

int cst = 0;
while (cst < nst) {
 System.out.println("n");
 cst++;
}

$$\begin{aligned} & \leq nst/2 - 1 \\ & \leq \\ \text{cst} & \leq nst/2 \\ & nst/4 + \\ & \text{else} \\ & nst - - \end{aligned}$$

0 1 2 3 4
1 1 1 1 1
1 2 3 2 1

1 + 1, 2
-- 3 ..

```

I
while (C) {
    // Koan
    update
}

for (I; C; U) {
    // Koan
}

```

$$0 \leq 2^n - 1$$

$$-2^{n-1} \leq 2^{n-1} - 1$$

Note 1: implicit type cast
 0 - 00011
 30 8 13
 byte bits transfers

$\begin{matrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ p_{21} & p_{12} & a_{10} & a_{01} & b_{10} & b_{01} & c_{10} & c_{01} & d_{10} & d_{01} & e_{10} \end{matrix}$
 $\begin{matrix} 6 \\ \parallel \end{matrix} \quad \begin{matrix} 1 \\ \parallel \end{matrix} \quad \begin{matrix} 1 \\ 2^1 \end{matrix} \quad \begin{matrix} 1 \\ 2^0 \end{matrix} \quad \begin{matrix} 1 \\ 2^3 \end{matrix} \quad \begin{matrix} 1 \\ 2^2 \end{matrix} \quad \begin{matrix} 1 \\ 2^3 \end{matrix}$
 boolean \rightarrow true [false]
 char \rightarrow 2 byte
 symbol

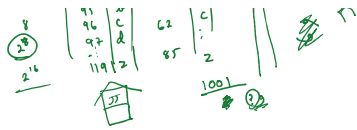
$$\begin{array}{c}
 8 \\
 \textcircled{2^8} \\
 \hline
 2^{16}
 \end{array}$$

$$\begin{array}{c}
 \text{ASCIT} \\
 \hline
 \begin{array}{cc}
 a & b \\
 94 & 95 \\
 96 & 97 \\
 98 & 99 \\
 \vdots & \vdots \\
 119 & 12
 \end{array}
 \end{array}$$

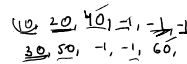
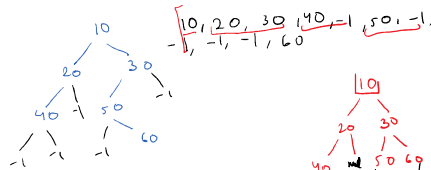
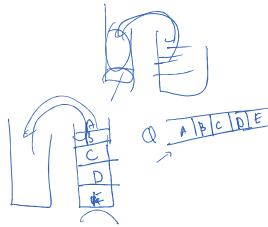
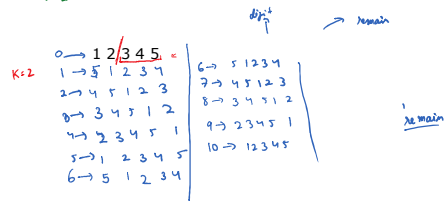
$$\begin{array}{c}
 \textcircled{08} \\
 61 \\
 62 \\
 85
 \end{array}$$

$$\begin{array}{c}
 \textcircled{A} \\
 B \\
 C \\
 \vdots \\
 Z
 \end{array}$$

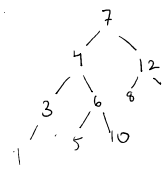
$$\begin{array}{c}
 1001 \\
 \textcircled{92}
 \end{array}$$



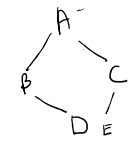
Input \rightarrow A-Z \rightarrow Upper
a-z \rightarrow Lower



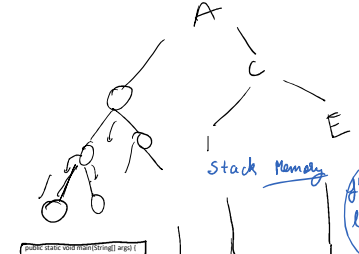
7, 4, 12, 3, 6, 8, 1, 5, 10



(A, B, C, D, E)

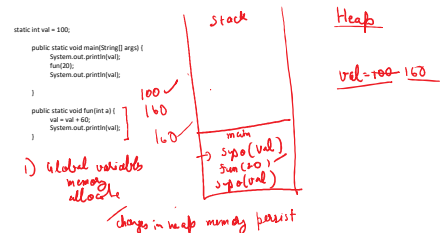
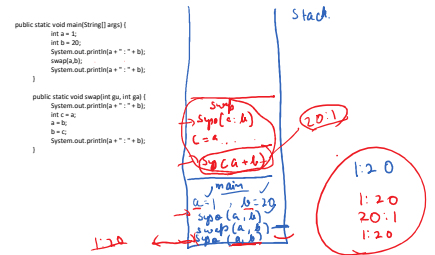
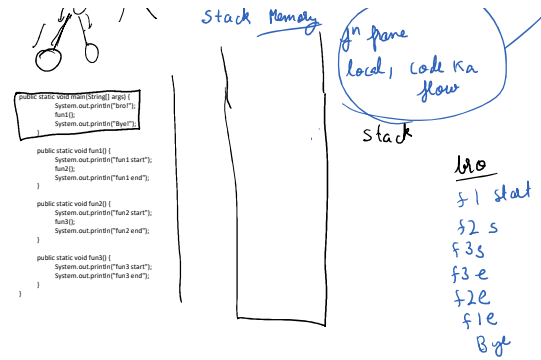


issue



Stack Memory

in frame local, code ka flow

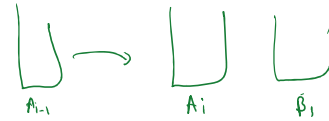


style

316
10K+20

0 → 1 and
 1 → 2 and

i → A_{i-1} Pile B_i → B₀ A₀



In P1 iteration, you start picking up the cards in A_{i-1} pile from the top one by one and check whether the number written on the card is divisible by the P1 prime number. If the number is divisible, you stack that card on pile B. Otherwise, you stack that card on pile A.

From <https://leetcode.com/problems/prime-number-divisors/>

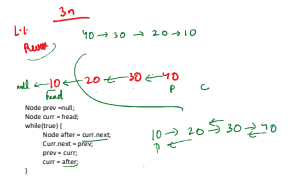
{ 2, 3, 5, 7, 11, 13, 17, 19, 23, ... }

A B C
 0 0 1 0
 0 0 1 0
 0 0 0 0
 0 0 1 0

mat [n][m] < 1
 > 0

From <https://leetcode.com/problems/prime-number-divisors/>

C



40 → 30 → 20 → 10
 10 → 20 → 30 → 40

```

private void rev2(Node prev) {
    if (prev.next == null) {
        return;
    }
    rev2(prev.next);
    Node curr = prev.next;
    curr.next = prev;
    prev = curr;
}

```

k = 3 0(n) 0(K)
 10 → 20 → 30 → 40 → 50 → 60 → 70 → 80



(out x side)
 10 → 20 → 30 → 40 →

10 → 20 → 30 → 40 →

// while (fast != null && fast.next != null) {
 while (fast.next != null && fast != null) {

10 → 20 → 30 → 40 →
 out K don't.

K = 1, 50

K = 2, 140

K = 3, 26

K = 4, 20

K = 5, 10

1 → 2 → 3

)

>

-> 90

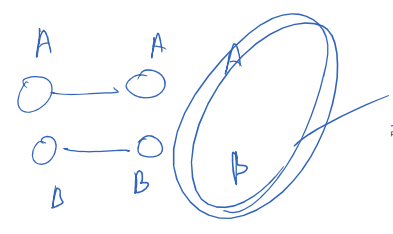
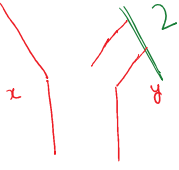
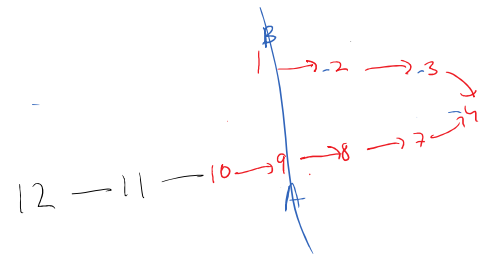
||


T_s

50

γ_j

50 A-
Hand



∂ 

$i \leq j < K$
 $\text{num}[i] < \text{num}[K] < \text{num}[j]$
 $\text{num}[j] > \text{num}[K] \rightarrow$
 $\text{ans}[j] \leftarrow K$
 next greater | greater
 next body AB
 $((((_)) ((_)))$
 $\text{if } T$
 $(\text{~~TTTT~~}) \rightarrow$
 $\text{Size} = 2 + 1 = 3 + 1 = 4 \times 2 =$

TECHNOLOGY

```

public int compare(char[] s1, char[] s2) {
    // Base Case
    if (s1 == null || s2 == null)
        return -1;
    // Base Case
    if (s1.length == 0 || s2.length == 0)
        return 0;
    // Base Case
    if (s1[0] < s2[0])
        return -1;
    if (s1[0] > s2[0])
        return 1;
    // Recursive Case
    return compare(s1.substring(1), s2.substring(1));
}

```

$\{(-, -), (-, -)\} \Rightarrow \text{jaadu}$
 \rightarrow
 $\text{for } (li : \text{jaadu}) \{$
 $\quad \text{sys}(li)$
 $\}$

yes $\left\{ \begin{array}{l} \text{arr2} = \text{arr1} \\ \text{arr2} = \text{arr1} + 10^k \\ \text{arr2} = 10^k \end{array} \right\} \quad \left| \quad \begin{array}{l} \text{A)} \quad 10 - 10 \\ \text{B)} \quad 10 - 0 \end{array} \right.$

[illegible]

Code Snippet:

```

int[] a1 = {1, 2, 3, 4};
int[] a2 = {12, 13, 14, 7};
System.out.println(a1[0] + " * " + a2[0]);
// swap(a1[0], a2[0]);
System.out.println(a1[0] + " * " + a2[0]);
}

public static void swap_arr(int[] a, int[] b) {
    int[] d = a;
    a = b;
    b = d;
}

```

Call Stack:

- main
 - swap_arr
 - swap_arr
 - swap_arr
 - swap_arr
 - swap_arr
 - swap_arr
 - swap_arr
 - swap_arr

Memory Diagram:

1	2	3	4
---	---	---	---

12	13	14	7
----	----	----	---

stack

```
int[] a1 = {1, 2, 3, 4};
int[] a2 = {2, 3, 2, 54, 7};
System.out.println(a1[0] + "" + a2[0]);
swap_in(a1, a2, 0);
System.out.println(a1[0] + "" + a2[0]);
```

public static void swap_in (int[] a, int[] b, int c) {
 int temp = a[c];
 a[c] = b[c];
 b[c] = temp;
}

1015

1	2	3	5
---	---	---	---

2015

0	2	14	2
---	---	----	---

3015

0	2	14	2
---	---	----	---

3015

0	2	14	2
---	---	----	---

int[] a1 = {1, 2, 3, 4};
int[] a2 = {2, 3, 2, 54, 7};
System.out.println(a1[0] + "" + a2[0]);
swap_in(a1, a2, 0);
System.out.println(a1[0] + "" + a2[0]);

1

a = new int[5];
a[0] = 301;
a[1] = 301;
b[0] = 0;

1

```
int[] arr = new int[10];
System.out.println(Max(arr));
}

public static int Max(int[] arr) {
    int job = Integer.MIN_VALUE;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] > job) {
            job = arr[i];
        }
    }
    return job;
}
```

[illegible]

FL
C
C
C

4

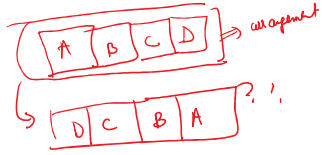
$\begin{array}{l} +5 \\ \textcircled{-1} \rightarrow 20, 30, 40, 50, 10 \\ +5 \\ \textcircled{-2} \rightarrow 30, 40, 50, 10, 20 \\ +5 \\ \textcircled{-3} \rightarrow 40, 50, 10, 20, 30 \\ +5 \\ \textcircled{-4} \rightarrow 50, 10, 20, 30, 40 \\ +5 \\ \textcircled{-5} \rightarrow 10, 20, 30, 40, 50 \end{array}$

$$\underline{-12} \div \underline{5}$$
$$\begin{aligned} \text{dividend} &= Q \cdot \text{divisor} + \text{rem} \\ -\text{dividend} &= (-Q) \cdot \text{divisor} - \text{rem} \end{aligned}$$

$\{10, 10, 20, 30, 40, 40\}$
 $\Rightarrow \{10, 20, 30, 40, 40\}$
 $\{10, 20, 30, 40\}$
 $\{10, 20, 40\}$
 $\{10, 40\}$
 $\{40\}$
 $\{10\}$

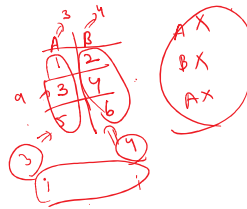
Pen

arr. { 10, 20, 30, 40, 50, ... }
 OK { 10, 20, 30, 40, 50 }
 i = 0 to n-1
 arr[0] = arr[n-1]
 arr[1] = arr[n-2]
 arr[2] = arr[n-3]
 ...
 arr[n-1] = arr[0]

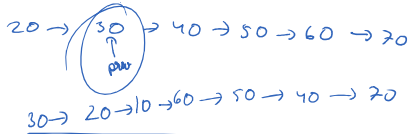


0 1 2 3 4

```
public static void func(int[] arr) {
    int n = arr.length;
    while (n > 0) {
        arr[0] = arr[n-1];
        arr[n-1] = arr[0];
        n--;
    }
}
```



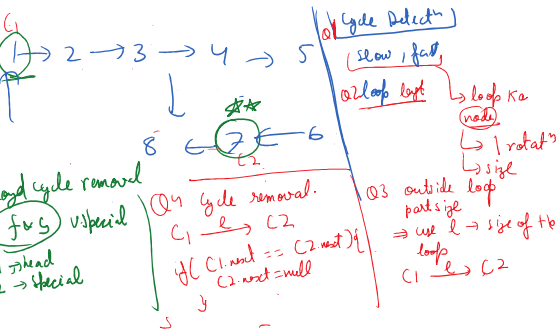
K=3



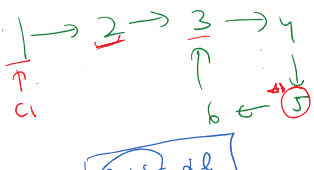
0-n → 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000



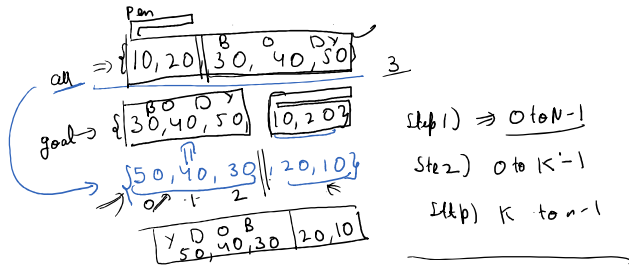
O(n) O(1)



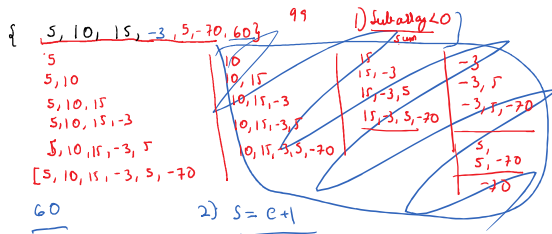
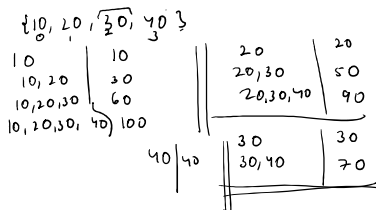
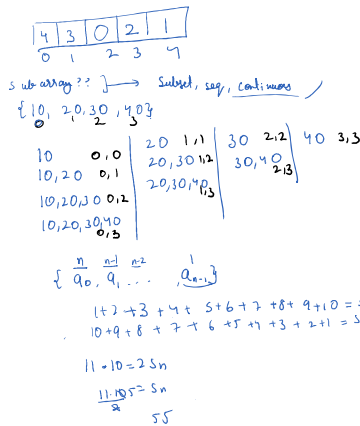
C delete = 3K



{10, 10, 20, 30, 40}



int[] arr = {2, 4, 3, 1, 0};
i 0, 1, 2, 3, 4

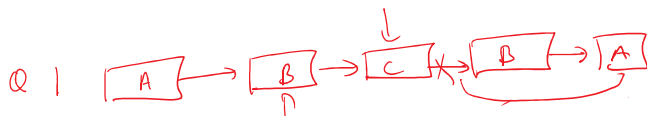


{ 5, 10, -12, 6, 60, -70, 100, -500, 2, 55, -6 }

sum = 51

max = 100

solve x \rightarrow Rain water

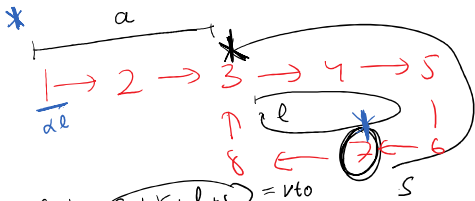


A1) $O(n)$ T $O(n)$ space

A2) $O(n)$ T $O(1)$ space mid, LL reverse

7.
C1

$$a + s = dl$$



slow
fast

$$Car 1 = a + K_1 \cdot l + s = v_{to}$$

$$Car 2 = a + K_2 \cdot l + s = 2v_{to}$$

$$a + K_2 l + s = 2(a + K_1 l + s)$$

$$(K_2 - 2K_1)l = a + s$$

$$a + K_1 \cdot l + s = 3(a + K_2 l + s)$$

$$(K_1 - 3K_2)l = 2(s + a)$$

$$\frac{(K_1 - 3K_2)l}{2} = (s + a)$$

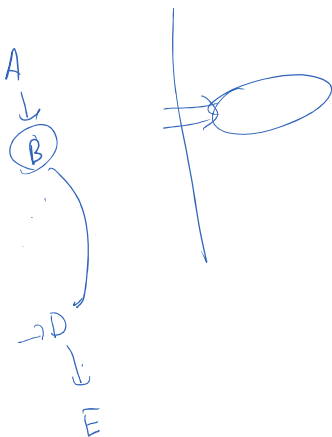
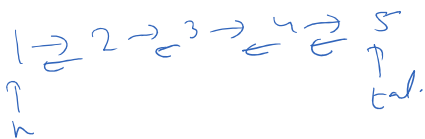
Floyd cycle removal

	Add First	Remove First	Add last	Remove last
L1, single head	$O(1)$	$O(1)$	$O(n)$	$O(n)$
L1, single head/tail	$O(1)$	$O(1)$	$O(1)$	$O(n)$
Doubly head/tail	$O(1)$	$O(1)$	$O(1)$	$O(1)$

Java

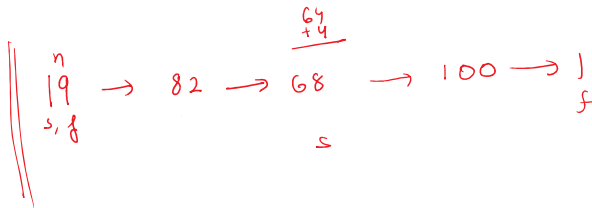
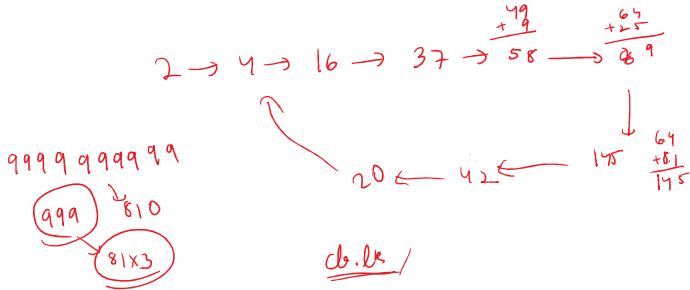
Doubly L.L.

data
next
prev

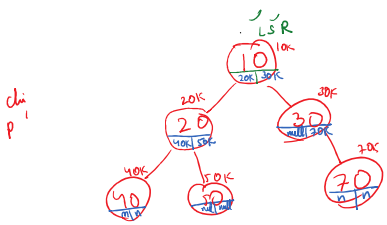


1 2 3 4 5 6 7 8
2 3 4 5 6 7 8 9
3 4 5 6 7 8 9 10
4 5 6 7 8 9 10 11
5 6 7 8 9 10 11 12
6 7 8 9 10 11 12 13
7 8 9 10 11 12 13 14
8 9 10 11 12 13 14 15
9 10 11 12 13 14 15 16
10 11 12 13 14 15 16 17
11 12 13 14 15 16 17 18
12 13 14 15 16 17 18 19
13 14 15 16 17 18 19 20
14 15 16 17 18 19 20 21
15 16 17 18 19 20 21 22
16 17 18 19 20 21 22 23
17 18 19 20 21 22 23 24
18 19 20 21 22 23 24 25
19 20 21 22 23 24 25 26
20 21 22 23 24 25 26 27
21 22 23 24 25 26 27 28
22 23 24 25 26 27 28 29
23 24 25 26 27 28 29 30
24 25 26 27 28 29 30 31
25 26 27 28 29 30 31 32
26 27 28 29 30 31 32 33
27 28 29 30 31 32 33 34
28 29 30 31 32 33 34 35
29 30 31 32 33 34 35 36
30 31 32 33 34 35 36 37
31 32 33 34 35 36 37 38
32 33 34 35 36 37 38 39
33 34 35 36 37 38 39 40
34 35 36 37 38 39 40 41
35 36 37 38 39 40 41 42
36 37 38 39 40 41 42 43
37 38 39 40 41 42 43 44
38 39 40 41 42 43 44 45
39 40 41 42 43 44 45 46
40 41 42 43 44 45 46 47
41 42 43 44 45 46 47 48
42 43 44 45 46 47 48 49
43 44 45 46 47 48 49 50
44 45 46 47 48 49 50 51
45 46 47 48 49 50 51 52
46 47 48 49 50 51 52 53
47 48 49 50 51 52 53 54
48 49 50 51 52 53 54 55
49 50 51 52 53 54 55 56
50 51 52 53 54 55 56 57
51 52 53 54 55 56 57 58
52 53 54 55 56 57 58 59
53 54 55 56 57 58 59 60
54 55 56 57 58 59 60 61
55 56 57 58 59 60 61 62
56 57 58 59 60 61 62 63
57 58 59 60 61 62 63 64
58 59 60 61 62 63 64 65
59 60 61 62 63 64 65 66
60 61 62 63 64 65 66 67
61 62 63 64 65 66 67 68
62 63 64 65 66 67 68 69
63 64 65 66 67 68 69 70
64 65 66 67 68 69 70 71
65 66 67 68 69 70 71 72
66 67 68 69 70 71 72 73
67 68 69 70 71 72 73 74
68 69 70 71 72 73 74 75
69 70 71 72 73 74 75 76
70 71 72 73 74 75 76 77
71 72 73 74 75 76 77 78
72 73 74 75 76 77 78 79
73 74 75 76 77 78 79 80
74 75 76 77 78 79 80 81
75 76 77 78 79 80 81 82
76 77 78 79 80 81 82 83
77 78 79 80 81 82 83 84
78 79 80 81 82 83 84 85
79 80 81 82 83 84 85 86
80 81 82 83 84 85 86 87
81 82 83 84 85 86 87 88
82 83 84 85 86 87 88 89
83 84 85 86 87 88 89 90
84 85 86 87 88 89 90 91
85 86 87 88 89 90 91 92
86 87 88 89 90 91 92 93
87 88 89 90 91 92 93 94
88 89 90 91 92 93 94 95
89 90 91 92 93 94 95 96
90 91 92 93 94 95 96 97
91 92 93 94 95 96 97 98
92 93 94 95 96 97 98 99
93 94 95 96 97 98 99 100

A1) $O(n) + O(n)$ space
 A2) $O(n) + O(1)$ space mid, LL reverse



→ cruce
 → Intra



Print (10K) f
 Print (20K)
 Print (30)
 Spac (10K data)

App?

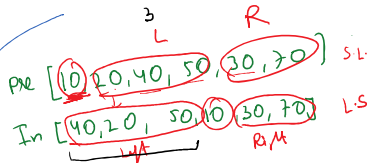
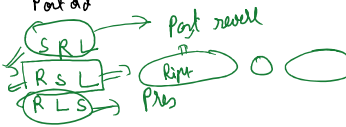
```

public void print(Node nn) {
    print(nn.left);
    print(nn.right);
    System.out.println(nn.data);
}

```

L: subtree R: subtree
 Post → 40, 50, 20, 70, 30, 10
 In → 40, 20, 50, 10, 30, 70
 Pre → 10, 20, 40, 50, 30, 70

Pre order : S-L-R
 In order : L-S-R
 Post order : L-R-S

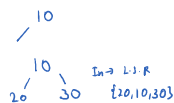


- i) $(ps+1, ps+5)$
 - e) $(ps+1, pe)$



In(0, 0-1)

In(0+1, 0)



```

private Node create(int[] pre, int pre_s, int pre_e, int[] in, int in_s, int in_e) {
    if (pre_s > pre_e) return null;
    return null;
}

```

	0	1
0 3 -1 3 5 6 7	3	
1 0 -1 -3 5 6 7	3	
1 3 0 -1 5 6 7	5	
1 3 0 1 5 6 7	5	
1 3 -1 3 5 6 7	6	
1 3 -1 3 5 6 7	7	

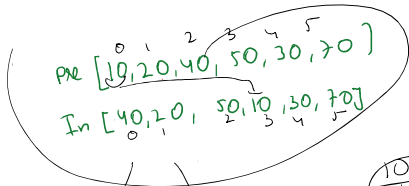
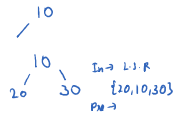
From <https://leetcode.com/problems/minimum-absolute-difference-in-pairs/>



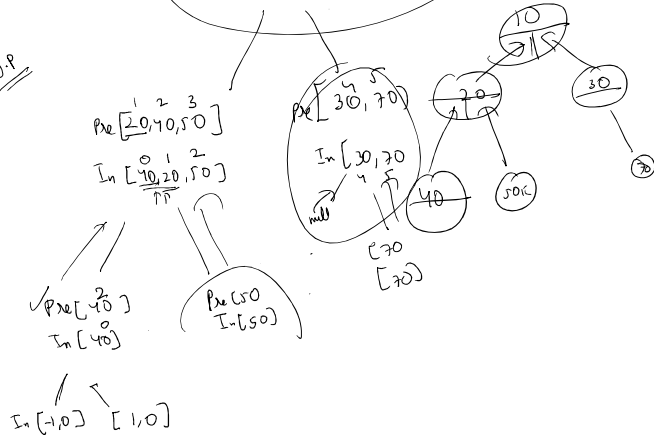
```

private Node create(int[] pre, int ps, int pe, int[] in, int is, int ie) {
    if (ps > pe || is > ie)
        return null;
    System.out.println(ps + " " + pe + " " + is + " " + ie);
    Node nn = new Node(pre[ps]);
    int found = -1;
    int left_size = 0;
    for (int i = is; i <= ie; i++) {
        if (pre[ps] == in[i]) {
            found = i;
            break;
        }
    }
    left_size = found - is + 1;
    nn.left = create(pre, ps + 1, ps + left_size, in, is, found - 1);
    nn.right = create(pre, ps + left_size + 1, pe, in, found + 1, ie);
    return nn;
}

```



D.P



```

private int Dia(Node nn) {
    if (nn == null) {
        return 0;
    }
    // TODO Auto-generated method stub
    int left = Dia(nn.left);
    int right = Dia(nn.right);
    int self = Math.max(left, right) + 1;
    return self;
}

private int Ht(Node nn) {
    if (nn == null) {
        return -1;
    }
    // TODO Auto-generated method stub
    int L = Ht(nn.left);
    int R = Ht(nn.right);
    return Math.max(L, R) + 1;
}

```

$$f(n) = 2f(n/2) + 2^n$$

$$f(n) = 2f(n/2) + 1$$

$$= 1 + f(n/2) + 1$$

$$f(n) = 4f(n/2) + 1$$

$$4(f(n/2)) = (4f(n/2) + 1) \cdot 4$$

$$4^2 f(n/2^2) = (4^2 f(n/2^2) + 1) \cdot 4^2$$

$$4^K f(n/2^K) = (4^K f(n/2^K) + 1) \cdot 4^K$$

$$f(n) = 1 + 4 + \dots + 4^K$$

$$f(n) = 4^K = 2^{2K} = 2^{2 \log_2 n} = n^2$$

$$2^K = n$$

$$K = \log_2 n$$

$$f(n) = 1 + 2 + 2^2 + \dots + 2^{K+1}$$

$$= 2^{K+2} - 2$$

$$2^{2 \log_2 n} = n^2$$

$$2^K = n$$

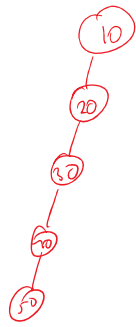
$$K = \log_2 n$$

$$\frac{2^{2 \log_2 n}}{2} = 4^{\log_2 n} = n^2$$

10

$$K = \log_2 n$$





$$K = \frac{1}{2}$$

$$\underline{O(n)}$$

$$2^{n^2} = (n)$$

$$\frac{2^{n^2}}{2} = n^{n^2}$$

