

Longest Nice Substring:

1763. Longest Nice Substring Solved

A string *s* is **nice** if every letter of the alphabet that *s* contains, it appears **both** in uppercase and lowercase. For example, "aAaBb" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "aAa" is not because 'B' appears, but 'b' does not.

Given a string *s*, return the **longest substring** of *s* that is **nice**. If there are multiple, return the **substring of the earliest occurrence**. If there are none, return an empty string.

Example 1:
Input: *s* = "aZaAa"
Output: "aAa"
Explanation: "aAa" is a nice string because 'A/a' is the only letter of the alphabet in *s*, and both 'A' and 'a' appear. "aAa" is the longest nice substring.

Example 2:
Input: *s* = "Bb"
Output: "Bb"
Explanation: "Bb" is a nice string because both 'B' and 'b' appear. The whole string is a substring.

Example 3:
Input: *s* = "c"
Output: ""
Explanation: There is no substring that is nice.

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if (s.length() < 2) return "";
        unordered_set<char> charSet(s.begin(), s.end());
        for (int i = 0; i < s.length(); i++) {
            if (charSet.count(s[i]) == 0 || charSet.count(trupper(s[i]) == 0) {
                string left = longestNiceSubstring(s.substr(0, i));
                string right = longestNiceSubstring(s.substr(i + 1));
                return left.length() > right.length() ? left : right;
            }
        }
        return s;
    }
};
```

Reverse Bits:

190. Reverse Bits Solved

Reverse bits of a given 32 bits unsigned integer.

Note:

- Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using **2's complement notation**, therefore, in **Example 2** above, the input represents the signed integer -43261596, and the output represents the signed integer -1874556429.

Example 1:
Input: *n* = 00000001000000000000000000000000
Output: 964176192 (00111001011100000100000100000000)
Explanation: The input binary string 00000001000000000000000000000000 represents the unsigned integer 43261596, so return 964176192, which its binary representation is 00111001011100000100000100000000.

Example 2:
Input: *n* = 11111111111111111111111111111111
Output: 3221225471 (10111111111111111111111111111111)
Explanation: The input binary string 11111111111111111111111111111111 represents the unsigned integer 4294967295, so return 3221225471, which its binary representation is 10111111111111111111111111111111.

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result |= ((n >> i) & 1) << (31 - i);
        }
        return result;
    }
};
```

Number of 1 Bits:

191. Number of 1 Bits Solved

Given a positive integer *n*, write a function that returns the number of **set bits** in its binary representation (also known as the **Hamming weight**).

Example 1:
Input: *n* = 11
Output: 3
Explanation: The input binary string 1011 has a total of three set bits.

Example 2:
Input: *n* = 128
Output: 1
Explanation: The input binary string 10000000 has a total of one set bit.

Example 3:
Input: *n* = 4294967295
Output: 32
Explanation: The input binary string 11111111111111111111111111111111 has a total of 32 set bits.

```
class Solution {
public:
    int hammingWeight(int n) {
        int count = 0;
        while (n) {
            n &= (n - 1); // clears the least significant set bit
            count++;
        }
        return count;
    }
};
```

Maximum Subarray:

53. Maximum Subarray

Given an integer array `nums`, find the **subarray** with the largest sum, and return its sum.

Example 1:
Input: `nums = [-2,1,3,4,-1,2,1,5,4]`
Output: 6
Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

Example 2:
Input: `nums = [1]`
Output: 1
Explanation: The subarray `[1]` has the largest sum 1.

Example 3:
Input: `nums = [5,4,-1,7,8]`
Output: 23
Explanation: The subarray `[5,4,-1,7,8]` has the largest sum 23.

Constraints:

- `1 <= nums.length <= 105`
- `-104 <= nums[i] <= 104`

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0];
        int currentSum = nums[0];
        for (int i = 1; i < nums.size(); i++) {
            currentSum = max(nums[i], currentSum + nums[i]);
            maxSum = max(maxSum, currentSum);
        }
        return maxSum;
    }
};
```

Search a 2D Matrix II:

240. Search a 2D Matrix II

Write an efficient algorithm that searches for a value `target` in an $n \times n$ integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    if (matrix.empty() || matrix[0].empty()) return false;
    int row = matrix.size(), col = matrix[0].size();
    int r = 0, c = col - 1;
    while (row < row && col >= 0) {
        if (matrix[r][c] == target) return true;
        else if (matrix[r][c] > target) col--;
        else row++;
    }
    return false;
}
```

Super Pow:

Accepted 11/17 testcases passed

228C511879 submitted at Mar 18, 2025 11:30

Runtime 0 ms | **Beats** 100.00% | **Memory** 15.34 MB | **Beats** 15.09%

Analysis Complete

Time	Memory
0ms	15.34 MB
0ms	15.34 MB
0ms	15.34 MB
0ms	15.34 MB
0ms	15.34 MB
0ms	15.34 MB
0ms	15.34 MB

```
class Solution {
public:
    const int MOD = 1337;
    int modPow(int base, int exp) {
        int result = 1;
        base %= MOD;
        while (exp > 0) {
            if (exp & 1) result = (result * base) % MOD;
            base = (base * base) % MOD;
            exp /= 2;
        }
        return result;
    }
    int superPow(vector<int>& a, int b) {
        int result = 1;
        for (int digit : a) {
            result = (result * modPow(digit, b)) % MOD;
        }
        return result;
    }
};
```

Skyline Problem:

The screenshot shows the LeetCode interface for the Skyline Problem. The submission is accepted, with a runtime of 8 ms and memory usage of 27.84 MB. A histogram shows the distribution of runtime results, with a peak around 10ms. The code is written in C++ and uses a sweep line algorithm to calculate the skyline from a set of buildings.

Accepted 44 / 44 testcases passed
228CS11879 submitted at Mar 18, 2025 11:55

Runtime: 8 ms | Beats: 94.50% | Memory: 27.84 MB | Beats: 63.38%

Code (C++)

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;

        for (auto& b : buildings) {
            events.push_back({b[0], -b[2]});
            events.push_back({b[1], b[2]});
        }

        sort(events.begin(), events.end(), [](pair<int, int> a, pair<int, int> b) {
            if (a.first == b.first) return a.second < b.second;
            return a.first < b.first;
        });

        multiset<int> heights = {0};
        vector<vector<int>> result;
        int prevMax = 0;

        for (auto& e : events) {
            if (e.second < 0) heights.erase(heights.find(e.second));
            else heights.insert(e.second);
            int curMax = *heights.rbegin();
            if (curMax > prevMax) {
                result.push_back({e.first, curMax});
                prevMax = curMax;
            }
        }

        return result;
    }
};
```

Test Result: Accepted Runtime: 0 ms

Input: buildings = [[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]

Beautiful Array:

The screenshot shows the LeetCode interface for the Beautiful Array problem. The submission is accepted, with a runtime of 1 ms and memory usage of 9.93 MB. A histogram shows the distribution of runtime results, with a peak around 2ms. The code is written in C++ and uses a recursive approach to generate a beautiful array.

Accepted 38 / 38 testcases passed
228CS11879 submitted at Mar 18, 2025 11:52

Runtime: 1 ms | Beats: 52.28% | Memory: 9.93 MB | Beats: 75.64%

Code (C++)

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1};
        for (int num : result) {
            if (2 * num - 1 < n) temp.push_back(2 * num - 1);
        }
        for (int num : result) {
            if (2 * num <= n) temp.push_back(2 * num);
        }
        result = temp;
        return result;
    }
};
```

Test Result: Accepted Runtime: 0 ms

Input: n = 4

Reverse Pairs:

Reverse Pairs - LeetCode

Accepted 140 / 140 testcases passed
228CS11879 submitted at Mar 18, 2025 11:57

Runtime: 551 ms, Beat: 26.11%
Memory: 240.21 MB, Beat: 59.79%

Code (C++)

```
class Solution {
public:
    int mergeSortAndCount(vector<int>& nums, int left, int right) {
        if (left >= right) return 0;

        int mid = left + (right - left) / 2;
        int count = mergeSortAndCount(nums, left, mid) + mergeSortAndCount(nums, mid + 1, right);

        int j = mid + 1;
        for (int i = left; i <= mid; i++) {
            while (j <= right && (long)nums[i] > 2LL * nums[j]) j++;
            count += (j - (mid + 1));
        }

        vector<int> temp;
        int l = left, k = mid + 1;
        while (l <= mid && k <= right) {
```

Test Result: Accepted, Runtime: 0 ms, Case 1

Input: nums = [1,3,2,3,1]

Longest Increasing Subsequence II:

Longest Increasing Subsequence II - LeetCode

Time Limit Exceeded 80 / 84 testcases passed

Last Executed Input

nums = [50001,50002,50003,50004,50005,50006,50007,50008,50009,50010,50011,50012,50013,50014,50015,50016,50017,50018,50019,50020,50021,50022,50023,50024,50025,50026,50027,50028,50029,50030,50031,50032,50033,50034,50035,50036,50037,50038,50039,50040,50041,50042,50043,50044,50045,50046,50047,50048,50049,50050,50051,50052,50053,50054,50055,50056,50057,50058,50059,50060,50061,50062,50063,50064,50065,50066,50067,50068,50069,50070,50071,50072,50073,50074,50075,50076,50077,50078,50079,50080,50081,50082,50083,50084,50085,50086,50087,50088,50089,50090,50091,50092,50093,50094,50095,50096,50097,50098,50099,50100,50101,50102,50103,50104,50105,50106,50107,50108,50109,50110,50111,50112,50113,50114,50115,50116,50117,50118,50119,50120,50121,50122,50123,50124,50125,50126,50127,50128,50129,50130,50131,50132,50133,50134,50135,50136,50137,50138,50139,50140,50141,50142,50143,50144,50145,50146,50147,50148,50149,50150,50151,50152,50153,50154,50155,50156,50157,50158,50159,50160,50161,50162,50163,50164,50165,50166,50167,50168,50169,50170,50171,50172,50173,50174,50175,50176,50177,50178,50179,50180,50181,50182,50183,50184,50185,50186,50187,50188,50189,50190,50191,50192,50193,50194,50195,50196,50197,50198,50199,50200,50201,50202,50203,50204,50205,50206,50207,50208,50209,50210,50211,50212,50213,50214,50215,50216,50217,50218,50219,50220,50221,50222,50223,50224,50225,50226,50227,50228,50229,50230,50231,50232,50233,50234,50235,50236,50237,50238,50239,50240,50241,50242,50243,50244,50245,50246,50247,50248,50249,50250,50251,50252,50253,50254,50255,50256,50257,50258,50259,50260,50261,50262,50263,50264,50265,50266,50267,50268,50269,50270,50271,50272,50273,50274,50275,50276,50277,50278,50279,50280,50281,50282,50283,50284,50285,50286,50287,50288,50289,50290,50291,50292,50293,50294,50295,50296,50297,50298,50299,50300,50301,50302,50303,50304,50305,50306,50307,50308,50309,50310,50311,50312,50313,50314,50315,50316,50317,50318,50319,50320,50321,50322,50323,50324,50325,50326,50327,50328,50329,50330,50331,50332,50333,50334,50335,50336,50337,50338,50339,50340,50341,50342,50343,50344,50345,50346,50347,50348,50349,50350,50351,50352,50353,50354,50355,50356,50357,50358,50359,50360,50361,50362,50363,50364,50365,50366,50367,50368,50369,50370,50371,50372,50373,50374,50375,50376,50377,50378,50379,50380,50381,50382,50383,50384,50385,50386,50387,50388,50389,50390,50391,50392,50393,50394,50395,50396,50397,50398,50399,50400,50401,50402,50403,50404,50405,50406,50407,50408,50409,50410,50411,50412,50413,50414,50415,50416,50417,50418,50419,50420,50421,50422,50423,50424,50425,50426,50427,50428,50429,50430,50431,50432,50433,50434,50435,50436,50437,50438,50439,50440,50441,50442,50443,50444,50445,50446,50447,50448,50449,50450,50451,50452,50453,50454,50455,50456,50457,50458,50459,50460,50461,50462,50463,50464,50465,50466,50467,50468,50469,50470,50471,50472,50473,50474,50475,50476,50477,50478,50479,50480,50481,50482,50483,50484,50485,50486,50487,50488,50489,50490,50491,50492,50493,50494,50495,50496,50497,50498,50499,50500,50501,50502,50503,50504,50505,50506,50507,50508,50509,50510,50511,50512,50513,50514,50515,50516,50517,50518,50519,50520,50521,50522,50523,50524,50525,50526,50527,50528,50529,50530,50531,50532,50533,50534,50535,50536,50537,50538,50539,50540,50541,50542,50543,50544,50545,50546,50547,50548,50549,50550,50551,50552,50553,50554,50555,50556,50557,50558,50559,50560,50561,50562,50563,50564,50565,50566,50567,50568,50569,50570,50571,50572,50573,50574,50575,50576,50577,50578,50579,50580,50581,50582,50583,50584,50585,50586,50587,50588,50589,50590,50591,50592,50593,50594,50595,50596,50597,50598,50599,50600,50601,50602,50603,50604,50605,50606,50607,50608,50609,50610,50611,50612,50613,50614,50615,50616,50617,50618,50619,50620,50621,50622,50623,50624,50625,50626,50627,50628,50629,50630,50631,50632,50633,50634,50635,50636,50637,50638,50639,50640,50641,50642,50643,50644,50645,50646,50647,50648,50649,50650,50651,50652,50653,50654,50655,50656,50657,50658,50659,50660,50661,50662,50663,50664,50665,50666,50667,50668,50669,50670,50671,50672,50673,50674,50675,50676,50677,50678,50679,50680,50681,50682,50683,50684,50685,50686,50687,50688,50689,50690,50691,50692,50693,50694,50695,50696,50697,50698,50699,50700,50701,50702,50703,50704,50705,50706,50707,50708,50709,50710,50711,50712,50713,50714,50715,50716,50717,50718,50719,50720,50721,50722,50723,50724,50725,50726,50727,50728,50729,50730,50731,50732,50733,50734,50735,50736,50737,50738,50739,50740,50741,50742,50743,50744,50745,50746,50747,50748,50749,50750,50751,50752,50753,50754,50755,50756,50757,50758,50759,50760,50761,50762,50763,50764,50765,50766,50767,50768,50769,50770,50771,50772,50773,50774,50775,50776,50777,50778,50779,50780,50781,50782,50783,50784,50785,50786,50787,50788,50789,50790,50791,50792,50793,50794,50795,50796,50797,50798,50799,50800,50801,50802,50803,50804,50805,50806,50807,50808,50809,50810,50811,50812,50813,50814,50815,50816,50817,50818,50819,50820,50821,50822,50823,50824,50825,50826,50827,50828,50829,50830,50831,50832,50833,50834,50835,50836,50837,50838,50839,50840,50841,50842,50843,50844,50845,50846,50847,50848,50849,50850,50851,50852,50853,50854,50855,50856,50857,50858,50859,50860,50861,50862,50863,50864,50865,50866,50867,50868,50869,50870,50871,50872,50873,50874,50875,50876,50877,50878,50879,50880,50881,50882,50883,50884,50885,50886,50887,50888,50889,50890,50891,50892,50893,50894,50895,50896,50897,50898,50899,50900,50901,50902,50903,50904,50905,50906,50907,50908,50909,50910,50911,50912,50913,50914,50915,50916,50917,50918,50919,50920,50921,50922,50923,50924,50925,50926,50927,50928,50929,50930,50931,50932,50933,50934,50935,50936,50937,50938,50939,50940,50941,50942,50943,50944,50945,50946,50947,50948,50949,50950,50951,50952,50953,50954,50955,50956,50957,50958,50959,50960,50961,50962,50963,50964,50965,50966,50967,50968,50969,50970,50971,50972,50973,50974,50975,50976,50977,50978,50979,50980,50981,50982,50983,50984,50985,50986,50987,50988,50989,50990,50991,50992,50993,50994,50995,50996,50997,50998,50999,51000,51001,51002,51003,51004,51005,51006,51007,51008,51009,51010,51011,51012,51013,51014,51015,51016,51017,51018,51019,51020,51021,51022,51023,51024,51025,51026,51027,51028,51029,51030,51031,51032,51033,51034,51035,51036,51037,51038,51039,51040,51041,51042,51043,51044,51045,51046,51047,51048,51049,51050,51051,51052,51053,51054,51055,51056,51057,51058,51059,51060,51061,51062,51063,51064,51065,51066,51067,51068,51069,51070,51071,51072,51073,51074,51075,51076,51077,51078,51079,51080,51081,51082,51083,51084,51085,51086,51087,51088,51089,51090,51091,51092,51093,51094,51095,51096,51097,51098,51099,51100,51101,51102,51103,51104,51105,51106,51107,51108,51109,51110,51111,51112,51113,51114,51115,51116,51117,51118,51119,51120,51121,51122,51123,51124,51125,51126,51127,51128,51129,51130,51131,51132,51133,51134,51135,51136,51137,51138,51139,51140,51141,51142,51143,51144,51145,51146,51147,51148,51149,51150,51151,51152,51153,51154,51155,51156,51157,51158,51159,51160,51161,51162,51163,51164,51165,51166,51167,51168,51169,51170,51171,51172,51173,51174,51175,51176,51177,51178,51179,51180,51181,51182,51183,51184,51185,51186,51187,51188,51189,51190,51191,51192,51193,51194,51195,51196,51197,51198,51199,51200,51201,51202,51203,51204,51205,51206,51207,51208,51209,51210,51211,51212,51213,51214,51215,51216,51217,51218,51219,51220,51221,51222,51223,51224,51225,51226,51227,51228,51229,51230,51231,51232,51233,51234,51235,51236,51237,51238,51239,51240,51241,51242,51243,51244,51245,51246,51247,51248,51249,51250,51251,51252,51253,51254,51255,51256,51257,51258,51259,51260,51261,51262,51263,51264,51265,51266,51267,51268,51269,51270,51271,51272,51273,51274,51275,51276,51277,51278,51279,51280,51281,51282,51283,51284,51285,51286,51287,51288,51289,51290,51291,51292,51293,51294,51295,51296,51297,51298,51299,51300,51301,51302,51303,51304,51305,51306,51307,51308,51309,51310,51311,51312,51313,51314,51315,51316,51317,51318,51319,51320,51321,51322,51323,51324,51325,51326,51327,51328,51329,51330,51331,51332,51333,51334,51335,51336,51337,51338,51339,51340,51341,51342,51343,51344,51345,51346,51347,51348,51349,51350,51351,51352,51353,51354,51355,51356,51357,51358,51359,51360,51361,51362,51363,51364,51365,51366,51367,51368,51369,51370,51371,51372,51373,51374,51375,51376,51377,51378,51379,51380,51381,51382,51383,51384,51385,51386,51387,51388,51389,51390,51391,51392,51393,51394,51395,51396,51397,51398,51399,51400,51401,51402,51403,51404,51405,51406,51407,51408,51409,51410,51411,51412,51413,51414,51415,51416,51417,51418,51419,51420,51421,51422,51423,51424,51425,51426,51427,51428,51429,51430,51431,51432,51433,51434,51435,51436,51437,51438,51439,51440,51441,51442,51443,51444,51445,51446,51447,51448,51449,51450,51451,51452,51453,51454,51455,51456,51457,51458,51459,51460,51461,51462,51463,51464,51465,51466,51467,51468,51469,51470,51471,51472,51473,51474,51475,51476,51477,51478,51479,51480,51481,51482,51483,51484,51485,51486,51487,51488,51489,51490,51491,51492,51493,51494,51495,51496,51497,51498,51499,51500,51501,51502,51503,51504,51505,51506,51507,51508,51509,51510,51511,51512,51513,51514,51515,51516,51517,51518,51519,51520,51521,51522,51523,51524,51525,51526,51527,51528,51529,51530,51531,51532,51533,51534,51535,51536,51537,51538,51539,51540,51541,51542,51543,51544,51545,51546,51547,51548,51549,51550,51551,51552,51553,51554,51555,51556,51557,51558,51559,51560,51561,51562,51563,51564,51565,51566,51567,51568,51569,51570,51571,51572,51573,51574,51575,51576,51577,51578,51579,51580,51581,51582,51583,51584,51585,51586,51587,51588,51589,51590,51591,51592,51593,51594,51595,51596,51597,51598,51599,51600,51601,51602,51603,51604,51605,51606,51607,51608,51609,51610,51611,51612,51613,51614,51615,51616,51617,51618,51619,51620,51621,51622,51623,51624,51625,51626,51627,51628,51629,51630,51631,51632,51633,51634,51635,51636,51637,51638,51639,51640,51641,51642,51643,51644,51645,51646,51647,51648,51649,51650,51651,51652,51653,51654,51655,51656,51657,51658,51659,51660,51661,51662,51663,51664,51665,51666,51667,51668,51669,51670,51671,51672,51673,51674,51675,51676,51677,51678,51679,51680,51681,51682,51683,51684,51685,51686,51687,51688,51689,51690,51691,51692,51693,51694,51695,51696,51697,51698,51699,51700,51701,51702,51703,51704,51705,51706,51707,51708,51709,51710,51711,51712,51713,51714,51715,51716,51717,51718,51719,51720,51721,51722,51723,51724,51725,51726,51727,51728,51729,51730,51731,51732,51733,51734,51735,51736,51737,51738,51739,51740,51741,51742,51743,51744,51745,51746,51747,51748,51749,51750,51751,51752,51753,51754,51755,51756,51757,51758,51759,51760,51761,51762,51763,51764,51765,51766,51767,51768,51769,51770,51771,51772,51773,51774,51775,51776,51777,51778,51779,51780,51781,51782,51783,51784,51785,51786,51787,51788,51789,51790,51791,51792,51793,51794,51795,51796,51797,51798,51799,51800,51801,51802,51803,51804,51805,51806,51807,51808,51809,51810,51811,51812,51813,51814,51815,51816,51817,51818,51819,51820,51821,51822,51823,51824,51825,51826,51827,51828,51829,51830,51831,51832,51833,51834,51835,51836,51837,51838,51839,51840,51841,51842,51843,51844,51845,51846,51847,51848,51849,51850,51851,51852,51853,51854,51855,51856,51857,51858,51859,51860,51861,51862,51863,51864,51865,51866,51867,51868,51869,51870,51871,51872,51873,51874,51875,51876,51877,51878,51879,51880,51881,51882,51883,51884,51885,51886,51887,51888,51889,51890,51891,51892,51893,51894,51895,51896,51897,51898,51899,51900,51901,51902,51903,51904,51905,51906,51907,51908,51909,51910,51911,51912,51913,51914,51915,51916,51917,51918,51919,51920,51921,51922,51923,51924,51925,51926,51927,51928,51929,51930,51931,51932,51933,51934,51935,51936,51937,51938,51939,51940,51941,51942,51943,51944,51945,51946,51947,51948,51949,51950,51951,51952,51