= inv;
120
         }
121
122
    }
    struct Poly {
123
         std::vector<Z> a;
124
         Poly() {}
125
         Poly(const std::vector<Z> &a) : a(a) {}
126
127
         int size() const {
128
             return a.size();
129
         void resize(int n) {
130
             a.resize(n);
131
132
         Z operator[](int idx) const {
133
             if (idx < 0 \mid idx >= size()) {
134
                 return 0;
135
             }
136
             return a[idx];
137
138
         Z & operator [] (int idx) {
139
             return a[idx];
140
141
         Poly mulxk(int k) const {
142
             auto b = a;
143
             b.insert(b.begin(), k, 0);
144
             return Poly(b);
145
146
147
         Poly modxk(int k) const {
             k = std::min(k, size());
148
             return Poly(std::vector<Z>(a.begin(), a.begin() + k));
149
150
         Poly divxk(int k) const {
151
             if (size() <= k) {
152
                 return Poly();
153
154
155
             return Poly(std::vector<Z>(a.begin() + k, a.end()));
156
         friend Poly operator+(const Poly &a, const Poly &b) {
157
             std::vector<Z> res(std::max(a.size(), b.size()));
158
159
             for (int i = 0; i < int(res.size()); i++) {
160
                 res[i] = a[i] + b[i];
161
             return Poly(res);
162
```

```
163
         friend Poly operator-(const Poly &a, const Poly &b) {
164
             std::vector<Z> res(std::max(a.size(), b.size()));
165
             for (int i = 0; i < int(res.size()); i++) {</pre>
166
                  res[i] = a[i] - b[i];
167
168
             return Poly(res);
169
170
         friend Poly operator*(Poly a, Poly b) {
171
             if (a.size() == 0 || b.size() == 0) {}
172
                  return Poly();
173
174
             }
             int sz = 1, tot = a.size() + b.size() - 1;
175
             while (sz < tot)</pre>
176
                 sz *= 2;
177
             a.a.resize(sz);
178
179
             b.a.resize(sz);
             dft(a.a);
180
             dft(b.a);
181
             for (int i = 0; i < sz; ++i) {
182
                 a.a[i] = a[i] * b[i];
183
184
             idft(a.a);
185
             a.resize(tot);
186
187
             return a;
188
         friend Poly operator*(Z a, Poly b) {
189
             for (int i = 0; i < int(b.size()); i++) {</pre>
190
                 b[i] *= a;
191
192
193
             return b;
194
         friend Poly operator*(Poly a, Z b) {
195
             for (int i = 0; i < int(a.size()); i++) {</pre>
196
                 a[i] *= b;
197
198
199
             return a;
200
         Poly &operator+=(Poly b) {
201
202
             return (*this) = (*this) + b;
203
         Poly &operator-=(Poly b) {
204
             return (*this) = (*this) - b;
205
206
207
         Poly &operator*=(Poly b) {
208
             return (*this) = (*this) * b;
209
         Poly deriv() const {
210
             if (a.empty()) {
211
212
                  return Poly();
213
214
             std::vector<Z> res(size() - 1);
             for (int i = 0; i < size() - 1; ++i) {
215
                 res[i] = (i + 1) * a[i + 1];
216
217
218
             return Poly(res);
219
         Poly integr() const {
220
             std::vector<Z> res(size() + 1);
221
```

```
222
             for (int i = 0; i < size(); ++i) {
223
                 res[i + 1] = a[i] / (i + 1);
224
             return Poly(res);
225
226
227
         Poly inv(int m) const {
             Poly x({a[0].inv()});
228
229
             int k = 1;
             while (k < m) {
230
231
                 k *= 2;
232
                 x = (x * (Poly(\{2\}) - modxk(k) * x)).modxk(k);
233
234
             return x.modxk(m);
235
         Poly log(int m) const {
236
             return (deriv() * inv(m)).integr().modxk(m);
237
238
239
         Poly exp(int m) const {
240
             Poly x(\{1\});
             int k = 1;
241
             while (k < m) {</pre>
242
                 k *= 2;
243
                 x = (x * (Poly(\{1\}) - x.log(k) + modxk(k))).modxk(k);
244
             }
245
246
             return x.modxk(m);
247
         Poly sqrt(int m) const {
248
249
             Poly x(\{1\});
250
             int k = 1;
251
             while (k < m) {
                 k *= 2;
252
                 x = (x + (modxk(k) * x.inv(k)).modxk(k)) * ((P + 1) / 2);
253
254
255
             return x.modxk(m);
256
         Poly mulT(Poly b) const {
257
258
             if (b.size() == 0) {
259
                  return Poly();
             }
260
             int n = b.size();
261
             std::reverse(b.a.begin(), b.a.end());
262
             return ((*this) * b).divxk(n - 1);
263
264
265
         std::vector<Z> eval(std::vector<Z> x) const {
             if (size() == 0) {
266
                  return std::vector<Z>(x.size(), 0);
267
268
             const int n = std::max(int(x.size()), size());
269
             std::vector<Poly> q(4 * n);
270
             std::vector<Z> ans(x.size());
271
272
             x.resize(n);
273
             std::function<void(int, int, int)> build = [&](int p, int l, int r) {
274
                 if (r - l == 1) {
                      q[p] = Poly({1, -x[l]});
275
                 } else {
276
277
                      int m = (l + r) / 2;
                      build(2 * p, l, m);
build(2 * p + 1, m, r);
278
279
                      q[p] = q[2 * p] * q[2 * p + 1];
280
```

```
}
281
282
             };
             build(1, 0, n);
283
             std::function<void(int, int, int, const Poly &)> work = [&](int p, int l, int r
284
        , const Poly &num) {
                 if (r - l == 1) {
285
286
                      if (l < int(ans.size())) {</pre>
                          ans[l] = num[0];
287
288
                 } else {
289
                      int m = (l + r) / 2;
290
                     work(2 * p, l, m, num.mulT(q[2 * p + 1]).modxk(m - l));
291
                     work(2 * p + 1, m, r, num.mulT(q[2 * p]).modxk(r - m));
292
293
                 }
294
             };
             work(1, 0, n, mulT(q[1].inv(n)));
295
296
             return ans;
297
        }
298 };
           多项式快速幂
    6.73
    #include "bits/stdc++.h"
   using namespace std;
 3
    const int N = 2e6 + 10;
 4
 5
 6
    const int P = 998244353, g3 = (P + 1) / 3;
 7
    int pow(int a,int b) {
 8
 9
         int r = 1;
        while(b) {
10
11
             if(b \& 1) r = (ll)r * a % P;
12
             a = (11)a * a % P;
             b >>= 1;
13
14
15
        return r;
16
    }
17
    namespace poly {
18
        int rev[N];
19
        void NTT(int *A, int n, int inv) {
             for(int i = 0; i < n; ++i)
20
                 if(i < rev[i]) swap(A[i], A[rev[i]]);</pre>
21
22
             for(int mid = 1; mid < n; mid <<= 1) {</pre>
                 int tmp = pow(inv == 1 ? 3 : g3, (P - 1) / (mid << 1));</pre>
23
                 for(int j = 0; j < n; j += (mid << 1)) {
24
25
                      int omega = 1;
                      for(int k = 0; k < mid; ++k, omega = (11)omega * tmp % P) {
26
                          int x = A[j + k], y = (ll)omega * A[j + k + mid] % P;
27
                          A[j + k] = (x + y) \% P;
28
29
                          A[j + k + mid] = (ll)(x - y + P) \% P;
                      }
30
                 }
31
32
             if(inv == 1) return;
33
             int invn = pow(n, P - 2);
34
             for(int i = 0; i < n; ++i)
35
                 A[i] = (ll)A[i] * invn % P;
36
```

```
37
        void Inv(int *a, int *b, int n) {
38
            static int B[N], A[N];
39
            b[0] = pow(a[0], P - 2);
40
            int len, lim;
41
            for(len = 1; len < (n << 1); len <<= 1) {
42
                 lim = len << 1;
43
                 for(int i = 0; i < len; i++)
44
                     A[i] = a[i], B[i] = b[i];
45
46
                 for(int i = 0; i < lim; i++)</pre>
                     rev[i] = (rev[i >> 1] >> 1) | ((i & 1) ? len : 0);
47
48
                 NTT(A, lim, 1), NTT(B, lim, 1);
                 for(int i = 0; i < lim; i++)</pre>
49
                     b[i] = ((2LL - 1LL * A[i] * B[i] % P) * B[i] % P + P) % P;
50
                 NTT(b, lim, -1);
51
                 for(int i = len; i < lim; i++)</pre>
52
53
                     b[i] = 0;
54
            for(int i = 0; i < len; i++)</pre>
55
                 A[i] = B[i] = 0;
56
            for(int i = n; i < len; i++)</pre>
57
                 b[i] = 0;
58
59
        void derivative(int *a, int *b, int n) {
60
61
            b[n - 1] = 0;
            for(int i = 1; i < n; ++i)</pre>
62
                 b[i - 1] = (ll)a[i] * i % P;
63
64
        void inter(int *a, int *b, int n) {
65
66
            *b = 0;
            for(int i = n - 1; i >= 0; --i)
67
                 b[i + 1] = a[i] * (ll)pow(i + 1, P - 2) % P;
68
69
        void ln(int *a, int *b, int n) {
70
            static int F[N];
71
            derivative(a, F, n);
72
73
            Inv(a, b, n);
74
            int lim = 1;
            while(lim < (n << 1)) lim <<= 1;</pre>
75
            for(int i = 1; i < lim; i++)</pre>
76
                 rev[i] = (rev[i >> 1] >> 1) | ((i & 1) ? (lim >> 1) : 0);
77
            for(int i = n; i < lim; ++i)
78
                 b[i] = F[i] = 0;
79
80
            NTT(F, lim, 1), NTT(b, lim, 1);
            for(int i = 0; i < lim; ++i)</pre>
81
82
                 F[i] = (ll)b[i] * F[i] % P;
            NTT(F, lim, 0);
83
            inter(F, b, n);
84
            for(int i = n; i < lim; ++i)</pre>
85
86
                 b[i] = 0;
87
88
        void exp(int*a, int*F, int n) {
            if(n == 1)
89
                 *F = 1;
90
            else {
91
92
                 \exp(a, F, n + 1 >> 1);
                 static int F0[N], A[N];
93
94
                 for(int i = 0; i <= (n << 1); ++i)
95
                     F0[i] = 0, A[i] = a[i];
```

```
ln(F, F0, n);
96
                 int lim = 1;
97
                 while(lim < (n << 1)) lim <<= 1;</pre>
98
                 for(int i = 1; i < lim; i++)</pre>
99
                      rev[i] = (rev[i >> 1] >> 1) | ((i & 1) ? (lim >> 1) : 0);
100
                 for(int i = n; i < lim; ++i)</pre>
101
                      A[i] = 0;
102
                 NTT(A, lim, 1), NTT(F0, lim, 1), NTT(F, lim, 1);
103
                 for(int i = 0; i < lim; ++i)</pre>
104
                      F[i] = F[i] * (A[i] + 1LL - F0[i] + P) % P;
105
                 NTT(F, lim, 0);
106
107
                 for(int i = n; i < lim; ++i)</pre>
                      F[i] = 0;
108
109
             }
         }
110
111
112
    using namespace poly;
113
114 int a[N], b[N];
115
116 void solve() {
      int n, m, k; cin >> n >> m >> k;
117
       for(int i = 0;i < n; i++) cin >> a[i];
118
119
       ln(a, b, m);
120
       for(int i = 0; i < m; i++) b[i] = b[i] * k % mod;
121
      exp(b, a, m);
122
       for(int i = 0;i < m; i++) cout << a[i] << " ";
123 }
    6.74 Geometry
 1
    using Point = std::complex<double>;
 2
 3 #define x real
 4 #define y imag
 5
 6
    double dot(const Point &a, const Point &b) {
 7
         return (std::conj(a) * b).x();
 8
    }
 9
10
    double cross(const Point &a, const Point &b) {
         return (std::conj(a) * b).y();
11
    }
12
13
    struct Line {
14
         Point a;
15
         Point b;
16
         Line(const Point &a, const Point &b) : a(a), b(b) {}
17
    };
18
19
    Point rotate(const Point &a) {
         return Point(-a.y(), a.x());
21
22
23
24 int sgn(const Point &a) {
         return a.y() > 0 \mid | (a.y() == 0 && a.x() > 0) ? 1 : -1;
25
26
    }
27
```

```
bool onLeft(const Point &a, const Line &l) {
        return cross(l.b - l.a, a - l.a) > 0;
29
   }
30
31
   Point intersection(const Line &l1, const Line &l2) {
32
        return l1.a + (l1.b - l1.a) * (cross(l2.b - l2.a, l1.a - l2.a) / cross(l2.b - l2.a,
33
        l1.a - l1.b));
   }
34
35
   std::vector<Point> hp(std::vector<Line> lines) {
36
37
        std::sort(lines.begin(), lines.end(), [&](auto l1, auto l2) {
            auto d1 = 11.b - 11.a;
38
            auto d2 = 12.b - 12.a;
39
40
            if (sgn(d1) != sgn(d2)) {
41
                return sgn(d1) == 1;
42
43
44
            return cross(d1, d2) > 0;
45
        });
46
47
        std::deque<Line> ls;
48
        std::deque<Point> ps;
49
        for (auto 1 : lines) {
50
51
            if (ls.empty()) {
                ls.push_back(l);
52
                continue;
53
            }
54
55
            while (!ps.empty() && !onLeft(ps.back(), 1)) {
56
57
                ps.pop_back();
                ls.pop_back();
58
            }
59
60
            while (!ps.empty() && !onLeft(ps[0], 1)) {
61
62
                ps.pop_front();
63
                ls.pop_front();
64
            }
65
            if (cross(l.b - l.a, ls.back().b - ls.back().a) == 0) {
66
                if (dot(l.b - l.a, ls.back().b - ls.back().a) > 0) {
67
68
                    if (!onLeft(ls.back().a, l)) {
69
70
                        assert(ls.size() == 1);
                        ls[0] = 1;
71
72
73
                    continue;
74
                return {};
75
76
            }
77
78
            ps.push_back(intersection(ls.back(), 1));
79
            ls.push_back(l);
        }
80
81
        while (!ps.empty() && !onLeft(ps.back(), ls[0])) {
82
83
            ps.pop_back();
84
            ls.pop_back();
85
        }
```

```
86     if (ls.size() <= 2) {
87         return {};
88     }
89     ps.push_back(intersection(ls[0], ls.back()));
90
91     return std::vector<Point>(ps.begin(), ps.end());
92 }
```

## 7 图论

## 7.1 graph

```
template <typename T>
1
   class graph {
3
   public:
        struct edge {
4
            int from;
5
6
            int to;
7
            T cost;
8
        };
9
10
        const int n;
        std::vector<edge> edges;
11
        std::vector<std::vector<int>> g;
12
13
        graph(int _n) : n(_n), g(n) {}
14
15
        virtual int add(int from, int to, T const) = 0;
16
   };
17
18
   template <typename T>
19
   class digraph : public graph<T> {
20
   public:
21
22
        using graph<T>::edges;
23
        using graph<T>::g;
24
        using graph<T>::n;
25
26
        digraph(int _n) : graph<T>(_n) {}
27
28
        int add(int from, int to, T cost = 1) {
29
            assert(0 \le from \&\& from < n \&\& 0 \le to \&\& to < n);
30
            int id = (int) edges.size();
31
            g[from].push_back(id);
            edges.push_back({from, to, cost});
32
            return id;
33
34
        }
35
36
        digraph<T> reverse() const {
37
            digraph<T> rev(n);
38
            for (auto &e : edges) {
39
                rev.add(e.to, e.from, e.cost);
40
41
            return rev;
42
        }
   };
43
44
45
   template <typename T>
46
   class undigraph : public graph<T> {
47
48
   public:
        using graph<T>::edges;
49
50
        using graph<T>::g;
51
        using graph<T>::n;
52
        undigraph(int _n) : graph<T>(_n) {}
53
54
        int add(int from, int to, T cost = 1) {
55
```

```
assert(0 \le from \&\& from < n \&\& 0 \le to \&\& to < n);
56
            int id = (int) edges.size();
57
            g[from].push_back(id);
58
            g[to].push_back(id);
59
            edges.push_back({from, to, cost});
60
            return id;
61
62
        }
  };
63
   7.2 isBipartiteGraph
   template <typename T>
1
   bool isBipartiteGraph(const graph<T>& g) {
        std::vector<int> color(g.n);
3
        bool flag = true;
4
        std::function<bool(int, int)> dfs = [&](int u, int x) -> bool {
5
            for (int id : g.g[u]) {
6
                auto& e = g.edges[id];
7
8
                int to = e.from ^ e.to ^ u;
                if (!color[to]) {
9
                    dfs(to, 3 - x);
10
11
                if (color[to] == color[u]) {
12
13
                    flag = false;
14
            }
15
16
        for (int i = 0; i < g.n; i++) {
17
            if (!color[i]) {
18
19
                dfs(i, 1);
20
21
22
        return flag;
   }
23
   7.3
        hungry
   template <typename T>
   int hungry(const digraph<T>& g) {
3
        std::vector<bool> was(g.n);
        std::vector<int> match(g.n, -1);
4
        std::function<bool(int)> dfs = [&](int u) -> bool {
5
            for (int id : g.g[u]) {
6
                auto& e : g.edges[id];
7
                int to = e.to;
8
                if (!was[to]) {
9
10
                    was[to] = true;
                    if (match[to] == -1 || dfs(match[to])) {
11
                        match[to] = u;
12
13
                         return true;
14
                    }
15
                }
16
            }
17
            return false;
18
        };
19
        int ans = 0;
20
```

```
for (int i = 0; i < n; i++) {
21
22
           vis.assign(g.n, false);
23
           if (dfs(i)) ans++;
24
25
       return ans;
26 }
   7.4 KM
1
   template <typename T>
3 class hungarian { // km
   public :
4
5
     int n;
     std::vector<int> matchx; // 左集合对应的匹配点
6
7
     std::vector<int> matchy; // 右集合对应的匹配点
                               // 连接右集合的左点
8
     std::vector<int> pre;
9
     std::vector<bool> visx;
                               // 拜访数组 左
     std::vector<bool> visy;
                               // 拜访数组 右
10
     std::vector<T> lx;
11
     std::vector<T> ly;
12
     std::vector<vector<T> > g;
13
     std::vector<T> slack;
14
15
     T inf;
     T res;
16
17
     std::queue<int> q;
18
     int org_n;
19
     int org_m;
20
     hungarian(int _n, int _m) {
21
22
       org_n = n;
       org_m = _m;
23
       n = max(n, m);
24
       inf = numeric_limits<T>::max();
25
26
       res = 0;
27
       g = vector<vector<T> >(n, vector<T>(n));
28
       matchx = vector < int > (n, -1);
29
       matchy = vector<int>(n, -1);
30
       pre = vector<int>(n);
31
       visx = vector<bool>(n);
32
       visy = vector<bool>(n);
33
       lx = vector<T>(n, -inf);
34
       ly = vector<T>(n)
35
       slack = vector<T>(n);
36
37
     void addEdge(int u, int v, int w) {
38
39
       g[u][v] = max(w, 0); // 负值还不如不匹配 因此设为0不影响
40
41
     bool check(int v) {
42
       visy[v] = true;
43
       if (matchy[v] != -1) {
44
45
         q.push(matchy[v]);
46
         visx[matchy[v]] = true; // in S
47
         return false;
48
       // 找到新的未匹配点 更新匹配点 pre 数组记录着"非匹配边"上与之相连的点
49
```

```
while (v != -1) {
50
          matchy[v] = pre[v];
51
          swap(v, matchx[pre[v]]);
52
53
54
        return true;
55
56
      void bfs(int i) {
57
        while (!q.empty()) {
58
59
           q.pop();
60
61
        q.push(i);
        visx[i] = true;
62
        while (true) {
63
          while (!q.empty()) {
64
             int u = q.front();
65
66
             q.pop();
             for (int v = 0; v < n; v++) {
67
               if (!visy[v]) {
68
                 T delta = lx[u] + ly[v] - g[u][v];
69
                 if (slack[v] >= delta) {
70
                   pre[v] = u;
71
                   if (delta) {
72
73
                     slack[v] = delta;
74
                   } else if (check(v)) { // delta=0 代表有机会加入相等子图 找增广路
75
                                            // 找到就return 重建交错树
76
                     return;
                   }
77
                 }
78
               }
79
            }
80
81
82
           // 没有增广路 修改顶标
          T a = inf;
83
           for (int j = 0; j < n; j++) {
84
             if (!visy[j]) {
85
86
               a = min(a, slack[j]);
87
             }
88
           for (int j = 0; j < n; j++) {
89
             if (visx[j]) { // S
90
               lx[j] -= a;
91
92
             if (visy[j]) { // T
93
94
               ly[j] += a;
             } else { // T'
95
96
               slack[j] -= a;
             }
97
98
99
           for (int j = 0; j < n; j++) {
100
             if (!visy[j] \&\& slack[j] == 0 \&\& check(j)) {
101
               return;
             }
102
103
          }
104
        }
105
106
107
      void solve() {
108
        // 初始顶标
```

```
for (int i = 0; i < n; i++) {
109
          for (int j = 0; j < n; j++) {
110
            lx[i] = max(lx[i], g[i][j]);
111
112
113
114
        for (int i = 0; i < n; i++) {
115
           fill(slack.begin(), slack.end(), inf);
116
           fill(visx.begin(), visx.end(), false);
117
118
          fill(visy.begin(), visy.end(), false);
          bfs(i);
119
120
        }
121
        // custom
122
        for (int i = 0; i < n; i++) {
123
           if (g[i][matchx[i]] > 0) {
124
125
            res += g[i][matchx[i]];
126
          } else {
127
            matchx[i] = -1;
          }
128
        }
129
        // cout << res << "\n";
130
        // for (int i = 0; i < org_n; i++) {
131
132
        // cout << matchx[i] + 1 << " ";
        // }
133
        // cout << "\n";
134
135
    };
136
         galeShapley
 1
    #include<iostream>
 2
    using namespace std;
 3
    const int N=4;
 4
 5
    void GaleShapley(const int (&man)[MAX][MAX], const int (&woman)[MAX][MAX], int (&match)
 6
        [XAM]
 7
        int wm[MAX][MAX];
                              // wm[i][j]: rank from girl i to boy j
 8
        int choose[MAX];
                             // choose[i]: current boyfriend of girl i
        int manIndex[MAX]; //
 9
                                  manIndex[i]: how many girls that have rejected boy i
10
        int i, j;
11
        int w, m;
        for (i = 0; i < N; i++) {
12
13
             match[i] = -1;
             choose[i] = -1;
14
            manIndex[i] = 0;
15
             for (j = 0; j < N; j++)
16
                 wm[i][woman[i][j]] = j;
17
        }
18
19
        bool bSingle = false;
20
21
        while (!bSingle) {
22
            bSingle = true;
             for (i = 0; i < N; i++) {
23
                 if (match[i] != -1) // boy i already have a girlfriend
24
25
                     continue;
                 bSingle = false;
26
```

```
j = manIndex[i]++; // the jth girl that boy i like most
27
                w = man[i][j];
28
                                   // current girl w's boyfriend
                m = choose[w];
29
                if (m == -1 | l | wm[w][i] < wm[w][m]) { // if girl w prefer boy i}
30
                    match[i] = w;
31
                     choose[w] = i;
32
                     if (m != -1)
33
                         match[m] = -1;
34
35
                }
            }
36
37
        }
38
   }
39
40
   void Print(const int(&match)[MAX], int N) {
41
        for (int i = 0; i < N; i++)
42
            cout << i << " " << match[i] << endl;
43
44
   }
45
46
   int main(){
47
        int man[N][N]={
48
            {2,3,1,0},
49
50
            {2,1,3,0},
51
            \{0,2,3,1\},
            {1,3,2,0},
52
        };
53
        int woman[N][N]={
54
            {0,3,2,1},
55
            {0,1,2,3},
56
            \{0,2,3,1\},\
57
            {1,0,3,2},
58
59
        };
60
        int match[N];
61
        GaleShapley(man, woman, match);
62
63
        Print(match,N);
64
        return 0;
65
66
  }
   7.6 bellman-ford
   template <typename T>
   std::vector<std::vector<T>> bellman_ford(const graph<T>& g, int st) {
3
        std::vector<T> dist(q.n, std::numeric_limits<T>::max());
        dist[st] = 0;
4
5
        // Relax all edges IVI - 1 times. A simple
6
        // shortest path from src to any other vertex can have
7
        // at-most |V| - 1 edges
8
        for (int i = 1; i < g.n; i++) {
9
10
            for (auto& e : g.edges) {
                int from = e.from, to = e.to, cost = e.cost;
11
                if (dist[from] != std::numeric_limits<T>::max() && dist[from] + cost < dist</pre>
12
       [to]) {
                     dist[to] = dist[from] + e.from;
13
14
                }
```

```
15
           }
       }
16
17
       for (auto& e : g.edges) {
18
19
            int from = e.from, to = e.to, cost = e.cost;
            if (dist[from] != std::numeric_limits<T>::max() && dist[from] + cost < dist[to</pre>
20
       } ([
                // Graph contains negative weight cycle
21
22
                return {};
23
            }
24
       }
25
26
       return dist;
27 }
   7.7
         dijkatra
   template <typename T>
   std::vector<T> dijkstra(const graph<T>& g, int st) {
       assert(0 \le st && st < g.n);
3
       std::vector<T> dist(g.n, std::numeric_limits<T>::max());
4
5
       std::priority_queue<std::pair<T, int>, std::vector<std::pair<T, int>>, std::greater
       dist[st] = 0;
6
       q.emplace(dist[st], st);
7
       while (!q.empty()) {
8
            T expected = q.top().first;
9
            int u = q.top().second; q.pop();
10
            if (dist[u] != expected) {
11
                continue ;
12
            }
13
            for (int id : g.g[u]) {
14
                auto& e = g.edges[id];
15
                int to = e.from ^ e.to ^ u;
16
                if (dist[to] > dist[u] + e.cost) {
17
18
                    dist[to] = dist[u] + e.cost;
                    q.emplace(dist[to], to);
19
                }
20
            }
21
22
       return dist;
23
24
       // returns numeric_limits<T>::max() if there's no path
25
  }
   7.8 floyd
   template <typename T>
   void floyd(std::vector<std::vector<T>>& dist) {
3
       for (int k = 0; k < dist.size(); k++) {
            for (int i = 0; i < dist.size(); i++) {</pre>
4
                for (int j = 0; j < dist.size(); j++) {</pre>
5
                    dist[i][j] = std::min(dist[i][j], dist[i][k], dist[k][j]);
6
7
                }
8
           }
9
       }
10 }
```

### 7.9 spfa

```
template <typename T>
   std::vector<T> spfa(const graph<T>& g, int st) {
3
        std::vector<T> dist(g.n, std::numeric_limits<T>::max());
4
        std::vector<bool> vis(q.n);
5
        std::vector<int> cnt(q.n);
        std::vector<int> x(1, st);
6
        dist[st] = 0; vis[st] = true;
7
        for (int ptr = 0; ptr < x.size(); ptr++) {</pre>
8
            int u = x[ptr];
9
            vis[u] = false;
10
            for (int id : g.g[u]) {
11
                auto& e = g.edges[id];
12
                int to = e.from ^ e.to ^ u;
13
                if (dist[to] > dist[u] + e.cost) {
14
                     dist[to] = dist[u] + e.cost;
15
                     if (!vis[to]) {
16
17
                         cnt[to]++;
18
                         vis[to] = true;
                         if (cnt[to] >= g.n) {
19
                             return std::vector<T>();
20
21
22
                         x.push_back(to);
                     }
23
24
                }
25
            }
26
27
        return dist;
28
  }
    7.10 Kruskal
   template <typename T>
    std::vector<int> find_mst(const undigraph<T> &g, T& ans) {
3
        std::vector<int> order(q.edges.size());
        iota(order.begin(), order.end(), 0);
4
5
        sort(order.begin(), order.end(), [&g](int a, int b) {
            return g.edges[a].cost < g.edges[b].cost;</pre>
6
7
        DSU d(g.n);
8
9
        std::vector<int> ans_list;
10
        ans = 0;
11
        for (int id : order) {
12
            auto &e = g.edges[id];
13
            if (!d.same(e.from, e.to)) {
                d.merge(e.from, e.to);
14
                ans_list.push_back(id);
15
16
                ans += e.cost;
17
            }
18
        }
```

// returns edge ids of minimum "spanning" forest

### 7.11 prim

return ans\_list;

19

20 21 }

```
template <typename T>
   bool find_mst(const undigraph<T> &g, T& ans) {
       std::vector<bool> vis(g.n);
3
       std::priority_queue<std::pair<T, int>, std::vector<std::pair<T, int>>, std::greater
4
       q.push({0, 0});
5
       int cnt = 0; ans = 0;
6
       while (!q.empty() && cnt < g.n) {</pre>
7
           T expected = q.top().first;
8
            int u = q.top().second; q.pop();
9
10
            if (vis[u]) continue;
11
           vis[u] = true;
           ans += expected; cnt++;
12
            for (int id : g.g[u]) {
13
                auto &e = g.edges[id];
14
                int to = e.from ^ e.to ^ u;
15
                if (!vis[to]) {
16
17
                    q.push({e.cost, to});
                }
18
19
           }
20
21
       return cnt == g.n;
22
       // returns false if there's not connected
23 }
   7.12 topSort
   template <typename T>
   std::vector<int> find_topsort(const digraph<T> &g) {
       std::vector<int> deg(g.n, 0);
3
       for (int id = 0; id < (int) g.edges.size(); id++) {</pre>
4
            deg[g.edges[id].to]++;
5
6
7
       std::vector<int> x;
8
       for (int i = 0; i < g.n; i++) {
9
            if (deg[i] == 0) {
                x.push_back(i);
10
11
            }
       }
12
13
       for (int ptr = 0; ptr < (int) x.size(); ptr++) {</pre>
14
            int i = x[ptr];
15
            for (int id : g.g[i]) {
16
                auto &e = g.edges[id];
17
                int to = e.to;
18
                if (--deg[to] == 0) {
19
20
                    x.push_back(to);
21
22
            }
       }
23
24
       if ((int) x.size() != q.n) {
25
26
            return std::vector<int>();
27
       }
28
       return x;
29 }
```

#### 7.13 scc

template <typename T>

```
std::vector<int> find_scc(const digraph<T> &g, int &cnt) {
3
       digraph<T> g_rev = g.reverse();
4
       std::vector<int> order;
5
       std::vector<bool> was(g.n, false);
       std::function<void(int)> dfs1 = [&](int v) {
6
7
            was[v] = true;
            for (int id : g.g[v]) {
8
                auto &e = g.edges[id];
9
                int to = e.to;
10
                if (!was[to]) {
11
                    dfs1(to);
12
13
            }
14
            order.push_back(v);
15
16
17
       for (int i = 0; i < g.n; i++) {
18
            if (!was[i]) {
                dfs1(i);
19
20
21
       std::vector<int> c(g.n, -1);
22
       std::function<void(int)> dfs2 = [&](int v) {
23
            for (int id : g_rev.g[v]) {
24
25
                auto &e = g_rev.edges[id];
                int to = e.to;
26
                if (c[to] == -1) {
27
28
                    c[to] = c[v];
29
                    dfs2(to);
30
                }
31
            }
       };
32
       cnt = 0;
33
       for (int id = g.n - 1; id >= 0; id--) {
34
            int i = order[id];
35
            if (c[i] != -1) {
36
37
                continue;
38
            c[i] = cnt++;
39
40
            dfs2(i);
41
42
       return c;
43
  }
   7.14 cycles
1 template <typename T>
   std::vector<std::vector<int>> find_cycles(const graph<T> &g, int bound_cnt = 1 << 30,
       int bound_size = 1 << 30) {</pre>
       std::vector<int> was(g.n, -1);
3
       std::vector<int> st;
4
5
       std::vector<std::vector<int>> cycles;
6
       int total_size = 0;
       std::function<void(int, int)> dfs = [&](int v, int pe) {
7
8
            if ((int) cycles.size() >= bound_cnt || total_size >= bound_size) {
9
                return ;
```

```
}
10
            was[v] = (int) st.size();
11
            for (int id : g.g[v]) {
12
13
                if (id == pe) {
                    continue;
14
15
                auto &e = g.edges[id];
16
                int to = e.from ^ e.to ^ v;
17
                if (was[to] >= 0) {
18
                    std::vector<int> cycle(1, id);
19
                    for (int j = was[to]; j < (int) st.size(); j++) {</pre>
20
21
                         cycle.push_back(st[j]);
                    }
22
                    cycles.push_back(cycle);
23
                    total_size += (int) cycle.size();
24
                    if ((int) cycles.size() >= bound_cnt || total_size >= bound_size) {
25
26
                         return;
                    }
27
28
                    continue;
29
                if (was[to] == -1) {
30
                    st.push_back(id);
31
32
                    dfs(to, id);
                    st.pop_back();
33
34
                }
35
            }
            was[v] = -2;
36
37
        for (int i = 0; i < g.n; i++) {
38
39
            if (was[i] == -1) {
                dfs(i, -1);
40
            }
41
42
        }
        return cycles;
43
        // cycles are given by edge ids, all cycles are simple
44
        // breaks after getting bound_cnt cycles or total_size >= bound_size
45
        // digraph: finds at least one cycle in every connected component (if not broken)
46
47
        // undigraph: finds cycle basis
   }
48
49
   template <typename T>
50
   std::vector<int> edges_to_vertices(const graph<T> &g, const std::vector<int> &
       edge_cycle) {
52
        int sz = (int) edge_cycle.size();
        std::vector<int> vertex_cycle;
53
        if (sz <= 2) {
54
            vertex_cycle.push_back(g.edges[edge_cycle[0]].from);
55
            if (sz == 2) {
56
                vertex_cycle.push_back(g.edges[edge_cycle[0]].to);
57
            }
58
59
        } else {
60
            for (int i = 0; i < sz; i++) {
61
                int j = (i + 1) \% sz;
                auto &e = g.edges[edge_cycle[i]];
62
                auto &other = g.edges[edge_cycle[j]];
63
64
                if (other.from == e.from || other.to == e.from) {
                    vertex_cycle.push_back(e.to);
65
66
                } else {
                    vertex_cycle.push_back(e.from);
67
```

```
}
68
            }
69
70
       return vertex_cycle;
71
72
       // only for simple cycles!
73 }
   7.15
          ternaryRingCount
   template <typename T>
   int ternaryRingCount(const digraph<T>& g) {
3
       std::vector<int> d(g.n);
       for (auto& e : g.edges) {
4
            int from = e.from, to = e.to;
5
            d[from]++;
6
7
            d[to]++;
8
9
       digraph<T>& ng(q.n);
       for (auto& e : g.edges) {
10
            int from = e.from, to = e.to;
11
            if (d[from] < d[to] \mid | (d[from] == d[to] && from > to)) std::swap(from, to);
12
            ng.add(from, to);
13
14
15
       int ans = 0;
       std::vector<bool> was(nq.n);
16
       for (int u = 0; u < ng.n; u++) {
17
            for (int id : ng.g[u]) {
18
                auto& e = ng.edges[id];
19
                int to = e.to;
20
21
                was[to] = u;
22
23
            for (int id1 : ng.g[u]) {
                auto& e1 = ng.edges[id1];
24
                int to1 = e1.to;
25
                for (int id2 : ng.g[to1]) {
26
                    auto& e2 = nq.q[id2];
27
28
                    int to 2 = e2.to;
29
                    if (was[to2] == u) ans++;
30
                }
            }
31
32
33
       return ans;
34
  }
   7.16 eulerian-path
   template <typename T>
   std::vector<int> find_eulerian_path(const graph<T> &g, int &root) {
3
       // in_deg and out_deg are fake for undigraph!
       std::vector<int> in_deg(q.n, 0);
4
       std::vector<int> out_deg(g.n, 0);
5
       int cnt_edges = 0;
6
       for (int id = 0; id < (int) g.edges.size(); id++) {</pre>
7
8
            cnt_edges++;
            auto &e = g.edges[id];
9
            out_deg[e.from]++;
10
            in_deg[e.to]++;
11
```

```
12
        }
        root = -1;
13
        int odd = 0;
14
        for (int i = 0; i < g.n; i++) {
15
            if ((in_deg[i] + out_deg[i]) % 2 == 1) {
16
17
                odd++;
                if (root == -1 || out_deg[i] - in_deg[i] > out_deg[root] - in_deg[root]) {
18
                     root = i;
19
20
                }
            }
21
22
        if (odd > 2) {
23
24
            root = -1;
25
            return std::vector<int>();
26
        if (root == -1) {
27
            root = 0;
28
            while (root < g.n && in_deg[root] + out_deg[root] == 0) {</pre>
29
30
                root++;
31
            if (root == g.n) {
32
                // an empty path
33
                root = 0;
34
35
                return std::vector<int>();
36
            }
        }
37
        std::vector<bool> used(g.edges.size(), false);
38
        std::vector<int> ptr(g.n, 0);
39
        std::vector<int> balance(g.n, 0);
40
        std::vector<int> res(cnt_edges);
41
42
        int stack_ptr = 0;
        int write_ptr = cnt_edges;
43
        int v = root;
44
        while (true) {
45
            bool found = false;
46
            while (ptr[v] < (int) g.g[v].size()) {</pre>
47
48
                int id = g.g[v][ptr[v]++];
49
                if (used[id]) {
                continue;
50
                }
51
                used[id] = true;
52
                res[stack_ptr++] = id;
53
                auto &e = g.edges[id];
54
55
                balance[v]++;
                v ^= e.from ^ e.to;
56
57
                balance[v]--;
                found = true;
58
                break;
59
60
61
            if (!found) {
62
                if (stack_ptr == 0) {
63
                     break;
64
                int id = res[--stack_ptr];
65
                res[--write_ptr] = id;
66
                auto &e = g.edges[id];
67
68
                v ^= e.from ^ e.to;
69
            }
        }
70
```

```
71
        int disbalance = 0;
        for (int i = 0; i < g.n; i++) {
72
            disbalance += abs(balance[i]);
73
74
        if (write_ptr != 0 || disbalance > 2) {
75
76
            root = -1;
            return std::vector<int>();
77
78
79
        return res;
80 }
   7.17 twoSat
   class twosat {
1
2
   public:
3
        digraph<int> g;
        int n;
4
5
6
        twosat(int _n) : g(digraph<int>(2 * _n)), n(_n) {}
7
        inline void add(int x, int value_x) {
8
            assert(0 \le x \&\& x < n);
9
10
            assert(0 <= value_x && value_x <= 1);
            g.add(2 * x + (value_x ^ 1), 2 * x + value_x);
11
12
13
        inline void add(int x, int value_x, int y, int value_y) {
14
            assert(0 \le x && x < n && 0 \le y && y < n);
15
            assert(0 \le value_x \& value_x \le 1 \& 0 \le value_y \& value_y \le 1);
16
            g.add(2 * x + (value_x ^ 1), 2 * y + value_y);
17
            g.add(2 * y + (value_y ^1), 2 * x + value_x);
18
        }
19
20
        inline std::vector<int> solve() {
21
22
            int cnt;
            std::vector<int> c = find_scc(q, cnt);
23
            std::vector<int> res(n);
24
            for (int i = 0; i < n; i++) {
25
                if (c[2 * i] == c[2 * i + 1]) {
26
                    return std::vector<int>();
27
28
29
                res[i] = (c[2 * i] < c[2 * i + 1]);
30
31
            return res;
32
        }
33 };
   7.18 maxAssignment
1 template<class T>
   struct MaxAssignment {
2
        public:
3
4
            T solve(int nx, int ny, std::vector<std::vector<T>> a) {
5
                assert(0 \le nx \& nx \le ny);
6
                assert(int(a.size()) == nx)
                for (int i = 0; i < nx; ++i) {
 7
8
                    assert(int(a[i].size()) == ny);
```

```
for (auto x : a[i])
9
                          assert(x >= 0);
10
                 }
11
12
13
                 auto update = [\&](int x) {
                      for (int y = 0; y < ny; ++y) {
14
                          if (lx[x] + ly[y] - a[x][y] < slack[y]) {
15
                               slack[y] = lx[x] + ly[y] - a[x][y];
16
                               slackx[y] = x;
17
18
                          }
19
                      }
20
                 };
21
22
                 costs.resize(nx + 1);
                 costs[0] = 0;
23
                 lx.assign(nx, std::numeric_limits<T>::max());
24
25
                 ly.assign(ny, 0);
26
                 xy.assign(nx, -1);
                 yx.assign(ny, -1);
27
28
                 slackx.resize(ny);
                 for (int cur = 0; cur < nx; ++cur) {</pre>
29
                      std::queue<int> que;
30
                      visx.assign(nx, false);
31
32
                      visy.assign(ny, false);
33
                      slack.assign(ny, std::numeric_limits<T>::max());
                      p.assign(nx, -1);
34
35
                      for (int x = 0; x < nx; ++x) {
36
                          if (xy[x] == -1) {
37
38
                               que.push(x);
39
                               visx[x] = true;
                               update(x);
40
41
                          }
                      }
42
43
                      int ex, ey;
44
                      bool found = false;
45
46
                     while (!found) {
                          while (!que.empty() && !found) {
47
                               auto x = que.front();
48
                               que.pop();
49
                               for (int y = 0; y < ny; ++y) {
   if (a[x][y] == lx[x] + ly[y] && !visy[y]) {</pre>
50
51
52
                                        if (yx[y] == -1) {
                                            ex = x;
53
                                            ey = y;
54
                                            found = true;
55
                                            break;
56
                                        }
57
                                        que.push(yx[y]);
58
59
                                        p[yx[y]] = x;
60
                                        visy[y] = visx[yx[y]] = true;
61
                                        update(yx[y]);
                                   }
62
                               }
63
                          }
if (found)
64
65
66
                               break;
67
```

```
T delta = std::numeric_limits<T>::max();
68
                          for (int y = 0; y < ny; ++y)
69
                              if (!visy[y])
70
                                   delta = std::min(delta, slack[y]);
71
                          for (int x = 0; x < nx; ++x)
72
73
                              if (visx[x])
                                   lx[x] -= delta;
74
                          for (int y = 0; y < ny; ++y) {
75
76
                              if (visy[y]) {
77
                                  ly[y] += delta;
                              } else {
78
79
                                  slack[y] -= delta;
80
                          }
81
                          for (int y = 0; y < ny; ++y) {
82
                              if (!visy[y] \&\& slack[y] == 0) {
83
84
                                  if (yx[y] == -1) {
                                       ex = slackx[y];
85
86
                                       ey = y;
                                       found = true;
87
                                       break;
88
89
                                  que.push(yx[y]);
90
                                  p[yx[y]] = slackx[y];
91
92
                                  visy[y] = visx[yx[y]] = true;
93
                                  update(yx[y]);
                              }
94
                          }
95
                     }
96
97
                      costs[cur + 1] = costs[cur];
98
                      for (int x = ex, y = ey, ty; x != -1; x = p[x], y = ty) {
99
100
                          costs[cur + 1] += a[x][y];
                          if (xy[x] != -1)
101
                              costs[cur + 1] -= a[x][xy[x]];
102
                          ty = xy[x];
103
104
                          xy[x] = y;
105
                          yx[y] = x;
                     }
106
                 }
107
                 return costs[nx];
108
109
             std::vector<int> assignment() {
110
111
                 return xy;
112
             std::pair<std::vector<T>, std::vector<T>> labels() {
113
                 return std::make_pair(lx, ly);
114
115
             std::vector<T> weights() {
116
117
                 return costs;
118
             }
119
         private:
120
             std::vector<T> lx, ly, slack, costs;
121
             std::vector<int> xy, yx, p, slackx;
122
             std::vector<bool> visx, visy;
123 };
```

#### 7.19 tarjan

```
template <typename T>
   class tarjan {
2
   public:
3
        digraph<T> &g;
4
        int tim;
5
        std::vector<int> dfn;
6
        std::vector<int> low;
7
        std::vector<bool> vis;
8
9
        std::stack<int> s;
10
        std::set<int> cutVertexs;
        std::vector<std::array<int, 2>> cutBridges;
11
12
        std::vector<std::vector<int>> cycles;
13
        int root;
        tarjan(digraph<T> \&_g) : g(_g) {
14
            dfn.resize(g.n);
15
            low.resize(g.n);
16
17
            vis.resize(g.n);
            tim = 1;
18
        }
19
20
        void dfs(int u) {
21
22
            s.push(u);
            dfn[u] = low[u] = tim++;
23
            vis[u] = true;
24
25
            int child = 0;
26
            for (int id : g.g[u]) {
27
                auto &e = g.edges[id];
28
                int to = e.to;
                if (!dfn[to]) {
29
30
                    dfs(to);
31
                    low[u] = std::min(low[u], low[to]);
32
                    // if u is not root and low value of one of its child is more than dfn
       value of u
                    // if u is root of DFS tree and has two or more children
33
                    if (low[to] >= dfn[u]) {
34
35
                         child++;
36
                         if (u != root || child >= 2) {
37
                             cutVertexs.insert(u);
38
39
                    }
                    // if the lowest vertex reachable from subtree under v is below u is
40
       DFS tree, the u-b is a bridge
                    if (low[to] > dfn[u]) {
41
                         cutBridges.push_back({std::min(u, to), std::max(u, to)});
42
43
                } else if (vis[to]) {
44
                    low[u] = std::min(low[u], dfn[to]);
45
                }
46
            }
47
48
            // find a strongly connected component
49
            if (dfn[u] == low[u]) {
50
                int from, cnt = 0;
51
                std::vector<int> cycle;
52
53
                do {
                    from = s.top();
54
                    cycle.push_back(from);
55
```

```
vis[from] = false;
56
57
                   s.pop();
                   cnt++;
58
                   // TODO ... num of connected component = cnt
59
               } while (from != u);
60
               cycles.push_back(cycle);
61
           }
62
       }
63
64
       void solve() {
65
           std::vector<std::vector<int>> cycles;
66
           for (int i = 0; i < g.n; i++) {
67
               if (!dfn[i]) {
68
                   root = i;
69
70
                   dfs(i);
               }
71
           }
72
       }
73
74
  };
          biconnected-components
   7.20
   template <typename T>
   std::vector<int> find_bicone(dfs_undigraph<T> &g, int &cnt) {
3
       q.dfs_all();
       std::vector<int> vertex_comp(g.n);
4
       cnt = 0;
5
       for (int i : g.order) {
6
           if (g.pv[i] == -1 \mid | g.min_depth[i] == g.depth[i]) {
7
8
               vertex_comp[i] = cnt++;
           } else {
9
10
               vertex_comp[i] = vertex_comp[g.pv[i]];
11
12
13
       return vertex_comp;
14
         differenceConstraints
   7.21
1
   /*
2
3
   差分约束是解决这样一类问题
   给出n个形如x[j] - x[i] <= k的式子,xx[n] - x[1]的最大/最小值
   最大值—>把所有式子整理为x[j] - x[i] <= k,从i向j连一条边权为k的边,跑最短路
   最小值—>把所有式子整理为x[j] - x[i] >= k, 从i向j连一条边权为k的边, 跑最长路
6
7
   注意初始化 有时候需要超级源点0
8
9
  //dfs跑差分约束最短路
10
   template <typename T>
   bool differenceConstraints(const graph<T>& q) {
12
       std::vector<bool> was(g.n);
13
14
       std::vector<T> dist(g.n, std::numeric_limits<T>::max());
       std::function<bool(int)> spfa = [&](int u) -> bool {
15
16
           was[u] = true;
           for (int id : g.g[u]) {
17
               auto& e = g.edges[id];
18
```

```
int to = e.from ^ e.to ^ to;
19
20
                if (dist[u] + e.cost < dist[to]) {</pre>
                    if (was[to]) return false;
21
                    dist[to] = dist[u] + e.cost;
22
23
                    if (!spfa(to)) return false;
                }
24
25
           }
           was[u] = false;
26
27
           return true;
28
       };
29
       return spfa(0);
30 }
   7.22 AHU
1 //用来判断两棵树是否同构
   // AHU:判断两棵树是否是同构
3 //同构:在更换节点的标号之后两棵树能完全相同
5 const int N = 1e5 + 5;
  const int maxn = N << 1;</pre>
6
7
   int n;
8
   struct Edge {
9
10
       int v, nxt;
11 } e[maxn << 1];</pre>
int head[maxn], sz[maxn], f[maxn], maxv[maxn], tag[maxn], tot, Max;
vector<int> center[2], L[maxn], subtree_tags[maxn];
14
  void addedge(int u, int v) {
       e[tot].v = v;
15
16
       e[tot].nxt = head[u];
       head[u] = tot++;
17
18
       e[tot].v = u;
19
       e[tot].nxt = head[v];
       head[v] = tot++;
20
21
   }
22
   void dfs_size(int u, int fa) {
23
24
       sz[u] = 1;
25
       \max v[u] = 0;
       for (int i = head[u]; i; i = e[i].nxt) {
26
           int v = e[i].v;
27
           if (v == fa) continue;
28
           dfs_size(v, u);
29
30
           sz[u] += sz[v];
           maxv[u] = max(maxv[u], sz[v]);
31
32
       }
33
   }
34
   void dfs_center(int rt, int u, int fa, int id) {
35
       maxv[u] = max(maxv[u], sz[rt] - sz[u]);
36
       if (Max > maxv[u]) {
37
38
           center[id].clear();
39
           Max = maxv[u];
40
       if (Max == maxv[u]) center[id].push_back(u);
41
       for (int i = head[u]; i; i = e[i].nxt) {
42
           int v = e[i].v;
43
```

```
if (v == fa) continue;
44
             dfs_center(rt, v, u, id);
45
46
    }
47
48
    int dfs_height(int u, int fa, int depth) {
49
         L[depth].push_back(u);
50
         f[u] = fa;
51
         int h = 0;
52
53
         for (int i = head[u]; i; i = e[i].nxt) {
             int v = e[i].v;
54
55
             if (v == fa) continue;
56
             h = max(h, dfs_height(v, u, depth + 1));
57
58
         return h + 1;
    }
59
60
    void init(int n) {
61
         for (int i = 1; i \le 2 * n; i++) head[i] = 0;
62
         tot = 1;
63
         center[0].clear();
64
         center[1].clear();
65
66
67
         int u, v;
68
         for (int i = 1; i <= n - 1; i++) {//在这里输入第一棵树的边
             scanf("%d %d", &u, &v);
69
             addedge(u, v);
70
71
         dfs_size(1, -1);
72
73
         Max = n;
74
         dfs_center(1, 1, -1, 0);
75
76
         for (int i = 1; i <= n - 1; i++) {//在这里输入第二棵树的边
             scanf("%d %d", &u, &v);
77
             addedge(u + n, v + n);
78
         }
79
80
         dfs_size(1 + n, -1);
81
         Max = n;
82
         dfs_center(1 + n, 1 + n, -1, 1);
    }
83
84
    bool cmp(int u, int v) { return subtree_tags[u] < subtree_tags[v]; }</pre>
85
86
    bool rootedTreeIsomorphism(int rt1, int rt2) {
   for (int i = 0; i <= 2 * n + 1; i++) L[i].clear(), subtree_tags[i].clear();</pre>
87
88
         int h1 = dfs_height(rt1, -1, 0);
89
         int h2 = dfs_height(rt2, -1, 0);
90
         if (h1 != h2) return false;
91
         int h = h1 - 1;
92
         for (int j = 0; j < (int)L[h].size(); <math>j++) tag[L[h][j]] = 0;
93
94
         for (int i = h - 1; i >= 0; i--) {
95
             for (int j = 0; j < (int)L[i + 1].size(); j++) {
                 int v = L[i + 1][j];
96
                 subtree_tags[f[v]].push_back(tag[v]);
97
             }
98
99
100
             sort(L[i].begin(), L[i].end(), cmp);
101
             for (int j = 0, cnt = 0; j < (int)L[i].size(); j++) {
102
```

```
if (j && subtree_tags[L[i][j]] != subtree_tags[L[i][j - 1]]) ++cnt;
103
104
                 tag[L[i][j]] = cnt;
            }
105
106
107
        return subtree_tags[rt1] == subtree_tags[rt2];
108
    }
109
    bool treeIsomorphism() {
110
        if (center[0].size() == center[1].size()) {
111
             if (rootedTreeIsomorphism(center[0][0], center[1][0])) return true;
112
            if (center[0].size() > 1)
113
114
                 return rootedTreeIsomorphism(center[0][0], center[1][1]);
115
        return false;
116
117
    }
118
    int main() {
119
        int T;
120
        scanf("%d", &T);
121
        while (T--) {
122
            scanf("%d", &n);
123
            init(n);
124
            puts(treeIsomorphism() ? "YES" : "NO");
125
126
127
        return 0;
128 }
    7.23 Astar
    #include "bits/stdc++.h"
 2
    using namespace std;
 3
    //A*
 4
   //用来计算点A到点B的第k短的路径
 5
   const int MAXN = 55;
 6
    const int MAXM = MAXN * MAXN;
 7
 8
 9 int dis[MAXN];
10 int n, m, k, a, b, cnt;
    bool hav = false;
11
12
13
    namespace G1{//反图
        int to[MAXM], val[MAXM], head[MAXN], nxt[MAXM], cnt;
14
15
        bool vis[MAXN];
16
        void AddEdge(int u, int v, int w) {
17
            cnt++;
18
            to[cnt] = v;
19
20
            val[cnt] = w;
            nxt[cnt] = head[u];
21
22
            head[u] = cnt;
        }
23
24
25
        void Spfa(int s, int t) {//SPFA+SLF跑最短路
            memset(dis, 0x7f, sizeof(dis)); dis[s] = 0;
26
            deque<int> q; q.push_back(s); vis[s] = true;
27
28
            while (!q.empty()) {
                 int u = q.front(); q.pop_front(); vis[u] = false;
29
```

```
for (int i = head[u]; i; i = nxt[i]) {
30
31
                     int v = to[i];
                     if (dis[v] > dis[u] + val[i]) {
32
33
                         dis[v] = dis[u] + val[i];
                         if (!vis[v]) {
34
35
                              vis[v] = true;
                              if (!q.empty() && dis[v] < dis[q.front()]) {</pre>
36
                                  q.push_front(v);
37
                              } else {
38
                                  q.push_back(v);
39
                              }
40
41
                         }
                     }
42
                }
43
            }
44
        }
45
   }
46
47
48
   namespace G2{//原图
        int to[MAXM], val[MAXM], nxt[MAXM], head[MAXN], cnt;
49
50
        void AddEdge(int u, int v, int w) {
51
52
            cnt++;
            to[cnt] = v;
53
54
            val[cnt] = w;
55
            nxt[cnt] = head[u];
            head[u] = cnt;
56
57
58
59
        struct Data{//当前位置,走过的距离,s->now->t总距离,走的步骤
            int now, pas, val;
60
61
            vector<int> route;
62
            bool operator < (const Data &b) const {return val > b.val;}
63
            */
64
            bool operator < (const Data &b) const {//重载
65
                if (val != b.val) return val > b.val;
66
67
                int sz = min(route.size(), b.route.size());
                for (int i = 0; i < sz; i++) {
68
                     if (route[i] != b.route[i]) return route[i] > b.route[i];
69
70
                return route.size() > b.route.size();
71
            }
72
        };
73
74
        void Astar(int s, int t) {//A*
75
            priority_queue<Data> q;
76
            Data st;
77
            st.now = s; st.pas = 0; st.val = dis[s]; st.route = vector<int>{s};
78
79
            q.push(st);
80
            vector<int> vec;
81
            while (!q.empty()) {
82
                Data u = q.top(); q.pop();
                if (u.now == t) {//更新路径数
83
84
                     :: cnt++;
85
                     if (:: cnt == k) {//最终答案
86
                          cout << u.route[0];</pre>
                         for (int i = 1, sz = u.route.size(); i < sz; i++)
    cout << '-' << u.route[i];</pre>
87
88
```

```
89
                          hav = true;
90
                          return;
                     }
91
92
                 for (int i = head[u.now]; i; i = nxt[i]) {//广搜
93
                      int v = to[i];
94
95
                     vec = u.route;
                     bool visit = false;
96
                      for (int j = 0, sz = vec.size(); j < sz; j++) {//记录是否重复经过
97
                          if (vec[j] == v) {
98
99
                              visit = true;
100
                              break;
                          }
101
                     }
102
                      if (visit) continue;
103
                     Data nx = u;
104
105
                     nx.now = v;
106
                     nx.pas = u.pas + val[i];
                     nx.val = dis[v] + nx.pas;
107
108
                     nx.route.push_back(v);
                      q.push(nx);
109
                 }
110
             }
111
112
         }
113 }
114
    int main() {
115
         cin >> n >> m >> k >> a >> b;
116
         for (int i = 1; i <= m; i++) {
117
             int u, v, w;
118
119
             cin >> u >> v >> w;
120
             G1 :: AddEdge(v, u, w);
             G2 :: AddEdge(u, v, w);
121
122
         }
123
         G1 :: Spfa(b, a);
124
         G2 :: Astar(a, b);
125
         if (!hav) cout << "No" << endl;</pre>
126
         return 0;
127 }
    7.24 dinic
    template <typename T>
    class flow_graph {
 3
    public:
 4
         static constexpr T eps = (T) 1e-9;
 5
         struct edge {
 6
             int from;
 7
             int to;
 8
 9
             T c;
10
             Tf;
         };
11
12
13
         const int n;
         std::vector<edge> edges;
14
         std::vector<std::vector<int>> g;
15
16
         int st;
```

```
int fin;
17
        T flow;
18
19
        flow_graph(int _n, int _st, int _fin) : n(n), st(_st), fin(_fin) {
20
21
            assert(0 <= st && st < n && 0 <= fin && fin < n && st != fin);
22
            g.resize(n);
23
            flow = 0;
        }
24
25
        void clear_flow() {
26
27
            for (const edge& e : edges) {
28
                e.f = 0;
29
30
            flow = 0;
        }
31
32
        int add(int from, int to, T forward_cap, T backward_cap) {
33
34
            assert(0 \le from && from < n && 0 <= to && to < n);
            int id = (int) edges.size();
35
            g[from].push_back(id);
36
            edges.push_back({from, to, forward_cap, 0});
37
            g[to].push_back(id + 1);
38
            edges.push_back({to, from, backward_cap, 0});
39
40
            return id;
41
        }
42
   };
43
   template <typename T>
44
   class dinic {
45
   public:
46
47
        flow_graph<T> &g;
        std::vector<int> ptr;
48
        std::vector<int> d;
49
        std::vector<int> q;
50
51
        dinic(flow\_graph < T > \&\_g) : g(\_g) {
52
53
            ptr.resize(g.n);
54
            d.resize(g.n);
55
            q.resize(g.n);
        }
56
57
        bool expath() {
58
            fill(d.begin(), d.end(), -1);
59
60
            q[0] = g.fin;
            d[g.fin] = 0;
61
            int beg = 0, end = 1;
62
            while (beg < end) {</pre>
63
                int i = q[beg++];
64
                for (int id : g.g[i]) {
65
                     const auto &e = g.edges[id];
66
67
                     const auto &back = g.edges[id ^ 1];
68
                     if (back.c - back.f > g.eps && d[e.to] != -1) {
                         d[e.to] = d[i] + 1;
69
                         if (e.to == g.st) {
70
                             return true;
71
72
73
                         q[end++] = e.to;
74
                     }
75
                }
```

```
}
76
77
              return false;
78
79
         T dfs(int v, int w) {
80
              if (v == g.fin) {
81
                  return w;
82
              }
83
              int &j = ptr[v];
84
             while (j \ge 0) {
85
                  int id = g.g[v][j];
86
87
                  const auto& e = g.edges[id];
                  if (e.c - e.f > g.eps \&\& d[e.to] == d[v] - 1) {
88
                       T t = dfs(e.to, std::min(e.c - e.f), w);
89
                       if (t > g.eps) {
    g.edges[id].f += t;
90
91
                           g.edges[id \land 1].f -= t;
92
93
                           return t;
                       }
94
                  }
95
96
                  j--;
              }
97
              return 0;
98
99
         }
100
         T max_flow() {
101
              while (expath()) {
102
                  for (int i = 0; i < g.n; i++) {
103
                       ptr[i] = (int) g.g[i].size() - 1;
104
105
                  T big_add = 0;
106
                  while (true) {
107
                       T add = dfs(g.st, std::numeric_limits<T>::max());
108
                       if (add \leftarrow g.eps) {
109
                         break;
110
111
112
                       big_add += add;
113
                  if (big_add <= g.eps) {</pre>
114
                       break;
115
116
                  g.flow += big_add;
117
118
119
              return g.flow;
120
         }
121
122
         std::vector<bool> min_cut() {
             max_flow();
123
              std::vector<bool> ret(g.n);
124
125
              for (int i = 0; i < g.n; i++) {
126
                  ret[i] = (d[i] != -1);
127
              }
128
              return ret;
129
         }
130 };
```

#### 7.25 ISAP

```
struct ISAP {
2
        const static int N = ...;//node size
3
        struct Edge {
            int from, to, cap, flow;
4
            bool operator < (const Edge &rhs) const {</pre>
5
                return from < rhs.from || (from == rhs.from && to < rhs.to);</pre>
6
7
        };
8
9
        int n, m, s, t;
10
        vector<Edge> edges;
        vector<int> g[N];
11
12
        bool vis[N];
        int dep[N], cur[N], p[N], num[N];
13
14
        void addEdge(int from, int to, int cap) {
15
            edges.push_back(Edge{from, to, cap, 0});
16
            edges.push_back(Edge{to, from, 0, 0});
17
18
            m = edges.size();
            g[from].push_back(m - 2);
19
20
            g[to].push_back(m - 1);
        }
21
22
23
        bool bfs() {
24
            memset(vis, 0, sizeof(vis));
25
            queue<int> q; q.push(t); vis[t] = 1, dep[t] = 0;
            while (!q.empty()) {
26
27
                int u = q.front(); q.pop();
                for (auto &v: g[u]) {
28
                     Edge &e = edges[v \land 1];
29
                     if (!vis[e.from] && e.cap > e.flow) {
30
                         dep[e.from] = dep[u] + (vis[e.from] = 1);
31
32
                         q.push(e.from);
33
                     }
                }
34
            }
35
            return vis[s];
36
37
        }
38
        void init(int siz) {
39
            n = siz;
40
            for (int i = 0; i < siz; i++) g[i].clear();</pre>
41
            edges.clear();
42
43
44
        int augment() {
45
46
            int u = t, a = INF;
            while (u != s) {
47
                Edge &e = edges[p[u]];
48
                a = min(a, e.cap - e.flow);
49
50
                u = edges[p[u]].from;
51
            }
52
            u = t;
            while (u != s) {
53
                edges[p[u]].flow += a;
54
                edges[p[u] ^1].flow -= a;
55
56
                u = edges[p[u]].from;
57
58
            return a;
        }
59
```

```
60
        int maxFlow(int S, int T) {
61
            s = S, t = T;
62
            int flow = 0; bfs();
63
            memset(num, 0, sizeof(num));
64
            for (int i = 0; i < n; i++) num[dep[i]]++;</pre>
65
            int u = S;
66
            memset(cur, 0, sizeof(cur));
67
            while (dep[S] < n) {</pre>
68
                 if (u == T) {
69
                     flow += augment();
70
71
                     u = S;
                 }
72
                 int ok = 0;
73
                 for (int i = cur[u]; i < g[u].size(); i++) {</pre>
74
                     Edge &e = edges[g[u][i]];
75
                     if (e.cap > e.flow && dep[u] == dep[e.to] + 1) {
76
77
                         ok = 1;
                         p[e.to] = g[u][i];
78
79
                         cur[u] = i;
80
                         u = e.to;
                         break;
81
                     }
82
83
                 if (!ok) {
84
                     int mn = n - 1;
85
                     for (int i = 0; i < g[u].size(); i++) {</pre>
86
                         Edge &e = edges[g[u][i]];
87
                         if (e.cap > e.flow) mn = min(mn, dep[e.to]);
88
89
                     if (--num[dep[u]] == 0) break;
90
                     num[dep[u] = mn + 1]++;
91
                     cur[u] = 0;
92
                     if (u != S) u = edges[p[u]].from;
93
                 }
94
            }
95
96
            return flow;
97
        }
98
   } flow;
99
   7.26 mcmf
   struct MCMF {
2
        struct Edge {
3
        int from, to, cap, flow, cost;
        Edge(int u, int v, int c, int f, int cc)
4
            : from(u), to(v), cap(c), flow(f), cost(cc) {}
5
        };
6
7
        static constexpr int INF = 1e9;
8
        int n, m;
9
10
        std::vector<Edge> edges;
11
        std::vector<std::vector<int>> G;
12
        std::vector<int> inq;
13
        std::vector<int> d;
14
        std::vector<int> p;
        std::vector<int> a;
15
```

```
16
17
        MCMF(int n) : n(n), G(n), inq(n), d(n), p(n), a(n) {}
18
19
        void add(int from, int to, int cap, int cost) {
20
            edges.emplace_back(Edge(from, to, cap, 0, cost));
21
            edges.emplace_back(Edge(to, from, 0, 0, -cost));
22
            m = int(edges.size());
23
            G[from].emplace_back(m - 2);
24
            G[to].emplace_back(m - 1);
25
26
        }
27
        bool spfa(int s, int t, int &flow, int &cost) {
28
            for (int i = 1; i < n; ++i) d[i] = INF;</pre>
29
            inq.assign(n, 0);
30
            d[s] = 0;
31
            inq[s] = 1;
32
33
            p[s] = 0;
            std::queue<int> q;
34
            a[s] = INF;
35
            q.push(s);
36
            while (!q.empty()) {
37
                int u = q.front();
38
39
                q.pop();
40
                inq[u] = 0;
                for (int i = 0; i < int(G[u].size()); ++i) {</pre>
41
                     Edge &e = edges[G[u][i];
42
                     if (e.cap > e.flow && d[e.to] > d[u] + e.cost) {
43
                         d[e.to] = d[u] + e.cost;
44
                         p[e.to] = G[u][i];
45
                         a[e.to] = std::min(a[u], e.cap - e.flow);
46
                         if (!inq[e.to]) {
47
48
                             q.push(e.to);
                             inq[e.to] = 1;
49
                         }
50
                     }
51
52
                }
53
            if (d[t] == INF) return false;
54
            flow += a[t];
55
            cost += d[t] * a[t];
56
            for (int u = t; u != s; u = edges[p[u]].from) {
57
                edges[p[u]].flow += a[t];
58
59
                edges[p[u] ^1].flow -= a[t];
60
            return true;
61
62
        }
63
        int MincostMaxflow(int s, int t, int &cost) {
64
65
            int flow = 0;
66
            cost = 0;
67
            while (spfa(s, t, flow, cost));
            return flow;
68
69
        }
   };
70
```

#### 7.27 conclusion.md

# 8 其他

## 8.1 simpleMO

```
// 0(n*sqrt(n))
   const int N = 1e5 + 10;
3
   int a[N], cnt[N], ans[N];
4
  int belong[N];
5
6
7
   struct Q {
8
        int 1, r, id;
9
   }q[N];
10
   int Size, bnum;
11
12
   bool cmp(Q a, Q b) {
13
        return (belong[a.l] ^ belong[b.l]) ? belong[a.l] < belong[b.l] : belong[a.l] & 1 ?</pre>
14
       a.r < b.r : a.r > b.r;
   }
15
16
   int now = 0;
17
18
   inline void add(int pos) {
19
        if(!cnt[a[pos]]) now++;
20
        ++cnt[a[pos]];
21
22
   }
23
24
   inline void del(int pos) {
25
        --cnt[a[pos]];
26
        if(!cnt[a[pos]]) --now;
27
   }
28
29
  int main() {
30
        int n, m;
31
        cin >> n >> m;
        Size = sqrt(n);
32
33
        bnum = ceil((double)n / Size);
        for(int i = 1;i <= bnum; i++) {</pre>
34
35
            for(int j = (i - 1) * Size + 1; j <= i * Size; j++) {
36
                belong[j] = i;
            }
37
38
        for(int i = 1;i <= n; i++) cin >> a[i];
39
40
        for(int i = 1; i <= m; i++) {
41
            cin >> q[i].l >> q[i].r;
            q[i].id = i;
42
43
        sort(q + 1, q + m + 1, cmp);
44
45
        int l = 1, r = 0;
        for(int i = 1;i <= m; i++) {
46
47
            int ql = q[i].l, qr = q[i].r;
            while(r < qr) add(++r);
48
            while(r > qr) del(r--);
49
50
            while(l < ql) del(l++);</pre>
            while(l > ql) add(--l);
51
            ans[q[i].id] = now;
52
53
        for(int i = 1;i <= m; i++) cout << ans[i] << endl;
54
```

55 }

## 8.2 modifyMO

```
//带修莫队模板题
1
   //查询[ql, qr]间不同颜色数量,带修改
3
  // 0(n^{5/3})
4
5
6
  int n, m;
7
  int c[N], cnt[N];
   int belong[N], size, totq, totm;
   int ans[N], res;
9
10
   struct Query {
11
12
       int l, r, t, id;
       bool operator < (const Query &rhs) const {</pre>
13
           return belong[l] ^ belong[rhs.l] ? belong[l] < belong[rhs.l] :</pre>
14
                  (belong[r] ^ belong[rhs.r] ? belong[r] < belong[rhs.r] : t < rhs.t);</pre>
15
16
   } q[N];
17
18
   struct Modify {
19
20
       int pos, val;
   } modify[N];
21
22
23
   void add(int x) {
24
       if (!cnt[c[x]]) res++;
25
       cnt[c[x]]++;
   }
26
27
28
   void del(int x) {
       cnt[c[x]]--;
29
30
       if (!cnt[c[x]]) res--;
31
   }
32
33
   void upd(int x, int ql, int qr) {
34
       int pos = modify[x].pos;
35
       if (ql <= pos && pos <= qr) {
36
           cnt[c[pos]]--; if (!cnt[c[pos]]) res--;
37
           if (!cnt[modify[x].val]) res++; cnt[modify[x].val]++;
38
       39
   }
40
41
   int main() {
42
   #ifdef ACM_LOCAL
43
       freopen("input.in", "r", stdin);
44
       freopen("output.out", "w", stdout);
45
   #endif
46
       scanf("%d%d", &n, &m);
47
       for (int i = 1; i <= n; i++) scanf("%d", &c[i]);</pre>
48
       for (int i = 1; i <= m; i++) {
49
           char op[10]; scanf("%s", op);
50
           if (op[0] == 'Q') {
51
               int ql, qr; scanf("%d%d", &ql, &qr); totq++;
52
53
               q[totq] = Query{ql, qr, totm, totq};
           }
54
```

```
55
           else {
                int pos, val; scanf("%d%d", &pos, &val); totm++;
56
57
                modify[totm] = Modify{pos, val};
58
           }
       }
59
60
       //size = N ^ (2 / 3), (N * totm) ^ (1 / 3)
61
       size = ceil(pow(n, (long double)2.0 / 3)); int num = ceil((long double)n / size);
62
       for (int i = 1, j = 1; i \le num; i++)
63
            while (j <= i * size && j <= n)
64
                belong[j++] = i;
65
66
67
       sort(q + 1, q + 1 + totq);
68
       int l = 1, r = 0, t = 0;
69
       for (int i = 1; i <= totq; i++) {
70
            int ql = q[i].l, qr = q[i].r, qt = q[i].t;
71
72
           while (l < ql) del(l++);
           while (l > ql) add(--l);
73
           while (r < qr) add(++r);
74
           while (r > qr) del(r--);
75
           while (t < qt) upd(++t, ql, qr);</pre>
76
           while (t > qt) upd(t--, ql, qr);
77
78
           ans[q[i].id] = res;
79
       }
80
       for (int i = 1; i <= totq; i++) printf("%d\n", ans[i]);</pre>
81
82
       return 0;
83
84 }
   8.3 rollbackMO
1 //问题可以莫队(询问可以离线,不带修改)
 2 //区间伸长的时候很好维护信息
 3 //区间缩短的时候不太好维护信息(如最大值,删除以后不知道次大值是多少)
 4 // 0(nsqrt(n))
5
6
   struct Hash {
7
       int b[N], tot;
8
       void init() { tot = 0; }
       void insert(int x) { b[++tot] = x; }
9
       void build() {
10
            sort(b + 1, b + 1 + tot);
11
12
            tot = unique(b + 1, b + 1 + tot) - (b + 1);
13
       int pos(int x) \{ return lower_bound(b + 1, b + 1 + tot, x) - b; \}
14
15 } ha;
16
  int n, m;
int c[N], pos[N], cnt[N], cntt[N];
19 int belong[N], sizz;
20 ll ans[N], res;
21
22
   struct Query {
23
       int 1, r, id;
       bool operator < (const Query &rhs) const {</pre>
24
            return belong[l] ^ belong[rhs.l] ? belong[l] < belong[rhs.l] : r < rhs.r;</pre>
25
```

```
26
   } q[N];
27
28
   ll bruteForce(int ql, int qr) {
29
30
        ll result = 0;
        for (int i = ql; i <= qr; i++) {</pre>
31
32
            cntt[pos[i]]++;
            result = max(result, 1ll * c[i] * cntt[pos[i]]);
33
34
35
        for (int i = ql; i <= qr; i++) cntt[pos[i]]--;
        return result;
36
37
   }
38
   void add(int x) {
39
        cnt[pos[x]]++;
40
        res = max(res, 1ll * c[x] * cnt[pos[x]]);
41
42
   }
43
44 void del(int x) {
        cnt[pos[x]]--;
45
   }
46
47
48 int main() {
49
50
        scanf("%d%d", &n, &m);
51
        for (int i = 1; i <= n; i++) scanf("%d", &c[i]), ha.insert(c[i]);</pre>
52
        ha.build();
53
        for (int i = 1; i \le n; i++) pos[i] = ha.pos(c[i]);
54
55
        sizz = sqrt(n); int num = ceil((long double)n / sizz);
56
        for (int i = 1, j = 1; i \le num; i++)
57
            while (j \le i * sizz \&\& j \le n)
58
                belong[j++] = i;
59
60
        for (int i = 1; i <= m; i++) scanf("%d%d", &q[i].1, &q[i].r), q[i].id = i;</pre>
61
62
        sort(q + 1, q + 1 + m);
63
        for (int i = 1, j = 1; i \le num; i++) {
64
            memset(cnt, 0, sizeof(cnt));
65
            int right = min(i * sizz, n);
66
67
            res = 0;
            for (int l = right + 1, r = right; j \le m & belong[q[j].l] == i; j++, l =
68
       right + 1) {
                int ql = q[j].l, qr = q[j].r;
69
70
                if (qr - ql + 1 \le sizz) {
71
                     ans[q[j].id] = bruteForce(ql, qr);
72
                     continue;
73
74
                while (r < qr) add(++r);
75
                11 tmp = res;
76
                while (l > ql) add(--l);
77
                ans[q[j].id] = res;
                res = tmp;
78
                while (l < right + 1) del(l++);</pre>
79
80
            }
81
        }
82
        for (int i = 1; i <= m; i++) printf("%lld\n", ans[i]);</pre>
83
```

```
84
        return 0;
85
86 }
   8.4 treeMO
   const int N = 1e5 + 10;
2
3
   struct Hash {
        int b[N], tot;
4
        void init() { tot = 0; }
5
        void insert(int x) { b[++tot] = x; }
6
7
        void build() {
8
            sort(b + 1, b + 1 + tot);
            tot = unique(b + 1, b + 1 + tot) - (b + 1);
9
10
        int pos(int x) \{ return lower_bound(b + 1, b + 1 + tot, x) - b; \}
11
   } ha;
12
13
14 int n, m;
15 int c[N], cnt[N];
16 vector<int> g[N];
  int st[N], ed[N], dfnt, nodeOf[N << 1], tag[N];</pre>
18 int belong[N], sizz;
  int ans[N], res;
19
20
   struct Query {
21
22
        int 1, r, id, k;
        bool operator < (const Query &rhs) const {</pre>
23
            return belong[1] ^ belong[rhs.1] ? belong[1] < belong[rhs.1] : r < rhs.r;</pre>
24
25
26
   } q[N];
27
  int son[N], siz[N], top[N], fa[N], dep[N];
28
   void dfs(int u, int par) {
        dep[u] = dep[fa[u] = par] + (siz[u] = 1);
30
        int max_son = -1; node0f[st[u] = ++dfnt] = u;
31
        for (auto &v: g[u])
32
33
            if (v != par) {
34
                dfs(v, u);
                siz[u] += siz[v];
35
                if (max_son < siz[v])</pre>
36
37
                     son[u] = v, max\_son = siz[v];
38
        node0f[ed[u] = ++dfnt] = u;
39
   }
40
   void dfs2(int u, int topf) {
41
        top[u] = topf;
42
        if (!son[u]) return;
43
        dfs2(son[u], topf);
44
        for (auto &v: q[u])
45
            if (v != fa[u] \&\& v != son[u]) dfs2(v, v);
46
47
   int lca(int x, int y) {
48
        while (top[x] != top[y]) {
49
            if (dep[top[x]] < dep[top[y]]) swap(x, y);
50
51
            x = fa[top[x]];
52
        }
```

```
return dep[x] < dep[y] ? x : y;</pre>
53
    }
54
55
    void upd(int x) {
56
57
         x = node0f[x];
         if (tag[x]) {
58
             cnt[c[x]]--;
59
             if (!cnt[c[x]]) res--;
60
         }
61
         else {
62
             if (!cnt[c[x]]) res++;
63
64
             cnt[c[x]]++;
65
         tag[x] \sim 1;
66
    }
67
68
69
    int main() {
70
    #ifdef ACM_LOCAL
         freopen("input.in", "r", stdin);
71
         freopen("output.out", "w", stdout);
72
    #endif
73
         scanf("%d%d", &n, &m);
74
         for (int i = 1; i <= n; i++) scanf("%d", &c[i]), ha.insert(c[i]);</pre>
75
76
         ha.build();
77
         for (int i = 1; i \le n; i++) c[i] = ha.pos(c[i]);
78
79
         for (int i = 1; i < n; i++) {
             int u, v; scanf("%d%d", &u, &v);
80
             g[u].push_back(v); g[v].push_back(u);
81
82
         int rt = 1; dfs(rt, 0); dfs2(rt, rt);
83
84
85
         sizz = sqrt(dfnt); int num = ceil((long double)dfnt / sizz);
         for (int i = 1, j = 1; i \le num; i++)
86
             while (j <= i * sizz && j <= dfnt)</pre>
87
                 belong[j++] = i;
88
89
         for (int i = 1; i <= m; i++) {
90
             int u, v; scanf("%d%d", &u, &v);
             int tlca = lca(u, v);
91
             if (st[u] > st[v]) swap(u, v);
92
             if (u == tlca) q[i] = Query{st[u], st[v], i, 0};
93
             else q[i] = Query{ed[u], st[v], i, tlca};
94
95
96
         sort(q + 1, q + 1 + m);
97
98
         int l = 1, r = 0;
99
         for (int i = 1; i \le m; i++) {
100
             int ql = q[i].l, qr = q[i].r;
101
             while (l < ql) upd(l++);
102
103
             while (l > ql) upd(--l);
104
             while (r < qr) upd(++r);
105
             while (r > qr) upd(r--);
             ans[q[i].id] = res + (q[i].k ? (cnt[c[q[i].k]] == 0) : 0);
106
107
108
109
         for (int i = 1; i \le m; i++) printf("%d\n", ans[i]);
110
         return 0;
111
```

112 }

#### 8.5 fastIO

```
namespace FastI0 {
1
2
        char gc(void) {
3
             const int S = 1 \ll 17;
             static char buf[S], *s = buf, *t = buf;
4
             if (s == t) t = buf + fread(s = buf, 1, S, stdin);
5
             if (s == t) return EOF;
6
7
             return *s++;
        }
8
9
10
        int read(void) {
             int a = 0, b = 1, c = gc();
11
             for (; !isdigit(c); c = gc()) b ^= (c == '-');
for (; isdigit(c); c = gc()) a = a * 10 + c - '0';
12
13
14
             return b ? a : -a;
15
16
   using namespace FastIO;
17
18
19
   static struct FastInput {
20
        static constexpr int BUF_SIZE = 1 << 20;</pre>
21
22
        char buf[BUF_SIZE];
23
        size_t chars_read = 0;
24
        size_t buf_pos = 0;
        FILE *in = stdin;
25
26
        char cur = 0;
27
28
        inline char get_char() {
29
             if (buf_pos >= chars_read) {
30
                 chars_read = fread(buf, 1, BUF_SIZE, in);
                 buf_pos = 0;
31
                 buf[0] = (chars\_read == 0 ? -1 : buf[0]);
32
33
34
             return cur = buf[buf_pos++];
35
        }
36
        inline void tie(int) {}
37
38
39
        inline explicit operator bool() {
40
             return cur != -1;
41
42
        inline static bool is_blank(char c) {
43
             return c <= ' ';</pre>
44
        }
45
46
        inline bool skip_blanks() {
47
             while (is_blank(cur) && cur != -1) {
48
                 get_char();
49
50
             return cur != -1;
51
52
        }
53
        inline FastInput& operator>>(char& c) {
54
```

```
skip_blanks();
55
             c = cur;
56
             return *this;
57
        }
58
59
        inline FastInput& operator>>(std::string& s) {
60
             if (skip_blanks()) {
61
                 s.clear();
62
                 do {
63
64
                     s += cur;
                 } while (!is_blank(get_char()));
65
66
             }
             return *this;
67
        }
68
69
        template <typename T>
70
        inline FastInput& read_integer(T& n) {
71
             // unsafe, doesn't check that characters are actually digits
72
73
             n = 0;
             if (skip_blanks()) {
74
                 int sign = +1;
75
                 if (cur == '-') {
76
                     sign = -1;
77
78
                     get_char();
79
                 }
80
                 do {
                     n += n + (n << 3) + cur - '0';
81
                 } while (!is_blank(get_char()));
82
                 n *= sign;
83
84
85
             return *this;
86
        }
87
        template <typename T>
88
        inline typename std::enable_if<std::is_integral<T>::value, FastInput&>::type
89
        operator>>(T& n) {
             return read_integer(n);
90
91
        }
92
        #if !defined(_WIN32) || defined(_WIN64)
93
        inline FastInput& operator>>(__int128& n) {
94
             return read_integer(n);
95
96
97
        #endif
98
        template <typename T>
99
        inline typename std::enable_if<std::is_floating_point<T>::value, FastInput&>::type
100
        operator>>(T& n) {
101
             // not sure if really fast, for compatibility only
102
             n = 0;
103
             if (skip_blanks()) {
104
                 std::string s;
105
                 (*this) >> s;
                 sscanf(s.c_str(), "%lf", &n);
106
107
             return *this;
108
109
110
    } fast_input;
111
```

```
112 #define cin fast_input
113
    static struct FastOutput {
114
         static constexpr int BUF_SIZE = 1 << 20;</pre>
115
         char buf[BUF_SIZE];
116
         size_t buf_pos = 0;
117
         static constexpr int TMP_SIZE = 1 << 20;</pre>
118
         char tmp[TMP_SIZE];
119
         FILE *out = stdout;
120
121
122
         inline void put_char(char c) {
123
             buf[buf_pos++] = c;
             if (buf_pos == BUF_SIZE) {
124
                  fwrite(buf, 1, buf_pos, out);
125
                  buf_pos = 0;
126
             }
127
         }
128
129
         ~FastOutput() {
130
             fwrite(buf, 1, buf_pos, out);
131
132
133
         inline FastOutput& operator<<(char c) {</pre>
134
135
             put_char(c);
136
             return *this;
         }
137
138
         inline FastOutput& operator<<(const char* s) {</pre>
139
             while (*s) {
140
                  put_char(*s++);
141
142
             return *this;
143
         }
144
145
         inline FastOutput& operator<<(const std::string& s) {</pre>
146
             for (int i = 0; i < (int) s.size(); i++) {
147
148
                  put_char(s[i]);
149
             }
             return *this;
150
         }
151
152
         template <typename T>
153
         inline char* integer_to_string(T n) {
154
             // beware of TMP_SIZE
155
             char* p = tmp + TMP_SIZE - 1;
156
             if (n == 0) {
157
                  *--p = '0';
158
             } else {
159
                  bool is_negative = false;
160
161
                  if (n < 0) {
162
                      is_negative = true;
163
                      n = -n;
164
                  while (n > 0) {
165
                      *--p = (char) ('0' + n % 10);
166
167
                      n /= 10;
168
                  if (is_negative) {
169
                      *--p = '-';
170
```

```
}
171
             }
172
173
             return p;
174
        }
175
176
        template <typename T>
        inline typename std::enable_if<std::is_integral<T>::value, char*>::type stringify(T
177
         n) {
             return integer_to_string(n);
178
        }
179
180
181
        #if !defined(_WIN32) || defined(_WIN64)
        inline char* stringify(__int128 n) {
182
             return integer_to_string(n);
183
184
        #endif
185
186
        template <typename T>
187
        inline typename std::enable_if<std::is_floating_point<T>::value, char*>::type
188
        stringify(T n) {
             sprintf(tmp, "%.17f", n);
189
             return tmp;
190
        }
191
192
193
        template <typename T>
        inline FastOutput& operator<<(const T& n) {</pre>
194
             auto p = stringify(n);
195
             for (; *p != 0; p++) {
196
                 put_char(*p);
197
198
199
             return *this;
200
201
    } fast_output;
202
    #define cout fast_output
203
         simulatedAnnealing
    const double DOWN = 0.996:
    const double START_T = 5000;
 3
    double ansx, ansy, ansz, anse;
 4
 5
    void initAns() {
 6
        //初始化一个答案点(可以选任意点)
    }
 7
 8
    double getEnergy(double x, double y, double z) {
 9
10
        //具体分析题目
11
12
    void SA() {
13
        double T = START_T;
14
15
        while (T > 1e-15) {
             double newx = ansx + (rand() * 2 - RAND_MAX) * T;
16
             double newy = ansy + (rand() * 2 - RAND_MAX) * T;
17
             double newz = ansz + (rand() * 2 - RAND_MAX) * T;
18
             double newe = getEnergy(newx, newy, newz);
19
20
             double delta = newe - anse;
```

```
if (delta < 0) ansx = newx, ansy = newy, ansz = newz, anse = newe;
21
           else if (exp(-delta / T) * RAND_MAX > rand())
22
23
               ansx = newx, ansy = newy, ansz = newz;
           T *= DOWN;
24
25
       }
   }
26
27
   void solve() {
28
29
       initAns();
       while ((double) clock() / CLOCKS_PER_SEC < 2.0) SA();</pre>
30
31
  }
   8.7 Cantor
   //主要应用为将N维的排列状态压缩成数字id
  //然后需要知道具体状态时用逆Cantor得到
4
5 int N;
6 int a[MAX], c[MAX];
7
   void upd(int p, int k) { for (; p \le N; p += lowbit(p)) c[p] += k; }
8
  int query(int p) {
9
10
       int res = 0;
       for (; p; p -= lowbit(p)) res += c[p];
11
12
       return res;
13
   }
14
15
   int cantor() {
       //ans = 1 + \sum_{i=1}^{N} fac[N - i] * (\sum_{j=i+1}^{N} x[i] > x[j])
16
       int res = 0, fac = 1;
17
       for (int i = N; i >= 1; i--) {
18
19
           upd(a[i], 1);
           res = (res + 1ll * fac * query(a[i] - 1) % mod) % mod;
20
           fac = 111 * fac * (N - i + 1) % mod;
21
22
23
       return res + 1;
24 }
25
26 //逆Cantor
27 #define lc u<<1
28 #define rc u<<1|1
29 #define mid (l+r)/2
30 int sum[MAX << 4];
   void push_up(int u) { sum[u] = sum[lc] + sum[rc]; }
   void build(int u, int l, int r) {
32
       if (l == r) {
33
           sum[u] = 1;
34
           return;
35
36
       build(lc, l, mid); build(rc, mid + 1, r);
37
       push_up(u);
38
39
   }
   int query(int u, int l, int r, int k) {//查找第k大并且删除该数
40
41
       sum[u]--;
42
       if (l == r) return l;
       if (k <= sum[lc]) return query(lc, l, mid, k);</pre>
43
       else return query(rc, mid + 1, r, k - sum[lc]);
44
```

```
45 }
46
   vector<int> inCantor(int x, int n) {
47
48
       x--;
       vector<int> res;
49
       ll fac = 1;
50
       build(1, 1, n);
51
       for (int i = 1; i <= n; i++) fac = fac * i;
52
       for (int i = 1; i <= n; i++) {
53
            fac = fac / (n - i + 1);
54
            int k = x / fac + 1; //比当前这位大的有x / fac位
55
56
            res.push_back(query(1, 1, n, k));//找到没被选的第k大
57
            x \% = fac;
       }
58
59
       return res;
60
   8.8 BigInteger
   struct BigInteger {
       typedef unsigned long long LL;
2
3
       static const int BASE = 100000000;
4
       static const int WIDTH = 8;
5
       vector<int> s;
6
7
8
       BigInteger& clean() { while (!s.back() && s.size()>1)s.pop_back(); return *this; }
9
       BigInteger(LL num = 0) { *this = num; }
       BigInteger(string s) { *this = s; }
10
       BigInteger& operator = (long long num) {
11
12
            s.clear();
13
            do {
14
                s.push_back(num % BASE);
                num /= BASE;
15
            } while (num > 0);
16
            return *this;
17
18
19
       BigInteger& operator = (const string& str) {
20
            s.clear();
21
            int x, len = (str.length() - 1) / WIDTH + 1;
22
            for (int i = 0; i < len; i++) {
                int end = str.length() - i*WIDTH;
23
                int start = max(0, end - WIDTH);
24
                sscanf(str.substr(start, end - start).c_str(), "%d", &x);
25
26
                s.push_back(x);
27
28
            return (*this).clean();
29
       }
30
       BigInteger operator + (const BigInteger& b) const {
31
32
            BigInteger c; c.s.clear();
            for (int i = 0, g = 0; i++) {
33
                if (g == 0 && i >= s.size() && i >= b.s.size()) break;
34
                int x = g;
35
                if (i < s.size()) x += s[i];</pre>
36
                if (i < b.s.size()) x += b.s[i];</pre>
37
38
                c.s.push_back(x % BASE);
                g = x / BASE;
39
```

```
}
40
            return c;
41
42
        BigInteger operator - (const BigInteger& b) const {
43
            assert(b <= *this); // 减数不能大于被减数
44
            BigInteger c; c.s.clear();
45
            for (int i = 0, g = 0; i++) {
46
                if (g == 0 \&\& i >= s.size() \&\& i >= b.s.size()) break;
47
                int x = s[i] + g;
48
                if (i < b.s.size()) x -= b.s[i];</pre>
49
                if (x < 0) \{ g = -1; x += BASE; \}
50
51
                else q = 0;
52
                c.s.push_back(x);
            }
53
            return c.clean();
54
55
       BigInteger operator * (const BigInteger& b) const {
56
            int i, j; LL g;
57
            vector<LL> v(s.size() + b.s.size(), 0);
58
            BigInteger c; c.s.clear();
59
            for (i = 0; i < s.size(); i++) for (j = 0; j < b.s.size(); j++) v[i + j] += LL(s[i + j])
60
       1)*b.s[j];
            for (i = 0, g = 0; i++) {
61
                if (q == 0 \&\& i >= v.size()) break;
62
63
                LL x = v[i] + q;
                c.s.push_back(x % BASE);
64
                g = x / BASE;
65
66
            return c.clean();
67
68
       BigInteger operator / (const BigInteger& b) const {
69
            assert(b > 0); // 除数必须大于0
70
            BigInteger c = *this;
                                         // 商:主要是让c.s和(*this).s的vector一样大
71
            BigInteger m;
72
                                         // 余数:初始化为0
            for (int i = s.size() - 1; i >= 0; i--) {
73
                m = m*BASE + s[i];
74
75
                c.s[i] = bsearch(b, m);
76
                m -= b*c.s[i];
            }
77
            return c.clean();
78
79
       BigInteger operator % (const BigInteger& b) const { //方法与除法相同
80
            BigInteger c = *this;
81
82
            BigInteger m;
            for (int i = s.size() - 1; i >= 0; i--) {
83
                m = m*BASE + s[i];
84
                c.s[i] = bsearch(b, m);
85
                m -= b*c.s[i];
86
            }
87
88
            return m;
89
90
       // 二分法找出满足bx<=m的最大的x
       int bsearch(const BigInteger& b, const BigInteger& m) const {
91
            int L = 0, R = BASE - 1, x;
92
            while (1) {
93
94
                x = (L + R) >> 1;
                if (b*x \le m) { if (b*(x + 1)>m) return x; else L = x; }
95
96
                else R = x;
97
            }
```

```
98
         BigInteger& operator += (const BigInteger& b) { *this = *this + b; return *this; } BigInteger& operator -= (const BigInteger& b) { *this = *this - b; return *this; }
99
100
         BigInteger& operator *= (const BigInteger& b) { *this = *this * b; return *this; } BigInteger& operator /= (const BigInteger& b) { *this = *this / b; return *this; }
101
102
         BigInteger& operator %= (const BigInteger& b) { *this = *this % b; return *this; }
103
104
         bool operator < (const BigInteger& b) const {</pre>
105
              if (s.size() != b.s.size()) return s.size() < b.s.size();</pre>
106
107
              for (int i = s.size() - 1; i >= 0; i--)
                   if (s[i] != b.s[i]) return s[i] < b.s[i];</pre>
108
109
              return false;
110
         }
         bool operator >(const BigInteger& b) const { return b < *this; }</pre>
111
         bool operator<=(const BigInteger& b) const { return !(b < *this); }</pre>
112
         bool operator>=(const BigInteger& b) const { return !(*this < b);</pre>
113
         bool operator!=(const BigInteger& b) const { return b < *this || *this < b; }</pre>
114
         bool operator==(const BigInteger& b) const { return !(b < *this) && !(b > *this); }
115
116
    };
117
    ostream& operator << (ostream& out, const BigInteger& x) {</pre>
118
         out << x.s.back();
119
         for (int i = x.s.size() - 2; i >= 0; i--) {
120
121
              char buf[20];
              sprintf(buf, "%08d", x.s[i]);
122
              for (int j = 0; j < strlen(buf); j++) out << buf[j];</pre>
123
124
125
         return out;
    }
126
127
128
     istream& operator >> (istream& in, BigInteger& x) {
129
         string s;
130
         if (!(in >> s)) return in;
131
         X = S;
132
         return in;
    }
133
134
135 int main()
136
137
          int t;
         scanf("%d", &t);
138
         while (t--)
139
140
              BigInteger a, b;
141
              cin >> a >> b;
142
              cout << a + b << endl;</pre>
143
         }
144
145 }
     8.9 debug
 1 using std::string;
 2 using std::to_string;
 3 template <typename A, typename B>
 4 std::string to_string(std::pair<A, B> p);
 5
 6 template <typename A, typename B, typename C>
 7 std::string to_string(std::tuple<A, B, C> p);
```

```
8
   template <typename A, typename B, typename C, typename D>
9
    std::string to_string(std::tuple<A, B, C, D> p);
10
11
   std::string to_string(const std::string& s) {
12
     return '"' + s + '"';
13
14
15
   std::string to_string(const char* s) {
16
      return to_string((std::string) s);
17
18
19
20
   std::string to_string(bool b) {
    return (b ? "true" : "false");
21
22
23
24 std::string to_string(std::vector<bool> v) {
      bool first = true;
25
      std::string res = "{";
26
27
      for (int i = 0; i < static_cast<int>(v.size()); i++) {
        if (!first) {
28
29
          res += ", ";
        }
30
31
        first = false;
32
        res += to_string(v[i]);
33
      res += "}";
34
35
      return res;
   }
36
37
   template <size_t N>
38
   std::string to_string(std::bitset<N> v) {
39
      std::string res = "";
40
      for (size_t i = 0; i < N; i++) {</pre>
41
        res += static_cast<char>('0' + v[i]);
42
43
44
      return res;
45
   }
46
   template <typename A>
47
   std::string to_string(A v) {
48
      bool first = true;
std::string res = "{";
49
50
      for (const auto &x : v) {
51
        if (!first) {
52
          res += ", ";
53
54
        first = false;
55
56
        res += to_string(x);
57
      }
58
      res += "}";
59
      return res;
60
   }
61
62
63 template <typename A, typename B>
   std::string to_string(std::pair<A, B> p) {
  return "(" + to_string(p.first) + ", " + to_string(p.second) + ")";
65
66 }
```

```
67
68 template <typename A, typename B, typename C>
   std::string to_string(std::tuple<A, B, C> p) {
   return "(" + to_string(std::get<0>(p)) + ", " + to_string(std::get<1>(p)) + ", " +
70
        to_string(std::get<2>(p)) + ")";
   }
71
72
73 template <typename A, typename B, typename C, typename D>
   std::string to_string(std::tuple<A, B, C, D> p) {
74
      return "(" + to_string(std::get<0>(p)) + ", " + to_string(std::get<1>(p)) + ", " +
        to_string(std::get<2>(p)) + ", " + to_string(std::get<3>(p)) + ")";
76
   }
77
78 void debug_out() { std::cerr << std::endl; }</pre>
79
80 template <typename Head, typename... Tail>
   void debug_out(Head H, Tail... T) {
      std::cerr << " " << to_string(H);
debug_out(T...);</pre>
83
84 }
85
86 #ifdef LOCAL
87 #define debug(...) std::cerr << "[" << #__VA_ARGS__ << "]:", debug_out(__VA_ARGS__)
89 #define debug(...) 42
90 #endif
    8.10 \quad \text{int} 128
1 using i128 = __int128;
   std::istream &operator>>(std::istream &is, i128 &n) {
3
        n = 0;
4
        std::string s;
5
6
        is >> s;
        for (auto c : s) {
7
            n = 10 * n + c - '0';
8
9
        }
10
        return is;
   }
11
12
   std::ostream &operator<<(std::ostream &os, i128 n) {</pre>
13
        if (n == 0) {
14
             return os << 0;
15
16
        std::string s;
17
        while (n > 0) {
18
             s += '0' + n \% 10;
19
            n /= 10;
20
21
22
        std::reverse(s.begin(), s.end());
23
        return os << s;
24 }
```