

Содержание

1	final/template/vimrc.txt
2	final/template/template.cpp
3	Practice round
4	final/template/fastIO.cpp
5	final/template/hashTable.cpp
6	final/template/optimizations.cpp
7	final/template/useful.cpp
8	final/template/Template.java
9	final/numeric/fft.cpp
10	final/numeric/fftint.cpp
11	final/numeric/blackbox.cpp
12	final/numeric/crt.cpp
13	final/numeric/mulMod.cpp
14	final/numeric/modReverse.cpp
15	final/numeric/pollard.cpp
16	final/numeric/poly.cpp
17	final/numeric/simplex.cpp
18	final/numeric/sumLine.cpp
19	final/geom/commonTangents.cpp
20	final/geom/halfplaneIntersection.cpp
21	final/geom/minDisc.cpp
22	final/geom/convexHull3D-N2.cpp
23	final/geom/polygonArcCut.cpp
24	final/strings/eertree.cpp
25	final/strings/sufAutomaton.cpp
26	final/graphs/centroid.cpp
27	final/graphs/dominatorTree.cpp
28	final/graphs/generalMatching.cpp
29	final/graphs/heavyLight.cpp
30	final/graphs/hungary.cpp
31	final/graphs/retro.cpp
32	final/graphs/smith.cpp
33	final/graphs/twoChinese.cpp
34	final/graphs/linkcut.cpp

final/template/vimrc.txt

```

2 1 map <F9> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -↵
2 2 Wno-unused-result -o %:r % -std=c++14 -DHOME -↵
2 3 D_GLIBCXX_DEBUG -fsanitize=address <CR>
2 4 map <F7> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -↵
2 5 Wno-unused-result -o %:r % -std=c++14 -DHOME -↵
2 6 02 <CR>
2 7 map <F8> :wall! <CR> :!ulimit -s 500000 && ./%:r <CR>↵
2 8 >
2 9 map <F10> :wall! <CR> :!g++ -Wall -Wextra -Wshadow -↵
2 10 Wno-unused-result -o %:r % -std=c++14 -DHOME -↵
2 11 D_GLIBCXX_DEBUG -fsanitize=address -g && gdb ↵
2 12 ./%:r <CR>
3 13
3 14 inoremap {<CR> {<CR>}<ESC>O
3 15 map <c-a> ggVG
3 16
3 17 set nu
3 18 set rnu
3 19 syntax on
4 20
4 21 map <c-t> :tabnew <CR>
4 22 map <c-l> :tabn <CR>
4 23 map <c-h> :tabp <CR>
4 24
4 25
4 26
4 27
4 28
4 29
4 30
4 31
4 32
4 33
4 34
4 35
4 36
4 37
4 38
4 39
4 40
4 41
4 42
4 43
4 44
4 45
4 46
4 47
4 48
4 49
4 50
4 51
4 52
4 53
4 54
4 55
4 56
4 57
4 58
4 59
4 60
4 61
4 62
4 63
4 64
4 65
4 66
4 67
4 68
4 69
4 70
4 71
4 72
4 73
4 74
4 75
4 76
4 77
4 78
4 79
4 80
4 81
4 82
4 83
4 84
4 85
4 86
4 87
4 88
4 89
4 90
4 91
4 92
4 93
4 94
4 95
4 96
4 97
4 98
4 99
4 100
4 101
4 102
4 103
4 104
4 105
4 106
4 107
4 108
4 109
4 110
4 111
4 112
4 113
4 114
4 115
4 116
4 117
4 118
4 119
4 120
4 121
4 122
4 123
4 124
4 125
4 126
4 127
4 128
4 129
4 130
4 131
4 132
4 133
4 134
4 135
4 136
4 137
4 138
4 139
4 140
4 141
4 142
4 143
4 144
4 145
4 146
4 147
4 148
4 149
4 150
4 151
4 152
4 153
4 154
4 155
4 156
4 157
4 158
4 159
4 160
4 161
4 162
4 163
4 164
4 165
4 166
4 167
4 168
4 169
4 170
4 171
4 172
4 173
4 174
4 175
4 176
4 177
4 178
4 179
4 180
4 181
4 182
4 183
4 184
4 185
4 186
4 187
4 188
4 189
4 190
4 191
4 192
4 193
4 194
4 195
4 196
4 197
4 198
4 199
4 200
4 201
4 202
4 203
4 204
4 205
4 206
4 207
4 208
4 209
4 210
4 211
4 212
4 213
4 214
4 215
4 216
4 217
4 218
4 219
4 220
4 221
4 222
4 223
4 224
4 225
4 226
4 227
4 228
4 229
4 230
4 231
4 232
4 233
4 234
4 235
4 236
4 237
4 238
4 239
4 240
4 241
4 242
4 243
4 244
4 245
4 246
4 247
4 248
4 249
4 250
4 251
4 252
4 253
4 254
4 255
4 256
4 257
4 258
4 259
4 260
4 261
4 262
4 263
4 264
4 265
4 266
4 267
4 268
4 269
4 270
4 271
4 272
4 273
4 274
4 275
4 276
4 277
4 278
4 279
4 280
4 281
4 282
4 283
4 284
4 285
4 286
4 287
4 288
4 289
4 290
4 291
4 292
4 293
4 294
4 295
4 296
4 297
4 298
4 299
4 300
4 301
4 302
4 303
4 304
4 305
4 306
4 307
4 308
4 309
4 310
4 311
4 312
4 313
4 314
4 315
4 316
4 317
4 318
4 319
4 320
4 321
4 322
4 323
4 324
4 325
4 326
4 327
4 328
4 329
4 330
4 331
4 332
4 333
4 334
4 335
4 336
4 337
4 338
4 339
4 340
4 341
4 342
4 343
4 344
4 345
4 346
4 347
4 348
4 349
4 350
4 351
4 352
4 353
4 354
4 355
4 356
4 357
4 358
4 359
4 360
4 361
4 362
4 363
4 364
4 365
4 366
4 367
4 368
4 369
4 370
4 371
4 372
4 373
4 374
4 375
4 376
4 377
4 378
4 379
4 380
4 381
4 382
4 383
4 384
4 385
4 386
4 387
4 388
4 389
4 390
4 391
4 392
4 393
4 394
4 395
4 396
4 397
4 398
4 399
4 400
4 401
4 402
4 403
4 404
4 405
4 406
4 407
4 408
4 409
4 410
4 411
4 412
4 413
4 414
4 415
4 416
4 417
4 418
4 419
4 420
4 421
4 422
4 423
4 424
4 425
4 426
4 427
4 428
4 429
4 430
4 431
4 432
4 433
4 434
4 435
4 436
4 437
4 438
4 439
4 440
4 441
4 442
4 443
4 444
4 445
4 446
4 447
4 448
4 449
4 450
4 451
4 452
4 453
4 454
4 455
4 456
4 457
4 458
4 459
4 460
4 461
4 462
4 463
4 464
4 465
4 466
4 467
4 468
4 469
4 470
4 471
4 472
4 473
4 474
4 475
4 476
4 477
4 478
4 479
4 480
4 481
4 482
4 483
4 484
4 485
4 486
4 487
4 488
4 489
4 490
4 491
4 492
4 493
4 494
4 495
4 496
4 497
4 498
4 499
4 500
4 501
4 502
4 503
4 504
4 505
4 506
4 507
4 508
4 509
4 510
4 511
4 512
4 513
4 514
4 515
4 516
4 517
4 518
4 519
4 520
4 521
4 522
4 523
4 524
4 525
4 526
4 527
4 528
4 529
4 530
4 531
4 532
4 533
4 534
4 535
4 536
4 537
4 538
4 539
4 540
4 541
4 542
4 543
4 544
4 545
4 546
4 547
4 548
4 549
4 550
4 551
4 552
4 553
4 554
4 555
4 556
4 557
4 558
4 559
4 560
4 561
4 562
4 563
4 564
4 565
4 566
4 567
4 568
4 569
4 570
4 571
4 572
4 573
4 574
4 575
4 576
4 577
4 578
4 579
4 580
4 581
4 582
4 583
4 584
4 585
4 586
4 587
4 588
4 589
4 590
4 591
4 592
4 593
4 594
4 595
4 596
4 597
4 598
4 599
4 600
4 601
4 602
4 603
4 604
4 605
4 606
4 607
4 608
4 609
4 610
4 611
4 612
4 613
4 614
4 615
4 616
4 617
4 618
4 619
4 620
4 621
4 622
4 623
4 624
4 625
4 626
4 627
4 628
4 629
4 630
4 631
4 632
4 633
4 634
4 635
4 636
4 637
4 638
4 639
4 640
4 641
4 642
4 643
4 644
4 645
4 646
4 647
4 648
4 649
4 650
4 651
4 652
4 653
4 654
4 655
4 656
4 657
4 658
4 659
4 660
4 661
4 662
4 663
4 664
4 665
4 666
4 667
4 668
4 669
4 670
4 671
4 672
4 673
4 674
4 675
4 676
4 677
4 678
4 679
4 680
4 681
4 682
4 683
4 684
4 685
4 686
4 687
4 688
4 689
4 690
4 691
4 692
4 693
4 694
4 695
4 696
4 697
4 698
4 699
4 700
4 701
4 702
4 703
4 704
4 705
4 706
4 707
4 708
4 709
4 710
4 711
4 712
4 713
4 714
4 715
4 716
4 717
4 718
4 719
4 720
4 721
4 722
4 723
4 724
4 725
4 726
4 727
4 728
4 729
4 730
4 731
4 732
4 733
4 734
4 735
4 736
4 737
4 738
4 739
4 740
4 741
4 742
4 743
4 744
4 745
4 746
4 747
4 748
4 749
4 750
4 751
4 752
4 753
4 754
4 755
4 756
4 757
4 758
4 759
4 760
4 761
4 762
4 763
4 764
4 765
4 766
4 767
4 768
4 769
4 770
4 771
4 772
4 773
4 774
4 775
4 776
4 777
4 778
4 779
4 780
4 781
4 782
4 783
4 784
4 785
4 786
4 787
4 788
4 789
4 790
4 791
4 792
4 793
4 794
4 795
4 796
4 797
4 798
4 799
4 800
4 801
4 802
4 803
4 804
4 805
4 806
4 807
4 808
4 809
4 810
4 811
4 812
4 813
4 814
4 815
4 816
4 817
4 818
4 819
4 820
4 821
4 822
4 823
4 824
4 825
4 826
4 827
4 828
4 829
4 830
4 831
4 832
4 833
4 834
4 835
4 836
4 837
4 838
4 839
4 840
4 841
4 842
4 843
4 844
4 845
4 846
4 847
4 848
4 849
4 850
4 851
4 852
4 853
4 854
4 855
4 856
4 857
4 858
4 859
4 860
4 861
4 862
4 863
4 864
4 865
4 866
4 867
4 868
4 869
4 870
4 871
4 872
4 873
4 874
4 875
4 876
4 877
4 878
4 879
4 880
4 881
4 882
4 883
4 884
4 885
4 886
4 887
4 888
4 889
4 890
4 891
4 892
4 893
4 894
4 895
4 896
4 897
4 898
4 899
4 900
4 901
4 902
4 903
4 904
4 905
4 906
4 907
4 908
4 909
4 910
4 911
4 912
4 913
4 914
4 915
4 916
4 917
4 918
4 919
4 920
4 921
4 922
4 923
4 924
4 925
4 926
4 927
4 928
4 929
4 930
4 931
4 932
4 933
4 934
4 935
4 936
4 937
4 938
4 939
4 940
4 941
4 942
4 943
4 944
4 945
4 946
4 947
4 948
4 949
4 950
4 951
4 952
4 953
4 954
4 955
4 956
4 957
4 958
4 959
4 960
4 961
4 962
4 963
4 964
4 965
4 966
4 967
4 968
4 969
4 970
4 971
4 972
4 973
4 974
4 975
4 976
4 977
4 978
4 979
4 980
4 981
4 982
4 983
4 984
4 985
4 986
4 987
4 988
4 989
4 990
4 991
4 992
4 993
4 994
4 995
4 996
4 997
4 998
4 999
4 1000
4 1001
4 1002
4 1003
4 1004
4 1005
4 1006
4 1007
4 1008
4 1009
4 1010
4 1011
4 1012
4 1013
4 1014
4 1015
4 1016
4 1017
4 1018
4 1019
4 1020
4 1021
4 1022
4 1023
4 1024
4 1025
4 1026
4 1027
4 1028
4 1029
4 1030
4 1031
4 1032
4 1033
4 1034
4 1035
4 1036
4 1037
4 1038
4 1039
4 1040
4 1041
4 1042
4 1043
4 1044
4 1045
4 1046
4 1047
4 1048
4 1049
4 1050
4 1051
4 1052
4 1053
4 1054
4 1055
4 1056
4 1057
4 1058
4 1059
4 1060
4 1061
4 1062
4 1063
4 1064
4 1065
4 1066
4 1067
4 1068
4 1069
4 1070
4 1071
4 1072
4 1073
4 1074
4 1075
4 1076
4 1077
4 1078
4 1079
4 1080
4 1081
4 1082
4 1083
4 1084
4 1085
4 1086
4 1087
4 1088
4 1089
4 1090
4 1091
4 1092
4 1093
4 1094
4 1095
4 1096
4 1097
4 1098
4 1099
4 1100
4 1101
4 1102
4 1103
4 1104
4 1105
4 1106
4 1107
4 1108
4 1109
4 1110
4 1111
4 1112
4 1113
4 1114
4 1115
4 1116
4 1117
4 1118
4 1119
4 1120
4 1121
4 1122
4 1123
4 1124
4 1125
4 1126
4 1127
4 1128
4 1129
4 1130
4 1131
4 1132
4 1133
4 1134
4 1135
4 1136
4 1137
4 1138
4 1139
4 1140
4 1141
4 1142
4 1143
4 1144
4 1145
4 1146
4 1147
4 1148
4 1149
4 1150
4 1151
4 1152
4 1153
4 1154
4 1155
4 1156
4 1157
4 1158
4 1159
4 1160
4 1161
4 1162
4 1163
4 1164
4 1165
4 1166
4 1167
4 1168
4 1169
4 1170
4 1171
4 1172
4 1173
4 1174
4 1175
4 1176
4 1177
4 1178
4 1179
4 1180
4 1181
4 1182
4 1183
4 1184
4 1185
4 1186
4 1187
4 1188
4 1189
4 1190
4 1191
4 1192
4 1193
4 1194
4 1195
4 1196
4 1197
4 1198
4 1199
4 1200
4 1201
4 1202
4 1203
4 1204
4 1205
4 1206
4 1207
4 1208
4 1209
4 1210
4 1211
4 1212
4 1213
4 1214
4 1215
4 1216
4 1217
4 1218
4 1219
4 1220
4 1221
4 1222
4 1223
4 1224
4 1225
4 1226
4 1227
4 1228
4 1229
4 1230
4 1231
4 1232
4 1233
4 1234
4 1235
4 1236
4 1237
4 1238
4 1239
4 1240
4 1241
4 1242
4 1243
4 1244
4 1245
4 1246
4 1247
4 1248
4 1249
4 1250
4 1251
4 1252
4 1253
4 1254
4 1255
4 1256
4 1257
4 1258
4 1259
4 1260
4 1261
4 1262
4 1263
4 1264
4 1265
4 1266
4 1267
4 1268
4 1269
4 1270
4 1271
4 1272
4 1273
4 1274
4 1275
4 1276
4 1277
4 1278
4 1279
4 1280
4 1281
4 1282
4 1283
4 1284
4 1285
4 1286
4 1287
4 1288
4 1289
4 1290
4 1291
4 1292
4 1293
4 1294
4 1295
4 1296
4 1297
4 1298
4 1299
4 1300
4 1301
4 1302
4 1303
4 1304
4 1305
4 1306
4 1307
4 1308
4 1309
4 1310
4 1311
4 1312
4 1313
4 1314
4 1315
4 1316
4 1317
4 1318
4 1319
4 1320
4 1321
4 1322
4 1323
4 1324
4 1325
4 1326
4 1327
4 1328
4 1329
4 1330
4 1331
4 1332
4 1333
4 1334
4 1335
4 1336
4 1337
4 1338
4 1339
4 1340
4 1341
4 1342
4 1343
4 1344
4 1345
4 1346
4 1347
4 1348
4 1349
4 1350
4 1351
4 1352
4 1353
4 1354
4 1355
4 1356
4 1357
4 1358
4 1359
4 1360
4 1361
4 1362
4 1363
4 1364
4 1365
4 1366
4 1367
4 1368
4 1369
4 1370
4 1371
4 1372
4 1373
4 1374
4 1375
4 1376
4 1377
4 1378
4 1379
4 1380
4 1381
4 1382
4 1383
4 1384
4 1385
4 1386
4 1387
4 1388
4 1389
4 1390
4 1391
4 1392
4 1393
4 1394
4 1395
4 1396
4 1397
4 1398
4 1399
4 1400
4 1401
4 1402
4 1403
4 1404
4 1405
4 1406
4 1407
4 1408
4 1409
4 1410
4 1411
4 1412
4 1413
4 1414
4 1415
4 1416
4 1417
4 1418
4 1419
4 1420
4 1421
4 1422
4 1423
4 1424
4 1425
4 1426
4 1427
4 1428
4 1429
4 1430
4 1431
4 1432
4 1433
4 1434
4 1435
4 1436
4 1437
4 1438
4 1439
4 1440
4 1441
4 1442
4 1443
4 1444
4 1445
4 1446
4 1447
4 1448
4 1449
4 1450
4 1451
4 1452
4 1453
4 1454
4 1455
4 1456
4 1457
4 1458
4 1459
4 1460
4 1461
4 1462
4 1463
4 1464
4 1465
4 1466
4 1467
4 1468
4 1469
4 1470
4 1471
4 1472
4 1473
4 1474
4 1475
4 1476
4 1477
4 1478
4 1479
4 1480
4 1481
4 1482
4 1483
4 1484
4 1485
4 1486
4 1487
4 1488
4 1489
4 1490
4 1491
4 1492
4 1493
4 1494
4 1495
4 1496
4 1497
4 1498
4 1499
4 1500
4 1501
4 1502
4 1503
4 1504
4 1505
4 1506
4 1507
4 1508
4 1509
4 1510
4 1511
4 1512
4 1513
4 1514
4 1515
4 1516
4 1517
4 1518
4 1519
4 1520
4 1521
4 1522
4 1523
4 1524
4 1525
4 1526
4 1527
4 1528
4 1529
4 1530
4 1531
4 1532
4 1533
4 1534
4 1535
4 1536
4 1537
4 1538
4 1539
4 1540
4 1541
4 1542
4 1543
4 1544
4 1545
4 1546
4 1547
4 1548
4 1549
4 1550
4 1551
4 1552
4 1553
4 1554
4 1555
4 1556
4 1557
4 1558
4 1559
4 1560
4 1561
4 1562
4 1563
4 1564
4 1565
4 1566
4 1567
4 1568
4 1569
4 1570
4 1571
4 1572
4 1573
4 1574
4 1575
4 1576
4 1577
4 1578
4 1579
4 1580
4 1581
4 1582
4 1583
4 1584
4 1585
4 1586
4 1587
4 1588
4 1589
4 1590
4 1591
4 1592
4 1593
4 1594
4 1595
4 1596
4 1597
4 1598
4 1599
4 1600
4 1601
4 1602
4 1603
4 1604
4 1605
4 1606
4 1607
4 1608
4 1609
4 1610
4 1611
4 1612
4 1613
4 1614
4 1615
4 1616
4 1617
4 1618
4 1619
4 1620
4 1621
4 1622
4 1623
4 1624
4 1625
4 1626
4 1627
4 1628
4 1629
4 1630
4 1631
4 1632
4 1633
4 1634
4 1635
4 1636
4 1637
4 1638
4 1639
4 1640
4 1641
4 1642
4 1643
4 1644
4 1645
4 1646
4 1647
4 1648
4 1649
4 1650
4 1651
4 1652
4 1653
4 1654
4 1655
4 1656
4 1657
4 1658
4 1659
4 1660
4 1661
4 1662
4 1663
4 1664
4 1665
4 1666
4 1667
4 1668
4 1669
4 1670
4 1671
4 1672
4 1673
4 1674
4 1675
4 1676
4 1677
4 1678
4 1679
4 1680
4 1681
4 1682
4 1683
4 1684
4 1685
4 1686
4 1687
4 1688
4 1689
4 1690
4 1691
4 1692
4 1693
4 1694
4 1695
4 1696
4 1697
4 1698
4 1699
4 1700
4 1701
4 1702
4 1703
4 1704
4 1705
4 1706
4 1707
4 1708
4 1709
4 1710
4 1711
4 1712
4 1713
4 1714
4 1715
4 1716
4 1717
4 1718
4 1719
4 1720
4 1721
4 1722
4 1723
4 1724
4 1725
4 1726
4
```

Practice round

1. Посабмитить задачи каждому человеку
2. IDE для джавы
3. Сравнить скорость локального компьютера и сервера
4. Проверить __int128
5. Проверить прагмы (например на битсетах)
6. Узнать максимально возможный размер отправляемого кода

final/template/fastIO.cpp

```

1 #include <cstdio>
2 #include <algorithm>
3
4 /** Interface */
5
6 inline int readInt();
7 inline int readUInt();
8 inline bool isEof();
9
10 /** Read */
11
12 static const int buf_size = 100000;
13 static char buf[buf_size];
14 static int buf_len = 0, pos = 0;
15
16 inline bool isEof() {
17     if (pos == buf_len) {
18         pos = 0, buf_len = fread(buf, 1, buf_size, stdin);
19     }
20     if (pos == buf_len) return 1;
21     return 0;
22 }
23
24 inline int getChar() { return isEof() ? -1 : buf[pos++]; }
25
26 inline int readChar() {
27     int c = getChar();
28     while (c != -1 && c <= 32) c = getChar();
29     return c;
30 }
31
32 inline int readUInt() {
33     int c = readChar(), x = 0;
34     while ('0' <= c && c <= '9') x = x * 10 + c - '0', c = getChar();
35     return x;
36 }
37
38 inline int readInt() {
39     int s = 1, c = readChar();
40     int x = 0;
41     if (c == '-') s = -1, c = getChar();
42     while ('0' <= c && c <= '9') x = x * 10 + c - '0', c = getChar();
43     return s == 1 ? x : -x;
44 }
45
46 // 10M int [0..1e9)
47 // cin 3.02
48 // scanf 1.2
49 // cin_sync_with_stdio(false) 0.71
50 // fastRead_getchar 0.53
51 // fastRead_fread 0.15

```

final/template/hashTable.cpp

```

1 template <const int max_size, class HashType, class Data, const Data default_value>
2 struct hashTable {
3     HashType hash[max_size];
4     Data f[max_size];
5     int size;
6
7     int position(HashType H) const {
8         int i = H % max_size;
9         while (hash[i] && hash[i] != H)
10             if (++i == max_size)
11                 i = 0;
12         return i;
13     }
14
15     Data & operator [] (HashType H) {
16         assert(H != 0);
17         int i = position(H);
18         if (!hash[i]) {
19             hash[i] = H;
20             f[i] = default_value;
21             size++;
22         }
23         return f[i];
24     }
25 };
26
27 hashTable<13, int, int, 0> h;

```

final/template/optimizations.cpp

```

1 inline void fasterLLDivMod(unsigned long long x, unsigned y, unsigned &out_d, unsigned &out_m) {
2     unsigned xh = (unsigned)(x >> 32), xl = (unsigned)x,
3         d, m;
4     #ifdef __GNUC__
5         asm(
6             "divl %4; \n\t"
7             : "=a" (d), "=d" (m)
8             : "d" (xh), "a" (xl), "r" (y)
9         );
10    #else
11        __asm {
12            mov edx, dword ptr[xh];
13            mov eax, dword ptr[xl];
14            div dword ptr[y];
15            mov dword ptr[d], eax;
16            mov dword ptr[m], edx;
17        };
18    #endif
19    out_d = d; out_m = m;
20 }
21
22 // have no idea what sse flags are really cool; list of some of them
23 // — very good with bitsets
24 #pragma GCC optimize("O3")
25 #pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mnmx")

```

final/template/useful.cpp

```

1 #include "ext/pb_ds/assoc_container.hpp"
2 using namespace __gnu_pbds;
3
4 template <typename T> using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
5 template <typename K, typename V> using ordered_map = tree<K, V, less<K>, rb_tree_tag, tree_order_statistics_node_update>;
6
7 // HOW TO USE ::
8 // — order_of_key(10) returns the number of elements in set/map strictly less than 10
9 // — *find_by_order(10) returns 10-th smallest element in set/map (0-based)
10
11 bitset<N> a;
12 for (int i = a._Find_first(); i != a.size(); i = a._Find_next(i)) {

```

```

13 cout << i << endl;
14 }

```

final/numeric/fft.cpp

final/template/Template.java

```

1 import java.util.*;
2 import java.io.*;
3
4 public class Template {
5     FastScanner in;
6     PrintWriter out;
7
8     public void solve() throws IOException {
9         int n = in.nextInt();
10        out.println(n);
11    }
12
13    public void run() {
14        try {
15            in = new FastScanner();
16            out = new PrintWriter(System.out);
17
18            solve();
19
20            out.close();
21        } catch (IOException e) {
22            e.printStackTrace();
23        }
24    }
25
26    class FastScanner {
27        BufferedReader br;
28        StringTokenizer st;
29
30        FastScanner() {
31            br = new BufferedReader(new InputStreamReader(↵
32            System.in));
33        }
34
35        String next() {
36            while (st == null || !st.hasMoreTokens()) {
37                try {
38                    st = new StringTokenizer(br.readLine());
39                } catch (IOException e) {
40                    e.printStackTrace();
41                }
42            }
43            return st.nextToken();
44        }
45
46        int nextInt() {
47            return Integer.parseInt(next());
48        }
49
50        public static void main(String[] arg) {
51            new Template().run();
52        }
53    }

```

```

1 namespace fft
2 {
3     const int maxBase = 21;
4     const int maxN = 1 << maxBase;
5
6     struct num
7     {
8         dbl x, y;
9         num() {}
10        num(dbl xx, dbl yy): x(xx), y(yy) {}
11        num(dbl alp): x(cos(alp)), y(sin(alp)) {}
12    };
13
14    inline num operator + (num a, num b) { return num(↵
15        a.x + b.x, a.y + b.y); }
16    inline num operator - (num a, num b) { return num(↵
17        a.x - b.x, a.y - b.y); }
18    inline num operator * (num a, num b) { return num(↵
19        a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); ↵
20    }
21    inline num conj(num a) { return num(a.x, -a.y); }
22
23    const dbl PI = acos(-1);
24
25    num root[maxN];
26    int rev[maxN];
27    bool rootsPrepared = false;
28
29    void prepRoots()
30    {
31        if (rootsPrepared) return;
32        rootsPrepared = true;
33        root[1] = num(1, 0);
34        for (int k = 1; k < maxBase; ++k)
35        {
36            num x(2 * PI / pw(k + 1));
37            for (int i = pw(k - 1); i < pw(k); ++i)
38            {
39                root[2 * i] = root[i];
40                root[2 * i + 1] = root[i] * x;
41            }
42        }
43    }
44
45    int base, N;
46
47    int lastRevN = -1;
48    void prepRev()
49    {
50        if (lastRevN == N) return;
51        lastRevN = N;
52        forn(i, N) rev[i] = (rev[i >> 1] >> 1) + ((i & ↵
53        1) << (base - 1));
54    }
55
56    void fft(num *a, num *f)
57    {
58        forn(i, N) f[i] = a[rev[i]];
59        for (int k = 1; k < N; k <= 1) for (int i = 0; ↵
60        i < N; i += 2 * k) forn(j, k)
61        {
62            num z = f[i + j + k] * root[j + k];
63            f[i + j + k] = f[i + j] - z;
64            f[i + j] = f[i + j] + z;
65        }
66    }
67
68    num a[maxN], b[maxN], f[maxN], g[maxN];
69    A[maxN], B[maxN], C[maxN];
70
71    void _multMod(int mod)
72    {
73        forn(i, N)
74        {
75            int x = A[i] % mod;
76            a[i] = num(x & (pw(15) - 1), x >> 15);
77        }
78        forn(i, N)
79        {
80            int x = B[i] % mod;
81            b[i] = num(x & (pw(15) - 1), x >> 15);
82        }
83        fft(a, f);
84        fft(b, g);
85
86        forn(i, N)
87        {
88            int j = (N - i) & (N - 1);
89
90

```

```

83     num a1 = (f[i] + conj(f[j])) * num(0.5, 0);
84     num a2 = (f[i] - conj(f[j])) * num(0, -0.5);
85     num b1 = (g[i] + conj(g[j])) * num(0.5 / N, 0) ←
;
86     num b2 = (g[i] - conj(g[j])) * num(0, -0.5 / N ←
);
87     a[j] = a1 * b1 + a2 * b2 * num(0, 1);
88     b[j] = a1 * b2 + a2 * b1;
89 }
90
91 fft(a, f);
92 fft(b, g);
93
94 forn(i, N)
95 {
96     ll aa = f[i].x + 0.5;
97     ll bb = g[i].x + 0.5;
98     ll cc = f[i].y + 0.5;
99     C[i] = (aa + bb % mod * pw(15) + cc % mod * pw ←
(30)) % mod;
100 }
101
102
103 void prepAB(int n1, int n2)
104 {
105     base = 1;
106     N = 2;
107     while (N < n1 + n2) base++, N <= 1;
108
109     for (int i = n1; i < N; ++i) A[i] = 0;
110     for (int i = n2; i < N; ++i) B[i] = 0;
111
112     prepRoots();
113     prepRev();
114 }
115
116 void mult(int n1, int n2)
117 {
118     prepAB(n1, n2);
119     forn(i, N) a[i] = num(A[i], B[i]);
120     fft(a, f);
121     forn(i, N)
122     {
123         int j = (N - i) & (N - 1);
124         a[i] = (f[j] * f[j] - conj(f[i] * f[i])) * num ←
(0, -0.25 / N);
125     }
126     fft(a, f);
127     forn(i, N) C[i] = (ll)round(f[i].x);
128 }
129
130
131 void multMod(int n1, int n2, int mod)
132 {
133     prepAB(n1, n2);
134     _multMod(mod);
135 }
136
137 int D[maxN];
138
139 void multLL(int n1, int n2)
140 {
141     prepAB(n1, n2);
142
143     int mod1 = 1.5e9;
144     int mod2 = mod1 + 1;
145
146     _multMod(mod1);
147
148     forn(i, N) D[i] = C[i];
149
150     _multMod(mod2);
151
152     forn(i, N)
153     {
154         C[i] = D[i] + (C[i] - D[i] + (ll)mod2) * (ll) ←
mod1 % mod2 * mod1;
155     }
156 }
157 // HOW TO USE ::
158 // — set correct maxBase
159 // — use mult(n1, n2), multMod(n1, n2, mod) and ←
multLL(n1, n2)
160 // — input : A[], B[]
161 // — output : C[]
162 }

```

final/numeric/fftint.cpp

```

1 namespace fft
2 {
3     const int mod = 998244353;
4     const int base = 20;
5     const int N = 1 << base;
6     const int ROOT = 646;
7
8     int root[N];
9     int rev[N];
10
11 void init()
12 {
13     forn(i, N) rev[i] = (rev[i >> 1] >> 1) + ((i & ←
1) << (base - 1));
14     int NN = N >> 1;
15     int z = 1;
16     forn(i, NN)
17     {
18         root[i + NN] = z;
19         z = z * (ll)ROOT % mod;
20     }
21     for (int i = NN - 1; i > 0; --i) root[i] = root ←
[2 * i];
22 }
23
24 void fft(int *a, int *f)
25 {
26     forn(i, N) f[i] = a[rev[i]];
27     for (int k = 1; k < N; k <= 1) for (int i = 0; ←
i < N; i += 2 * k) forn(j, k)
28     {
29         int z = f[i + j + k] * (ll)root[j + k] % mod;
30         f[i + j + k] = (f[i + j] - z + mod) % mod;
31         f[i + j] = (f[i + j] + z) % mod;
32     }
33 }
34
35 int A[N], B[N], C[N];
36 int F[N], G[N];
37
38 void _mult(int eq)
39 {
40     fft(A, F);
41     if (eq) forn(i, N) G[i] = F[i];
42     else fft(B, G);
43     int invN = inv(N);
44     forn(i, N) A[i] = F[i] * (ll)G[i] % mod * invN % ←
mod;
45     reverse(A + 1, A + N);
46     fft(A, C);
47 }
48
49 void mult(int n1, int n2, int eq = 0)
50 {
51     for (int i = n1; i < N; ++i) A[i] = 0;
52     for (int i = n2; i < N; ++i) B[i] = 0;
53
54     _mult(eq);
55
56     //forn(i, n1 + n2) C[i] = 0;
57     //forn(i, n1) forn(j, n2) C[i + j] = (C[i + j] + ←
A[i] * (ll)B[j]) % mod;
58 }
59 }

```

final/numeric/blackbox.cpp

```

1 namespace blackbox
2 {
3     int A[N];
4     int B[N];
5     int C[N];
6
7     int magic(int k, int x)
8     {
9         B[k] = x;
10        C[k] = (C[k] + A[0] * (ll)B[k]) % mod;
11        int z = 1;
12        if (k == N - 1) return C[k];
13        while ((k & (z - 1)) == (z - 1))
14        {
15            //mult B[k - z + 1 ... k] x A[z .. 2 * z - 1]
16            forn(i, z) fft::A[i] = A[z + i];
17            forn(i, z) fft::B[i] = B[k - z + 1 + i];
18            fft::multMod(z, z, mod);
19            forn(i, 2 * z - 1) C[k + 1 + i] = (C[k + 1 + i] ←
+ fft::C[i]) % mod;

```

```

20     z <=<= 1;
21 }
22 return C[k];
23 }
24 // A — constant array
25 // magic(k, x):: B[k] = x, returns C[k]
26 // !! WARNING !! better to set N twice the size ←
27 // needed
28 }

```

final/numeric/crt.cpp

```

1 int CRT(int a1, int m1, int a2, int m2) {
2     return (a1 - a2 % m1 + m1) * (ll)rev(m2, m1) % m1 ←
3     * m2 + a2;
4 }

```

final/numeric/mulMod.cpp

```

1 ll mul(ll a, ll b, ll m) { // works for MOD 8e18
2     ll k = (ll)((long double)a * b / m);
3     ll r = a * b - m * k;
4     if (r < 0) r += m;
5     if (r >= m) r -= m;
6     return r;
7 }

```

final/numeric/modReverse.cpp

```

1 int rev(int x, int m) {
2     if (x == 1) return 1;
3     return (1 - rev(m % x, x) * (ll)m) / x + m;
4 }

```

final/numeric/pollard.cpp

```

1 namespace pollard
2 {
3     using math::p;
4
5     vector<pair<ll, int>> getFactors(ll N)
6     {
7         vector<ll> primes;
8
9         const int MX = 1e5;
10        const ll MX2 = MX * (ll)MX;
11
12        assert(MX <= math::maxP && math::pc > 0);
13
14        function<void(ll)> go = [&go, &primes](ll n)
15        {
16            for (ll x : primes) while (n % x == 0) n /= x;
17            if (n == 1) return;
18            if (n > MX2)
19            {
20                auto F = [&](ll x) {
21                    ll k = ((long double)x * x) / n;
22                    ll r = (x * x - k * n + 3) % n;
23                    return r < 0 ? r + n : r;
24                };
25                ll x = mt19937_64()() % n, y = x;
26                const int C = 3 * pow(n, 0.25);
27
28                ll val = 1;
29                for(it, C)
30                {
31                    x = F(x), y = F(y);
32                    if (x == y) continue;
33                    ll delta = abs(x - y);
34                }
35            }
36        };
37    }
38 }

```

```

34 ll k = ((long double)val * delta) / n;
35 val = (val * delta - k * n) % n;
36 if (val < 0) val += n;
37 if (val == 0)
38 {
39     ll g = __gcd(delta, n);
40     go(g), go(n / g);
41     return;
42 }
43 if ((it & 255) == 0)
44 {
45     ll g = __gcd(val, n);
46     if (g != 1)
47     {
48         go(g), go(n / g);
49         return;
50     }
51 }
52 }
53 }
54 primes.pb(n);
55 };
56
57 ll n = N;
58
59 for (int i = 0; i < math::pc && p[i] < MX; ++i) ←
60 if (n % p[i] == 0)
61 {
62     primes.pb(p[i]);
63     while (n % p[i] == 0) n /= p[i];
64 }
65
66 go(n);
67
68 sort(primes.begin(), primes.end());
69
70 vector<pair<ll, int>> res;
71 for (ll x : primes)
72 {
73     int cnt = 0;
74     while (N % x == 0)
75     {
76         cnt++;
77         N /= x;
78     }
79     res.push_back({x, cnt});
80 }
81 return res;
82 }

```

final/numeric/poly.cpp

```

1 struct poly
2 {
3     vi v;
4     poly() {}
5     poly(vi vv)
6     {
7         v = vv;
8     }
9     int size()
10    {
11        return (int)v.size();
12    }
13    poly cut(int maxLen)
14    {
15        if (maxLen < sz(v)) v.resize(maxLen);
16        return *this;
17    }
18    poly norm()
19    {
20        while (sz(v) > 1 && v.back() == 0) v.pop_back();
21        return *this;
22    }
23    inline int& operator [] (int i)
24    {
25        return v[i];
26    }
27    void out(string name="")
28    {
29        stringstream ss;
30        if (sz(name)) ss << name << "=";
31        int fst = 1;
32        for(it, sz(v)) if (v[i])
33        {
34            int x = v[i];
35        }
36    }
37 }

```

```

35     int sgn = 1;
36     if (x > mod / 2) x = mod - x, sgn = -1;
37     if (sgn == -1) ss << "-";
38     else if (!fst) ss << "+";
39     fst = 0;
40     if (!i || x != 1)
41     {
42         ss << x;
43         if (i > 0) ss << "*x";
44         if (i > 1) ss << "^" << i;
45     }
46     else
47     {
48         ss << "x";
49         if (i > 1) ss << "^" << i;
50     }
51 }
52 if (fst) ss << "0";
53 string s;
54 ss >> s;
55 eprintf("%s\n", s.data());
56 }
57 };
58
59 poly operator + (poly A, poly B)
60 {
61     poly C;
62     C.v = vi(max(sz(A), sz(B)));
63     forn(i, sz(C))
64     {
65         if (i < sz(A)) C[i] = (C[i] + A[i]) % mod;
66         if (i < sz(B)) C[i] = (C[i] + B[i]) % mod;
67     }
68     return C.norm();
69 }
70
71 poly operator - (poly A, poly B)
72 {
73     poly C;
74     C.v = vi(max(sz(A), sz(B)));
75     forn(i, sz(C))
76     {
77         if (i < sz(A)) C[i] = (C[i] + A[i]) % mod;
78         if (i < sz(B)) C[i] = (C[i] + mod - B[i]) % mod;
79     }
80     return C.norm();
81 }
82
83 poly operator * (poly A, poly B)
84 {
85     poly C;
86     C.v = vi(sz(A) + sz(B) - 1);
87
88     forn(i, sz(A)) fft::A[i] = A[i];
89     forn(i, sz(B)) fft::B[i] = B[i];
90     fft::multMod(sz(A), sz(B), mod);
91     forn(i, sz(C)) C[i] = fft::C[i];
92     return C.norm();
93 }
94
95 poly inv(poly A, int n) // returns A^{-1} mod x^n
96 {
97     assert(sz(A) && A[0] != 0);
98     A.cut(n);
99
100     auto cutPoly = [](poly &from, int l, int r)
101     {
102         poly R;
103         R.v.resize(r - l);
104         for (int i = l; i < r; ++i)
105         {
106             if (i < sz(from)) R[i - l] = from[i];
107         }
108         return R;
109     };
110
111     function<int(int, int)> rev = [&rev](int x, int m) -> int
112     {
113         if (x == 1) return 1;
114         return (1 - rev(m % x, x) * (ll)m) / x + m;
115     };
116
117     poly R({rev(A[0], mod)});
118     for (int k = 1; k < n; k <= 1)
119     {
120         poly A0 = cutPoly(A, 0, k);
121         poly A1 = cutPoly(A, k, 2 * k);
122         poly H = A0 * R;
123         H = cutPoly(H, k, 2 * k);
124         poly R1 = (((A1 * R).cut(k) + H) * (poly({0}) - R)).cut(k);
125         R.v.resize(2 * k);

```

```

126         forn(i, k) R[i + k] = R1[i];
127     }
128     return R.cut(n).norm();
129 }
130
131 pair<poly, poly> divide(poly A, poly B)
132 {
133     if (sz(A) < sz(B)) return {poly({0}), A};
134
135     auto rev = [](poly f)
136     {
137         reverse(all(f.v));
138         return f;
139     };
140
141     poly q = rev((inv(rev(B), sz(A) - sz(B) + 1) * rev(A)).cut(sz(A) - sz(B) + 1));
142     poly r = A - B * q;
143
144     return {q, r};
145 }

```

final/numeric/simplex.cpp

```

1 vector<double> simplex(vector<vector<double>> > a) {
2     int n = a.size() - 1;
3     int m = a[0].size() - 1;
4     vector<int> left(n + 1), up(m + 1);
5     iota(up.begin(), up.end(), 0);
6     iota(left.begin(), left.end(), m);
7     auto pivot = [&](int x, int y) {
8         swap(left[x], up[y]);
9         double k = a[x][y];
10        a[x][y] = 1;
11        vector<int> vct;
12        for (int j = 0; j <= m; j++) {
13            a[x][j] /= k;
14            if (!eq(a[x][j], 0)) vct.push_back(j);
15        }
16        for (int i = 0; i <= n; i++) {
17            if (eq(a[i][y], 0) || i == x) continue;
18            k = a[i][y];
19            a[i][y] = 0;
20            for (int j : vct) a[i][j] -= k * a[x][j];
21        }
22    };
23    while (1) {
24        int x = -1;
25        for (int i = 1; i <= n; i++) if (ls(a[i][0], 0) <=
26            && (x == -1 || a[i][0] < a[x][0])) x = i;
27        if (x == -1) break;
28        int y = -1;
29        for (int j = 1; j <= m; j++) if (ls(a[x][j], 0) <=
30            && (y == -1 || a[x][j] < a[x][y])) y = j;
31        if (y == -1) assert(0); // infeasible
32        pivot(x, y);
33    }
34    while (1) {
35        int y = -1;
36        for (int j = 1; j <= m; j++) if (ls(0, a[0][j]) <=
37            && (y == -1 || a[0][j] > a[0][y])) y = j;
38        if (y == -1) break;
39        int x = -1;
40        for (int i = 1; i <= n; i++) if (ls(0, a[i][y]) <=
41            && (x == -1 || a[i][0] / a[i][y] < a[x][0] / a[x][y])) x = i;
42        if (x == -1) assert(0); // unbounded
43        pivot(x, y);
44    }
45    vector<double> ans(m + 1);
46    for (int i = 1; i <= n; i++) if (left[i] <= m) ans[
47        left[i]] = a[i][0];
48    ans[0] = -a[0][0];
49    return ans;
50 }
51 // j=1..m: x[j]>=0
52 // i=1..n: sum(j=1..m) A[i][j]*x[j] <= A[i][0]
53 // max sum(j=1..m) A[0][j]*x[j]
54 // res[0] is answer
55 // res[1..m] is certificate

```

final/numeric/sumLine.cpp

```

1 // sum(i=0..n-1) (a+b*i) div m
2 ll solve(ll n, ll a, ll b, ll m) {
3     if (b == 0) return n * (a / m);
4     if (a >= m) return n * (a / m) + solve(n, a % m, b ←
5         , m);
6     if (b >= m) return n * (n - 1) / 2 * (b / m) + ←
7         solve(n, a, b % m, m);
8     return solve((a + b * n) / m, (a + b * n) % m, m, ←
9         b);
10 }

```

final/geom/commonTangents.cpp

```

1 vector<Line> commonTangents(pt A, dbl rA, pt B, dbl ←
2     rB) {
3     vector<Line> res;
4     pt C = B - A;
5     dbl z = C.len2();
6     for (int i = -1; i <= 1; i += 2) {
7         for (int j = -1; j <= 1; j += 2) {
8             dbl r = rB * j - rA * i;
9             dbl d = z - r * r;
10            if (ls(d, 0)) continue;
11            d = sqrt(max(0.01, d));
12            pt magic = pt(r, d) / z;
13            pt v(magic % C, magic * C);
14            dbl CC = (rA * i - v % A) / v.len2();
15            pt O = v * -CC;
16            res.pb(Line(O, O + v.rotate()));
17        }
18    }
19    return res;
20 }
21
22 // HOW TO USE ::
23 // --- *D*-----*F*
24 // --- *...* - - *...*
25 // --- *.....* - - *.....*
26 // --- *.....* - - *.....*
27 // --- *...A...* - - *...B...*
28 // --- *.....* - - *.....*
29 // --- *.....* - - *.....*
30 // --- *...* - - *...*
31 // --- *C*-----*E*
32 // --- res = {CE, CF, DE, DF}
33

```

final/geom/halfplaneIntersection.cpp

```

1 int getPart(pt v) {
2     return less(0, v.y) || (equal(0, v.y) && less(v.x, ←
3         0));
4 }
5
6 int cmpV(pt a, pt b) {
7     int partA = getPart(a);
8     int partB = getPart(b);
9     if (partA < partB) return -1;
10    if (partA > partB) return 1;
11    if (equal(0, a * b)) return 0;
12    if (0 < a * b) return -1;
13    return 1;
14 }
15
16 double planeInt(vector<Line> l) {
17     int n = l.size();
18     sort(all(l), [](Line a, Line b) {
19         int r = cmpV(a.v, b.v);
20         if (r != 0) return r < 0;
21         return a.O % a.v.rotate() < b.O % a.v.rotate() ←
22             ;
23     });
24     int cur = 0;
25     for (int i = 0; i < n; i++) {
26         int j = i;
27         for (; i < n && cmpV(l[j].v, l[i].v) == 0 && ←
28             cmpV(l[i].v, l[j].v) == 0; i++);
29         l[cur++] = l[i - 1];
30     }
31     n = cur;
32     for (int i = 0; i < n; i++)
33         l[i].id = i;
34
35     int flagUp = 0;
36     int flagDown = 0;
37     for (int i = 0; i < n; i++) {
38         int part = getPart(l[i].v);
39         if (part == 1) flagUp = 1;
40         if (part == 0) flagDown = 1;
41     }
42     if (!flagUp || !flagDown) return -1;
43 }

```

```

43 for (int i = 0; i < n; i++) {
44     pt v = l[i].v;
45     pt u = l[(i + 1) % n].v;
46     if (equal(0, v * u) && less(v % u, 0)) {
47         pt dir = l[i].v.rotate();
48         if (lessE(l[(i + 1) % n].0 % dir, l[i].0 % dir ←
49     )) return 0;
50         return -1;
51     }
52     if (less(v * u, 0))
53         return -1;
54 }
55
56 cur = 0;
57 vector<Line> st(n * 2);
58 for (int tt = 0; tt < 2; tt++) {
59     for (int i = 0; i < n; i++) {
60         for (; cur >= 2; cur--) {
61             pt G = st[cur - 1] * l[i];
62             if (!lessE(st[cur - 2].v * (G - st[cur - 2]. ←
63                 0), 0))
64                 break;
65             st[cur++] = l[i];
66             if (cur >= 2 && lessE(st[cur - 2].v * st[cur ←
67                 1].v, 0)) return 0;
68         }
69     }
70     vector<int> use(n, -1);
71     int left = -1, right = -1;
72     for (int i = 0; i < cur; i++) {
73         if (use[st[i].id] == -1) {
74             use[st[i].id] = i;
75         }
76         else {
77             left = use[st[i].id];
78             right = i;
79             break;
80         }
81     }
82     vector<Line> tmp;
83     for (int i = left; i < right; i++)
84         tmp.pb(st[i]);
85     vector<pt> res;
86     for (int i = 0; i < (int)tmp.size(); i++)
87         res.pb(tmp[i] * tmp[(i + 1) % tmp.size()]);
88     double area = 0;
89     for (int i = 0; i < (int)res.size(); i++)
90         area += res[i] * res[(i + 1) % res.size()];
91     return area / 2;
92 }

```

final/geom/minDisc.cpp

```

1 pair<pt, dbl> minDisc(vector<pt> p) {
2     int n = p.size();
3     pt 0 = pt(0, 0);
4     dbl R = 0;
5     random_shuffle(all(p));
6     for (int i = 0; i < n; i++) {
7         if (ls(R, (0 - p[i]).len())) {
8             0 = p[i];
9             R = 0;
10            for (int j = 0; j < i; j++) {
11                if (ls(R, (0 - p[j]).len())) {
12                    0 = (p[i] + p[j]) / 2;
13                    R = (p[i] - p[j]).len() / 2;
14                    for (int k = 0; k < j; k++) {
15                        if (ls(R, (0 - p[k]).len())) {
16                            Line l1((p[i] + p[j]) / 2, (p[i] + p[j] ←
17                                ]) / 2 + (p[i] - p[j]).rotate());
18                            Line l2((p[k] + p[j]) / 2, (p[k] + p[j] ←
19                                ]) / 2 + (p[k] - p[j]).rotate());
20                            0 = l1 * l2;
21                            R = (p[i] - 0).len();
22                        }
23                    }
24                }
25            }
26        }
27        return {0, R};
28    }

```

final/geom/convexHull3D-N2.cpp

```

1 struct Plane {
2     pt 0, v;
3     vector<int> id;
4 };
5
6 vector<Plane> convexHull3(vector<pt> p) {
7     vector<Plane> res;
8     int n = p.size();
9     for (int i = 0; i < n; i++)
10         p[i].id = i;
11     for (int i = 0; i < 4; i++) {
12         vector<pt> tmp;
13         for (int j = 0; j < 4; j++)
14             if (i != j)
15                 tmp.pb(p[j]);
16         res.pb({tmp[0], (tmp[1] - tmp[0]) * (tmp[2] - ←
17             tmp[0]), {tmp[0].id, tmp[1].id, tmp[2].id}});
18         if ((p[i] - res.back().0) % res.back().v > 0) {
19             res.back().v = res.back().v * -1;
20             swap(res.back().id[0], res.back().id[1]);
21         }
22     }
23     vector<vector<int>> use(n, vector<int>(n, 0));
24     int tnr = 0;
25     for (int i = 4; i < n; i++) {
26         int cur = 0;
27         tnr++;
28         vector<pair<int, int>> curEdge;
29         for (int j = 0; j < sz(res); j++) {
30             if ((p[i] - res[j].0) % res[j].v > 0) {
31                 for (int t = 0; t < 3; t++) {
32                     int v = res[j].id[t];
33                     int u = res[j].id[(t + 1) % 3];
34                     use[v][u] = tnr;
35                     curEdge.pb({v, u});
36                 }
37             }
38             else {
39                 res[cur++] = res[j];
40             }
41         }
42         res.resize(cur);
43         for (auto x: curEdge) {
44             if (use[x.S][x.F] == tnr) continue;
45             res.pb({p[i], (p[x.F] - p[i]) * (p[x.S] - p[i] ←
46                 )}, {x.F, x.S, i});
47         }
48     }
49     return res;
50 }
51
52 // plane in 3d
53 // (A, v) * (B, u) -> (O, n)
54 pt n = v * u;
55 pt m = v * n;
56 double t = (B - A) % u / (u % m);
57 pt 0 = A - m * t;

```

final/geom/polygonArcCut.cpp

```

1 struct Meta {
2     int type; // 0 - seg, 1 - circle
3     pt 0;
4     dbl R;
5 };
6
7 const Meta SEG = {0, pt(0, 0), 0};
8
9 vector<pair<pt, Meta>> cut(vector<pair<pt, Meta>> p, ←
10     Line l) {
11     vector<pair<pt, Meta>> res;
12     int n = p.size();
13     for (int i = 0; i < n; i++) {
14         pt A = p[i].F;
15         pt B = p[(i + 1) % n].F;
16         if (le(0, l.v * (A - l.0))) {
17             if (eq(0, l.v * (A - l.0)) && p[i].S.type == 1 ←
18                 && ls(0, l.v % (p[i].S.0 - A)))
19                 res.pb({A, SEG});
20         }
21     }
22 }

```



```

19     else
20         res.pb(p[i]);
21     }
22     if (p[i].S.type == 0) {
23         if (sign(l.v * (A - l.0)) * sign(l.v * (B - l.0)) == -1) {
24             pt FF = Line(A, B) * l;
25             res.pb(make_pair(FF, SEG));
26         }
27     }
28     else {
29         pt E, F;
30         if (intCL(p[i].S.O, p[i].S.R, l, E, F)) {
31             if (onArc(p[i].S.O, A, E, B))
32                 res.pb({E, SEG});
33             if (onArc(p[i].S.O, A, F, B))
34                 res.pb({F, p[i].S});
35         }
36     }
37     return res;
38 }

```

final/strings/eertree.cpp

```

1 namespace eertree {
2     const int INF = 1e9;
3     const int N = 5e6 + 10;
4     char _s[N];
5     char *s = _s + 1;
6     int to[N][2];
7     int suf[N], len[N];
8     int sz, last;
9
10     const int odd = 1, even = 2, blank = 3;
11
12     void go(int &u, int pos) {
13         while (u != blank && s[pos - len[u] - 1] != s[pos]) {
14             u = suf[u];
15         }
16     }
17
18     int add(int pos) {
19         go(last, pos);
20         int u = suf[last];
21         go(u, pos);
22         int c = s[pos] - 'a';
23         int res = 0;
24         if (!to[last][c]) {
25             res = 1;
26             to[last][c] = sz;
27             len[sz] = len[last] + 2;
28             suf[sz] = to[u][c];
29             sz++;
30         }
31         last = to[last][c];
32         return res;
33     }
34
35     void init() {
36         to[blank][0] = to[blank][1] = even;
37         len[blank] = suf[blank] = INF;
38         len[even] = 0, suf[even] = odd;
39         len[odd] = -1, suf[odd] = blank;
40         last = even;
41         sz = 4;
42     }
43 }

```

```

10     memset(nxt, -1, sizeof(nxt));
11     memset(link, -1, sizeof(link));
12     memset(len, 0, sizeof(len));
13     last = 0;
14     sz = 1;
15 }
16
17 void add(int c) {
18     int cur = sz++;
19     len[cur] = len[last] + 1;
20     pos[cur] = len[cur];
21     int p = last;
22     last = cur;
23     for (; p != -1 && nxt[p][c] == -1; p = link[p]) ←
24         nxt[p][c] = cur;
25     if (p == -1) {
26         link[cur] = 0;
27         return;
28     }
29     int q = nxt[p][c];
30     if (len[p] + 1 == len[q]) {
31         link[cur] = q;
32         return;
33     }
34     int clone = sz++;
35     memcpy(nxt[clone], nxt[q], sizeof(nxt[q]));
36     len[clone] = len[p] + 1;
37     pos[clone] = pos[q];
38     link[clone] = link[q];
39     link[q] = link[cur] = clone;
40     for (; p != -1 && nxt[p][c] == q; p = link[p]) ←
41         nxt[p][c] = clone;
42 }
43
44 int n;
45 string s;
46 int l[MAXN], r[MAXN];
47 int e[MAXN][SIGMA];
48
49 void getSufTree(string _s) {
50     memset(e, -1, sizeof(e));
51     s = _s;
52     n = s.length();
53     reverse(s.begin(), s.end());
54     init();
55     for (int i = 0; i < n; i++) add(s[i] - 'a');
56     reverse(s.begin(), s.end());
57     for (int i = 1; i < sz; i++) {
58         int j = link[i];
59         l[i] = n - pos[i] + len[j];
60         r[i] = n - pos[i] + len[i];
61         e[j][s[l[i]] - 'a'] = i;
62     }
63 }
64
65 namespace duval {
66     string s;
67     int n = (int) s.length();
68     int i = 0;
69     while (i < n) {
70         int j = i + 1, k = i;
71         while (j < n && s[k] <= s[j]) {
72             if (s[k] < s[j])
73                 k = i;
74             else
75                 ++k;
76             ++j;
77         }
78         while (i <= k) {
79             cout << s.substr(i, j - k) << ' ';
80             i += j - k;
81         }
82     }
83 }

```

final/strings/sufAutomaton.cpp

```

1 namespace SA {
2     const int MAXN = 1 << 18;
3     const int SIGMA = 26;
4
5     int sz, last;
6     int nxt[MAXN][SIGMA];
7     int link[MAXN], len[MAXN], pos[MAXN];
8
9     void init() {

```

final/graphs/centroid.cpp

```

1 // original author: burunduk1, rewritten by me (←
2 // !!! warning !!! this code is not tested well
3 const int N = 1e5, K = 17;
4
5 int pivot, level[N], parent[N];
6 vector<int> v[N];
7
8 int get_pivot( int x, int xx, int n ) {
9     int size = 1;
10    for ( int y : v[x] )
11    {
12        if ( y != xx && level[y] == -1 ) size += get_pivot(
13            (y, x, n);
14    }
15    if ( pivot == -1 && (size * 2 >= n || xx == -1) ) ←
16        pivot = x;
17    return size;
18 }
19
20 void build( int x, int xx, int dep, int size ) {
21     assert( dep < K );
22     pivot = -1;
23     get_pivot( x, -1, size );
24     x = pivot;
25     level[x] = dep, parent[x] = xx;
26     for ( int y : v[x] ) if ( level[y] == -1 )
27     {
28         build( y, x, dep + 1, size / 2 );
29     }
30 }

```

final/graphs/dominatorTree.cpp

```

1 namespace domtree {
2     const int K = 18;
3     const int N = 1 << K;
4
5     int n, root;
6     vector<int> e[N], g[N];
7     int sdom[N], dom[N];
8     int p[N][K], h[N], pr[N];
9     int in[N], out[N], tmr, rev[N];
10
11     void init( int _n, int _root ) {
12         n = _n;
13         root = _root;
14         tmr = 0;
15         for ( int i = 0; i < n; i++ ) {
16             e[i].clear();
17             g[i].clear();
18             in[i] = -1;
19         }
20     }
21
22     void addEdge( int u, int v ) {
23         e[u].push_back( v );
24         g[v].push_back( u );
25     }
26
27     void dfs( int v ) {
28         in[v] = tmr++;
29         for ( int to : e[v] ) {
30             if ( in[to] != -1 ) continue;
31             pr[to] = v;
32             dfs( to );
33         }
34         out[v] = tmr - 1;
35     }
36
37     int lca( int u, int v ) {
38         if ( h[u] < h[v] ) swap( u, v );
39         for ( int i = 0; i < K; i++ ) if ( (h[u] - h[v]) & ←
40             (1 << i) ) u = p[u][i];
41         if ( u == v ) return u;
42         for ( int i = K - 1; i >= 0; i-- ) {
43             if ( p[u][i] != p[v][i] ) {
44                 u = p[u][i];
45                 v = p[v][i];
46             }
47         }
48         return p[u][0];
49     }
50 }

```

```

48 }
49
50 void solve( int _n, int _root, vector<pair<int, int ←
51 >> _edges ) {
52     init( _n, _root );
53     for ( auto ed : _edges ) addEdge( ed.first, ed. ←
54         second );
55
56     dfs( root );
57     for ( int i = 0; i < n; i++ ) if ( in[i] != -1 ) rev ←
58         [in[i]] = i;
59     segtree tr( tmr ); // a[i] := min( a[i], x ) and return ←
60         a[i]
61     for ( int i = tmr - 1; i >= 0; i-- ) {
62         int v = rev[i];
63         int cur = i;
64         for ( int to : g[v] ) {
65             if ( in[to] == -1 ) continue;
66             if ( in[to] < in[v] ) cur = min( cur, in[to] );
67             else cur = min( cur, tr.get( in[to] ) );
68         }
69         sdom[v] = rev[cur];
70         tr.upd( in[v], out[v], in[sdom[v]] );
71     }
72     for ( int i = 0; i < tmr; i++ ) {
73         int v = rev[i];
74         if ( i == 0 ) {
75             dom[v] = v;
76             h[v] = 0;
77         } else {
78             dom[v] = lca( sdom[v], pr[v] );
79             h[v] = h[dom[v]] + 1;
80         }
81         p[v][0] = dom[v];
82         for ( int j = 1; j < K; j++ ) p[v][j] = p[p[v][j ←
83             - 1][j - 1];
84     }
85     for ( int i = 0; i < n; i++ ) if ( in[i] == -1 ) dom ←
86         [i] = -1;
87 }
88 }

```

final/graphs/generalMatching.cpp

```

1 //COPYPASTED FROM E-MAXX
2 namespace GeneralMatching {
3     const int MAXN = 256;
4     int n;
5     vector<int> g[MAXN];
6     int match[MAXN], p[MAXN], base[MAXN], q[MAXN];
7     bool used[MAXN], blossom[MAXN];
8
9     int lca( int a, int b ) {
10         bool used[MAXN] = { 0 };
11         for ( ;; ) {
12             a = base[a];
13             used[a] = true;
14             if ( match[a] == -1 ) break;
15             a = p[match[a]];
16         }
17         for ( ;; ) {
18             b = base[b];
19             if ( used[b] ) return b;
20             b = p[match[b]];
21         }
22     }
23
24     void mark_path( int v, int b, int children ) {
25         while ( base[v] != b ) {
26             blossom[base[v]] = blossom[base[match[v]]] = ←
27                 true;
28             p[v] = children;
29             children = match[v];
30             v = p[match[v]];
31         }
32     }
33
34     int find_path( int root ) {
35         memset( used, 0, sizeof used );
36         memset( p, -1, sizeof p );
37         for ( int i = 0; i < n; ++i )
38             base[i] = i;
39
40         used[root] = true;
41         int qh = 0, qt = 0;
42         q[qt++] = root;
43         while ( qh < qt ) {

```

```

43     int v = q[qh++];
44     for (size_t i=0; i<g[v].size(); ++i) {
45         int to = g[v][i];
46         if (base[v] == base[to] || match[v] == to) ←
47             continue;
48         if (to == root || (match[to] != -1 && p[←
49             match[to]] != -1)) {
50             int curbase = lca(v, to);
51             memset(blossom, 0, sizeof blossom);
52             mark_path(v, curbase, to);
53             mark_path(to, curbase, v);
54             for (int i=0; i<n; ++i)
55                 if (blossom[base[i]]) {
56                     base[i] = curbase;
57                     if (!used[i]) {
58                         used[i] = true;
59                         q[qt++] = i;
60                     }
61                 }
62             else if (p[to] == -1) {
63                 p[to] = v;
64                 if (match[to] == -1)
65                     return to;
66                 to = match[to];
67                 used[to] = true;
68                 q[qt++] = to;
69             }
70         }
71     }
72     return -1;
73 }
74
75 vector<pair<int, int>> solve(int _n, vector<pair<←
76     int, int>> edges) {
77     n = _n;
78     for (int i = 0; i < n; i++) g[i].clear();
79     for (auto o : edges) {
80         g[o.first].push_back(o.second);
81         g[o.second].push_back(o.first);
82     }
83     memset(match, -1, sizeof match);
84     for (int i=0; i<n; ++i) {
85         if (match[i] == -1) {
86             int v = find_path(i);
87             while (v != -1) {
88                 int pv = p[v], ppv = match[pv];
89                 match[v] = pv, match[pv] = v;
90                 v = ppv;
91             }
92         }
93     }
94     vector<pair<int, int>> ans;
95     for (int i = 0; i < n; i++) {
96         if (match[i] > i) {
97             ans.push_back(make_pair(i, match[i]));
98         }
99     }
100     return ans;
}

```

final/graphs/heavyLight.cpp

```

1 namespace hld {
2     const int N = 1 << 17;
3     int par[N], heavy[N], h[N];
4     int root[N], pos[N];
5     int n;
6     vector<vector<int>> e;
7     segtree tree;
8
9     int dfs(int v) {
10         int sz = 1, mx = 0;
11         for (int to : e[v]) {
12             if (to == par[v]) continue;
13             par[to] = v;
14             h[to] = h[v] + 1;
15             int cur = dfs(to);
16             if (cur > mx) heavy[v] = to, mx = cur;
17             sz += cur;
18         }
19         return sz;
20     }
21
22     template <typename T>
23     void path(int u, int v, T op) {

```

```

24         for (; root[u] != root[v]; v = par[root[v]]) {
25             if (h[root[u]] > h[root[v]]) swap(u, v);
26             op(pos[root[v]], pos[v] + 1);
27         }
28         if (h[u] > h[v]) swap(u, v);
29         op(pos[u], pos[v] + 1);
30     }
31
32     void init(vector<vector<int>> _e) {
33         e = _e;
34         n = e.size();
35         tree = segtree(n);
36         memset(heavy, -1, sizeof(heavy[0]) * n);
37         par[0] = -1;
38         h[0] = 0;
39         dfs(0);
40         for (int i = 0, cpos = 0; i < n; i++) {
41             if (par[i] == -1 || heavy[par[i]] != i) {
42                 for (int j = i; j != -1; j = heavy[j]) {
43                     root[j] = i;
44                     pos[j] = cpos++;
45                 }
46             }
47         }
48     }
49
50     void add(int v, int x) {
51         tree.add(pos[v], x);
52     }
53
54     int get(int u, int v) {
55         int res = 0;
56         path(u, v, [&](int l, int r) {
57             res = max(res, tree.get(l, r));
58         });
59         return res;
60     }
61 }

```

final/graphs/hungary.cpp

```

1 namespace hungary
2 {
3     const int N = 210;
4
5     int a[N][N];
6     int ans[N];
7
8     int calc(int n, int m)
9     {
10         ++n, ++m;
11         vi u(n), v(m), p(m), prev(m);
12         for (int i = 1; i < n; ++i)
13         {
14             p[0] = i;
15             int x = 0;
16             vi mn(m, inf);
17             vi was(m, 0);
18             while (p[x])
19             {
20                 was[x] = 1;
21                 int ii = p[x], dd = inf, y = 0;
22                 for (int j = 1; j < m; ++j) if (!was[j])
23                 {
24                     int cur = a[ii][j] - u[ii] - v[j];
25                     if (cur < mn[j]) mn[j] = cur, prev[j] = x;
26                     if (mn[j] < dd) dd = mn[j], y = j;
27                 }
28                 forn(j, m)
29                 {
30                     if (was[j]) u[p[j]] += dd, v[j] -= dd;
31                     else mn[j] -= dd;
32                 }
33                 x = y;
34             }
35             while (x)
36             {
37                 int y = prev[x];
38                 p[x] = p[y];
39                 x = y;
40             }
41         }
42         for (int j = 1; j < m; ++j)
43         {
44             ans[p[j]] = j;
45         }
46         return -v[0];

```

```

47 }
48 // HOW TO USE ::
49 // -- set values to a[1..n][1..m] (n <= m)
50 // -- run calc(n, m) to find MINIMUM
51 // -- to restore permutation use ans[]
52 // -- everything works on negative numbers
53 //
54 // !! i don't understand this code, it's ←
55 // copy-pasted from e-maxx (and rewrited by enot110←
56 )
57 }

```

final/graphs/retro.cpp

```

1 namespace retro
2 {
3     const int N = 4e5 + 10;
4
5     vi v[N];
6     vi vrev[N];
7
8     void add(int x, int y)
9     {
10         v[x].pb(y);
11         vrev[y].pb(x);
12     }
13
14     const int UD = 0;
15     const int WIN = 1;
16     const int LOSE = 2;
17
18     int res[N];
19     int moves[N];
20     int deg[N];
21     int q[N], st, en;
22
23     void calc(int n)
24     {
25         forn(i, n) deg[i] = sz(v[i]);
26         st = en = 0;
27         forn(i, n) if (!deg[i])
28         {
29             q[en++] = i;
30             res[i] = LOSE;
31         }
32         while (st < en)
33         {
34             int x = q[st++];
35             for (int y : vrev[x])
36             {
37                 if (res[y] == UD && (res[x] == LOSE || (--←
38                     deg[y] == 0 && res[x] == WIN)))
39                 {
40                     res[y] = 3 - res[x];
41                     moves[y] = moves[x] + 1;
42                     q[en++] = y;
43                 }
44             }
45         }
46     }

```

final/graphs/smith.cpp

```

1 const int N = 1e5 + 10;
2
3 struct graph
4 {
5     int n;
6
7     vi v[N];
8     vi vrev[N];
9
10    void read()
11    {
12        int m;
13        scanf("%d", &m);
14        forn(i, m)
15        {
16            int x, y;
17            scanf("%d%d", &x, &y);

```

```

18         --x, --y;
19         v[x].pb(y);
20         vrev[y].pb(x);
21     }
22 }
23
24 int deg[N], cnt[N], used[N], f[N];
25 int q[N], st, en;
26
27 set<int> s[N];
28
29 void calc()
30 {
31     forn(x, n) f[x] = -1, cnt[x] = 0;
32     int val = 0;
33     while (1)
34     {
35         st = en = 0;
36         forn(x, n)
37         {
38             deg[x] = 0;
39             used[x] = 0;
40             for (int y : v[x]) if (f[y] == -1) deg[x]++;
41         }
42         forn(x, n) if (!deg[x] && f[x] == -1 && cnt[x] ←
43             == val)
44         {
45             q[en++] = x;
46             f[x] = val;
47         }
48         if (!en) break;
49         while (st < en)
50         {
51             int x = q[st];
52             st++;
53             for (int y : vrev[x])
54             {
55                 if (used[y] == 0 && f[y] == -1)
56                 {
57                     used[y] = 1;
58                     cnt[y]++;
59                     for (int z : vrev[y])
60                     {
61                         deg[z]--;
62                         if (f[z] == -1 && deg[z] == 0 && cnt[z] ←
63                             == val)
64                         {
65                             f[z] = val;
66                             q[en++] = z;
67                         }
68                     }
69                 }
70             }
71             val++;
72         }
73         forn(x, n) eprintf("%d%c", f[x], " \n"[x + 1 == ←
74             n]);
75         forn(x, n) if (f[x] == -1)
76         {
77             for (int y : v[x]) if (f[y] != -1) s[x].insert←
78                 (f[y]);
79         }
80     }
81     g1, g2;
82
83     string get(int x, int y)
84     {
85         int f1 = g1.f[x], f2 = g2.f[y];
86         if (f1 == -1 && f2 == -1) return "draw";
87         if (f1 == -1) {
88             if (g1.s[x].count(f2)) return "first";
89             return "draw";
90         }
91         if (f2 == -1) {
92             if (g2.s[y].count(f1)) return "first";
93             return "draw";
94         }
95         if (f1 ^ f2) return "first";
96         return "second";
97     }

```

final/graphs/twoChinese.cpp

```

1 const int INF = 1e9;
2 struct Edge {
3     int from, to, w, id;

```

```

4 };
5 namespace dmst {
6     int n;
7     vector<int> p;
8     vector<Edge> edges;
9
10    int get(int x) {
11        if (x == p[x]) return x;
12        return p[x] = get(p[x]);
13    }
14
15    void uni(int u, int v) {
16        p[get(v)] = get(u);
17    }
18
19    vector<Edge> solve() {
20        vector<int> id(n, -1);
21        vector<int> vert;
22        int cn = 0;
23        for (int i = 0; i < n; i++) if (get(i) == i) {
24            vert.push_back(i);
25            id[i] = cn++;
26        }
27        if (cn == 1) return vector<Edge>();
28
29        vector<vector<int>> e(cn);
30        for (int i = 0; i < (int)edges.size(); i++) {
31            if (get(edges[i].to) != get(edges[i].from)) {
32                e[id[get(edges[i].to)]].push_back(i);
33            }
34        }
35
36        vector<int> nxtId(cn, -1);
37        for (int i = 0; i < cn; i++) {
38            int mn = INF;
39            for (int id : e[i]) mn = min(mn, edges[id].w);
40            for (int id : e[i]) {
41                edges[id].w -= mn;
42                if (edges[id].w == 0) nxtId[i] = id;
43            }
44        }
45
46        vector<char> vis(cn);
47        vis[0] = 1;
48        int cur = 1;
49        while (!vis[cur]) {
50            vis[cur] = 1;
51            cur = id[get(edges[nxtId[cur]].from)];
52        }
53        vector<Edge> ans;
54        if (cur == 0) {
55            for (int i = 0; i < cn; i++) {
56                if (vis[i] && i != 0) {
57                    ans.push_back(edges[nxtId[i]]);
58                    uni(0, vert[i]);
59                }
60            }
61            auto nans = solve();
62            for (auto ee : nans) ans.push_back(ee);
63            return ans;
64        }
65        vector<int> cp = p;
66        int o = cur;
67        while (1) {
68            uni(vert[o], vert[cur]);
69            ans.push_back(edges[nxtId[cur]]);
70            int to = id[get(edges[nxtId[cur]].from)];
71            if (to == o) break;
72            cur = to;
73        }
74        vector<Edge> nedges = solve();
75        p = cp;
76        vector<char> covered(cn);
77        for (auto ee : nedges) covered[id[get(ee.to)]] +=
78            1;
79        for (auto ee : ans) if (!covered[id[get(ee.to)]] <=
80            1) nedges.push_back(ee);
81        return nedges;
82    }
83
84    // root is 0
85    vector<Edge> getMst(int _n, vector<Edge> _edges) {
86        n = _n;
87        edges = _edges;
88        p.resize(n);
89        for (int i = 0; i < n; i++) p[i] = i;
90
91        return solve();
92    }
93 }

```

final/graphs/linkcut.cpp

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cassert>
4
5  using namespace std;
6
7  // BEGIN ALGO
8
9  const int MAXN = 110000;
10
11  typedef struct _node{
12      _node *l, *r, *p, *pp;
13      int size; bool rev;
14      _node();
15      explicit _node(nullptr_t){
16          l = r = p = pp = this;
17          size = rev = 0;
18      }
19      void push(){
20          if (rev){
21              l->rev ^= 1; r->rev ^= 1;
22              rev = 0; swap(l, r);
23          }
24      }
25      void update();
26  }* node;
27  node None = new _node(nullptr);
28  node v2n[MAXN];
29  _node::_node(){
30      l = r = p = pp = None;
31      size = 1; rev = false;
32  }
33  void _node::update(){
34      size = (this != None) + l->size + r->size;
35      l->p = r->p = this;
36  }
37  void rotate(node v){
38      assert(v != None && v->p != None);
39      assert(!v->rev); assert(!v->p->rev);
40      node u = v->p;
41      if (v == u->l){
42          u->l = v->r; v->r = u;
43      }
44      else{
45          u->r = v->l; v->l = u;
46          swap(u->p, v->p); swap(v->pp, u->pp);
47          if (v->p != None){
48              assert(v->p->l == u || v->p->r == u);
49              if (v->p->r == u) v->p->r = v;
50              else v->p->l = v;
51          }
52          u->update(); v->update();
53      }
54  }
55  void bigRotate(node v){
56      assert(v->p != None);
57      v->p->p->push();
58      v->p->push();
59      v->push();
60      if (v->p->p != None){
61          if ((v->p->l == v) ^ (v->p->p->r == v->p))
62              rotate(v->p);
63          else
64              rotate(v);
65      }
66      rotate(v);
67  }
68  inline void Splay(node v){
69      while (v->p != None) bigRotate(v);
70  }
71  inline void splitAfter(node v){
72      v->push();
73      Splay(v);
74      v->r->p = None;
75      v->r->pp = v;
76      v->r = None;
77      v->update();
78  }
79
80  void expose(int x){
81      node v = v2n[x];
82      splitAfter(v);
83      while (v->pp != None){
84          assert(v->p == None);
85          splitAfter(v->pp);
86          assert(v->pp->r == None);
87          assert(v->pp->p == None);
88          assert(!v->pp->rev);
89          v->pp->r = v;
90          v->pp->update();
91          v = v->pp;
92      }
93  }

```

```

89     v->r->pp = None;
90 }
91 assert(v->p == None);
92 Splay(v2n[x]);
93 }
94 inline void makeRoot(int x){
95     expose(x);
96     assert(v2n[x]->p == None);
97     assert(v2n[x]->pp == None);
98     assert(v2n[x]->r == None);
99     v2n[x]->rev ^= 1;
100 }
101 inline void link(int x,int y){
102     makeRoot(x); v2n[x]->pp = v2n[y];
103 }
104 inline void cut(int x,int y){
105     expose(x);
106     Splay(v2n[y]);
107     if (v2n[y]->pp != v2n[x]){
108         swap(x,y);
109         expose(x);
110         Splay(v2n[y]);
111         assert(v2n[y]->pp == v2n[x]);
112     }
113     v2n[y]->pp = None;
114 }
115 inline int get(int x,int y){
116     if (x == y) return 0;
117     makeRoot(x);
118     expose(y); expose(x);
119     Splay(v2n[y]);
120     if (v2n[y]->pp != v2n[x]) return -1;
121     return v2n[y]->size;
122 }
123 // END ALGO
124
125 _node mem[MAXN];
126
127 int main(){
128     freopen("linkcut.in","r",stdin);
129     freopen("linkcut.out","w",stdout);
130
131     int n,m;
132     scanf("%d %d",&n,&m);
133
134     for (int i = 0; i < n; i++){
135         v2n[i] = &mem[i];
136     }
137
138     for (int i = 0; i < m; i++){
139         int a,b;
140         if (scanf("link %d %d",&a,&b) == 2)
141             link(a-1,b-1);
142         else if (scanf("cut %d %d",&a,&b) == 2)
143             cut(a-1,b-1);
144         else if (scanf("get %d %d",&a,&b) == 2)
145             printf("%d\n",get(a-1,b-1));
146         else
147             assert(false);
148     }
149     return 0;
150 }

```

```

dbl Simpson() { return (F(-1) + 4 * F(0) + F(1)) / 6;
} dbl Runge2() { return (F(-sqrtl(1.0 / 3)) + F(sqrtl(1.0 /
3))) / 2; } dbl Runge3() { return (F(-sqrtl(3.0 / 5)) * 5 +
F(0) * 8 + F(sqrtl(3.0 / 5)) * 5) / 18; }

```

Simpson и Runge2 – точны для полиномов степени ≤ 3
 Runge3 – точен для полиномов степени ≤ 5

—
 Явный Рунге-Кутты четвертого порядка, ошибка $O(h^4)$

$y' = f(x, y)$ $y_{n+1} = y_n + (k_1 + 2 * k_2 + 2 * k_3 + k_4) * h / 6$

$k_1 = f(x_n, y_n)$ $k_2 = f(x_n + h/2, y_n + h/2 * k_1)$ $k_3 = f(x_n + h/2, y_n + h/2 * k_2)$ $k_4 = f(x_n + h, y_n + h * k_3)$

Методы Адамса-Башфорта

$y_{n+3} = y_{n+2} + h * (23/12 * f(x_{n+2}, y_{n+2}) - 4/3 * f(x_{n+1}, y_{n+1}) + 5/12 * f(x_n, y_n))$
 $y_{n+4} = y_{n+3} + h * (55/24 * f(x_{n+3}, y_{n+3}) - 59/24 * f(x_{n+2}, y_{n+2}) + 37/24 * f(x_{n+1}, y_{n+1}) - 3/8 * f(x_n, y_n))$
 $y_{n+5} = y_{n+4} + h * (1901/720 * f(x_{n+4}, y_{n+4}) - 1387/360 * f(x_{n+3}, y_{n+3}) + 109/30 * f(x_{n+2}, y_{n+2}) - 637/360 * f(x_{n+1}, y_{n+1}) + 251/720 * f(x_n, y_n))$

—
 Извлечение корня по простому модулю (от Серёжи) $3 \leq p$, $1 \leq a < p$, найти $x^2 = a$

1) Если $a^{((p-1)/2)} \neq 1$, return -1
 2) Выбрать случайный $1 \leq i < p$
 3) $T(x) = (x+i)^{((p-1)/2)} \bmod (x^2 - a) = bx + c$
 4) Если $b \neq 0$ то вернуть c/b , иначе к шагу 2)

—
 Иногда вместо того чтобы считать первообразный у простого числа, можно написать чекер ответа и перебирать случайный первообразный.

Не заходит FFT по TL-ю – чекнуть что стоит double, а не long double

mt19937 генерит случайный unsigned int, если хочется больше есть mt19937_64

Иногда можно представить ответ в виде многочлена и вместо подсчета самих k-тов посчитать значения и проинтерполировать

Перед сабмитом чекнуть что все выводится в printf, а не eprintf!!!

—
 Лемма Бернсайда:

Группа G действует на множество X Тогда число классов эквивалентности $= (\sum |f(g)| \text{ for } g \in G) / |G|$ где $f(g) = \text{число } x \text{ (из } X) : g(x) = x$

—
 Число простых быстрее $O(n)$:

$dp(n, k)$ – число чисел от 1 до n в которых все простые $\geq p[k]$
 $dp(n, 1) = n$ $dp(n, j) = dp(n, j+1) + dp(n / p[j], j)$, т. е. $dp(n, j+1) = dp(n, j) - dp(n / p[j], j)$

Если $p[j], p[k] > \sqrt{n}$ то $dp(n, j) + j == dp(n, k) + k$

Хуяришь все оптимайзы сверху, но не считаешь глубже $dp(n, k)$, $n < K$ Потом фенвиком+сортировкой подсчитываешь за $(K+Q)\log$ все эти запросы Хуяришь во второй раз, но на этот раз берешь прекальканные значения

Если $\sqrt{n} < p[k] < n$ то (число простых до n) $= dp(n, k) + k - 1$

—
 Чиселки:

Table of Integrals*

Basic Forms

$$\int x^n dx = \frac{1}{n+1} x^{n+1} \quad (1)$$

$$\int \frac{1}{x} dx = \ln |x| \quad (2)$$

$$\int u dv = uv - \int v du \quad (3)$$

$$\int \frac{1}{ax+b} dx = \frac{1}{a} \ln |ax+b| \quad (4)$$

Integrals of Rational Functions

$$\int \frac{1}{(x+a)^2} dx = -\frac{1}{x+a} \quad (5)$$

$$\int (x+a)^n dx = \frac{(x+a)^{n+1}}{n+1}, n \neq -1 \quad (6)$$

$$\int x(x+a)^n dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \quad (7)$$

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x \quad (8)$$

$$\int \frac{1}{a^2+x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a} \quad (9)$$

$$\int \frac{x}{a^2+x^2} dx = \frac{1}{2} \ln |a^2+x^2| \quad (10)$$

$$\int \frac{x^2}{a^2+x^2} dx = x - a \tan^{-1} \frac{x}{a} \quad (11)$$

$$\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln |a^2+x^2| \quad (12)$$

$$\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \quad (13)$$

$$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a} \ln \frac{a+x}{b+x}, a \neq b \quad (14)$$

$$\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln |a+x| \quad (15)$$

$$\int \frac{x}{ax^2+bx+c} dx = \frac{1}{2a} \ln |ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \quad (16)$$

Integrals with Roots

$$\int \sqrt{x-a} dx = \frac{2}{3} (x-a)^{3/2} \quad (17)$$

$$\int \frac{1}{\sqrt{x \pm a}} dx = 2\sqrt{x \pm a} \quad (18)$$

$$\int \frac{1}{\sqrt{a-x}} dx = -2\sqrt{a-x} \quad (19)$$

$$\int x\sqrt{x-a} dx = \frac{2}{3} a(x-a)^{3/2} + \frac{2}{5} (x-a)^{5/2} \quad (20)$$

$$\int \sqrt{ax+bd} dx = \left(\frac{2b}{3a} + \frac{2x}{3} \right) \sqrt{ax+b} \quad (21)$$

$$\int (ax+b)^{3/2} dx = \frac{2}{5a} (ax+b)^{5/2} \quad (22)$$

$$\int \frac{x}{\sqrt{x \pm a}} dx = \frac{2}{3} (x \mp 2a) \sqrt{x \pm a} \quad (23)$$

$$\int \sqrt{\frac{x}{a-x}} dx = -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a} \quad (24)$$

$$\int \sqrt{\frac{x}{a+x}} dx = \sqrt{x(a+x)} - a \ln [\sqrt{x} + \sqrt{x+a}] \quad (25)$$

$$\int x\sqrt{ax+bd} dx = \frac{2}{15a^2} (-2b^2 + abx + 3a^2x^2) \sqrt{ax+b} \quad (26)$$

$$\int \sqrt{x(ax+b)} dx = \frac{1}{4a^{3/2}} \left[(2ax+b) \sqrt{ax(ax+b)} - b^2 \ln |a\sqrt{x} + \sqrt{a(ax+b)}| \right] \quad (27)$$

$$\int \sqrt{x^3(ax+b)} dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3} \right] \sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}} \ln |a\sqrt{x} + \sqrt{a(ax+b)}| \quad (28)$$

$$\int \sqrt{x^2 \pm a^2} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln |x + \sqrt{x^2 \pm a^2}| \quad (29)$$

$$\int \sqrt{a^2 - x^2} dx = \frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2 - x^2}} \quad (30)$$

$$\int x\sqrt{x^2 \pm a^2} dx = \frac{1}{3} (x^2 \pm a^2)^{3/2} \quad (31)$$

$$\int \frac{1}{\sqrt{x^2 \pm a^2}} dx = \ln |x + \sqrt{x^2 \pm a^2}| \quad (32)$$

$$\int \frac{1}{\sqrt{a^2 - x^2}} dx = \sin^{-1} \frac{x}{a} \quad (33)$$

$$\int \frac{x}{\sqrt{x^2 \pm a^2}} dx = \sqrt{x^2 \pm a^2} \quad (34)$$

$$\int \frac{x}{\sqrt{a^2 - x^2}} dx = -\sqrt{a^2 - x^2} \quad (35)$$

$$\int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln |x + \sqrt{x^2 \pm a^2}| \quad (36)$$

$$\int \sqrt{ax^2+bx+cd} dx = \frac{b+2ax}{4a} \sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}} \ln |2ax+b+2\sqrt{a(ax^2+bx+c)}| \quad (37)$$

$$\int x\sqrt{ax^2+bx+c} dx = \frac{1}{48a^{5/2}} \left(2\sqrt{a}\sqrt{ax^2+bx+c} \times (-3b^2+2abx+8a(c+ax^2)) + 3(b^3-4abc) \ln |b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}| \right) \quad (38)$$

$$\int \frac{1}{\sqrt{ax^2+bx+c}} dx = \frac{1}{\sqrt{a}} \ln |2ax+b+2\sqrt{a(ax^2+bx+c)}| \quad (39)$$

$$\int \frac{x}{\sqrt{ax^2+bx+c}} dx = \frac{1}{a} \sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}} \ln |2ax+b+2\sqrt{a(ax^2+bx+c)}| \quad (40)$$

$$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}} \quad (41)$$

Integrals with Logarithms

$$\int \ln ax dx = x \ln ax - x \quad (42)$$

$$\int \frac{\ln ax}{x} dx = \frac{1}{2} (\ln ax)^2 \quad (43)$$

$$\int \ln(ax+b) dx = \left(x + \frac{b}{a} \right) \ln(ax+b) - x, a \neq 0 \quad (44)$$

$$\int \ln(x^2+a^2) dx = x \ln(x^2+a^2) + 2a \tan^{-1} \frac{x}{a} - 2x \quad (45)$$

$$\int \ln(x^2-a^2) dx = x \ln(x^2-a^2) + a \ln \frac{x+a}{x-a} - 2x \quad (46)$$

$$\int \ln(ax^2+bx+c) dx = \frac{1}{a} \sqrt{4ac-b^2} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left(\frac{b}{2a} + x \right) \ln(ax^2+bx+c) \quad (47)$$

$$\int x \ln(ax+b) dx = \frac{bx}{2a} - \frac{1}{4} x^2 + \frac{1}{2} \left(x^2 - \frac{b^2}{a^2} \right) \ln(ax+b) \quad (48)$$

$$\int x \ln(a^2-b^2x^2) dx = -\frac{1}{2} x^2 + \frac{1}{2} \left(x^2 - \frac{a^2}{b^2} \right) \ln(a^2-b^2x^2) \quad (49)$$

Integrals with Exponentials

$$\int e^{ax} dx = \frac{1}{a} e^{ax} \quad (50)$$

$$\int \sqrt{x} e^{ax} dx = \frac{1}{a} \sqrt{x} e^{ax} + \frac{i\sqrt{\pi}}{2a^{3/2}} \operatorname{erf}(i\sqrt{ax}),$$

where $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ (51)

$$\int x e^x dx = (x-1)e^x \quad (52)$$

$$\int x e^{ax} dx = \left(\frac{x}{a} - \frac{1}{a^2} \right) e^{ax} \quad (53)$$

$$\int x^2 e^x dx = (x^2 - 2x + 2) e^x \quad (54)$$

$$\int x^2 e^{ax} dx = \left(\frac{x^2}{a} - \frac{2x}{a^2} + \frac{2}{a^3} \right) e^{ax} \quad (55)$$

$$\int x^3 e^x dx = (x^3 - 3x^2 + 6x - 6) e^x \quad (56)$$

$$\int x^n e^{ax} dx = \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} dx \quad (57)$$

$$\int x^n e^{ax} dx = \frac{(-1)^n}{a^{n+1}} \Gamma[1+n, -ax],$$

where $\Gamma(a, x) = \int_x^\infty t^{a-1} e^{-t} dt$ (58)

$$\int e^{ax^2} dx = -\frac{i\sqrt{\pi}}{2\sqrt{a}} \operatorname{erf}(i\sqrt{a}x) \quad (59)$$

$$\int e^{-ax^2} dx = \frac{\sqrt{\pi}}{2\sqrt{a}} \operatorname{erf}(x\sqrt{a}) \quad (60)$$

$$\int x e^{-ax^2} dx = -\frac{1}{2a} e^{-ax^2} \quad (61)$$

$$\int x^2 e^{-ax^2} dx = \frac{1}{4} \sqrt{\frac{\pi}{a^3}} \operatorname{erf}(x\sqrt{a}) - \frac{x}{2a} e^{-ax^2} \quad (62)$$

*© 2014. From <http://integral-table.com>, last revised June 14, 2014. This material is provided as is without warranty or representation about the accuracy, correctness or suitability of the material for any purpose, and is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

