Massive Algorithms

Data Structures and Algorithm Interview Questions from the internet. Help programmers to get hired by dream companies like Google or Facebook.

Newer Post Slider(Newer 20) Home Random Post Slider(Random 20) Slider(Older 20) Older Post

LeetCode 287 - Perfect Squares

Perfect Squares [LeetCode] | Training dragons the hard way - Programming Every Day!

Given a positive integer *n*, find the least number of perfect square numbers (for example, `1, 4, 9, 16, ...`) which sum to *n*.

For example, given *n* = `12`, return `3` because `12 = 4 + 4 + 4`; given *n* = `13`, return `2` because `13 = 4 + 9`.

http://www.zrzahid.com/least-number-of-perfect-squares-that-sums-to-n/

maximum perfect square less then n will be √n. So, we can check for each numbers in j=1 to √n whether we can break n into two parts such that one part is a perfect square j*j and the remaining part n-j*j can be broken into perfect squares in similar manner. Clearly it has a recurrence relation `ps(n)=j*j+ps(n-j*j), for all possible 1≤j≤√n`. We need to find such j that minimizes number of perfect squares generated.

```
Let, PSN(i) is minimum number of perfect squares that sum to i
PSN(i) = min{1+PSN(i-j*j)}, for all j, 1≤j≤√n
```

O(nlogn) DP solution
https://discuss.leetcode.com/topic/26400/an-easy-understanding-dp-solution-in-java/

```
public static int perfectSquareDP(int n){
 if(n <= 0){
  return 0;
 }

 int[] dp = new int[n+1];
 Arrays.fill(dp, Integer.MAX_VALUE);
 dp[0] = 0;
 dp[1] = 1;

 //to compute least perfect for n we compute top down for each
 //possible value sum from 2 to n
 for(int i = 2; i<=n; i++){
  //for a particular value i we can break it as sum of a perfect square j*j and
  //all perfect squares from solution of the remainder (i-j*j)
  for(int j = 1; j*j<=i; j++){
   dp[i] = Math.min(dp[i], 1+dp[i-j*j]);
  }
 }

 return dp[n];
}
```

Fastest Solution – Lagrange's four-square theorem

Popular Posts

LeetCode 425 - Word Squares
LeetCode 415 - Add Strings
LeetCode 438 - Find All Anagrams in a String
LeetCode 418 - Sentence Screen Fitting
Dynamic Programming - Subset Sum Problem
LeetCode 388 - Longest Absolute File Path
LeetCode 445 - Add Two Numbers II
LeetCode 448 - Find All Numbers Disappeared in an Array
First pair non matching leaves
HackerRank: Equal

Labels

GeeksforGeeks (1020)
Algorithm (811)
LeetCode (696)
to-do (601)
Review (412)
Classic Algorithm (334)
Classic Interview (298)
Dynamic Programming (269)
LeetCode - Review (248)
Google Interview (234)

Blog Archive

► 2016 (1011)
▼ 2015 (1762)
► December (135)
► November (321)
► October (185)
▼ September (111)
编程之美 2013 全国挑战赛 资格赛 题目二 长方形 - CYJB - 博客园
[CTCI] 最小调整有序 - Eason Liu - 博客园
LeetCode 287 - Find the Duplicate Number - Google ...
Rearrange a given linked list in-place. - Geeksfor...
San Fransisco Onsite面经 - ---☆心随风飞 - 博客频道 - CSDN.NET
Airbnb电面面经 - ---☆心随风飞 - 博客频道 - CSDN.NET
Snapchat Interview
Phone number to Letters Dropbox
Json Encoding
LeetCode 286 - Walls and Gates
LIKE CODING: MJ [40] Count squares
每日一贴: http://www.mitbbs.com/mitbbs_article_t.php?b...
Check if two given strings are isomorphic to each ...
Print all non-increasing sequences of sum equal to...
Suffix Array
Programming Pearls

Fastest Solution – Lagrange's four-square theorem

```
private static boolean is_square(int n){
    int sqrt_n = (int)(Math.sqrt(n));
    return (sqrt_n*sqrt_n == n);
}

// Based on Lagrange's Four Square theorem, there
// are only 4 possible results: 1, 2, 3, 4.
public static int perfectSquaresLagrange(int n)
{
    // If n is a perfect square, return 1.
    if(is_square(n))
    {
        return 1;
    }

    // The result is 4 if n can be written in the
    // form of 4^k*(8*m + 7).
    while ((n & 3) == 0) // n%4 == 0
    {
        n >>= 2;
    }
    if ((n & 7) == 7) // n%8 == 7
    {
        return 4;
    }

    // Check whether 2 is the result.
    int sqrt_n = (int)(Math.sqrt(n));
    for(int i = 1; i <= sqrt_n; i++)
    {
        if (is_square(n - i*i))
        {
            return 2;
        }
    }

    return 3;
}
```

X 动态规划（**Dynamic Programming**） **-时间复杂度都是O(n3/2)**

http://bookshadow.com/weblog/2015/09/09/leetcode-perfect-squares/

时间复杂度：**O(n * sqrt n)**

初始化将dp数组置为无穷大；令dp[y * y] = 1，其中：y * y <= n

状态转移方程：

```
dp[x + y * y] = min(dp[x + y * y], dp[x] + 1)
```

其中：x + y * y <= n

```
    public int numSquares(int n) {
        int dp[] = new int[n + 1];
        Arrays.fill(dp, Integer.MAX_VALUE);
        for (int i = 1; i * i <= n; i++) {
```

```
            dp[i * i] = 1;
        }
        for (int i = 1; i <= n; i++) {
            for (int j = 1; i + j * j <= n; j++) {
                dp[i + j * j] = Math.min(dp[i] + 1, dp[i + j * j]);
            }
        }
        return dp[n];
    }
```

```
int numSquares(int n) {
      static vector<int> dp {0};
      int m = dp.size();
      dp.resize(max(m, n+1), INT_MAX);
      for (int i=1, i2; (i2 = i*i)<=n; ++i)
        for (int j=max(m, i2); j<=n; ++j)
          if (dp[j] > dp[j-i2] + 1) dp[j] = dp[j-i2] + 1;
      return dp[n];
}
```

http://buttercola.blogspot.com/2015/09/leetcode-perfect-squares.html

```
      public int numSquares(int n) {
          if (n <= 0) {
              return 0;
          }

          int[] dp = new int[n + 1];

          for (int i = 1; i <= n; i++) {
              dp[i] = Integer.MAX_VALUE;//\\
              for (int j = 1; j * j <= i; j++) {
                  dp[i] = Math.min(dp[i], dp[i - j * j] + 1);
              }
          }

          return dp[n];
      }
```

https://asanchina.wordpress.com/2015/09/11/perfect-squares/

```
      public int numSquares(int n) {
          if(n==1 || n==2 || n==3) return n;

          //dp[i] store the result of "i"
```

to-do-must (28)
Random (27)
Sliding Window (27)
GeeksQuiz (25)
Logic Thinking (25)
hihocoder (25)
High Frequency (23)
Palindrome (23)
Algorithm Game (22)
Company - LinkedIn (22)
Graph (22)
Hash (22)
Queue (22)
Binary Indexed Trees (21)
DFS + Review (21)
Pre-Sort (21)
TopCoder (21)
Brain Teaser (20)
CareerCup (20)
Company - Twitter (20)
Company-Facebook (19)
UVA (19)
Follow Up (18)
Merge Sort (18)
Probabilities (18)
Codercareer (16)
Company-Uber (16)
Game Theory (16)
Heap (16)
Shortest Path (16)

```
        int[] dp = new int[n+1];

        dp[0] = 0; dp[1] = 1; dp[2] = 2; dp[3] = 3;
        for(int i = 4; i < n+1; ++i)
        {
            //now divide i to two parts: i = left + right
            int right = 1;
            dp[i] = i;
            while(i >= right*right)
            {
                dp[i] = dp[i]>dp[i-right*right]+1?dp[i-right*right]+1:dp[i];
                ++right;
            }
        }
        return dp[n];
    }
```

https://github.com/shawnfan/LeetCode/blob/master/Java/Perfect%20Squares.java

```
public int numSquares(int n) {
    if (n <= 0) {
      return 0;
    }
    int[] dp = new int[n + 1];
    dp[0] = 0;

    for (int i = 1; i <= n; i++) {
      int maxSqrNum = (int)Math.floor(Math.sqrt(i));
      int min = Integer.MAX_VALUE;
      for (int j = 1; j <= maxSqrNum; j++) {
        min = Math.min(min, dp[i - j * j] + 1);
      }
      dp[i] = min;
    }
    return dp[n];
}
```

Googol: English To 'Pig Latin'
【新提醒】Uber Intern Onsite四轮附求职感想【一亩三分地论坛面经版】 - Power...
Find all possible outcomes of a given expression -...
LeetCode Q209 Minimum Size Subarray Sum | Lei Jian...
LeetCode 214 - Shortest Palindrome
[leetcode]Invert Binary Tree
LeetCode - Product of Array Except Self (Java)
leetcode 228: Summary Ranges
Zenefits Interview - mitbbs
Group sort
Distributing Reservoir Sampling
Weighted Reservoir Sampling
Gregable: Majority Voting Algorithm - Find the maj...
贡献一道G家onsite题吧 - 未名空间(mitbbs.com)
Word Abbreviation
Check String Order - Google Intreview
Leetcode: 总结
graph, detect the longest cycle in an array
Detect cycle in an array
find the minimal difference between two sorted arr...
4*4 matrix, walk from top left to right bottom, an...
Print Common Nodes in Two Binary Search Trees - Ge...
google面经 - 未名空间(mitbbs.com)
Leetcode: Google interview questions #3
Algorithm Game - Fishing -

http://shibaili.blogspot.com/2015/09/day-125-279-perfect-squares.html

```
int numSquares(int n) {
    vector<int> dp(n + 1,INT_MAX);
    dp[0] = 0;

    for (int i = 0; i <= n; i++) {
        for (int j = 1; j *j + i <= n; j++) {
            dp[i + j * j] = min(dp[i + j * j],dp[i] + 1);
        }
    }
    return dp[n];
}
```

Recursive Version: O(n * sqrt(n)) dp递归, 同样复杂度，但过不了oj

```
int helper(unordered_map<int,int> &dic,int n) {
    if (n < 4) return n;
    if (dic.find(n) != dic.end()) return dic[n];

    int minLen = INT_MAX;
    int i = (int)sqrt(n);
 for (; i > 0; i--) {
  minLen = min(minLen,helper(dic,n - i * i) + 1);
 }
 dic[n] = minLen;
 return minLen;
}

int numSquares(int n) {
    unordered_map<int,int> dic;
 return helper(dic,n);
}
```

由于该题有许多大型的测试样例，因此对于速度比较慢的编程语言，在测试样例之间共享计算结果，可以节省时间开销。

参考链接：https://leetcode.com/discuss/56993/static-dp-c-12-ms-python-172-ms-ruby-384-ms

```
class Solution(object):
    _dp = [0]
    def numSquares(self, n):
        dp = self._dp
```

```
        while len(dp) <= n:
            dp += min(dp[-i*i] for i in range(1, int(len(dp)**0.5+1))) + 1,
        return dp[n]
```

https://leetcode.com/discuss/58056/summary-of-different-solutions-bfs-static-and-mathematics

A Static DP solution in Java: 15ms.

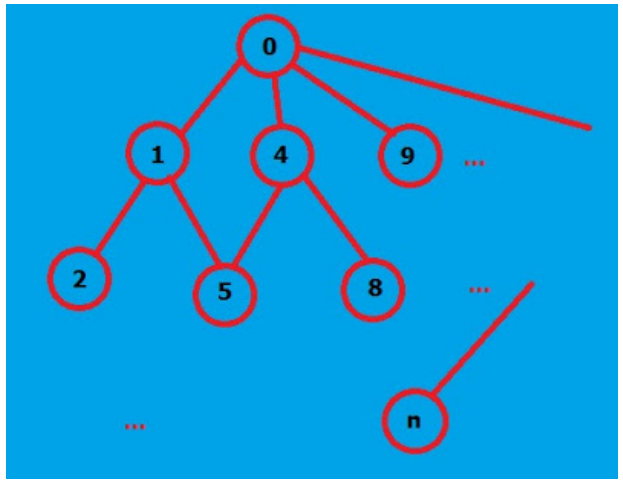
```
    static List<Integer> result = new ArrayList<>();
    public int numSquares(int n) {
        if (result.size() == 0) {
            result.add(0);
        }
        while (result.size() <= n) {
            int m = result.size();
            int tempMin = Integer.MAX_VALUE;
            for (int j = 1; j * j <= m; j++) {
                tempMin = Math.min(tempMin, result.get(m - j * j) + 1);
            }
            result.add(tempMin);
        }
        return result.get(n);
    }
```

II. Breadth First Search

Summary of 4 different solutions (BFS, DP, static DP and mathematics)

http://traceformula.blogspot.com/2015/09/perfect-squares-leetcode.html

In general, the time complexity of BFS is O(|V| + |E|) where |V| is the number of vertices in the graph and |E| is the number of edges in the graph. As in the constructed graph, |V| = n and |E| <= n^2. The time complexity of the BFS here is O(n^2).



Picture 1: Graph of numbers

In this problem, we define a graph where each number from *0* to *n* is a node. Two numbers *p* < *q* is connected if *(q-p)* is a perfect square.

So we can simply do a Breadth First Search from the node *0*.

```
01. class Solution(object):
02.     _dp = [0]
03.     def numSquares(self, n):
```

```
        """
        :type n: int
        :rtype: int
        """

        q1 = [0]
        q2 = []
        level = 0
        visited = [False] * (n+1)
        while True:
            level += 1
            for v in q1:
                i = 0
                while True:
                    i += 1
                    t = v + i * i
                    if t == n: return level
                    if t > n: break
                    if visited[t]: continue
                    q2.append(t)
                    visited[t] = True
            q1 = q2
            q2 = []

        return 0
```

http://nb4799.neu.edu/wordpress/?p=1010

Don't forget to use a set `visited` to record which nodes have been visited. A node can be reached through multiple ways but BFS always makes sure whenever it is reached with the least steps, it is flagged as visited.

Why can't we use DFS? Why can't we report steps when we reach n by DFS? Because unlike BFS, DFS doesn't make sure you are always using the least step to reach a node.

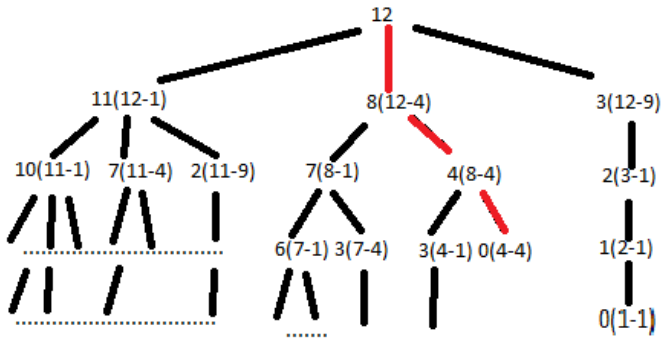
if you want to find "…."s least number to do something", you can try to think about solving it in BFS way.

```
    def numSquares(self, n):
        """
        :type n: int
        :rtype: int
        """
        # corner case 1
        if n < 0:
            return -1
        # corner case 2
        if n==0:
```

```
            return 1

        q = deque()
        visited = set()
        # val, step
        q.append((0,0))
        visited.add(0)

        while q:
            val, step = q.popleft()
            for i in xrange(1, n+1):
                tmp = val + i**2
                if tmp > n:
                    break
                if tmp == n:
                    return step+1
                else:
                    if tmp not in visited:
                        visited.add(tmp)
                        q.append((tmp, step+1))

        # Should never reach here
        return -1
```

https://leetcode.com/discuss/62229/short-python-solution-using-bfs

The basic idea of this solution is a BSF search for shortest path, take 12 as an example, as shown below, the shortest path is 12-8-4-0:




```
def numSquares(self, n):
    if n < 2:
        return n
    lst = []
    i = 1
    while i * i <= n:
        lst.append( i * i )
        i += 1
    cnt = 0
    toCheck = {n}
    while toCheck:
        cnt += 1
        temp = set()
        for x in toCheck:
            for y in lst:
                if x == y:
                    return cnt
                if x < y:
                    break
                temp.add(x-y)
        toCheck = temp

    return cnt
```

https://leetcode.com/discuss/58056/summary-of-different-solutions-bfs-static-and-mathematics

Java BFS implementation: Start from node 0 in queue, and keep pushing in perfect square number + curr value, once we reach number n, we found the solution.

```
public int numSquares(int n) {
    Queue<Integer> q = new LinkedList<>();
    Set<Integer> visited = new HashSet<>();
    q.offer(0);
    visited.add(0);
    int depth = 0;
    while(!q.isEmpty()) {
        int size = q.size();
        depth++;
        while(size-- > 0) {
```

```
            int u = q.poll();
            for(int i = 1; i*i <= n; i++) {
                int v = u+i*i;
                if(v == n) {
                    return depth;
                }
                if(v > n) {
                    break;
                }
                if(!visited.contains(v)) {
                    q.offer(v);
                    visited.add(v);
                }
            }
        }
    }
    return depth;
}
```

http://nb4799.neu.edu/wordpress/?p=1010

```
    def numSquares(self, n):
        """
        :type n: int
        :rtype: int
        """
        # corner case 1
        if n < 0:
            return -1
        # corner case 2
        if n==0:
            return 1

        q = deque()
        visited = set()
        # val, step
        q.append((0,0))
        visited.add(0)

        while q:
            val, step = q.popleft()
            for i in xrange(1, n+1):
                tmp = val + i**2
```

```
                if tmp > n:
                    break
                if tmp == n:
                    return step+1
                else:
                    if tmp not in visited:
                        visited.add(tmp)
                        q.append((tmp, step+1))

        # Should never reach here
        return -1
```

Use BFS to solve this problem. We can think of a tree graph in which nodes are value 0, 1,…,n. A link only exists between i and j (i<j) if j=i+square_num. We start from node 0 and add all nodes reachable from 0 to queue. Then we start to fetch elements from the queue to continue to explore until we reach node n. Since BFS guarantees that for every round all nodes in the queue have same depth, so whenever we reach node n, we know the least depth needed to reach n, i.e., the least number of square number to add to n.

Don't forget to use a set `visited` to record which nodes have been visited. A node can be reached through multiple ways but BFS always makes sure whenever it is reached with the least steps, it is flagged as visited.

Why can't we use DFS? Why can't we report steps when we reach n by DFS? Because unlike BFS, DFS doesn't make sure you are always using the least step to reach a node. Imagine DFS may traverse all nodes to reach n becaues each node is its previous node plus one (because one is a square number.) When you reach n, you use n steps. But it may or may not be the least number required to reach n.

This teaches us a lesson that if you want to find "….'s least number to do something", you can try to think about solving it in BFS way.

In this problem, we define a graph where each number from *0* to *n* is a node. Two numbers $p < q$ is connected if $(q-p)$ is a perfect square.
So we can simply do a Breadth First Search from the node *0*.

```
01. class Solution(object):
02.     _dp = [0]
03.     def numSquares(self, n):
04.         q1 = [0]
05.         q2 = []
06.         level = 0
07.         visited = [False] * (n+1)
08.         while True:
09.             level += 1
10.             for v in q1:
11.                 i = 0
```

```
                while True:
                    i += 1
                    t = v + i * i
                    if t == n: return level
                    if t > n: break
                    if visited[t]: continue
                    q2.append(t)
                    visited[t] = True
            q1 = q2
            q2 = []

        return 0
```

X. DFS

http://blog.csdn.net/elton_xiao/article/details/49021623

模型同上，进行深度优先搜索。

从n从发，减去一个平方，即得到下一层的一个节点。 直到找到节点0为止。

遍历所有路径，记录下最短的路径数。

1. 遍历时，使用变量level，限制递归的深度。以节省时间。

2. 先减去较大的平方，再减去较小的平方。 这样，当搜索到节点0时，所用的步数（即深度），不会特别大。从而用此深度，限制后续的搜索深度。

因为我们要找的是最短路径。

反之如果从1个平方开始递减的话，会出现大量的无用搜索。比如每次只减掉1，搜索n，将第一次会达到n层。

https://leetcode.com/discuss/58056/summary-of-different-solutions-bfs-static-and-mathematics

```
    int numSquares(int n) {
        vector<int> nums(n+1);
        return helper(n, nums, n);
    }

    int helper(int n, vector<int>& nums, int level) {
        if (!n)
            return 0;
        if (nums[n])
            return nums[n];
        if (!level)
            return -1;

        --level;
        const int up = sqrt(n);
        int ans = n;
        int res = n;
        for (int i=up; i>=1 && res; i--) {
            res = helper(n-i*i, nums, min(ans, level));
            if (res >= 0)
                ans = min(ans, res);
        }
        nums[n] = ans + 1;
        return nums[n];
    }
```

https://leetcode.com/discuss/62782/recursive-dfs-solution-in-java-which-i-believe-has-o-run-time

I'm not certain on my math so please correct me if I'm wrong, but the run-time should be O(n^(1/2) * n^(1/4) * n^(1/8) * ...), repeating as many times as there are layers in the minimum, which regardless of the number of layers approaches O(n).

```
public int numSquares(int n) {
    return dfs(n,0,n);
}
public int dfs(int n, int level, int min) {
    if(n == 0 || level >= min) return level;
    for(int i = (int) Math.sqrt(n); i > 0; i--) {
        min = dfs(n - ((int)Math.pow(i,2)), level+1, min);
    }
    return min;
}
```

1. 暴力枚举 brute force time complexity: exponential

```
int numSquares(int n) {
    int digit = sqrt(n);
    if(digit*digit == n) return 1;

    int minimum = n;//this is the maximum number
    dfs(n, 0, digit, 0, minimum);
    return minimum;
}
void dfs(int n, int curSum, int curDigit, int curNum, int& minimum)
{
    if(curSum > n) return;
    if(curSum == n)
    {
        minimum = minimum>curNum?curNum:minimum;
        return;
    }
    if(curDigit <= 0) return;
    if(curDigit == 1)
    {
        curNum += n-curSum;
        minimum = minimum>curNum?curNum:minimum;
        return;
    }
    //either use curDigit or not use
    if(n-curSum >= curDigit*curDigit)
```

```
            dfs(n, curSum+curDigit*curDigit, curDigit, curNum+1, minimum);
        dfs(n, curSum, curDigit-1, curNum, minimum);
    }
```

http://blog.csdn.net/u012290414/article/details/48730819
This doesn't use visited - not efficient.
http://traceformula.blogspot.com/2015/09/perfect-squares-leetcode-part-2-solve.html
I want to confirm that all the returned values will always be in range [1,4] inclusively. Why is that? It is because we have **Lagrange's Four Square Theorem**, also known as **Bachet's conjecture**:
Every natural numbers can be expressed as a sum of four square numbers. (*)
As a note, many proofs of the theorem use the Euler's four square identity:

$$(a_1^2 + a_2^2 + a_3^2 + a_4^2)(b_1^2 + b_2^2 + b_3^2 + b_4^2) =$$
$$(a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4)^2 +$$
$$(a_1b_2 - a_2b_1 + a_3b_4 - a_4b_3)^2 +$$
$$(a_1b_3 - a_2b_4 - a_3b_1 + a_4b_2)^2 +$$
$$(a_1b_4 + a_2b_3 - a_3b_2 - a_4b_1)^2.$$

Picture 1: Euler's Four Square Identity

http://www.alpertron.com.ar/4SQUARES.HTM
From the article, you can find that if a number is in the form *n = 4^r (8k+7)*(Where ^ is *power),* then *n* cannot be represented as a sum of less than 4 perfect squares. If *n* is not in the above form, then *n* can be represented as a sum of 1, 2, or 3 perfect squares.

```
01.     public boolean is_square(int n){
02.         int temp = (int) Math.sqrt(n);
03.         return temp * temp == n;
04.     }
05.     public int numSquares(int n) {
06.         while ((n & 3) == 0) //n % 4 == 0
07.             n >>= 2;
08.         if ((n & 7) == 7) return 4; //n% 8 == 7
09.
10.         if(is_square(n)) return 1;
11.         int sqrt_n = (int) Math.sqrt(n);
12.         for (int i = 1; i<= sqrt_n; i++){
13.             if (is_square(n-i*i)) return 2;
14.         }
15.         return 3;
16.     }
```

https://discuss.leetcode.com/topic/24255/summary-of-4-different-solutions-bfs-dp-static-dp-and-mathematics/2
Read full article from Perfect Squares [LeetCode] | Training dragons the hard way - Programming Every Day!

You might also like

Twitter Interview Misc

Missing Number - Cracking the coding interview

HackerRank: Cutting boards

HackerRank - Non-Divisible Subset

Find Maximum number possible by doing at-most K swaps - GeeksforGeeks

Find the only repeating element in a sorted array of size n - GeeksforGeeks

LeetCode 468 - Validate IP Address

Check if given sorted sub-sequence exists in binary search tree - GeeksforGeeks

leetcode 475 - Heaters

Maximum distance between two occurrences of same element in array

at 10:48 PM

Labels: BFS, Classic Interview, Different Solutions, Dynamic Programming, Google Interview, LeetCode

No comments:

Post a Comment

Labels

GeeksforGeeks (1020) Algorithm (811) LeetCode (696) to-do (601) Review (412) Classic Algorithm (334) Classic Interview (298) Dynamic Programming (269) LeetCode - Review (248) Google Interview (234) Tree (146) POJ (137) Difficult Algorithm (136) EPI (127) Different Solutions (119) Bit Algorithms (113) Cracking Coding Interview (110) Smart Algorithm (110) Math (95) HackerRank (88) Lintcode (83) Binary Search (75) Graph Algorithm (75) Greedy Algorithm (61) Interview Corner (61) Binary Tree (59) DFS (58) List (58) Codility (54) Algorithm Interview (53) Advanced Data Structure (52) ComProGuide (52) LeetCode - Extended (47) USACO (46) Geometry Algorithm (45) BFS (43) Data Structure (42) Mathematical Algorithm (42) ACM-ICPC (41) Jobdu (39) Knapsack (39) Stack (39) Binary Search Tree (38) Interval (38) Recursive Algorithm (38) String Algorithm (38)

Codeforces (36) Introduction to Algorithms (36) Matrix (36) Must Known (36) Space Optimization (36) Beauty of Programming (35) Sort (35) Trie (34) Array (33) prismoskills (33) Backtracking (32) Segment Tree (32) Union-Find (32) HDU (31) Google Code Jam (30) Permutation (30) Puzzles (30) Array O(N) (29) Data Structure Design (29) Company-Zenefits (28) Microsoft 100 - July (28) to-do-must (28) Random (27) Sliding Window (27) GeeksQuiz (25) Logic Thinking (25) hihocoder (25) High Frequency (23) Palindrome (23) Algorithm Game (22) Company - LinkedIn (22) Graph (22) Hash (22) Queue (22) Binary Indexed Trees (21) DFS + Review (21) Pre-Sort (21) TopCoder (21) Brain Teaser (20) CareerCup (20) Company - Twitter (20) Company-Facebook (19) UVA (19) Follow Up (18) Merge Sort (18) Probabilities (18) Codercareer (16) Company-Uber (16) Game Theory (16) Heap (16) Shortest Path (16) String Search (16) Topological Sort (16) Tree Traversal (16) itint5 (16) Bisection Method (15) Iterator (15) O(N) (15) Post-Order Traverse (15) Priority Quieue (15) Difficult (14) Number (14) Number Theory (14) Amazon Interview (13) BST (13) Basic Algorithm (13) Codechef (13) KMP (13) Long Increasing Sequence(LIS) (13) Majority (13) Ordered Stack (13) mitbbs (13) Combination (12) Computational Geometry (12) Euclidean GCD (12) Modify Tree (12) Reconstruct Tree (12) Reservoir Sampling (12) 尺取法 (12) AOJ (11) Binary Search - Bisection (11) DFS+Backtracking (11) Fast Power Algorithm (11) Graph DFS (11) LCA (11) LeetCode - DFS (11) Princeton (11) Tree DP (11) 挑战程序设计竞赛 (11) Company - Microsoft (10) Company-Airbnb (10) Facebook Hacker Cup (10) HackerRank Easy (10) Reverse Thinking (10) Rolling Hash (10) SPOJ (10) Theory (10) Time Complexity (10) Tutorialhorizon (10) X Sum (10) Coin Change (9) Divide and Conquer (9) Lintcode - Review (9) Mathblog (9) Max-Min Flow (9) Stack Overflow (9) Stock (9) Two Pointers (9) Use XOR (9) Book Notes (8) Bottom-Up (8) DP-Space Optimization (8) Graph BFS (8) LeetCode - DP (8) LeetCode Hard (8) O(1) Space (8) Prefix Sum (8) Prime (8) Quick Sort (8) Suffix Tree (8) System Design (8) Tech-Queries (8) 穷竭搜索 (8) Algorithm Problem List (7) DFS+BFS (7) Facebook Interview (7) Fibonacci Numbers (7) Game Nim (7) HackerRank Difficult (7) Hackerearth (7) Interval Tree (7) Linked List (7) Longest Common Subsequence(LCS) (7) Math-Divisible (7) Minimum Spanning Tree (7) Miscs (7) Probability DP (7) Radix Sort (7) Simulation (7) Xpost (7) n00tc0d3r (7) 蓝桥杯 (7) Bucket Sort (6) Catalan Number (6) Classic Data Structure Impl (6) DFS+Cache (6) DFS+DP (6) DP - Tree (6) How To (6) Interviewstreet (6) Inversion (6) Kadane's Algorithm (6) Knapsack - MultiplePack (6) Level Order Traversal (6) Manacher (6) One Pass (6) Programming Pearls (6) Quick Select (6) Rabin-Karp (6) Randomized Algorithms (6) Sampling (6) Schedule (6) Suffix Array (6) Threaded (6) reddit (6) AI (5) Art Of Programming-July (5) Big Data (5) Brute Force (5) Code Kata (5) Codility-lessons (5) Coding (5) Company - WMware (5) Crazyforcode (5) DP-Multiple Relation (5) DP-Print Solution (5) Dutch Flag (5) Fast Slow Pointers (5) Graph Cycle (5) Hash Strategy (5) Immutability (5) Java (5) Kadane - Extended (5) Matrix Chain Multiplication (5) Microsoft Interview (5) Morris Traversal (5) Pruning (5) Quadtrees (5) Quick Partition (5) Quora (5) SPFA(Shortest Path Faster Algorithm) (5) Subarray Sum (5) Sweep Line (5) Traversal Once (5) TreeMap (5) jiuzhang (5) 单调栈 (5) 树形DP (5) 1point3acres (4) Anagram (4) Approximate Algorithm (4) Backtracking-Include vs Exclude (4) Brute Force - Enumeration (4) Chess Game (4) Company-Amazon (4) Consistent Hash (4) Convex Hull (4) Cycle (4) DP-Include vs Exclude (4) Dijkstra (4) Distributed (4) Eulerian Cycle (4) Flood fill (4) Graph-Classic (4) HackerRank AI (4) Histogram (4) Kadane Max Sum (4) Knapsack - Mixed (4) Knapsack - Unbounded (4) Left and Right Array (4) MinMax (4) Multiple Data Structures (4) N Queens (4) Nerd Paradise (4) Parallel Algorithm (4) Practical Algorithm (4) Pre-Sum (4) Probability (4) Programcreek (4) Spell Checker (4) Stock Maximize (4) Subsets (4) Sudoku (4) Symbol Table (4) TreeSet (4) Triangle (4) Water Jug (4) Word Ladder (4) algnotes (4) fgdsb (4) to-do-2 (4) 最大化最小值 (4) A Star (3) Abbreviation (3) Algorithm - Brain Teaser (3) Algorithm Design (3) Anagrams (3) B Tree (3) Big Data Algorithm (3) Binary Search - Smart (3) Caterpillar Method (3) Coins (3) Company - Groupon (3) Company - Indeed (3) Cumulative Sum (3) DP-Fill by Length (3) DP-Two Variables (3) Dedup (3) Dequeue (3) Dropbox (3) Easy (3) Edit Distance (3) Expression (3) Finite Automata (3) Forward && Backward Scan (3) Github (3) GoLang (3) Graph - Bipartite (3) Include vs Exclude (3) Joseph (3) Jump Game (3) Knapsack-多重背包 (3) LeetCode - Bit (3) LeetCode - TODO (3) Linked List Merge Sort (3) LogN (3) Master Theorem (3) Maze (3) Min Cost Flow (3) Minesweeper (3) Missing Numbers (3) NP Hard (3) Online Algorithm (3) Pascal's Triangle (3) Pattern Match (3) Project Euler (3) Rectangle (3) Scala (3) SegmentFault (3) Stack - Smart (3) State Machine (3) Streaming Algorithm (3) Subset Sum (3) Subtree (3) Transform Tree (3) Two Pointers Window (3) Warshall Floyd (3) With Random Pointer (3) Word Search (3) bookkeeping (3) codebytes (3) Activity Selection Problem (2) Advanced Algorithm (2) AnAlgorithmADay (2) Application of Algorithm (2) Array Merge (2) BOJ (2) BT - Path Sum (2) Balanced Binary Search Tree (2) Bellman Ford (2) Binomial Coefficient (2) Bit Mask (2) Bit-Difficult (2) Bloom Filter (2) Book Coding Interview (2) Branch and Bound Method (2) Clock (2) Codesays (2) Company - Baidu (2) Complete Binary Tree (2) DFS+BFS， Flood Fill (2) DP - DFS (2) DP-3D Table (2) DP-Classical (2) DP-Output Solution (2) DP-Slide Window Gap (2) DP-i-k-j (2) DP-树形 (2) Distributed Algorithms (2) Divide and Conqure (2) Doubly Linked List (2) GoHired (2) Graham Scan (2) Graph BFS+DFS

(2) Graph Coloring (2) Graph-Cut Vertices (2) Hamiltonian Cycle (2) Huffman Tree (2) In-order Traverse (2) Include or Exclude Last Element (2) Information Retrieval (2) Interview - Linkedin (2) Invariant (2) Islands (2) Knuth Shuffle (2) LeetCode - Recursive (2) Linked Interview (2) Linked List Sort (2) Longest SubArray (2) Lucene-Solr (2) MST (2) MST-Kruskal (2) Math-Remainder Queue (2) Matrix Power (2) Minimum Vertex Cover (2) Negative All Values (2) Number Each Digit (2) Numerical Method (2) Object Design (2) Order Statistic Tree (2) Palindromic (2) Parentheses (2) Parser (2) Peak (2) Programming (2) Range Minimum Query (2) Reuse Forward Backward (2) Robot (2) Rosettacode (2) Scan from right (2) Search (2) Shuffle (2) Sieve of Eratosthenes (2) SimHash (2) Simple Algorithm (2) Skyline (2) Spatial Index (2) Stream (2) Strongly Connected Components (2) Summary (2) TV (2) Tile (2) Traversal From End (2) Tree Sum (2) Tree Traversal Return Multiple Values (2) Word Break (2) Word Graph (2) Word Trie (2) Young Tableau (2) 剑指Offer (2) 数位DP (2) 1-X (1) 51Nod (1) Akka (1) Algorithm - How To (1) Algorithm - New (1) Algorithm Series (1) Algorithms Part I (1) Analysis of Algorithm (1) Array-Element Index Negative (1) Array-Rearrange (1) Auxiliary Array (1) Auxiliary Array: Inc&Dec (1) BACK (1) BK-Tree (1) BZOJ (1) Basic (1) Bayes (1) Beauty of Math (1) Big Integer (1) Big Number (1) Binary (1) Binary String (1) Binary Tree Variant (1) Bipartite (1) Bit-Missing Number (1) BitMap (1) BitMap index (1) BitSet (1) Bug Free Code (1) BuildIt (1) C/C++ (1) CC Interview (1) Cache (1) Calculate Height at Same Recusrion (1) Cartesian tree (1) Check Tree Property (1) Chinese (1) Circular Buffer (1) Code Quality (1) Codesolutiony (1) Company - Alibaba (1) Company - Palantir (1) Company - WalmartLabs (1) Company-Apple (1) Company-Epic (1) Company-Salesforce (1) Company-Snapchat (1) Company-Yelp (1) Compression Algorithm (1) Concurrency (1) Convert BST to DLL (1) Convert DLL to BST (1) Custom Sort (1) Cyclic Replacement (1) DFS-Matrix (1) DP - Probability (1) DP Fill Diagonal First (1) DP-Difficult (1) DP-End with 0 or 1 (1) DP-Fill Diagonal First (1) DP-Graph (1) DP-Left and Right Array (1) DP-MaxMin (1) DP-Memoization (1) DP-Node All Possibilities (1) DP-Optimization (1) DP-Preserve Previous Value (1) DP-Print All Solution (1) Database (1) Detect Negative Cycle (1) Directed Graph (1) Do Two Things at Same Recusrion (1) Domino (1) Dr Dobb's (1) Duplicate (1) Equal probability (1) External Sort (1) FST (1) Failure Function (1) Fraction (1) Front End Pointers (1) Funny (1) Fuzzy String Search (1) Game (1) Generating Function (1) Generation (1) Genetic algorithm (1) GeoHash (1) Geometry - Orientation (1) Google APAC (1) Graph But No Graph (1) Graph Transpose (1) Graph Traversal (1) Graph-Coloring (1) Graph-Longest Path (1) Gray Code (1) HOJ (1) Hanoi (1) Hard Algorithm (1) How Hash (1) How to Test (1) Improve It (1) In Place (1) Inorder-Reverse Inorder Traverse Simultaneously (1) Interpolation search (1) Interview (1) Interview - Easy (1) Interview - Facebook (1) Isomorphic (1) JDK8 (1) K Dimensional Tree (1) Knapsack - Fractional (1) Knapsack - ZeroOnePack (1) Knight (1) Kosaraju's algorithm (1) Kruskal (1) Kruskal MST (1) Kth Element (1) Least Common Ancestor (1) LeetCode - Binary Tree (1) LeetCode - Coding (1) LeetCode - Detail (1) LeetCode - Related (1) LeetCode Diffcult (1) Linked List Reverse (1) Linkedin (1) Linkedin Interview (1) Local MinMax (1) Logic Pattern (1) Longest Common Subsequence (1) Longest Common Substring (1) Longest Prefix Suffix(LPS) (1) Manhattan Distance (1) Map && Reverse Map (1) Math - Induction (1) Math-Multiply (1) Math-Sum Of Digits (1) Matrix - O(N+M) (1) Matrix BFS (1) Matrix Graph (1) Matrix Search (1) Matrix+DP (1) Matrix-Rotate (1) Max Min So Far (1) Median (1) Memory-Efficient (1) MinHash (1) MinMax Heap (1) Monotone Queue (1) Monto Carlo (1) Multi-Reverse (1) Multiple DFS (1) Multiple Tasks (1) Next Successor (1) Offline Algorithm (1) PAT (1) Parent-Only Tree (1) Partition (1) Path Finding (1) Patience Sort (1) Persistent (1) Pigeon Hole Principle (1) Power Set (1) Pratical Algorithm (1) Probabilistic Data Structure (1) Proof (1) Python (1) Queue & Stack (1) RSA (1) Ranking (1) Rddles (1) ReHash (1) Realtime (1) Recurrence Relation (1) Recursive DFS (1) Recursive to Iterative (1) Red-Black Tree (1) Region (1) Regular Expression (1) Resources (1) Reverse Inorder Traversal (1) Robin (1) Selection (1) Self Balancing BST (1) Similarity (1) Sort && Binary Search (1) String Algorithm. Symbol Table (1) String DP (1) String Distance (1) SubMatrix (1) Subsequence (1) System of Difference Constraints(差分约束系统) (1) TSP (1) Ternary Search Tree (1) Test (1) Thread (1) TimSort (1) Top-Down (1) Tournament (1) Tournament Tree (1) Transform Tree in Place (1) Tree Diameter (1) Tree Rotate (1) Trie + DFS (1) Trie and Heap (1) Trie vs Hash (1) Trie vs HashMap (1) Triplet (1) Two Data Structures (1) Two Stacks (1) USACO - Classical (1) USACO - Problems (1) UyHiP (1) Valid Tree (1) Vector (1) Wiggle Sort (1) Wikipedia (1) Yahoo Interview (1) ZOJ (1) baozitraining (1) codevs (1) cos126 (1) javabeat (1) jum (1) namic Programming (1) sqrt(N) (1) 两次dijkstra (1) 九度 (1) 二进制枚举 (1) 夹逼法 (1) 归一化 (1) 折半枚举 (1) 枚举 (1) 状态压缩DP (1) 男人八题 (1) 英雄会 (1) 逆向思维 (1)

Popular Posts

LeetCode 415 - Add Strings
https://leetcode.com/problems/add-strings/ Given two non-negative numbers num1 and num2 represented as string, return the sum of num...

LeetCode 445 - Add Two Numbers II
https://leetcode.com/problems/add-two-numbers-ii/ You are given two linked lists representing two non-negative numbers. The most signific...

LeetCode 448 - Find All Numbers Disappeared in an Array
https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/ Given an array of integers where 1 ≤ a[i] ≤ n (n = size of ar...

LeetCode 418 - Sentence Screen Fitting
http://bookshadow.com/weblog/2016/10/09/leetcode-sentence-screen-fitting/ Given a rows x cols screen and a sentence represented by a li...

LeetCode 438 - Find All Anagrams in a String
https://leetcode.com/problems/find-all-anagrams-in-a-string/ Given a string s and a non-empty string p , find all the start indices ...

LeetCode 388 - Longest Absolute File Path
http://bookshadow.com/weblog/2016/08/21/leetcode-longest-absolute-file-path/ Suppose we abstract our file system by a string in the follo...

LeetCode 407 - Trapping Rain Water II
https://leetcode.com/problems/trapping-rain-water-ii/ Given an m x n matrix of positive integers representing the height of each unit c...

LeetCode 458 - Poor Pigs
https://leetcode.com/problems/poor-pigs/ There are 1000 buckets, one and only one of them contains poison, the rest are filled with water...

LeetCode 461 - Hamming Distance
http://bookshadow.com/weblog/2016/12/18/leetcode-hamming-distance/ The Hamming distance between two integers is the number of positions...

LeetCode - Remove Duplicates from Sorted Array | Darren's Blog
Given a sorted array, remove the duplicates in place such that each element appear only once and return the new length. Do not allocate ext...