bright sunlight

$h_m \in \mathcal{H}$

$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + \text{const}$

$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$

$F_m(x) = F_{m-1}(x) + \arg\min_{h_m \in \mathcal{H}} [\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i))]$

$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)),$

$$\forall k = 0, 1, \quad \mu_k = \sum_{i=1}^{n_k} p_{k,i} \delta_{x_{k,i}} \tag{1}$$

$$X_k = (x_{k,i})_{i=1}^{n_i} \subset \mathbb{R}^d \tag{2}$$

$$\mathcal{P} = \enscond(\gamma_{i,j})_{i,j} \in (\RR^+)^{n_0 \times n_1} \forall i, \sum_j \ga_{i,j} = p_{0,i}\,,\ \forall j, \sum_i \ga_{i,j} = p_{1,j} \tag{3}$$

In the bright sunlight, he caught sight of a terrific shadow/outline of a figure whom he didn't expect to be Yang, his dream girl. And actually he didn't even immediately recognize her at the first sight.

Looking at his left elbow, he saw that there was a big worm enjoying his fresh blood silently, but he didn't squeeze the greedy devil's big belly before it was full and stopped.

Listening to the noise outside the window, he heard the quarrel between a young couple, but they both began laughing loudly before he felt obliged to give some advice about life.

2019年11月14日00:37:10

I saw the news that Citibank would have a campus talk at our school on its WeChat official account.

So I have come here for an interview opportunity. I sent my Chinese resume on your company's website and email a couple of days ago but got no response. I thought maybe it's because Citi is an international company, so I have also taken my English resume here today.

Well, my name is Jiang Qizhi. I was born on July 16, 1994, in a peaceful countryside. I am currently a last-year graduate student at the School of Computer Science in Fudan University. From 2013 to 2017, I spent four years studying Financial Mathematics at the School of Mathematics and Statistics in Henan University and got my science bachelor degree there. So, I have a little background of economics and finance and a relatively solid background of advanced mathematics. Although I like mathematics, I don't think I am talented enough to have some great findings in this area full of genius. So, I decided to change my major to IT related subject. Because I studied hard and had a good rank in my major, finally I got the qualification of being a graduate candidate at the School of Computer Science in Fudan University without examination. Over the last 2 and a half years, I have been studying computer technology under the supervision of professor Zhou. Since my research area is privacy preservation, my daily work is to read related thesis and propose novel solutions. In addition to general literature reading, I also perform programming to implement new ideas, or to validate existing algorithms in the literature. I am a **self-starter**, my advisor values my diligence and initiative in a varied of situations. And I expect to graduate from here next year if everything goes right.

Just a few years ago when I was younger, I thought that I have the ability to see through a person's heart, that is, I could understood the joy and sorrow and jealousy and hatred and kind silently going on in their little heart. I acknowledged my ignorance after many times introspections afterwards and realizing that those emotions quietly and secretly happened in my heart were actually even not fully recognized by myself immediately.

Preserving Differential Privacy in Degree-Correlation based Graph Generation

Enabling accurate analysis of social network data while preserving differential privacy has been challenging since graph features such as cluster coefficient often have high sensitivity, which is different from traditional aggregate functions (e.g., count and sum) on tabular data. In this paper, we study the problem of enforcing edge differential privacy in graph generation. The idea is to enforce differential privacy on graph model parameters learned from the original network and then generate the graphs for releasing using the graph model with the private parameters. In particular, we develop a differential privacy preserving graph generator based on the dK-graph generation model. We first derive from the original graph various parameters (i.e., degree correlations) used in the dK-graph model, then enforce edge differential privacy on the learned parameters, and finally use the dK-graph model with the perturbed parameters to generate graphs. For the 2K-graph model, we enforce the edge differential privacy by calibrating noise based on the smooth sensitivity, rather than the global sensitivity. By doing this, we achieve the strict differential privacy guarantee with smaller magnitude noise. We conduct experiments on four real networks and compare the performance of our private dK-graph models with the stochastic Kronecker graph generation model in terms of utility and privacy tradeoff. Empirical evaluations show the developed private dK-graph generation models significantly outperform the approach based on the stochastic Kronecker generation model.

Enabling accurate analysis of human genetic data while preserving participants' privacy has been challenging since the intrinsic sensitive and high dimensional characteristics of a person's genotype profiles. In this paper, we study the problem of enforcing DP in subpopulation partitioning task. The idea is to enforce DP in the process of dimensionality reduction step on genetic data and then partition the population into several subpopulations using a clustering algorithm. Concretely, we developed a general privacy-preserving framework PSP.

2019年11月17日14:29:20

What am I looking and waiting for? Who am I and what kind of person I want to be? As far as I am concerned, those aren questions of utmost importance. As for questions like why am I so unhappy, are relatively not so essential to my future prosperity. Because when I ask the latter question, situations are tend to be that I know clearly what is suspending on my mind makes me so worried and unhappy. I know it, but I just avoid to make choice of fighting or fleeing. I am thus stuck there and harmfully repeat what am I gonna do, what am I gonna do. I doubt the harmfulness of doing so is even more severe than just fleeing without thinking too much. I have the choice of fleeing but I don't realize it in most circumstances, because the idea that I really really don't want to do this fuck thing, but I must do it is stubbornly planted in my mind and repeated over and over again.

2019年11月18日17:32:43

forgive myself with a Flower to miss you

原谅捧花的我未能出席心疼人民币

it's a kind of obsession, obsession with self-addiction.

As far as I am concerned, the existence that could be called systematic influences to children is their parents, partners and teachers(but last two tend to be changeable), so basically only their parents could comprehensively and systematically affect (even decide?) the early years happiness and development of their lives.

I felt a thrill of fear for the little girl when I heard "systematic" plans from her mother who must loves her unconditionally.

My mother is surely kind to me, but I had seen a miserable peer who rarely came out to play with us.

Thanks.By the way, what's the story behind your profile picture, the person i liked secretly has the same one.

2019年11月28日13:09:23

I beat my laziness in a stage-wise fashion like many other anti-procrastination books suggest, and I practice them by doing one thing that is only a little beyond my comfortzone at a time without thinking too much and controlling my distraction intentionally.

2019年11月28日19:59:29

2019-11 IEEEBIBM 2019 Award Committee Chairs Best Student Paper

```
//binary-tree-level-order-traversal
/**2
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        if(root==NULL)
            return {};
        else
        {
            queue<TreeNode*> q;
            q.push(root);
            vector<int> inter;
            vector<vector<int>> result;
            while(q.size()!=NULL)
            {
               int count=q.size();
                while(count>0)
                {   TreeNode* curr=q.front();
                    q.pop();
                    inter.push_back(curr->val);
                    if(curr->left!=NULL)
                        q.push(curr->left);
                    if(curr->right!=NULL)
                        q.push(curr->right);
                    count--;
                }
                result.push_back(inter);
                inter={};
            }
             return result;
         }
    }
};


* https://leetcode-cn.com/problems/binary-tree-level-order-traversal-ii
* 题目描述：
*     给定一个二叉树，返回其节点值自底向上的层次遍历。
```

```
*     即按从叶子节点所在层到根节点所在的层，逐层从左向右遍历
*
* 例如：
*     给定二叉树 [3,9,20,null,null,15,7]
*         3
*        / \
*       9  20
*         /  \
*        15   7
*     返回其自底向上的层次遍历为：
*     [
*       [15,7],
*       [9,20],
*       [3]
*     ]
*
* 解题思路：
*     由于需要按层进行组成，且是vector<vector<int>>
*     所以必须把每个层分开记录，另一点就是还要实现逆序
*     分层可以通过单独记录每层节点来完成，而逆序有两种方式
*     第一种是直接正序记录，然后再颠倒顺序
*     第二种是进行递归记录每层节点，最后在每层中添加数据
*     这样第二种就省去了第一种方法中颠倒顺序这步的操作
*/

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<vector<int>> levelOrderBottom(TreeNode* root) {
        if (root == NULL)
            return {};

        vector<vector<int>> res;
        vector<TreeNode *> nodes;
        nodes.push_back(root);
        do_order(nodes, res);

        return res;
    }

    void do_order(vector<TreeNode *> &nodes, vector<vector<int>> &res) {
        if (nodes.size() == 0)
            return;

        vector<TreeNode *> next_level;
        for (int i=0; i<nodes.size(); i++) {
            if (nodes[i]->left)
                next_level.push_back(nodes[i]->left);
            if (nodes[i]->right)
                next_level.push_back(nodes[i]->right);
        }

        do_order(next_level, res);
```

```
        vector<int> temp;
        for (int i=0; i<nodes.size(); i++) {
            temp.push_back(nodes[i]->val);
        }
        res.push_back(temp);
    }
};

/*4
//minimum-depth-of-binary-tree

/*
* https://leetcode-cn.com/problems/minimum-depth-of-binary-tree
* 题目描述：
*     给定一个二叉树，找出其最小深度。
*     最小深度是从根节点到最近叶子节点的最短路径上的节点数量。
*
* 说明:
*     叶子节点是指没有子节点的节点。
*
* 示例:
*     给定二叉树 [3,9,20,null,null,15,7],
*             3
*            / \
*           9  20
*             /  \
*            15   7
*     返回它的最小深度  2.
*
* 解题思路：
*     使用递归的方法，求最短路径，也就是左右子树的最短路径
*     主要考虑几种特殊情况：
*     1. 递归终止条件，即无子节点时终止，即空指针
*     2. 没有左子树或右子树时如何处理
*     3. 左右子树都存在时如何处理
*     4. 左右子树都不存在时如何处理
*     总之，只要进入递归的不是空指针，当节点都算一个深度值
*     这也就是为什么返回值后都 +1 的原因
*/

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    int minDepth(TreeNode* root) {
        if (root == nullptr)
            return 0;

        int left, right;
        left = minDepth(root->left);
        right = minDepth(root->right);

        if (right == 0 && left == 0)
            return 1;
```

```
        if (right == 0)
            return left+1;
        if (left == 0)
            return right+1;

        return left>right ? (right+1) : (left+1);
    }
};

/*5
//two-sum
/*
 https://leetcode-cn.com/problems/two-sum
 给定一个整数数组 nums 和一个目标值 target，
     请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。
     你可以假设每种输入只会对应一个答案。但是，你不能重复利用这个数组中同样的元素。

 示例:
     给定 nums = [2, 7, 11, 15], target = 9
     因为 nums[0] + nums[1] = 2 + 7 = 9
     所以返回 [0, 1]

 解题思路:
     1. 最简单的方法就是暴力搜索了，两个嵌套循环就搞定了
     2. 为了降低复杂度，努力的方向就是在有了第一个数之后
        如何快速的判断是否有另一个数存在，这里就可以看出其实是一种映射
     3. 映射方式可以使用 map、hash table 等，而 hash table 更快
        所以这里使用 unordered_map，因为其底层实现就是 hash table
*/

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> data;
        construct_map(nums, data);

        int i=0;
        vector<int> res;
        while (i < nums.size()) {
            /*
                这里是不对的，两个数的和不一定月假越大，比如负数!
                if (nums[i] > target) {
                    i++;
                    continue;
                }
            */
            int left = target - nums[i];
            if (data.count(left) && data[left] != i) {
                res.push_back(i);
                res.push_back(data[left]);
                break;
            }

            i++;
        }

        return res;
    }

    void construct_map(vector<int> &nums, unordered_map<int, int> &data) {
        int i = 0;
```

```
        while (i < nums.size()) {
            data[nums[i]] = i;
            i++;
        }
    }
};

/*6
* https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array
* 题目描述：
*     给定一个排序数组，你需要在原地删除重复出现的元素，使得每个元素只出现一次，
*     返回移除后数组的新长度。
*     不要使用额外的数组空间，你必须在原地修改输入数组
*     并在使用 O(1) 额外空间的条件下完成。
*
* 示例 1:
* 给定数组 nums = [1,1,2],
*     函数应该返回新的长度 2, 并且原数组 nums 的前两个元素被修改为 1, 2。
*     你不需要考虑数组中超出新长度后面的元素。
*
* 示例 2:
*     给定 nums = [0,0,1,1,1,2,2,3,3,4],
*     函数应该返回新的长度 5, 并且原数组 nums 的前五个元素被修改为 0, 1, 2, 3, 4。
*
* 注：
*     你不需要考虑数组中超出新长度后面的元素。
*
* 解题思路：
*     1. 题目的意思其实就是将不同的数字向前紧密排列，因此涉及到下一个新的数字
*        放在哪里，即需要一个索引值指定，还需要知道最新的数字在哪里，即又需要
*        一个索引值
*     2. 因此使用两个索引值，相同时一个前移、一个不动；
*        遇到不同时，前边的指针前移并将新的数字复制过来，然后走的快的
*        指针继续前移，如此反复即可
*/

class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if (nums.size() <= 1)
            return nums.size();

        int head = 0, tail = 0;
        while (tail < nums.size()) {
            if (nums[head] == nums[tail]) {
                tail++;
            } else {
                head++;
                if (head < tail) {
                    nums[head] = nums[tail];
                    tail++;
                }
            }
        }

        nums.resize(head+1);

        return head+1;
    }
};
```

```
/*7
* https://leetcode-cn.com/problems/remove-element
* 题目描述：
*     给定一个数组 nums 和一个值 val，你需要原地移除所有数值等于 val 的元素，
*     返回移除后数组的新长度。
*     不要使用额外的数组空间，你必须在原地修改输入数组
*     并在使用 O(1) 额外空间的条件下完成。
*     元素的顺序可以改变。你不需要考虑数组中超出新长度后面的元素。
*
* 示例 1:
*     给定 nums = [3,2,2,3], val = 3,
*     函数应该返回新的长度 2, 并且 nums 中的前两个元素均为 2。
*     你不需要考虑数组中超出新长度后面的元素。
*
* 示例 2:
*     给定 nums = [0,1,2,2,3,0,4,2], val = 2,
*     函数应该返回新的长度 5, 并且 nums 中的前五个元素为 0, 1, 3, 0, 4。
*     注意这五个元素可为任意顺序。
*
* 注：
*     你不需要考虑数组中超出新长度后面的元素。
*
* 解题思路：
*     1. 解题思路与题目 0026_remove_duplicates_from_sorted_array 类似
*        不同的是这里需要覆盖的是指定数字，同样需要双指针
*     2. 注意指针的含义即可
*/

class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        if (nums.size() == 0)
            return 0;

        int head = 0, tail = 0;
        while (tail < nums.size()) {
            if (nums[tail] == val) {
                tail++;
            } else {
                if (head < tail) {
                    nums[head] = nums[tail];
                }
                head++;
                tail++;
            }
        }

        nums.resize(head);

        return head;
    }
};

/*8
* https://leetcode-cn.com/problems/search-insert-position
* 题目描述：
*     给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。
*     如果目标值不存在于数组中，返回它将会被按顺序插入的位置。
*     你可以假设数组中无重复元素。
*
* 示例 1:
```

```
*     输入: [1,3,5,6], 5
*     输出: 2
*
* 示例 2:
*     输入: [1,3,5,6], 2
*     输出: 1
*
* 示例 3:
*     输入: [1,3,5,6], 7
*     输出: 4
*
* 示例 4:
*     输入: [1,3,5,6], 0
*     输出: 0
*
* 解题思路:
*     1. 最简单粗暴的方法就是逐个遍历，直到找到目标数字或者第一个
*        大于目标数的值，不过时间复杂度为 O(N)
*     2. 由于给定的数组是有序数组，所以可以使用二分法进行查找
*        这样时间复杂度就可以变成 O(logN)
*/

class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        if (nums.size() == 0)
            return 0;

        int start = 0, end = nums.size() - 1;
        if (nums[start] > target)
            return 0;
        if (nums[end] < target)
            return end + 1;

        int mid;
        while (start < end) {
            /*
                本来想优化一下，结果这句造成了死循环
                按理说有符号的正数不应该出现这种情况的呀!
                mid = start + (end - start) >> 1;
            */
            mid = start + (end - start) / 2;
            if (nums[mid] > target) {
                end = mid - 1;
            } else if (nums[mid] < target) {
                start = mid + 1;
            } else {
                return mid;
            }
        }

        if (nums[start] >= target)
            return start;
        return start + 1;
    }
};

/*9
jump-game-ii
题目:
    给定一个非负整数数组，你最初位于数组的第一个位置。
```

```
    数组中的每个元素代表你在该位置可以跳跃的最大长度。
    你的目标是使用最少的跳跃次数到达数组的最后一个位置。
示例:
    输入: [2,3,1,1,4]
    输出: 2
    解释: 跳到最后一个位置的最小跳跃数是 2。
         从下标为 0 跳到下标为 1 的位置，跳 1 步，然后跳 3 步到达数组的最后一个位置。
说明:
    假设你总是可以到达数组的最后一个位置。
*/

/*
    贪婪算法(Greedy Algorithm), 时间复杂度 O(n)
*/
class Solution {
public:
    int jump(vector<int>& nums) {
        int step = 0, cur = 0, prev = 0;
        while (cur < nums.size()-1) {
            step++;
            int temp = cur;
            int index = 0;
            // 因为当前点可达，则它前方的点中上次的产生最大可达点之间都可达了
            // 故检查这段中的元素最大的可达距离
            for (int i=prev; i<=temp; i++) {
                if (cur < nums[i] + i) {
                    // 记录上次最大可达 与 当前点 之间的点中产生最大可达的点坐标
                    index = i;
                    cur = nums[i] + i;
                }
            }
            if (cur == temp)
                return -1;

            prev = index;
        }

        return step;
    }
};

/*10
* https://leetcode-cn.com/problems/maximum-subarray
* 题目描述:
*     给定一个整数数组 nums ，找到一个具有最大和的连续子数组(子数组最少包含一个元素)
*     返回其最大和
*
* 示例:
*     输入: [-2,1,-3,4,-1,2,1,-5,4],
*     输出: 6
*     解释: 连续子数组 [4,-1,2,1] 的和最大，为 6。
*
* 进阶:
*     如果你已经实现复杂度为 O(n) 的解法，尝试使用更为精妙的分治法求解。
*
* 解题思路:
*     1. 这里使用的方法不是分治法，分治法时间复杂度反而变高了
*     2. 直接一次性扫描数组即可，每当当前和变为负数时，就重新计数即可
*        同时记录最大值
*/
```

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int max = INT_MIN, sum = 0;
        for (int i=0; i<nums.size(); i++) {
            if (sum < 0) {
                sum = nums[i];
            } else {
                sum += nums[i];
            }

            if (sum > max)
                max = sum;
        }

        return max;
    }
};

/*11
jump-game
题目:
    给定一个非负整数数组，你最初位于数组的第一个位置。
    数组中的每个元素代表你在该位置可以跳跃的最大长度。
    判断你是否能够到达最后一个位置。
示例 1:
    输入: [2,3,1,1,4]
    输出: true
    解释: 从位置 0 到 1 跳 1 步, 然后跳 3 步到达最后一个位置。
示例 2:
    输入: [3,2,1,0,4]
    输出: false
    解释: 无论怎样，你总会到达索引为 3 的位置。但该位置的最大跳跃长度是 0,
          所以你永远不可能到达最后一个位置。
*/

/*
    动态规划(DP)方法，复杂度 O(n^2)
    解题思路:
        1. 该问题要解决的是最后一个元素是否可达，故需要判断最后一个元素
           前边可达的元素是否可达最后一个元素
        2. 由此问题变为从 第 1 个元素开始扫描，判断当前元素是否可达
           直到扫描完整个数组
*/
class Solution {
public:
    bool canJump(vector<int>& nums) {
        if (nums.size() <= 1)
            return true;

        bool can[nums.size()];
        can[0] = true;

        for (int i=1; i<nums.size(); i++) {
            for (int j=0; j<i; j++) {
                // 判断当前元素前方所有的元素是否有可达到当前元素的
                if (can[j] && j+nums[j]>=i) {
                    can[i] = true;
                    break;
                }
            }
```

```
        }

        return can[nums.size()-1];
    }
};

/*
    贪婪算法(Greedy Algorithm), 时间复杂度 O(n)
*/
class Solution {
public:
    bool canJump(vector<int>& nums) {
        if (nums.size() <= 1)
            return true;

        int reach = 0;
        for (int i=0; i<nums.size(); i++) {
            // 第一个条件表示当前点不可达，即前方元素的最远可达无法到达该元素
            // 第二个条件为最后一个元素可达
            if (i > reach || reach >= nums.size() - 1)
                break;

            reach = max(reach, i+nums[i]);
        }

        return reach >= nums.size() - 1;
    }
};
```

ll

```
# -*- coding:utf-8 -*-
class Solution:
    def minNumberInRotateArray(self, rArr):
        if not rArr:
            return 0
        low,hi = 0,len(rArr)-1
        while low<hi :
            mid = low + (hi-low)/2
            if rArr[mid]<rArr[hi]:
                hi = mid
            elif rArr[mid]>rArr[hi]:
                low = mid +1
            else:
                hi-=1
        return rArr[low]

        # write code here
```

ll