知识类问题

1．一个系统里面有多个线程，其中一个线程想终止另外一个线程，该怎么实现．

2．为什么要用 virtual destructor

3．什么是 heap corruption

4．Semaphore VS Mutex

5．C++中的 static，怎么用，内存在哪里分配，等等

6．sizeof 一个指针和 sizeof 一个指针指向的空间有什么不同？

7．char* 和 char[]有什么区别？

8．操作系统的 Page 是怎么做的？

9．什么是 Thrashing

10．什么是 volatileu 关键字？在什么情况下需要使用？http://drdobbs.com/cpp/184403766

11．什么是 critical section?

12．C++里面的 dynamic_cast 如何实现，为什么 dynamic_cast 要求被 cast 的指针指向的类至少要有一个 virtual method?

13．virtual inheritance 为什么能够避免导出类中有多于一个的 virtual base class 的 copy?

14．deadlock's four condition

15．关于 C++ 处理异常的方法

16．static 变量最大的问题是什么（考虑多线程）？

17．state Machine 的几种实现方法， 各自的优缺点是什么

18．多线程程序里，你最喜欢哪种同步操作？ Mutex 是大家用得最多的，为什么？Mutex 有什么不好？(overhead ,serialize ,deadlock)，给例子。如何避免这些坏处？(lock-free 算法，比如 lock-free 的 queue，stack，hashtable) 一般用什么办法来写 lock-free 算法？(用 CAS 等原子操作，等等)，如何实现一个 lock-free 的 queue? (coding ?)

19．strlen(“hello”)=? ， sizeof(“hello”)=?

20. 如果 main 函数只有如下结构

```
int main(…)
{
   Try{…}
   catch(){…}
}
```

并且这个 catch 语句 catch 了所有的 exception，但是这个程序还是 coredump 了，问可能是什么问题引起的。

21. singleton，多线程，如何做 double check?

22. Command Pattern

23. 同步/异步 IO 和阻塞/非阻塞 IO 的区别？

24. 写一个 thread safe lazy initialization 的 singleton

25. Map Reduce 和 Multi-thread 的 trade-off

编程，数据结构，算法

1. 在 linked list 中找倒数第 N 个结点

```
Node* get_last_nth(Node* head, int n)
{
     if (n < 1) return NULL;

     Node* p1 = head;
     Node* p2 = head;

     while (p1 && n-- > 0) p1 = p1->next;

     if (n > 0) return NULL;

     while (p1)
     {
          p1 = p1->next;
          p2 = p2->next;
     }

     return p2;
}
```

2. 倒转 linked list

```
void reverse_list(Node*& head)
{
     Node* remain = head->next;

     head->next = NULL;

     while (remain)
     {
          Node* tmp = remain;
          remain = remain->next;

          tmp->next = head;
          head = tmp;
     }
}
```

3．二叉树的结点有指向 parent 的指针，求最近公共祖先

```
TreeNode* search_tree_node_with_ancestors(TreeNode* root, int value, vector<TreeNode*>& ancestors)
{
     if (root == NULL) return NULL;

     if (root->value == value)
     {
          ancestors.push_back(root);
          return root;
     }
     else
     {
          ancestors.push_back(root);
          TreeNode* tn = search_tree_node_with_ancestors(root->left, value, ancestors);
          if (tn != NULL)    return tn;

          tn = search_tree_node_with_ancestors(root->right, value, ancestors);
          if (tn != NULL) return tn;

          ancestors.pop_back();
          return NULL;
     }
}

TreeNode* find_lowest_common_ancestor(TreeNode* root, int val1, int val2)
{
     vector<TreeNode*> ancestor1;
     TreeNode* n1 = search_tree_node_with_ancestors(root, val1, ancestor1);

     vector<TreeNode*> ancestor2;
     TreeNode* n2 = search_tree_node_with_ancestors(root, val2, ancestor2);

     int minLen = min(ancestor1.size(), ancestor2.size());

     for (int i = 1; i < minLen; ++i)
     {
          if (ancestor1[i] != ancestor2[i])
               return ancestor1[i-1];
     }

     return ancestor1.size() > ancestor2.size() ? ancestor2[minLen - 1] : ancestor2[minLen - 1];
}

int _tmain(int argc, _TCHAR* argv[])
{
     int in_order[] = {6,2,7,1,5,3,4,8,9,10};
     int pre_order[]  = {5,2,6,1,7,4,3,9,8,10};
     TreeNode* tree = create_tree(pre_order, in_order, 10);

     TreeNode *tn = find_lowest_common_ancestor(tree, 7, 8);
     tn = find_lowest_common_ancestor(tree, 7, 6);
     tn = find_lowest_common_ancestor(tree, 7, 1);
     tn = find_lowest_common_ancestor(tree, 4, 8);
     tn = find_lowest_common_ancestor(tree, 3, 10);

     return 0;
}
```

4．给一个数组，如何打印该数组成员构成集合的全部子集合．

```
void _print_all_subset(const vector<int>& col, int i, vector<int>& output)
{
     if (i >= col.size())
          print_vector(output);
     else
     {
          output.push_back(col[i]);
          _print_all_subset(col, i+1, output);
          output.pop_back();

          _print_all_subset(col, i+1, output);
     }
}

void print_all_subset(const vector<int>& col)
{
     vector<int> output;
     _print_all_subset(col, 0, output);
}
```

5．有两个字符串，一个是 text,一个是 command，Command 有四种：

'+'：在 text 中前进一位

'-'：在 text 中后退一位

'a'：在当前位置插入一个字符，字符由 command 中的后一位决定

'd'：删除当前字符

实现函数 Process(string& text, string& command, string& result);

Coding 题，大致要点：

1. 扫描一遍 command,看看有多少加字符的 command,再建一个满足大小要求的临时数组，copy text
2. 在临时数组上进行操作，注意插入和删除的复杂度都是 O(N)

6．实现一个 LRU 的 cache

数据结构：

```
/// Typedef for URL/Entry pair
typedef std::pair< std::string, Entry > EntryPair;

/// Typedef for Cache list
typedef std::list< EntryPair > CacheList;

/// Typedef for URL-indexed map into the CacheList
typedef boost::unordered_map< std::string, CacheList::iterator > CacheMap;

/// Cache LRU list
CacheList mCacheList;

/// Cache map into the list
CacheMap mCacheMap;
```

插入新 cache 的算法：

3.

```
    // create new entry
    Entry iEntry( ... );

    // push it to the front;
    mCacheList.push_front( std::make_pair( aURL, iEntry ) );

    // add it to the cache map
    mCacheMap[ aURL ] = mCacheList.begin();

    // increase count of entries
    mEntries++;

    // check if it's time to remove the last element
    if ( mEntries > mMaxEntries )
    {
        // erease from the map the last cache list element
        mCacheMap.erase( mCacheList.back().first );

        // erase it from the list
        mCacheList.pop_back();

        // decrease count
        mEntries--;
    }
```

如果找到了，用 splice 函数将刚刚被访问的 CacheEntry 移到队首。

关于多线程，一般来说 reader/writer lock 不适用，因为 reader 也会更改 LRU cache. 一种解决的办法是让每个线程拥有自己的 cache.

更多内容参考 http://www.bottlenose.demon.co.uk/article/lru.htm

7．两个排序的数组，求它们的交集

```
vector<int> find_intersect(const vector<int>& v1, const vector<int>& v2)
{
    int i = 0;
    int j = 0;
    vector<int> ret;
    while (i < v1.size() && j < v2.size())
    {
        if (v1[i] == v2[j])
        {
            ret.push_back(v1[i]);
            ++i;++j;
        }
        else if (v1[i] > v2[j]) ++j;
        else ++i;
    }

    return ret;
}
```

8．在二叉树中添加额外的两个指针（树可能非满），遍历整棵树并将同一层的结点用这两个额外指针连接起来

```
void link_same_level(TreeNode* root)
{
    queue<TreeNode*> q1;
    q1.push(root);
    while (!q1.empty())
    {
        queue<TreeNode*> q2;
        TreeNode* prev = NULL;
        TreeNode* cur = q1.front();
        q1.pop();
        cur->ll = NULL;

        if (cur->left) q2.push(cur->left);
        if (cur->right) q2.push(cur->right);

        while (!q1.empty())
        {
            prev = cur;
            cur = q1.front();
            q1.pop();

            if (cur->left) q2.push(cur->left);
            if (cur->right) q2.push(cur->right);

            cur->ll = prev;
            prev->rr = cur;
        }
        cur->rr = NULL;

        q1 = q2;
    }
}
```

9．用一个给定的值 partition 一个数组，注意这个值不一定在数组中出现

```
void partition(int v[], int n, int key)
{
    int i = 0;
    int j = n - 1;

    while (j > i)
    {
        if (v[i] >= key)
        {
            swap(v[i], v[j]);
            --j;
        }
        else
            ++i;

        if (v[j] < key)
        {
            swap(v[i],v[j]);
            ++i;
        }
        else
            --j;
    }
}
```

10. 用数组实现一个 queue，考虑以下一些内容：

a) **实现固定 size**

b) **实现可变 size** 每次 size 不够用时，建一个更大的 array 并复制原有数据

c) **与 linked list 的实现相比，有什么好处和坏处？**保证了操作恒定为 O(1),但是内存有浪费，且不连续

d) **如何处理 thread safe**

1. 在 queue 被更改的情况下，使用 locker
2. Lock-free code，见：
   a) http://drdobbs.com/high-performance-computing/208801974
   b) http://drdobbs.com/high-performance-computing/210604448

11. 洗牌算法

For i = 0 to n-1,

生成一个 i 到 n-1 之间的随机数 j，将 v[i]与 v[j]交换

12. [Microsoft] 对 stack 上的元素排序，可以使用的方法有 pop()，top()，push()，isEmpty()，isFull().

```
void sort_stack(stack<int>& s)
{
    stack<int> dest;

    while(!s.empty())
    {
        if (dest.empty() || s.top() >= dest.top())
        {
            dest.push(s.top());
            s.pop();
        }
        else
        {
            stack<int> tmp;
            while (!dest.empty() && dest.top() > s.top())
            {
                tmp.push(dest.top());
                dest.pop();
            }
            dest.push(s.top());
            s.pop();
            while (!tmp.empty())
            {
                dest.push(tmp.top());
                tmp.pop();
            }
        }
    }

    s = dest;
}
```

13. [Microsoft] 有一个 M*N 行的矩阵，如果第(i，j)个元素是 0，则把 i 行和 j 列都设为零，注意尽量少使用额外空间

分成如下几步：

1. 扫描第 M 行和第 N 列，看(M,N)是否需要设为零

2. 扫描每行和每列，在第 M 行和第 N 列记录对应的列和行的结果
3. 扫描第 M 行和第 N 列，将其所对应的列和行记为零
4. 处理(M,N)

14. [Microsoft] 一个二维空间第一象限有很多点，怎么找出最外围的那些点？

Graham 扫描算法：

1. 选出 y 最小的起始点 p0
2. 将其它所有点按相对于 p0 的极角排序，记为 p1,p2,…pN-1
3. 将 p0, p1, p2 push 到栈
4. 对余下的所有点：
   a) Px 为栈顶的下一个点，Py 为栈顶，当前点为 Pi
   b) 如果 Py->Pi，相对于 Px->Py 向右
      i. Pop 栈
      ii. Push Pi 到栈

算法复杂为 O(NlogN)（第二步的排序）

15. [Google] 返回一组字符串的最长公共前缀，如“abc”，“abcdef”，“abcd”，则返回”abc”

```
string common_prefix(vector<string>& vs)
{
    string ret;

    int i = 0;
    while (1)
    {
        if (i >= vs[0].size()) break;
        char c = vs[0][i];
        for (int j = 1; j < vs.size(); ++j)
        {
            if (vs[j].size() <= i) break;
            if (vs[j][i] != c) break;
        }
        ++i;
        ret.push_back(c);
    }

    return ret;
}
```

16. [Microsft] 给出平面上第一象限内 landscape 的轮廓，也就是一些列的(x,y)坐标，x=0,1,...,N，以及 Y 轴上光源坐标(0,H)。问这 N+1 个点钟那些被照亮那些是阴影。（叉乘）

——计算光源到(x,y)的角度，再与左边的角度对比即可知是否被遮挡，复杂度 O(N)

17. [Microsoft] 一个 linked list，每个节点除了正常 next 指针外，还有一个 extra 指针，这个指针可以指向链表中的任一节点，不同的 extra 指针可以指向同一个节点，extra 指针也可能形成 loop。问怎么复制这个结构。

```
ListNode* copy_ll_with_extra_pointer(ListNode* list)
{
    // 1. duplicate each node
    ListNode* n = list;
    while (n)
    {
        ListNode* tmp = new ListNode(n->value);
        tmp->next = n->next;
        n->next = tmp;

        n = tmp->next;
    }
    // 2. duplicate extra pointer
    n = list;
    while (n)
    {
        n->next->extra = n->extra->next;
        n = n->next->next;
    }

    // 3. split duplicated list
    ListNode* n1 = list;
    ListNode* ret = list->next;
    ListNode* n2 = list->next;
    while (n2->next)
    {
        n1->next = n2->next;
        n2->next = n1->next->next;

        n2 = n2->next;
    }

    return ret;
}
```

18. [Microsoft] 怎么组织字典，使得在解 cross puzzle 时可以很快得到满足条件的所有单词(比如所有第二个字母是 o，第 5 个是 H 的单词）。不过这题算 brain storm，不用写 code.

按单词的长度不同，构造多个 container

对某一组长度相同的单词，构造多个 index，从(2,o)，(5，H) 映射到单词（id），每一个 collection 保持有序，可以加快 merge 的速度

19. [Google] 如何设计 binary tree 和 hash table 的 iterator

Binary Tree Iterator:

假设是中序遍历的话，在 iterator 中保存一个遍历的状态(parent node stack).

Hash Table Iterator:

取决于 hash table 的数据结构，一般直接按 array 或者 bucket 顺序遍历就可以了。

20. [Google] 设计一个 class，类似于 stack，但可以是 O(1)时间内返回 min()

给 stack 加一个只用来保存当前最小值的 stack，Push 时，如果当前值比 minStack 栈顶小，则也 push 到 MinStack，Pop 时，如果 minStack 栈顶与当前 pop 元素一样大，则也 pop minStack

21. [Google] 比较两个 binary tree 是不是完全一致

递归比较 if (tree1->value == tree2->value) && is_equal(tree1->left, tree2->left) &&

```
    is_equal(tree1->right, tree2->right)
```

22. [Google] 一个整数数组里怎么同时找最大和最小的数，尽量优化比较次数

考虑二个数 a,b， a 与 b 先比，大的与当前最大比，小的与当前最小比。两个数共需要比较三次。

23. [Google] 在一个循环有序的数组里查找一个数

```
int rotated_search(int v[], int b, int e, int key)
{
    int m = (b+e)/2;

    if (v[m] == key) return m;

    if (b >= e) return -1;

    if (v[m] >= v[b])
    {
        if (v[m] > key && v[b] <= key)
            return rotated_search(v, b, m-1, key);
        else
            return rotated_search(v, m+1, e, key);
    }
    else
    {
        if (v[m] < key && key <= v[e])
            return rotated_search(v, m+1, e, key);
        else
            return rotated_search(v, b, m-1, key);
    }
}
```

24. [Google] 给一个 array 和一个 target value，检查 array 里是否存在两个数之和为 target

两种做法：

1. 先对数组排序，然后从两头开始 scan
2. 建一个 hash table，然后 scan 数组，去查找，注意要处理正好有一个数等于 target 的一半的情况

25. [Google] 给一个文本， 然后给出几个关键词及他们所出现的位置，比如

this: 1, 16, 55....

is: 5, 33, 77...

要求找出最短的一段文章使其具备给出的关键词。

大致算法：按位置往后找，直到所有的词都出现，然后再尝试把左边的位置缩减。如此直到找到更短的区间。

见后面的 find_min_window 的程序，这里需要处理 inverted index

26. [Google] 给出一棵 tree，该 tree 没有任何特征， 即可以有多个子节点， 父节点和左右子节点也没有大小关系。但每个节点的值不相等。现给出几个值， 如(12， 24）请找出从根节点到值为 12 和 24 的节点的 subtree.

```
TreeNode* get_sub_tree(TreeNode* root, set<int>& nums)
{
    if (root == NULL)
        return NULL;

    TreeNode* left = get_sub_tree(root->left, nums);
    TreeNode* right = get_sub_tree(root->right, nums);

    if (left || right || nums.find(root->value) != nums.end())
    {
        TreeNode* node = new TreeNode(root->value);
        node->left = left;
        node->right = right;

        return node;
    }
    else
        return NULL;
}
```

27.[Google] 给一个 array，再给一个 sh 值， 设计函数将数组内的所有元素向右偏移 sh 个位置(将数组看成一个圈)。

见 Programming Pearls，先把[a,c] reverse，再 reverse[a,b],[b,c]

28.[Microsoft] 删除数组中的重复元素

略。。。

29.[Microsoft] 按如下规则转化数字的字符串

(integers that appear >=1 times)

(integers that appear >=2 times)

...

(integers that appear >=n times)

并保持字符原来的顺序

例如：12223314->12342312

略。。。

30.[Microsoft] 检查一个表达式中的括号是否合法，括号包括 {，[，(，)，]，}

简单的栈的应用

31.[Microsoft] 如何高效地用堆栈模拟队列.

使用两个 stack，s1 和 s2

Push 时，push 到 s1

Pop 时，若 s2 非空，则从 s2 中 pop，若 s2 为空，则将 s1 的全部元素 pop 到 s2 中，再从 s2 中 pop

分摊复杂度为 O(1)

32.[Microsoft] 打印中两个整数范围内的所有素数，例如：(12，15) ->13

1．单个验证是否为素数

2．筛法

33. [Google] 求直方图的最大内接矩形

TODO

34. [Google] NxN 行列有序的矩阵查找一个数

两种方法，1 从角上开始 search， 2， Divide and Conquer

```
bool search_2d_sorted_matrix_1(const vector<vector<int>>& mat, int target, int
&targetRow, int &targetCol)
{
    int row = 0;
    int col = mat[0].size() - 1;

    while (row < mat.size() && col >= 0)
    {
        if (mat[row][col] == target)
        {
            targetRow = row;targetCol = col;
            return true;
        }
        else if (mat[row][col] > target)
            col--;
        else if (mat[row][col] < target)
            row++;
    }
    return false;
}


bool search_2d_sorted_matrix_2(const vector<vector<int>>& mat, int target, int l, int
u, int r, int d, int &targetRow, int& targetCol)
{
    if (l > r || u > d) return false;
    if (target < mat[u][l] || target > mat[d][r]) return false;

    int mid = l + (r-l)/2;
    int row = u;
    while (row <= d && mat[row][mid] <= target)
    {
        if (mat[row][mid] == target)
        {
            targetRow = row;
            targetCol = mid;
            return true;
        }
        ++row;
    }

    return search_2d_sorted_matrix_2(mat, target, l, row, mid, d, targetRow, targetCol)
||
           search_2d_sorted_matrix_2(mat, target, mid, u, r, row, targetRow,
targetCol);
}
```

分治法可以用来解决另外一个问题，在行列有序的二维数组中，大于/小于 0 的元素有多少个？

```
int count_negative_in_sorted_matrix(const vector<vector<int>>& mat, int l, int u, int
r, int d)
{
    if (l > r || u > d) return 0;

    if (mat[u][l] >= 0) return 0;
    if (mat[d][r] < 0) return (r - l + 1) * (d - u + 1);

    int mid = l + (r-l)/2;
    int row = u;
    while (row <= d && mat[row][mid] < 0)
    {
        ++row;
    }

    return count_negative_in_sorted_matrix(mat, l, row, mid - 1, d) +
           count_negative_in_sorted_matrix(mat, mid + 1, u, r, row - 1) +
           (mid - l + 1) * (row -u);
}
```

35. [Google] 将 MxN 的矩阵转秩,要求 O(1)的空间复杂度.参考群论中 cyclic group,group generator

TO LEARN

36. Inplace perfect shuffle

TO LEARN

37. 有一个矩阵 A，找出这个矩阵中所有的 A(i,j)，它所在的行和列都是 0.

依次扫描？略。。。

38. 有一个变长的 characters system，每个 character 所占的 bytes 数不固定。每个 character 的最后一个 byte 的值是 0. 一个字符串由这些变长的 characters 组成。字符串的最后两个 bytes 是 0. 要求反转这个字符串。额外空间使用越少越好。

先将整个字符串反转，再按个字符反转

39. 有 n 张扑克牌，从中随机选出几张。要求找出所有的选法，使得所选扑克牌的点数的和是 s. 不用 recursion,代码行数越少越好。

TODO 如何不用 recursion?

40. [Google] 有 1G 内存和 4 billion 个整数,输出一个不在这些整数内的数， 如果只有 10MB 内存呢？

1. 1G 内存有 1024*1024*1024*8 个位，扫描一遍记位即可
2. 如果内存不够，可以依次处理 10*1024*1024*8 个数，需要扫描 4*1024/10/8 大概 50 趟

41. [Google] Merge N 个 sorted array

```
// assume the input has the following property:
// v[i] <= v[i+N]
// which is a intermediant state of shell sort
void N_way_merge(vector<int>& v, int N)
{
    priority_queue<OneQueue> pq;
    vector<int> ret;

    int* indices = new int[N];
    for (int i = 0; i < N; ++i)
    {
        pq.push(OneQueue(v[i], i));
        indices[i] = i;
    }

    while (!pq.empty())
    {
        OneQueue nextQ = pq.top();
        pq.pop();

        ret.push_back(nextQ._top);

        indices[nextQ._id] += N;

        if (indices[nextQ._id] < v.size())
        {
            pq.push(OneQueue(v[indices[nextQ._id]], nextQ._id));
        }
    }
    // copy back
    v = ret;
    delete[] indices;
}
```

42. [Google] 给一个大小为 N 的数组，输出所有大小为 K 的子集（K<=N）

TODO

43. [Microsoft] 已知 bst 和两个数，求在此范围内的节点数。递归和非递归

中序遍历，加一个 collection 作为参数即可

44. [Microsoft] 已知 bst 和一个数，找到 next larger number，O(logn)时间

TO CODE

45. N*N 的正方形内有黑白两色，求四边都是黑色的最大的子正方形

TO LEARN

46. 平面上有一组点集，求穿过点最多的直线

计算两两点够成的直线方程 x+By+C=0， 找出出现次数最多的方程即可

47. 实现 itoa

```
void _itoa(int n, char s[])
 {
     int i, sign;

     if ((sign = n) < 0)  /* record sign */
         n = -n;          /* make n positive */
     i = 0;
     do{                       /* generate digits in reverse order */
         s[i++] = n % 10 + '0';   /* get next digit */
     } while ((n /= 10) > 0);     /* delete it */
     if (sign < 0)
         s[i++] = '-';
     s[i] = '\0';
     reverse(s, s + i);
 }
```

48. 给一个 2D 的 matrix，按 spiral order 打印

```
void _print_matrix_spiral(int m[][5], int L, int R, int U, int D)
{
    if (L == R && U == D) {cout << m[L][U] << ' '; return;}
    if (L == R){ for (int i = U; i <= D; ++i) cout << m[i][L] << ' '; return;  }
    if (U == D){ for (int i = L; i <= R; ++i) cout << m[L][i] << ' '; return;}
    for (int i = L; i < R; ++i) cout << m[U][i] << ' ';
    for (int i = U; i < D; ++i) cout << m[i][R] << ' ';
    for (int i = R; i > L; --i) cout << m[D][i] << ' ';
    for (int i = D; i > U; --i) cout << m[i][L] << ' ';
}

void print_matrix_spiral(int m[][5], int N)
{
    int L = 0, U = 0, R = N - 1, D = N - 1;
    while (L <= R && U <= D)
        _print_matrix_spiral(m, L++, R--, U++, D--);
}
```

49. [Google] 一个字符串，复制所有的'x'，再删除所有的'z'，其它不变

略。。。

50. [Google] 实现 memcpy(void *src, int size, void *dest);

判断参数合法性，判断 src,dest 是否相等， 是否 overlapping 时会出错，

注意 memcpy 和 memmove 的区别

```
void *(_memcpy)(void * s1, const void * s2, size_t n)
 {
     char *dst = (char*)s1;
     const char *src = (const char*)s2;
     /* Loop and copy.  */
     while (n-- != 0)
         *dst++ = *src++;
     return s1;
 }

void *(_memmove)(void *s1, const void *s2, size_t n)
 {
    /* note: these don't have to point to unsigned chars */
    char *p1 = (char*)s1;
    const char *p2 = (const char*)s2;
    /* test for overlap that prevents an ascending copy */
    if (p2 < p1 && p1 < p2 + n) {
        /* do a descending copy */
        p2 += n;
        p1 += n;
        while (n-- != 0)
            *--p1 = *--p2;
    } else
        while (n-- != 0)
            *p1++ = *p2++;
    return s1;
 }
```

51. [Google] 大小为 N 的数组，给出一个大小为 K 的随机子集

作一个 shuffle,然后选取前 K 个（因此 shuffle 时只要进行到第 K 个即可）

52. [Microsoft]判断这四个点是不是构成了一个矩形

TODO

53. 实现 char* strtok(char* str, const char* delimeter)

略。。。应该需要用到 static 变量

54. 美国的 coin 设计为 1，5，10，25，任意给定一个 change，用 greedy algorithm 可以算出最少所需要的 coins，如何判断一组 coin 是否可以用 greedy algorithm?

TO LEARN

找硬币的 DP 程序:

```
vector<int> get_coins(set<int>& coins, int sum)
{
    vector<int> coin_num(sum+1);
    vector<int> last_coin(sum+1);

    for (int i = 0; i <= sum; ++i) coin_num[i] = INT_MAX;

    coin_num[0] = 0;
    last_coin[0] = 1;

    for (int i = 1; i <= sum; ++i)
    {
        for (int j = 0; j < i; ++j)
        {
            if (coins.find(i - j) != coins.end())
            {
                if (coin_num[j] + 1 < coin_num[i])
                {
                    coin_num[i] = coin_num[j] + 1;
                    last_coin[i] = (i - j);
                }
            }
        }
    }

    vector<int> ret;
    while (sum > 0)
    {
        ret.push_back(last_coin[sum]);
        sum -= last_coin[sum];
    }

    return ret;
}
```

55．给定 5 张牌，写一个函数判断是否有两对

略。。。

56．在 BST 中删除一个节点

```
TreeNode* RotateRight(TreeNode* h)
{
    TreeNode* x = h->left;
    h->left = x->right;
    x->right = h;

    return x;
}

TreeNode* RotateLeft(TreeNode* h)
{
    TreeNode* x = h->right;
    h->right = x->left;
    x->left = h;

    return x;
}

TreeNode* MoveUpSmallest(TreeNode* root)
{
    if (root == NULL) return NULL;

    root->left = MoveUpSmallest(root->left);
    if (root->left != NULL)
        root = RotateRight(root);

    return root;
}

TreeNode* JoinBST(TreeNode* a, TreeNode* b)
{
    if (b == NULL) return a;

    b = MoveUpSmallest(b);
    b->left = a;
    return b;
}

TreeNode* DeleteBST(TreeNode* root, int key)
{
    if (root == NULL)
        return NULL;
    if (root->value > key)
        root->left = DeleteBST(root->left, key);
    else if (root->value < key)
        root->right = DeleteBST(root->right, key);
    else if (root->value == key)
    {
        TreeNode* x = root;
        root = JoinBST(root->left, root->right);
        delete x;
    }

    return root;
}
```

57. [Microsoft] 给你一个地址/a/b/../c/./d.txt，让你把它 normalize。

```
void normalize_path(string& path)
{
    stack<string> s;
    const char* b = path.c_str();
    const char* e = b + path.size();

    const char* p1 = b;
    const char* p2 = p1 + 1;
    while (p2 < e)
    {
        while (p2 < e && *p2 != '/') ++p2;
        string str(p1+1, p2);
        p1 = p2;p2 = p1+1;
        if (str == "..") {s.pop();continue;}
        else if (str == ".") continue;
        else
            s.push(str);
    }
    s.push(string(p1+1,p2));

    // concatenate strings in stack...
}
```

58. [Microsoft] 给出一个 BST 和一个值，求所有和等于这个值的 Path.

```
void find_sum(TreeNode* root, int sum, vector<int>& nums)
{
    int tmp = sum;
    for (int i = nums.size() - 1; i >= 0; --i)
    {
        tmp -= nums[i];
        if (tmp == 0)
            cout << "found" << endl;
    }
    if (root->left)
    {
        nums.push_back(root->left->value);
        find_sum(root->left, sum, nums);
        nums.pop_back();
    }
    if (root->right)
    {
        nums.push_back(root->right->value);
        find_sum(root->right, sum, nums);
        nums.pop_back();
    }
}
```

59. [Microsoft] 按层打印树

略

60. [Google] 五个非常大的数组求交集（考虑时间，空间，I/O）

略

61. [Google] 两个排序好的数组，求和最小的 M 个 pair，比如 A={1，2，4，5，6}，B={3，5，7，9} m=3 那么 Results 就是(1，3),(2，3),(1，5)

这个结构形成了一个行列有序的矩阵，这个题就类似于在行列有序的矩阵中找第 k 个元素。

结论是：

对于一个 n×m(n≤m)的矩阵，若每行和每列都是递增的，则可以在 O(nlog2m/n)找到第 k 大的数。论文题目为“Generalized Selection and Ranking: Sorted Matrices”

如果是查中位数：

先找出每一行（列）的中位数，再找出中位数的中位数，这样可以去掉接近一半的数，再在剩下的数里找中位数即可，复杂度 O(N(logN)^2)

62. [Microsoft] 一个数组，有大小写字母和数字，一次编历，大写字母在一起， 小写字母在一起，数字在一起.

见 138. Dutch National Flag Problem (DNFP)

63. [Google] 1. 给定一个树和任意 2 个 nodes，找出来最小的 subtree 包含这 2 个 nodes 第 26 题

2. 写 BFS 算法打印 subtree 的 nodes 等价于分层访问树

3. 如果把树变成了 graph（可能有 loop），怎么改刚写的 BFS 需要标记已经访问过的点

64. 实现 next_permutation

算法：

1. 从后往前找，找到第一个 x1,使 a[x1]<a[x1+1]
2. 从后往前找，找到第一个 x2,使 a[x2]>a[x1]
3. 交换 a[x1],a[x2]，并且反置 a[x1+1 … end]

```
bool _next_permutation(int* begin, int* end)
{
    if (begin == end) return false;

    int* x1 = end - 1;
    while (x1 >= begin && *x1 >= *(x1+1)) --x1;

    if (x1 == begin)
    {
        reverse(begin, end+1);
        return false;
    }

    int* x2 = end;
    while (*x2 < *x1) --x2;

    swap(*x1, *x2);
    reverse(x1+1, end+1);
    return true;
}
```

65. 最长上升子序列

```cpp
// O(N^2)
void longest_increasing_subsequence(const vector<int>& v)
{
    vector<int> counts;
    counts.push_back(1);

    vector<int> prev;
    prev.push_back(0);

    int all_max = -1;
    int last_index = -1;

    for (int i = 1; i < v.size(); ++i)
    {
        int max = 1;
        int index = i;
        for (int j = 0; j < i; ++j)
        {
            if (v[j] <= v[i] && counts[j] + 1 > max)
            {
                max = counts[j] + 1;
                index = j;
            }
        }

        counts.push_back(max);
        prev.push_back(index);

        if (max > all_max)
        {
            all_max = max;
            last_index = i;
        }
    }

    while (true)
    {
        cout << v[last_index] << ' ';
        if (prev[last_index] == last_index)
            break;
        last_index = prev[last_index];
    }

    cout << endl << all_max << endl;
}
```

```
// O(NlogN)
void longest_increasing_subsequence2(vector<int> &a, vector<int> &b)
{
    vector<int> p(a.size());
    int u, v;

    if (a.empty()) return;

    b.push_back(0);

    for (size_t i = 1; i < a.size(); i++)
    {
       // If next element a[i] is greater than last element of current longest
subsequence a[b.back()], just push it at back of "b" and continue
        if (a[b.back()] < a[i])
       {
            p[i] = b.back();
            b.push_back(i);
            continue;
        }

       // Binary search to find the smallest element referenced by b which is just bigger
than a[i]
       // Note : Binary search is performed on b (and not a). Size of b is always <=k
and hence contributes O(log k) to complexity.
        for (u = 0, v = b.size()-1; u < v;)
       {
            int c = (u + v) / 2;
            if (a[b[c]] < a[i])
                u = c + 1;
            else
                v = c;
        }

       // Update b if new value is smaller then previously referenced value
        if (a[i] < a[b[u]])
       {
            if (u > 0)
                p[i] = b[u-1];
            b[u] = i;
        }
    }

    for (u = b.size(), v = b.back(); u--; v = p[v]) b[u] = v;
}
```

66. 给一个 single linked list， 里面有 true 和 false 两类的 node，写一个程序把 true node 和 false node 分类，并且中间生成一个新的 node 把两类分开。

将 true 和 false 两类 node 接到两个不同的 list,最后再合并即可

67. 1 billion query 里选出时间最近 5 分钟内最 frequent 的 1000 个 query，one pass

精确解：选出所有 5 分钟内的 query,count 每个 query 的个数，同时维护一个最大堆，最后得到查询

扩展：如果 query 的数量非常大，则可以在一个个 time windows 里面做一些 sample，最后把这五分钟内的 sample 结果合并起来

68. 两个排序数组找共同中值。

```
int find_kth_in_combined_array(int v1[], int n1, int v2[], int n2, int k)
{
    if (k == 1) return min(v1[0], v2[0]);
    if (k == n1 + n2) return max(v1[n1-1],v2[n2-1]);

    int i1 = k * (double)n1 / (double)(n1 + n2);
    if (i1 > n1 - 1) i1 = n1 - 1;
    int i2 = k - i1;

    if (v1[i1-1] > v2[i2-1] && (i2 == n2 - 1 || v2[i2] > v1[i1-1])) return v1[i1-1];
    if (v2[i2-1] > v1[i1-1] && (i1 == n1 - 1 || v1[i1] > v2[i2-1])) return v2[i2-1];

    if (v1[i1-1] > v2[i2-1])
        return find_kth_in_combined_array(v1, n1, v2 + i2, n2 - i2, k - i2);
    else
        return find_kth_in_combined_array(v1 + i1, n1 - i1, v2, n2, k - i1);
}
```

69. 实现 strstr(str1, str2)

```
char* StrStr(const char* str, const char* target)
{
    if (!*target) return (char*)str;

    while (*str)
    {
        const char* p = str;
        const char* q = target;
        while (*q && *p++ == *q++);
        if (!*q) return (char*)str;

        ++str;
    }

    return NULL;
}
```

70. 给定一个排好序的 linked list，删除其中所有的重复元素。比如给定 1->2->3->3->4->4->5，返回 1->2->5。给定 1->1->1->2->3，返回 2->3

```
void remove_dup(ListNode*& list)
{
    ListNode* dummy = new ListNode(0);
    dummy->next = list;

    set<int> values;
    ListNode *n = dummy;
    while (n->next != NULL)
    {
        if (values.find(n->next->value) != values.end())
        {
            ListNode* tmp = n->next;
            n->next = tmp->next;
            delete tmp;
        }
        else
        {
            values.insert(n->next->value);
            n = n->next;
        }
    }
    list = dummy->next;
    delete dummy;
}
```

71. 返回给定字符串的第一个不符合字母顺序的 index，比如 abcdaf 就需要返回第二个 a 的 index，比如 aZe 就返回 e 的 index

依次扫描即可。。。

72. 检查 sudoku 的输入是 valid，允许 solution 是不完全的

略

73. 实现 wildcast string matching.

```
bool wildchar_match(const char *str, const char *pattern)
{
    if (*str == NULL)
        return (*pattern == NULL || (*pattern == '*' && *(pattern+1) == NULL)) ? true :
false;

    if (*pattern != '*')
       return (*pattern == '.' || *pattern == *str) ? wildchar_match(str+1, pattern+1) :
false;
    else
        return wildchar_match(str+1, pattern) ? true : wildchar_match(str, pattern+1);

}
```

74. 给你一本 dictionary，任意给你七个 letters，让你找出包含这七个字母的、最长的单词。条件：可以 pre-processing，这样每次给你不同的 letters 时，可以 very efficient

将单词按长度从长到短编号

对每一个字母，建一个 collection，按编号排序。

对给出的七个字母，找到它们的 collection 中的第一个共同的字母。

75. 表达式求值

```
double eval_expr(const string& expr)
{
    stack<double> nums;
    stack<char> ops;

    string::const_iterator i = expr.begin();

    for (; i != expr.end(); ++i)
    {
        if ('0' <= *i && *i <= '9')
            nums.push((double)(*i-'0'));
        else
        {
            char op = *i;
            if (op == ')')
            {
                while (ops.top() != '(')
                    calc_one(nums,ops);
                ops.pop();
            }
            else if (op == '(')
            {
                ops.push(op);
            }
            else
            {
                while (!ops.empty() && get_level(op) <= get_level(ops.top()))
                    calc_one(nums, ops);

                ops.push(op);
            }
        }
    }

    while (!ops.empty())
        calc_one(nums, ops);

    return nums.top();
}
```

76. 两个 string，给出它们的两个 substring，定义它们的距离为 distance=\sum_i|s1[i]-s2[i]|，怎么找距离最大的两个 substring?

穷举... 有没有更好的办法？

77 N*N 的 0/1 矩阵，找出最大的全 0 矩阵

最大直方图的应用， O(N^3)时间复杂度

78. 将一个 linked list 按不同元素的值分组

用多个 head 放不同元素的组，最后合并

79. serialize and re-construct binary tree.

按 pre-order 遍历，写入结点，包括 NULL 的子结点

Re-construct 的时候，读入结点，构建 node,再递归构建 child,(当该 Node 不为空)

80. 手机上的电话号码提示，用 prefix tree

CODE prefix tree

81. [Facebook] 给定一个数组，删除最少的元素使得剩下的元素有序

等价于找最长上升（下降）子序列，见 65 题

82. [Facebook] BST 中找中间节点

中序遍历一遍放到数组中，最后拿中间数

83. [Facebook] implement char *remove_badchars(char string[], char bad_chars[]) in place。

将 bad_chars 放到 hash 中查询，用两个指针来 remove bad char，code 略。。。

84. [Facebook] implement adding two unsigned numbers without using "+"

```
int add_no_arithm(int a, int b) {
    if (b == 0) return a;
    int sum = a ^ b; // add without carrying
    int carry = (a & b) << 1; // carry, but don’t add
    return add_no_arithm(sum, carry); // recurse
}
```

85. [Facebook] How to implement a smart_pointer

TO LEARN

86. [Facebook] implement sqrt

```
double _sqrt(double s)
{
    double b,e;
    if ( s >= 1.0){b = 1.0;e=s;}
    else{b=0.0;s=1.0;}

    for (int i = 0; i < 32; ++i)
    {
        double m = (b+e)/2;
        if (m*m > s) e = m; else b = m;
    }

    return b;
}
```

87. [Facebook] implement reader/writer lock

TO LEARN

88. given a word a and a word b, all are 6-letter. Also given a dictionary. Find a transformation from a to b, such that: (a) each time you can changeone letter; (b) all the changed word should appear in the dictionary

图的 search 问题，见 crack the interview. 略…

89. 给定一个硬币集合{10,5,2,1},给定一个 input amount，找出所有的硬币组合其 sum 可以等于这个数额，硬币可以被重复使用，就是说 amount = 4，集合可以是{2,2}.

```
void get_combination_sum(int input[], int n, int target, vector<int>& v, int& sum)
{
    for (int i = 0; i < n; ++i)
    {
        if (sum + input[i] < target)
        {
            v.push_back(input[i]);
            sum += input[i];
            get_combination_sum(input + i, n - i, target, v, sum);
            sum -= input[i];
            v.pop_back();
        }
        else if (sum+input[i] == target)
        {
            v.push_back(input[i]);
            print_vector(v);
            v.pop_back();
        }
        else
        {
            break;
        }
    }
}
```

90. 集合的 intersection, union

TO LEARN

91. Given an int n, print all the numbers which are less than n and in the following pattern: 1,4,5,9,14... Write a recursive program.

看不懂…

92. How to sort a large collection of string?

因为 string 的比较开销比较大，所以可以考虑用 radix sort. 见 138 题， America flag sort 问题

93. How to serialize a very sparse tree?

保存 parent->child 关系

94. Given an arbitrarily long string, design an algorithm to find the longest repetitive substring. For example, the longest repetitive substring of "abcabcabc" is "abcabc".

TO LEARN. Suffix tree 的应用

95. reverse a link list within each k blocks

```
void reverse_linked_list_each_K_nodes(ListNode*& head, int k)
{
    ListNode* dummy = new ListNode(0);
    dummy->next = head;

    ListNode* block_head = dummy->next;
    ListNode* block_end = block_head;
    ListNode* last_end = dummy;
    while(last_end->next != NULL)
    {
        int count = k - 1;
        block_end = block_head;
        while (block_end && block_end->next != NULL && count > 0)
        {
            block_end = block_end->next;
            --count;
        }

        ListNode* tmp = block_head;
        ListNode* reversed = block_end->next;
        while (tmp != block_end)
        {
            ListNode* tmp_next = tmp->next;
            tmp->next = reversed;
            reversed = tmp;
            tmp = tmp_next;
        }
        last_end->next = block_end;
        block_end->next = reversed;
        last_end = block_head;

        block_head = last_end->next;
    }

    head = dummy->next;
    delete dummy;
}
```

96. BST，排序双链表互相转换

```
void BST2BiList(TreeNode* root, BiListNode*& start, BiListNode*& end)
{
    if (root == NULL)
    {
        start = end = NULL;
        return;
    }

    BiListNode* lstart = NULL;
    BiListNode* lend = NULL;
    BiListNode* rstart = NULL;
    BiListNode* rend = NULL;

    BST2BiList(root->left, lstart, lend);
    BST2BiList(root->right, rstart, rend);

    BiListNode* self = new BiListNode(root->value);

    if (lend != NULL)
    {
        lend->next = self;
        self->prev = lend;
        start = lstart;
    }
    else
        start = self;

    if (rstart != NULL)
    {
        rstart->prev = self;
        self->next = rstart;
        end = rend;
    }
    else
        end = self;
}
```

CODE

97．字符表格找单词，比如下面的 3*3 字符表格

1 2 3

4 5 6

7 8 9

每一个位置都是随机生成的 char,给你一个字典然后找到表格里面所有可能的单词．

单词的定义是任意个连续字符组合,一个位置用过之后就不能再用．

回溯， 略。。。

98．给一个 string，输出所有由这个 sting 中字符组成的所有可能 strings。然后，如果有重复的字符怎么办。如果给你一个 string，和输出 string 长度，找出由这个 sting 中字符组成的所有可能 string

生成子集的问题，略

99．给一个 log 文件，找最长 session。session 定义：同一个 userid，两 log 间隔时间小于一小时

不是很理解，用 map<userid, list<time>> 来记录用户的登录时间，然后再扫描？

100. 不用乘法实现两数相乘 m*n， O(lgn)

```
int multiple(int m, int n)
{
    if (n == 1) return m;
    int k = multiple(m, n / 2);
    return n & 1 ? k + k + m : k + k;
}
```

101. 一个返回所有 n 比特格雷码的函数 vector<int> getGrayCode(n) 比如 getGrayCode(2)， 应该返回{0,1,3,2}

略。。。

102. 两个 sorted array A，B， 问能否从 A，B 中各取且只取一个数，是他们的和为 x

从两头 scan，略

103. 一个数组有 N 个元素，都大于 0.将数组分成 K 段，求使最大的每段数组和的最小值

初始条件：f(x,y,1) = a[x] + … + a[y]

递推： f(1, n, k) = min(1<=i<=n+1-k)max{f(1,i,1),f(i+1,n,k-1)}

该问题的一个变体是：有一个包括 N 个整数的数组,求 k 个数，使得这 k 个数排序后，相邻两个数的差的最小值最大。

先将数组排序

初始条件：f(x,y,2) = a[y] – a[x]

递推：f (1,n,k) = max(2<=i<=n+2-k)min(f(1,i,2), f(i,n,k-1))

104. 判断某个点是否在多边形的内部。按逆时针方向依次给出多边形的所有顶点。

图形学，考虑依次形成的所有夹角的和，如果在内为 2pi，否则为 0

105. 判断一个 set 里是否有四个数的和等于一个 target number.

预先计算所有两数之和，得到 map<sum, vector<pair<index1,index2>>>

然后再这个 map 中搜索有没有和为 target number 的 pair(并且 index 不能重复)，时间和空间复杂度 O(N^2).

另外一种做法是当成子集合问集，按 target number(T)做 DP，复杂度 O(NT)

106. how to implement priority queue (describe) ?

用最大/最小堆来实现，堆的 heapify 操作

```
void heapify(int v[], int k, int N)
{
    while (k*2 <= N)
    {
        int j = k * 2;
        if (j < N && v[j-1] < v[j]) ++j;
        if (v[k-1] >= v[j-1]) break;

        exch(v[k-1], v[j-1]);
        k = j;
    }
}
```

107．找到数组中的第二大的元素

```
int find_second_largest(int v[], int N)
{
    if (N == 1) return v[0];

    int first = max(v[0], v[1]);
    int second = min(v[0], v[1]);

    for (int i = 2; i < N; ++i)
    {
        if (v[i] >= first)
        {
            second = first;
            first = v[i];
        }
        else if (v[i] >= second)
            second = v[i];
    }
    return second;
}
```

108．两个人（A，B）参与一个游戏，规则如下：

1）一个随机的整数数列有偶数个数,a1,a2,...a2n

2）A 先从数列取数，但只能从两头取,a1 or a2n

3)然后 B 取数，也是只能从剩下的两头取，依此类推，两个人轮流，都只能从两头取

４）最后谁手里的数和最大赢。

1. 先拿的人有必胜策略，把所有的数按在奇数位和偶数位分成两组，则先拿的人可以选择拿到所有奇数位或者所有偶数位的数。
2. 用 DP 求最后能拿到的最大的和：

   设 v[x,y]是当某人在数列剩下 x 到 y 位时，能拿到的最大值，n[x,y]表示需要拿的位置(x 或者 y)则

   初始化：v[x,x] = a[x], n[x,x] = x

   递推：v[x,y] = max(v[x] + (v[x+2,y]或者 v[x+1,y-1]，由 n[x-1,y]决定)，

   v[y] +（v[x,y-2]或者 v[x+1,y-1]，由 n[x,y-1]决定））

   n[x,y]由上一步取 v[x]还是取 v[y]决定。

109．最大回文的详细解法

Suffix tree 的应用， TO LEARN

110．假定有个 graph，怎么找出不带 circle 的最长 path

有向无环图可以用 DP 解，一般情形下是 NP 完全问题

算法：

```
algorithm dag-longest-path is
    input:
        Directed acyclic graph G
    output:
        Length of the longest path
```

```
length_to = array with |V(G)| elements of type int with default value 0

for each vertex v in topOrder(G) do
    for each edge (v, w) in E(G) do
        if length_to[w] <= length_to[v] + weight(G,(v,w)) then
            length_to[w] = length_to[v] + weight(G, (v,w))

return max(length_to[v] for v in V(G))
```

111. 关于外部排序

一般做法：

1. 将输出数据分成 K 份，使得每一份都能放到内存中排序，然后将每一份排好序后写到文件
2. 从每一份排好序的数据中读一部分到 buffer,对 buffer 中的数据进行 K-way merge 后写到最终的文件。（这里存在一个多路归并的开销，和反复读取文件的开销的权衡）
3. 如何改进性能：
   a) 使用多块磁盘同时进行读/写
   b) 使用多线程提高内存里的 sort 的性能
   c) 使用异步的 IO 使 sort 和磁盘读/写同时进行
   d) 多机的并行（map reduce ?）
   e) 如果 key 较大，则可以使用 radix sort 提高速度

112. 正态随机

http://en.wikipedia.org/wiki/Normal_distribution#Generating_values_from_normal_distribution

根据中心极限定义，一种简单的做法是,生成 2N 个(0,1)之间的随机数，然后将它们的和减去 N，得到一个近似正态分布的数

113. 实现 linkedIn 里查找两个人之间 connection 的功能。（如果每人有 100 个熟人，假设任何两个人之间只隔 6 个人，需要 space 100^6，内存放不下。所以改用同时从两边 bfs，需要 space 2*100^3）

略。。。

114. 两个 Sorted Array ,找 K smallest element，array1 长度 M ,array2 长度 N ,要求 O(logM+logN)

见 68 题

115. [Facebook] Given a string and a dictionary that maps a character to several characters, print all combination and tell the complexity.
i.e., string = "face", f=> f, @, 4, h a=> a, c, e
print: face, @ace, 4ace, ...

```
void variant_string(string& s, map<char, vector<char>>& mapping, int index)
{
    if (index >= s.size())
        cout << s << endl;
    else
    {
        if (mapping.find(s[index]) != mapping.end())
        {
            char tmp = s[index];
            vector<char> chars = mapping[s[index]];
            for (int i = 0; i < chars.size(); ++i)
            {
                s[index] = chars[i];
                variant_string(s, mapping, index + 1);
            }
            s[index] = tmp;
        }
        else
            variant_string(s, mapping, index + 1);
    }
}
```

116. Merge sort linked list.

略。。。

详见 http://www.chiark.greenend.org.uk/~sgtatham/algorithms/listsort.html

Merge sort linked list 的特点：不需要额外空间，时间复杂度 O(NlogN),并且是 stable 的

117. example:

char *words[] = {"foo", "bar", "baz", NULL};

setup(words);

1) First step:

isMember("foo") -> true

isMember("garply") -> false

2) Second step:

isMember("f.o") -> true

isMember("..") -> false

***/**

1. 用 map 即可。。。
2. 需要对 words 里面的每一个 elements 依次匹配。为了加速 ,可以预先对 words 构建一棵字典树。

118. Given an integer, print the next smallest and next largest number that has the same number of 1 bits in their binary representation.

xxx011100 -> xxx100011

1. 从右往左扫，将第一个在 1 后面出现的 0 置 1，xxx011100 -> xxx111100
2. 将这个 1 后面的 1 置 0，xxx111100 -> xxx101100
3. 将剩下的 1 放到最右边，xxx101100 -> 100011

CODE 略

119. 一个有 n 个整数数列，如果有符合下面条件，就返回 1，如果没有返回 0.

要求：a[i]+a[j]>a[k]; a[i]+a[k]>a[j]; a[j]+a[k]>a[i]

先排序，再比较相邻的三个数即可

120 很长很长的 string/file，要我找出中间重复次数最多的 character，character set 可能是任何 char set，unicode.（map reduce，multi-thread）

TO LEARN， 写一下用 map reduce 怎么做

121. [Apple] You are given a deck containing n cards. While holding the deck:
1. Take the top card off the deck and set it on the table
2. Take the next card off the top and put it on the bottom of the deck in your hand.
3. Continue steps 1 and 2 until all cards are on the table. This is around.
4. Pick up the deck from the table and repeat steps 1-3 until the deck is in the original order.
Write a program to determine how many rounds it will take to put a deck backinto the original order. This will involve creating a data structure to represent the order of the cards. This program should be written in Python. It should take a number of cards in the deck as a command line argument and write the result to stdout.

略。。。

122. Suppose there is a binary tree having millions of nodes and by mistake one node has two indegree (i.e. There becomes a cycle/loop in that tree. Tou have to find that node which is having two indegree) Constraint is no extra memory can be used and tree representation is in Linked list format.

？？？可能吗？没有额外 memory，否则做个 DFS/BFS 即可

123. Print the nodes on the exterior of a binary tree in a anti-clockwise order，i.e.，nodes on left edge，then leaf nodes，then nodes on right edge.

先按层遍历，将每一层的开始放到 left edge，结尾放到 right edge．再中序遍历打出所有的 leaf node（这里取决于 leaf node 的定义）， CODE 取自：
http://www.leetcode.com/2010/10/print-edge-nodes-boundary-of-binary.html

```
void printLeftEdges(BinaryTree *p, bool print) {
  if (!p) return;
  if (print || (!p->left && !p->right))
    cout << p->data << " ";
  printLeftEdges(p->left, print);
  printLeftEdges(p->right, (print && !p->left ? true : false));
}

void printRightEdges(BinaryTree *p, bool print) {
  if (!p) return;
  printRightEdges(p->left, (print && !p->right ? true : false));
  printRightEdges(p->right, print);
  if (print || (!p->left && !p->right))
    cout << p->data << " ";
}

void printOuterEdges(BinaryTree *root) {
  if (!root) return;
  cout << root->data << " ";
  printLeftEdges(root->left, true);
  printRightEdges(root->right, true);
}
```

124. 已知整数 m ，其二进制形式有 k 位为 1，打印出 0≤x≤m 所有满足以下条件的 x
x^(m-x) = m，其中^是异或运算符。在 0≤m<2^n 范围内对每一个 m ，打印出所有的 x ，并求总复杂度。

TO LEARN， 像数学题

125. You can win three kinds of basketball points, 1 point, 2 points, and 3 points. Given a total score X, print out all the combination to compose X. (recursion/ Dp)

略。。。

126. 有 n 个 job，分配给 3 个打印机，求所有任务完成的最短时间。例如：3，5，4 每个打印机占 1 个 job，最短时间是 5. 15，7，8，10， 4，15 给 1 号打印机，7，8 给 2 号打印机，10，4 给 3 号打印机，最短时间是 15.

http://en.wikipedia.org/wiki/Partition_problem

见 133 题

127. 设计一个算法，判断一个 integer n 是不是可以表示成 k(k>=2)个连续正整数的和

假设所要分解的数为 x ，分解成 n 个数，那么我们可以这样表示：

x=m+(m+1)+(m+2)+.....+(m+n-1)

其中 m 为分解成的连续整数中最小的那一个，并且我们知道 m 大于等于 1 的正整数。易知：

x=(2m+n-1)*n/2， 变换一下的 m=（2*x/n-n+1)/2

由 m 的范围我们知道（2*x/n-n+1)/2>=1 以上就是 x 和 n 的关系。给定一个 n 看是否 x 能分解成 n 个连续整数的和可以判断是否存在 m，也就是转换成（2*x/n-n+1)是否是偶数。

代码略

128. how to serialize and deserialize a n ary tree?

见 79 题

129. 如何刷屏最快

G(n) = max{G(n-1)+1, g(n-k)*(k-2)}

130. Given a sorted array with duplicates and a number, find the range in the form of (startIndex, endIndex) of that number. For example, given 0 2 3 3 3 10 10 and 3, return (2,4). Given the same array and 6, return (-1,-1).

Binary Search 的扩展，见 142 题

131. 有 N 个整数，M 个 bucket，每个 bucket 都可以装至少 N 个整数，一个 bucket 的 value 等于放入它的所有整数的和。现求解一种分配方法，使之 minimize(the max value of all buckets)

NP 完全问题，见：

http://en.wikipedia.org/wiki/Job_shop_scheduling

132. Given pairs of connected items ((A, B), (B, C), (C, D)...), find the root node of this tree.

找到入度为零的 node 即可

133. There's a book and each sentence consists of several words. How to find the minimal set of words which can used by the reader to understand half of total number of the sentences. Each sentence is readable if the reader knows all the words.

TO LEARN， 有点类似于 dancing links 用到的满足问题

134. [facebook]求一段时间内出现最多的 ip address(es)

只能是搞几张 aggregate 表，按秒，分钟，小时等做为粒度。。。

135. 两个文件里面存着行程安排，开始时间以及结束时间，设计算法找所有 conflicts。

如果要找所有 conflicts，则复杂度为 O(N^2)， 略。。。

136. what's the best data structure for storing and looking up a huge amount of url's (presented in a prefix format) like these:
com.google.www -> value1
com.yahoo.finance -> value2
com.yahoo.finance/stocks -> value3
com.yahoo/finance -> value2
1.2.3.4/node/123 -> value4
....
the requirements are:
1. it has to be compact (compression if necessary).
2. lookup time should be fast (constant would be ideal, but a few level of tree lookup is fine too).

有点类似于设计题，可以用一个优化过的 prefix tree?

137. Subset sum problem

可以转换为背包问题

138. Dutch National Flag Problem (DNFP)

http://www.iis.sinica.edu.tw/~scm/ncs/2010/10/dutch-national-fl

```
void three_way_partition(int a[], int n)
{
    int p = -1;
    int q = n;
    int i = 0;
    while (i < q)
    {
        if (a[i] == 1)
        {
            ++p; swap(a[i], a[p]); ++i;
        }
        else if (a[i] == 2)
        {
            ++i;
        }
        else
        {
            --q; swap(a[i], a[q]);
        }
    }
}
```

如果有多于三种元素，则称为 America Flag Problem，本质上是 radix sort

```
void radix_sort(int *array, int offset, int end, int shift) {
    int x, y, value, temp;
    int last[256] = { 0 }, pointer[256];

    for (x=offset; x<end; ++x) {
        ++last[(array[x] >> shift) & 0xFF];
    }

    last[0] += offset;
    pointer[0] = offset;
    for (x=1; x<256; ++x) {
        pointer[x] = last[x-1];
        last[x] += last[x-1];
    }

    for (x=0; x<256; ++x) {
        while (pointer[x] != last[x]) {
            value = array[pointer[x]];
            y = (value >> shift) & 0xFF;
            while (x != y) {
                temp = array[pointer[y]];
                array[pointer[y]++] = value;
                value = temp;
                y = (value >> shift) & 0xFF;
            }
            array[pointer[x]++] = value;
        }
    }

    if (shift > 0) {
        shift -= 8;
        for (x=0; x<256; ++x) {
            temp = x > 0 ? pointer[x] - pointer[x-1] : pointer[0] - offset;
            if (temp > 64) {
                radix_sort(array, pointer[x] - temp, pointer[x], shift);
            } else if (temp > 1) {
                // std::sort(array + (pointer[x] - temp), array + pointer[x]);
                insertion_sort(array, pointer[x] - temp, pointer[x]);
            }
        }
    }
}
```

139. 一个单词的列表，要找出两个单词，它们没有相同的字母出现，而且长度乘积最大

难题， TO LEARN

140. You are given N blocks of height 1…N. In how many ways can you arrange these blocks in a row such that when viewed from left you see only L blocks (rest are hidden by taller blocks) and when seen from right you see only R blocks? Example given N=3, L=2, R=1 there is only one arrangement {2, 1, 3} while for N=3, L=2, R=2 there are two ways {1, 3, 2} and {2, 3, 1}.

DP 题，定义 g(n,l,r)为题目的答案，而 f(n,l)为 n 个 block,从左边看到 l 个 block,则递推公式为：

g(n,l,r) = (1<=k<=n)sum(C(n-1,k-1)*f(k-1,l-1)*f(n-k,r-1))

f(n,l) = (1<=k<=n)sum(C(n-1,k-1)*f(K-1,l-1)*(n-k)!)

f(n,1) = (n-1)!

F(n,n) = 1

F(n,m) = 0 if n < m

141. There is a binary tree(Not a BST) in which you are given three nodes x,y,z. Write a function which finds whether y lies in the path b/w x

如何定义 in the path?...略。。。

142. binary search 的各种变种

1) 如果找不到元素，返回应该插入的位置

```
int binary_search(int a[], int n, int key)
{
    int b = 0;
    int e = n - 1;
    while (b <= e)
    {
        int m = (b + e) / 2;
        if (a[m] == key)
            return m;
        if (a[m] > key)
            e = m - 1;
        else
            b = m + 1;
    }
    return e + 1;
}
```

2) 如果数组允许有重复，寻找最小的那个 i，使得 arr[i] = v，（第一次出现的位置）

```
int binary_search2(int a[], int n, int key)
{
    if (a[0] == key) return 0;
    if (a[0] > key) return -1;
    int b = 0;
    int e = n - 1;
    while (b <= e)
    {
        int m = (b + e) / 2;
        if (a[m] == key && a[m-1] < key) return m;
        if (a[m] >= key)
            e = m - 1;
        else
            b = m + 1;
    }
    return -1;
}
```

3) 如果数组允许有重复，寻找最大的那个 i，使得 arr[i] = v

与 2)类似

4) 如果数组允许有重复，寻找最小的那个 i，使得 arr[i] > v

```
int binary_search3(int a[], int n, int key)
{
    if (a[0] > key) return 0;
    int b = 0;
    int e = n - 1;
    while (b <= e)
    {
        int m = (b + e) / 2;
        if (a[m] > key && a[m-1] <= key) return m;
        if (a[m] > key)
            e = m - 1;
        else
            b = m + 1;
    }
    return -1;
}
```

5) 如果数组允许有重复，寻找最大的那个 i，使得 arr[i] < v

与 4)类似

6) 给 2 个有序数组，大小一致，如何求他们的 median

见第 68 题

7) 循环数组的二分查找

```
//int a[10] = {7, 8, 9, 10, 1, 2, 3, 4, 5, 6 };
// return 3 (10);
int find_sorted_array_rotation(int A[], int N) {
  int L = 0;
  int R = N - 1;

  while (A[L] > A[R]){
    int M = (L + R) / 2;
    if (A[M] > A[R]) // A[M] > A[R] return 1, while A[M] < A[L] return 10
        L = M+1;
    else
        R = M;
  }

  return L;
}
```

143. 怎样 de-dup 一个 sorted array？

可以直接线性扫描，也可以用 binary search 来优化（如果重复数比较多）

follow-up: 如果有一个 big single file，many servers，how to use these servers to compute this problem? 要求尽量 balance load

balance load,可以将 big file 分成比较小的块，每台机器完成若干块，根据计算情况进行调度。

144. Given a number n, find the nearest palindrome of n just greater than n.

算法：

1. 看左边一半的反是不是等于右边一半，如是，直接返回
2. 如果不是，看左边一半的反是不是大于右边一半，如果是，将右边一半改成左边一半的反，再返回
3. 否则，将左边一半的最后一位加 1，再重复第二步（这次不用考虑大小）

CODE

145. 有一个不定长的 char string (S1)，都是 0,1 组成；另外一个 string (S2)，也是 0,1 组成，可以认为是 pattern,问 S1 中是否含有 S2

e.g S1 = 11100011 10101011 11111111 10000100 (4 bytes, '11100011' is 1 byte)

S2 = 00111010

the answer is ture, 1110"00111010"1011

http://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_string_search_a

Rabin Karp 算法 ? 略。。。

146. Write a function for a linked list that exchanges the positions of the nodes after the nodes referenced by two given pointers t and u. (assume t and u are either in the list or NULL).

Example:

1 2 4 3 6 8 5 7 9 10

Exchange nodes after 2 and 3:

1 2 6 3 4 8 5 7 9 10

简单题，略。。。

147. Design a data structure for a text editor. Require better than linear for both line search and insert a new line.

Notepad++使用gap buffer，即在一段buffer的中间留下空间 这样插入和删除操作都是O(1)．关于 search，可以用 Rabin karp 算法。

Rope 也是一种可以考虑的数据结构，见：

http://en.wikipedia.org/wiki/Rope_%28computer_science%29

148. Given an array of 0's and 1's. only, find the maximum length of the subarray such that the number of 0's and 1's in that subarray are equal.

将所有的 0 换成-1，先计算累加和,cum[i] = a[0]+…+a[i]，则问题转化为：求 cum[i]相同时的最大的 index 差。扫描计算即可，复杂度 O(N)

类似的问题有（利用累加和）:

1.一个有正有负的数组中，求一个 subarray 使其和最接近 0

2.对数组进行 add(l,r,v)操作，即对 a[l]到 a[r]的每一个元素加上 v，现有 N 个这种操作，怎么在 O(N)时间内完成？

149. 一个数组有很多 key，已知所有 key 属于三个组，存在先后顺序，现在要求把所有 key 按照组排序，比如{1， 2， 3}， {4， 5}， {6， 7}，key 之间不需要有顺序，只要不同组之间的元素在数组里满足组规定的顺序就行了(DNFP?)

DNFP 问题,见 138 题。

150. 游戏算法，给定一个 N*N 格的板块，往上面放不规则的 element。

如何表达这个板块，用什么数据结构来表达 element，这个 element 有可能是任何形状，like "–"，X"，"Y"，"田"（参考俄罗斯方块）

如何判断这个 element 可以放在某个 cell 里面，放在 cell 的条件是这个 element 可以覆盖这个

cell。(element 之间不能重叠)

象俄罗斯方块一样，这个 element 可以 rotate 四个角度，问如何判断 rotate 后可以放入。

TO LEARN，俄罗斯方块?

这里的描述不是很清楚，如果是俄罗斯方块的话：(取自 http://wintris.codeplex.com/)

```
//每个俄罗斯方块的旋转数，每个方向的形状及颜色
struct shape_t
{
    int count;
    int shape[4];
    enum color_type color;
};

struct shape_t shapes[7] =
{
    { 0x0001, 0xCC00, 0x0000, 0x0000, 0x0000, RED },
    { 0x0004, 0x4444, 0x0F00, 0x2222, 0x0F00, ORANGE },
    { 0x0004, 0x4E00, 0x4C40, 0x0E40, 0x4640, YELLOW },
    { 0x0004, 0x4460, 0x0E80, 0xC440, 0x2E00, GREEN },
    { 0x0004, 0x44C0, 0x8E00, 0x6440, 0x0E20, BLUE },
    { 0x0002, 0x4C80, 0xC600, 0x0000, 0x0000, WHITE },
    { 0x0002, 0x8C40, 0x6C00, 0x0000, 0x0000, MAGENTA }
};

// 代表一个俄罗斯方块的位置，方向和形状
struct piece_t
{
    int x, y;
    int rotation;
    int shape;
};

// 检查当前这个方块是否可以被放置， 其中field表示整个背景（包括边界）
inline BOOL check_piece(struct piece_t *piece)
{
    int row = 0, col = 0;

    for (int bit = 0x8000; bit >= 0x0001; bit >>= 1)
    {
        if (shapes[piece->shape].shape[piece->rotation] & bit) {
            if (field[piece->y + row][piece->x + col].color != BLACK) {
                return FALSE;
            }
        }

        col++;
        if (col == 4) {
            row++;
            col = 0;
        }
    }

    return TRUE;
}
```

151．给你一串数字，让你写程序，把操作符（可以是任意多个 + - * / %）放进各个数字之间，同时还可以在任意位置放进括号，让这个算术表达式的值等于一个给定的数字。比如：给你 5 3 8 9 = 6 你的

程序应该输出 5 * (4 + 8) % 9 = 6

CODE，24 点

```
bool calc_24_point(vector<double> nums)
{
    for (int i = 0; i < nums.size(); ++i)
    {
        for (int j = 0; j < nums.size(); ++j)
        {
            if (i != j)
            {
                for (int k = 0; k < 4; ++k)
                {
                    double val = 0.0;
                    switch(k)
                    {
                    case 0:
                        val = nums[i] + nums[j];
                        break;
                    case 1:
                        val = nums[i] - nums[j];
                        break;
                    case 2:
                        val = nums[i] * nums[j];
                        break;
                    case 3:
                        if (abs(nums[j]) < 1e-6) continue;
                        val = nums[i] / nums[j];
                        break;
                    default:
                        break;
                    }

                    if (abs(val-24) < 1e-6 && nums.size() == 2)
                        return true;

                    if (nums.size() > 2)
                    {
                        vector<double> next;
                        for (int p = 0; p < nums.size(); ++p)
                            if (p != i && p!= j)
                                next.push_back(nums[p]);
                        next.push_back(val);

                        if (calc_24_point(next))
                            return true;
                    }
                }
            }
        }
    }

    return false;
}
```

152. 一道编程题，大意是给定一个类 read1，它有一个函数 read4096，每次调用它可以从文件中读取 4K 个字节，同时移动文件指针 4K 个位置（若文件中剩余数据不足 4K，则读取剩下的所有数据），这个函数返回实际读取的字节数，int 型，要求实现另一个类 read2 中的一个函数 read，它有一个参数 int n_byte，

这个函数可以从文件中读取由 n_byte 指定的字节数，同样返回实际读取的字节数；然后又给出一个函数 reset，它可以将文件指针重置到起始位置，要求实现 read2 中的另一个函数 seek，有一个参数 int pos ，它可以将缓冲区的指针移动到第 pos 个字节的位置，返回实际指针移动到的位置。可以在 read2 中添加任意变量来完成这两个函数。

多次做 read4096 即可，为了加速，可以重用上次 seek 的结果

153. 编程题问的是 boggle 游戏的问题：给定一个 4*4 的矩阵，每个位置有一个字母，可以从一个位置跳转到周围八个相邻位置中的任何一个，但不能跳到已经访问过的位置，要求找出所有的单词（假设给定了一个词典）。http://en.wikipedia.org/wiki/Boggle

构造一个单词的 prefix 字典，然后再递归+回溯。。。

154. Find median for k sorted arrays

设 k 个 sorted array 为 a1, a2, …, ak.

先找出这 k 个 sorted array 的 median, m1, m2, … mk.

再找出这 k 个 median 的 median: mm

然后可以把所有比 mm 小的数和大的数都去掉，大致有一半

然后再找剩下的数的 median（递归），复杂度 O(klogn)

155. "Count and Say problem" Write a code to do following:

n String to print

0 1

1 1 1

2 2 1

3 1 2 1 1

...

Base case: n = 0 print "1"

for n = 1, look at previous string and write number of times a digit is seen and the digit itself. In this case, digit 1 is seen 1 time in a row... so print "1 1"

for n = 2, digit 1 is seen two times in a row, so print "2 1"

for n = 3, digit 2 is seen 1 time and then digit 1 is seen 1 so print "1 2 1 1"

for n = 4, you will print "1 1 1 2 2 1"

Consider the numbers as integers for simplicity. e.g. if previous string is "10 1" then the next will be "1 10 1 1" and the next one will be "1 1 1 10 2 1" 10

CODE

156. 给定一个数组， A[0 .... N]作为输入数组。给定一个函数 f(i,j) ,这个函数以两个下表(i,j)为输入， 返回一个值。(这个函数是个 blackbox，唯一的信息就是输入两个整数返回一个值)。要求把数组 A 分为 3 份，使得 f(0,a) + f(a,b) + f(b,N)最小。

LEARN

157. 给 n * m 的字符矩阵。然后给你一个字符串。问是否可以在矩阵中找到他的 track。track 是指从其中一个符出发，可以向四周走，可以重复，可以回头。

e.g:

a b

c d

string: 'bdba' could be found but not for 'bcd'.

递归+回溯，也可以用 DP 优化

158. Given three linked list of integers, all sorted. Find the first shared integer in all three. What is the time complexity?

略...

159. Initially you have a 2x2 matrix, say zoom1:

a b

c d

zooming it results in a 4x4 matrix (zoom2) as follows:

aa ab ba bb

ac ad bc bd

ca cb da db

cc cd dc dd

zooming it again will result in an 8x8 matrix and so on..

aaaa aaab abaa abab baba babb bbba bbbb

aaac aaad abac abad babc babd bdba bdbb

acaa acab adaa adab bcba bcbb bdba bdbb

acac acad adac adad bcbc bcbd bdbc bdbd

caca cacb cbca cbcb dada dadb dbda dbda

cacc cacd cbcc cbcd dadc dadd dbdc dbdd

ccca cccb cdca cdcb dcda dcdb ddda dddb

cccc cccd cdcc cdcd dcdc dcdd dddc dddd

The question is, given a sequence say abaaccda... we need to find out the sequence coming just left to it. For e.g. if the given sequence is "bd" for zoom2, the sequence coming just left to it is "bc". For "cd" it's "cc" etc.

CODE

160. 如何在 binary tree 找一个 path 从 root 到 leaf，和是 sum?

2) 如何序列化一个 binary tree 到一个文件

3) 如果有一个已经序列化的 tree，很大，要做 1)的算法，怎么做，2) 中如果有多个方法选择哪中序列化的方法比较好？

4) 如果有 1000w 个已经序列化的小文件，对他们都要做 3)，如何提高性能，系统是 5 台机器

TO LEARN

161. Programming: interval halving. Given a continuous function 'f(x)' and an interval on the x-axis from 'start' to 'end'. It is know that 'f(x)=0' for exactly one value of 'x' between 'start' and 'end', and that ‘f(x)’ crosses the x-axis at this point. Write a program that repeatedly cuts in half the interval until the interval containing 'f(x)=0' is equal or less than 'epsilon' wide.

略...

162 [Facebook] You are given N ranges of date offsets when N employees are present in an organization. Something like
1-4 (i.e. employee will come on 1st, 2nd, 3rd and 4th day)
2-6
8-9
..
1-14
You have to organize an event on minimum number of days such that each employee can attend the event at least twice. Write an algorithm (there is apparently an O(n) algorithm for this).

先按结束的时候排序，然后依次选取

162. Search in skip list

TO LEARN

163. 给定一个 string，可以任意删除其中的 char,以使的剩下的 string 成为 palindrome,求最长的这样的 palindrome。问有啥 dp 算法可以解？

与 reverse 求 longest common subsequence

164. 把一个有大写字母和小写字母的字符串变换成小写字母在前面大写字母在后面的字符串

略，partition

165. 给很多 date ranges,一个 array，每个 date range 有开始日期和结束日期，判断连续不连续

CODE

166. 实现 hash 表

CODE

167. 判断两二叉树全等（在可以交换左右子树的条件下），进一步给出需要多少次交换。

CODE

168. 一个 NxN 矩阵，每个格子有一个整型数，从左上角到右下角找一条路径使得经过的格子数字和最大。只能向右和下移动。时间复杂度，如何优化。

DP，复杂度 O(N*N)

169. 现在假设有一堆整数数组，有一个 flip 函数，接受一个数组下标 i 为参数，作用是将数组 index 从 0 到 i 的元素反转。eg. 假设数组为 5，6，-1，3，2，如果调用 flip(3)，那么就将数组下标 0，1，2，3 反转，数组变为 3，-1，6，5，2。问：只使用 flip 函数(不能直接用 swap 或数组下标操作[]等对数组进行修改)，来对该数组排序。

IDEA：每做两次 flip 将当前最大的 item 放到位上

CODE

170. 一个大小为 N 的数组，其中有 N-1 个不同的整数，范围从 0-N，问找出 missing 的那个整数。然后又扩展，问如果有 k 个 missing，如果用 O(1)space 去找出来。

IDEA：将整数 i 放到数的第 i 位上

CODE

171. Suppose we have a stream of web clicks. Now you need to provide at any time the count of web clicks during the past 1 minute. To be specific, you get informed whenever a web click happens. You need to provide a function "getCount()" such that once called, return the count of clicks during the past 1 minute. The solution will be both constant in time and space.

用一个 circular buffer 来保存每一秒的 click 数，再维护一个 total 数即可。

172. 一个有序序列，从某个地方 rotate，求在 rotate 的位置，比如 1 3 5 0 0 0，那么 rotate 的位置是 5，他后来只用了 5 行就写出来了，很 nb，被 bs 了~~

```
//average: O(log N) worst case: O(N)
int search(int *arr, int st, int en)
{
    if(arr[st] < arr[en] || st == en) return 0;
    if(en - st == 1 ){
        if (arr[st] == arr[en])
            return 0;
        return en;
    }
    return search(arr, st, (st+en)/2) + search(arr, (st+en)/2, en);
}
```

173. 二分图最大匹配

略。。。

174. [Facebook] implement copy-on-write string class.

TODO

175. 给你 n 个数字 a1,a2...an，表示坐标上面（i，ai）个点，i=1..n（i，ai）到（i，0）共 n 个线段，从中挑两条，和 x 轴一起构成一个容器，让容器能装的液体容量最大(容器不能倾斜)。

穷举？。。。

176. Describe an algorithm that takes an unsorted array of axis-aligned rectangles and returns any pair of rectangles that overlaps, if there is such a pair. Axis-aligned means that all the rectangle sides are either parallel or perpendicular to the x- and y-axis. You can assume that each rectangle object has two variables in it: the x-y coordinates of the upper-left corner and the bottom-right corner.

穷举？。。。

177. Given a list of intervals, 比如 (10,20),(30,50)...,and a target interval, 比如(18,35) 找有多少 overlap.

http://programmers.stackexchange.com/questions/64132/interestin

INTERVAL TREE

178. 给定一个 integer array，a1,a2...an，找出所有 a,b,c,d 使得 a+b = c+d. 很容易找到 O(n^2)

空间，O(n^2)时间的算法，不知道有没有更快更好的。

如果所有的数都相等，那至少需要 O(n^2)时间复杂度。另外排序后，二重循环加两头扫描，不需要额外的空间复杂度了。

179. n 个球，n 很大 > 1 billion，k 个颜色，k 相对小， 比如 10. 要求 in space sort 最 efficient 的做法 白板 coding. （hint，k<< n 所以最后算法争取比 nlog(n)小）
该题的第二问 k = 1000 实现比 nlogn 更 efficient 的 in space 的算法

见 138 题

180. If the Fibonacci series is 1,2,3,5,8,13,….. then 10 can be written as 8 + 2 ==> 10010 and 17 can be written as 13 + 3 + 1 ==> 100101. The Question was, given n, I need to get all possible representations of n in Fibonacci Binary Number System. as 10 = 8 + 2 ==> 10010 also 10 = 5 + 3 + 2 ==> 1110 (Zeckendorf's theorem)

如果要得到所有的组合，先计算出该数范围内的 Fibonacci 数，再当成找零问题来计算就可以了。

Zeckendorf’s theorem:

Every positive integer can be represented uniquely as the sum of one or more distinct Fibonacci numbers in such a way that the sum does not include any two consecutive Fibonacci numbers.

181. void reversePairsOfLinkList(Node*& head) {}
[] => []
[A,B] => [B, A]
[A, B, C] => [B, A, C]
[A, B, C, D, E, F, G] => [B, A, D, C, F, E, G]

见 195 题

182. You have to paint N boards of length {B1, B2, B3… BN}. There are K painters available and you are also given how much time a painter takes to paint 1 unit of board. You have to get this job done as soon as possible under the constraints that any painter will only paint continuous sections of board, say board {2, 3, 4} or only board {1} or nothing but not board {2, 4, 5}.
know it could be solved by DP. But solution space seems quite big. What is the optimal solution? Thx.

DP,类似于 103 题

183. 一个 range 的序列(链表或数组) 如[1,3], [2,6], [8,10] ,[15,18] 写程序合并有重叠的 range，比如上面的序列合并为[1,6], [8,10], [15,18] 如果这个序列不是静态的，而是一个数据流，如何 处理？

INTERVAL TREE

184. 实现 atoi，要考虑特殊的情况，比如不合法的输入等等。参照这个定义
http://www.cplusplus.com/reference/clibrary/cstdlib/atoi/

```
int _atoi (const char *str)
{
    //异常时返回零，觉得不是很合适，但又没想到什么好的方法，标准函数也是这么定义的
    if (str == NULL) return 0; //指针为空

    int flag = 1;            //定义符号位
    if (*str == '-')     //取符号位，下同
    {
        flag = -1;
        str++;
    }
    else if (*str == '+')
        str++;

    int result = 0;          //保存迭代结果
    while (*str)         //迭代，算法很简单
    {
        if (*str > '9' || *str < '0')   //处理异常
            return 0;
        result = result * 10 + (*str++ - '0');
    }
    return result * flag;    //确定符号并返回
}
```

185. word edit distance.

伪代码如下：

```
int LevenshteinDistance(char s[1..m], char t[1..n])
{
  // for all i and j, d[i,j] will hold the Levenshtein distance
between
  // the first i characters of s and the first j characters
of t;
  // note that d has (m+1)x(n+1) values
  declare int d[0..m, 0..n]

  for i from 0 to m
    d[i, 0] := i // the distance of any first string to an empty
second string
  for j from 0 to n
    d[0, j] := j // the distance of any second string to an
empty first string

  for j from 1 to n
  {
    for i from 1 to m
    {
      if s[i] = t[j] then
        d[i, j] := d[i-1, j-1]       // no operation required
      else
        d[i, j] := minimum
                   (
                     d[i-1, j] + 1,  // a deletion
                     d[i, j-1] + 1,  // an insertion
                     d[i-1, j-1] + 1 // a substitution
                   )
    }
  }

  return d[m,n]
}
```

186. longest palindrome substring.

http://www.akalin.cx/2007/11/28/finding-the-longest-palindromic-substring-in-linear-time/

如果是 substring,那可以按中心位置依次搜索。

如果是 subsequence，与它的反求 longest common subsequence 即可

187. Describe and analyze an algorithm to find the longest prefix of a given string that is also a palindrome. For example, the longest palindrome prefix of ILLINOISURBANACHAMPAIGN is ILLI, and the longest palindrome prefix of HYAKUGOJYUUICHI is the single letter S. For full credit, your algorithm should run in O(n) time.

TO LEARN， 后缀树？

188. 给一个数据结构数组，(parent，child)， 重建二叉树，总是先遇见 leftchild，再遇见 right child，假设输入没有问题。要求返回 root。

用一个 hashtable 来保存所有见到过的结点。 Root 结点就是没有在 child 位置上出现过的结点。

189. 1. 两个 sorted array，如果 merge 成一个 array。

2. 如果这两个 array 没有 sort 呢？并分析复杂度。

3. 如果有 K 个没有 sorted 的 array 怎么办呢？

4. 如果当前机器有 K 个 cpu， 怎么处理问题 3 呢？复杂度分析。(考虑 multithreading)

后面几问假设是 merge 成一个 sorted array，那这个题就类似于外部排序的多路归并。

190. 给定一个 tree， 每个节点有一个数值， 如果找到一个从 root 到 leaf 的 path，使得这个 path 上的所有节点的 sum 最小。 Interviewer 所要的答案是和 hashtable 联系上的，因为考虑到树很大的时候需要很长的时间。这个题很容易用 recursive 的方式解答，可是这个不是 interviewer 所要的答案。后来按照 interviewer 的意见，还是基本写出了用 hashtable 的算法。

不理解题目的意思。。

191. 给定一个没有通往父节点的连接的 BST，找到大于 x 的最小的那个节点

中序遍历。略。。

192. 技术问题是找一个 binary tree 的叶子的最少 depth

分层遍历即可，略。。

193. Integer to Roman number

CODE

194. 有一行 animal cages，每个 cage 的动物的用水量为数组,有两个 pipe 给所有动物供水，pipe 给当前 cage 的 cost 是 这个 cage 动物的用水量，给其他 cage 的动物供水的 cost 是(distance to that cage)*那个 cage 动物的用水量， 求两个 pipe 供水的位置使 cost 最小。

TO LEARN

195. 问把整数分成 sum of square 的经典问题

TO LEARN，数论题。。。

196. longest increase consecutive subsequence. e.g. 3, 2, 4, 5, 1, 6. Return {2, 4, 5}

CODE

197. 用 1*2 的瓷砖覆盖 8*8 的地板，有多少种方式呢？如果是 N*M 的地板呢?

中等难度的 DP，有个解题报告见：

http://www.cnblogs.com/PureMilk/archive/2008/07/21/1247492.html

公式的论文见：

http://arxiv.org/abs/math/0310326

198. 生成 Dyck word list

与生成所有合法的括号组合类似，见：

http://en.wikipedia.org/wiki/Catalan_number

199. one integer array A with ascending order, for example 1 ,2 ,3,4, please generate another array B with any sum of any 3 different numbers picked from the A with ascending order, for example for 1,2,3,4, the result is (1+2+3),(1+2+4),(1+3+4),(2+3+4)"

三重循环即可，代码略。。

200. int array a[n] with all element 0<=a[i]<=1000,find the subarray with the largest sum that is <= max and output the starting and ending index of this subarray and the sum. If there is more than one such subarray, output the one with smallest starting index.

算法：

先求出累加和 cum[i] = a[0]+…+a[i]，从 a[k]>max 开始，在前面二分查找第一个 cum[l]>=(a[k]-max)，直到找到范围最大的 range. 复杂度 O(NlogN)

201. given is an unsorted integer array, how to divide it into two subarrays, such that the averages of these two arrays are the same (or have minimum difference).what if those are positive integers only, and what happens when it is mixed with positive and negative integers?

线性扫描并计算一下即可。正负数是否有关系？

202. 一个 arraylist,里面有很多 string,它们按照未知的 alphabetical 顺序排列,要求输出所有可能的 alphabetical order,(后来又问了如何判断是否有 conflict)

例子:it is a good day today,

it<is -- t<s

is<a -- i<a

a<good -- a<g

good<day--g<d

day<today--d<t

两两比较找第一个不同即可。

判断是否有冲突？整个关系形成一个图，这个图应该是无环的，做一次 DFS 即可。

203. 有一堆的工作,每个工作有开始和结束时间,假设表示为[2,3],[3,7]等等,现在要求在[1,n]的时间段内选出尽可能多的工作.

贪心，尽量选择结束时间最早的工作即可

204. 给一个数组 里边都是整数 求最大乘积的连续子数组

先按零把数组分成若干个子数组，再在子数组里找包含偶数个负数的最大的范围。

205. f(i,j)=2^i*5^j 给出一个长度为 N 的(i,j)序列，使得 f 值递增，i>=0,j>=0

设 f(i,j)是第 k 个数 f_k，则下一个数取使 i+1 或者 j+1 中间比较小的那个即可

```
void dummy()
{
    int num[100];
    int i[100];
    int j[100];
    num[0] = 1;
    i[0] = 0;
    j[0] = 0;
    int _max = 1;

    for (int k = 1; k < 100; ++k)
    {
        int tmp = INT_MAX;
        int tmpi,tmpj;
        for (int l = k - 1; l >= 0; --l)
        {
            if (num[l] * 2 < tmp && num[l] * 2 > _max)
            {
                num[k] = num[l] * 2;
                i[k] = i[l] + 1;
                j[k] = j[l];
                tmp = num[k];
            }

            if (num[l] * 5 < tmp && num[l] * 5 > _max)
            {
                num[k] = num[l] * 5;
                i[k] = i[l];
                j[k] = j[l] + 1;
                tmp = num[k];
            }
        }

        _max = num[k];
    }

    for (int k = 0; k < 100; ++k)
    {
        cout << '(' << i[k] << ',' << j[k] << ')' << ':' << num[k] << endl;
    }
}
```

206. f(N): return round(reduce(lambda x,y: int(x)+int(y), list(str(N))))/len(N) N 为整数，f 函数返回 N 各位数值的平均数，现在给出一个正整数范围[begin, end]，要求得出该范围中符合 f(N)>=7 的数的集合，希望算法尽可能比 end-begin+1 次 test 快。

TO LEARN，

发信人: bluesun (老小虎), 信区: JobHunting
标　题: Re: 问两道面试题
发信站: BBS 未名空间站 (Sat May 28 18:38:26 2011, 美东)

I have got a semi-baked solution. It deals with input [A, B] if A = 10^n
and B= 10^(n+1) - 1, e.g. [10, 99], or [100, 999].

Starting with B, store B into a vector(vBEnd) for easy processing.

Set CurSumB to the sum of each digits in B;
Set SumNeededB to 7 * number of digits in B;
Set extra = CurSumB - SumNeededB. Here extra means how much we can
decrease for the digits in B and still make the average >= 7;
e.g. if B=999, CurSumB = 27, SumNeededB = 7*3 = 21, extra = 6.

Let i = index of LSB of B, decreasing to the index of MSB of B (e.g. if
B=999, i=2, 1, 0)
  call function: Solve(A, vBEnd, i, extra)

The main thing Solve does is: (please look at the example couple of
lines below, I know I may not express this clearly)

set digit to the ith item of vBEnd;
Let j = digit, decreasing to max(digit-extra, 0)
  set ith item in vBEnd to j;
  if i == index of LSB of B
    print current number (from vBEnd);
  else
    call Solve(A, vBEnd, i+1, extra-(digit-j));
end

What Solve really does is to decrease the digit at index i from its
current value all the way to digit-extra or 0. That way it guarantees
the average >= 7. When the index i is the LSB, we can immediately print
out the number. If the index i is not the LSB, we have to recursively
call the function to decrease the digits from i+1 to LSB till extra
reaches 0.

For example for B=99
When i = 1, extra=4, we print out 99, 98, 97, 96, 95(where 9-4 = 5).
When i = 0, extra=4,
  first: we decrease the 10-digit to 8, now the vector vBEnd has item
of 8, and 9.
    Now we call Solve(A, vBEnd, 1， 3), here i=1, extra = 3.

207. There is a straight road with 'n' number of milestones. You are given an array with the distance between all the pairs of milestones in some random order. Find the position of milestones.

Example:

Consider a road with 4 milestones(a,b,c,d) :

a <--- 3Km --->b<--- 5Km --->c<--- 2Km --->d

Distance between a and b = 3

Distance between a and c = 8

Distance between a and d = 10

Distance between b and c = 5

Distance between b and d = 7

Distance between c and d = 2

All the above values are given in a random order say 7, 10, 5, 2, 8, 3.

The output must be 3,5,2 or 2,5,3

对这个数组里的每一个数，如果它不能表示为该数组里两个数之和，则是一个要求的两个 milestone 之间的距离

208. 找二叉树两个最大的相同子树

不是很理解题意。。。两两比较即可，复杂度 O(N^2),如要加速，可以先计算每个结点的深度和子结点数，相同再进行比较。

209. Given an array of elements find the largest possible number that can be formed by using the elements of the array.

eg: 10 9

ans: 910

2 3 5 78

ans: 78532

100 9

ans: 9100

先按如下规定排序： a > b if ab > ba,

然后再从高到低拼接起来即可。

210. You have an array of 0s and 1s and you want to output all the intervals (i, j) where the number of 0s and numbers of 1s are equal. Example index = 0 1 2 3 4 5 6 7 8 array = 0 1 0 0 1 1 1 1 0 One interval is (0, 1) because there the number of 0 and 1 are equal. There are many other intervals, find all of them in linear time. How can this be done in O(n)?? find all intervals, not find the longest interval.

O(N)好像不可能做到，因为所有的 intervals 可能就是 O(N^2)级别的

211. Rabin Karp Algorithm

略。。。

212. Given a list of presentations with begin and end time that all need to use a conference

room. We want to optimize the utilization of the room by allowing maximum hours of the conference room being used.

DP 题,先把所有的 presentation 按 endtime 排序，然后设 q(endtime)为在 endtime 之间 conference room 被使用的最长时间。然后依次用 presentation 与之间所有的 q 相比，得到新的 q.总体复杂度 O(N^2)

213. Given an array of int, each int appears exactly TWICE in the array. Find and return the int such that this pair of int has the max distance between each other in this array.
e.g. [2, 1, 1, 3, 2, 3]
2: d = 5-1 = 4;
1: d = 3-2 = 1;
3: d = 6-4 = 2;
return 2

用 hashtable 保存 num->index,然后线性扫描一遍即可

214. 二进制加法

```
/**
* i.e.
*
* char a[] = "11";
* char b[] =  "1";
* char *c = bstradd(a, b); // c is a pointer to "100"
**/
```

```
char* add_binary(char* a, char* b)
{
    int sa = strlen(a);
    char* _a = new char[sa];
    for (int i = 0; i < sa; ++i) _a[i] = a[sa - i - 1];

    int sb = strlen(b);
    char* _b = new char[sb];
    for (int i = 0; i < sb; ++i) _b[i] = b[sb - i - 1];

    int sr = (sa > sb ? sa : sb) + 1;
    char* ret = new char[sr+1];

    int sum = 0;
    int carry = 0;
    int j = 0;
    for (j = 0; j < sr; ++j)
    {
        int na = j < sa ? _a[j] - '0' : 0;
        int nb = j < sb ? _b[j] - '0' : 0;
        sum = na + nb + carry;
        carry = sum > 1 ? 1 : 0;

        ret[j] = (carry == 1 ? sum - 2 : sum) + '0';
    }
    if (carry > 0) ret[j++] = carry;

    ret[j] = 0;

    // reverse string
    int begin = 0, end = j - 1;
    while (begin < end)
    {
        char tmp = ret[begin];
        ret[begin] = ret[end];
        ret[end] = tmp;
        ++begin;--end;
    }

    printf("%s\n", ret);

    return ret;
}
```

215. 给你一列单词 / 字符串（内部字符范围:unicode），例如：banana，cat，dog，elephant，type，middle，lake， 让你把这些单词排列成任意相邻单词不能有任何相同字符的序列，如果确定无法满足这个要求，返回 false.

考虑一个图，满足的首尾字母不同即有一条 a->b 的边，那该问题等价于找 Hamilton 圈，是一个 NP 完全问题

216. 判断（二叉）树是否镜像对称

镜像对称的定义？

217. Given a binary tree, find 2 leaf nodes say X and Y such that F(X,Y) is maximum where F(X,Y) = sum of nodes in the path from root to X + sum of nodes in the path from root to Y - sum of nodes in the common path from root to first common ancestor of the Nodes

X and Y

除了穷举，没啥好 idea… 求所有的和 O(N),求最近公共父结点可以优化到 O(1)具体比较为 O(N^2)，总体复杂度为 O(N^2).

218. x^n = y，x/n/y 都是整数，n>1，y 叫做一个啥数来着，姑且叫做 Super Cool 数吧，
比如，1^2 = 1×1=1， 1^3 = 1×1×1 = 1 ...
2^2 = 2×2=4， 2^3 = 2×2×2 = 8 ...
现在给你一个整数 y，请返回最近的那个 Super Cool 数，写 Code。

对每一个小于 sqrt(y)的数，求对数后找一个最接近的即可。。

219. 题目是给定一个 unsorted int array，怎么找到第一个大于 0，并且不在此 array 的 integer。
比如[1,2,0] return 3, [3,4,-1,1] return 2. (time O(n), constant space?)

将第 i 个数放到 i 位，再从头扫描

```
int find_nonexist(int a[], int len)
{
    for (int i = 0; i < len; ++i){
        if (a[i] != i+1)
        {
            int k = a[i];
            while (k <= len && k > 0 && a[k-1] != k)
            {
                int t = a[k-1];
                a[k-1]=k;
                k=t;
            }
        }
    }

    for (int i = 0; i < len; ++i)
        if (a[i] != i+1)
            return (i+1);

    return -1;
}
```

220. 给定一个数字数组（Let's call it count-array），其中每个元素是从末端数小于原数组中该元素的个数。求原数组。原数组中元素是从 1 到 n。
Example:
原数组 4,1, 3, 2
Count array 3, 0, 1, 0
求 nlogn 的算法。

CODE，考虑通过 merge sort 的变形

221 将整型变量 x 中数字左右翻转后存到另外一个整型变量 y 中，例如 x = 12345 时，y 为 54321，x = -123 时，y 为-321。其中 x 的个位不为 0。 void reverse (int x, int* y);

CODE

222. 对集合{1, 2, 3, …, n}中的数进行全排列，可以得到 n!个不同的排列方式。现在我们用字母序把它们列出来，并一一标上序号，如当 n=3 时：

0．123

1．132

2．213

3．231

4．312

5．321

现在，请书写一个函数 void print (int n, int k)， （函数原型是用 C 语言写的，你可以用你熟悉的语言）在已知 n 和序号 k 的情况下，输出对应的排列，并简要阐述思路

```
void print_kth_permutation(int n, int k)
{
    vector<int> nums;
    for (int i = 0; i < n; ++i) nums.push_back(i+1);

    vector<int> result;

    for (int i = 0; i < n; ++i)
    {
        int count = factory(n-i-1);
        int index = k / count;
        k = k % count;

        int _index = 0;
        for (int j = 0; j < n; ++j)
        {
            if (nums[j] != 0){
                if (_index == index)
                {
                    result.push_back(nums[j]);
                    nums[j] = 0;
                    break;
                }
                _index++;
            }
        }
    }

    print_vector(result);
}
```

223．一维数轴上有 n 条线段，它们的端点都是已知的。请设计一个算法，计算出这些线段的并集在数轴上所覆盖的长度，并分析时间复杂度。例如，线段 A 的坐标为[4，8]，线段 B 的坐标为[1，5.1]， 那么它们共同覆盖的长度为 7。 请尽量找出最优化的算法， 解释算法即可，不必写代码。

TO LEARN, INTERVAL TREE

224. 3 sorted arrays, A, B, C, find indexes i, j, k, so that max(a-b, b-c, c-a) is minimized. (a = A[i], b = B[j], c = C[k]) Another version is to minimize max(|a-b|, |b-c|, |c-a|).

IDEA:类似于三路归并，反复地将当前最小的元素往前移，并 Update 目标值即可。

225．一条直线上有 N 个站台，已知任何两点间直达列车的票价，求出从起点到终点的票价最优的乘车方案。因为从 A 到 B，再从 B 到 C 的价格可能比直接从 A 到 C 便宜

DP。。。。

226. N 个 job，要求分配到 M 台机器上，每个机器可以被分配 0-N 个 job，但有些 job 相互排斥不能被放到一起执行，给出所有可能的分配方案

要给出所有方案只能递归穷举吧

227. 给 N 个元素，第 i 个元素有一个大于 0 的 score(i)，要求随机选出 k 个，每个元素可以被选择任意多次，但保证被选择的概率要和 score(i)成比例

将 0-1 之间按 scroe(i)的比例划分成区间，然后生成 0-1 之间的随机数，按其所在区间决定为哪个元素

228. N 个矩形，所有矩形都有一条边在同一条直线上，他们相互可能有 overlap，找出最后得到的这个不规则图形的所有边界点

TO LEARN，

也是 INTERVAL TREE 或者 SEGMENT TREE?先取出所有的点，再把被任意矩形所包含的点去掉。

229. 给两颗树，如果节点深度相同且 value 相同，则这两个 node 是 match 的，两棵树上的节点如果相互 match，则它们的父节点必须也要 match。假设一棵树上所有 node 的 value 都不同，并且兄弟节点间不用考虑顺序，问给两棵树，如何求最大 match 的 node 数目。如果 value 有重复，并且要求兄弟节点 match 的顺序一致，问如何求最大 match 数。

CODE

230. Write a program to find the largest possible rectangle of letters such that every row forms a word (reading left to right) and every column forms a word (reading top to bottom). Words should appear in this dictionary: WORD.LST (1.66MB). Heuristic solutions that may not always produce a provably optimal rectangle will be accepted: seek a reasonable tradeoff of efficiency and optimality. (Hint: Use a B-Tree)
http://www.itasoftware.com/careers/work-at-ita/hiring-puzzles.h

TO LEARN, hard.

231. 直方图盛水

CODE

232. We have been given a deck of cards and each combination has a rating eg 3 As is higher than 2 As 1K etc. Write an algorithm such that the probability of picking 3 or 5 or 7 cards from the deck results in high rating

DP,写递推公式

233. 求一个 unsorted 数组中最长的等差数列（int 数组，可递增或者递减）
http://compgeom.cs.uiuc.edu/~jeffe/pubs/pdf/arith.pdf

难题。。。略

Jump Game:
Given an array start from the first element and reach the last by jumping. The jump length can be at most the value at the current position in the array. Optimum result is when u reach the goal in minimum number of jumps.

For ex:
Given array A = {2,3,1,1,4}
possible ways to reach the end (index list)
i) 0,2,3,4 (jump 2 to index 2, then jump 1 to index 3 then 1 to index 4)
ii) 0,1,4 (jump 1 to index 1, then jump 3 to index 4)
Since second solution has only 2 jumps it is the optimum result.
要求：O(N) time，O(1) space.
类似于 DP？反复 update jump N+1 步后，记录到达当前位置的最小步数

234.一个数组，不考虑重复数字，找出数组中有多少个 cycle。cycle 每个 element 去找他对应的 index，如果 index 在界内，就继续找这个 index 的值所对应的下一个 index 直到找到一个 cycle，或者出界。
比方说 0，2，1，9，10
对于 0， index 0 就是 val 0， 所以是一个 cycle
对于 2， index 2 是 1， 在界内，就找 index 1，which is 2，所以又是一个 cycle
总共 4 个 cycles： 0-->0，2 -> 1，9 -> 界外， 10->界外
先要求写了个可以有 extra space 的，然后要求 no extra space
No extra space 需要修改原数组吧。。。CODE

235. bool isOverlap(int a, int b, int c, int d)参数取值范围 0~6 代表星期几。能不能不用大于号和小于号
用一个七个数的数组来标记就行了吧

236.给一个吸地毯的 irobot，和一个长方形的屋子，四面有墙，四个指令：
Bool moveForward ( ) //向前走一格，走不了的话返回 false
Void Rotate (int degree) //就是左拐右拐
Bool isClean ( ) //当前单元格是否干净
Void clean ( )
把 irobot 扔在屋子任意位置，写代码让 irobot 清理房间，每一格都要走过（单元格没有坐标）
先走到角落，然后来回扫就行了

237. Edit Distance
见 185 题

238. {1，5， -5， -8，2， -1，15 } 要把负的扫到左边，正的扫到后边。不能改变顺序得到{-5 -8 -1 1 5 2 15} 这个题有 time 低于 n^2 space=O（1）的解法吗
应该没有这样的解？

239.给一个通常的表达式，转成后缀表达式
CODE

240. Given n arrays, find n number such that sum of their differences is minimum. For e.g. if there are three arrays
A = {4, 10, 15, 20}
B = {1, 13, 29}

C = {5, 14, 28}
find three numbers a, b, c such that |a-b| + |b-c| + |c-a| is minimum. Here the answer is a = 15, b = 13, and c = 14

同 224 题

241. reverse double linked list.

CODE

242. [Facebook] Given an array A of positive integers. Convert it to a sorted array with minimum cost. The only valid operation are:
1) Decrement with cost = 1
2) Delete an element completely from the array with cost = value of element

DP，写递推公式，使[0,i]中的元素有序，update 时，加入[0,i+1]，要么 decrement 以前的，要么删掉 i+1

243. 在一个平面上有 n 个点，设计算法看能不能找出四个点构成一个正方形，分析时间复杂度。

TO LEARN

244. Algorithm to find the two numbers whose difference is minimum among the set of numbers. For example the sequence is 5, 13, 7, 0, 10, 20, 1, 15, 4, 19
The algorithm should return min diff = 20-19 = 1.
Constraint - Time Complexity O(N) & Space is not a constraint [upto O(3N)]
Assumption - Sorting O(nlogn) & comparison of adjacent numbers is already known & is not an option. Try to keep it linear

想不出来。。。

245. Given an array of integers (both positive and negative) divide the array into two parts (sub-arrays) such that the difference between the sum of elements in each array is minimum?

如果 subarray 连续的话很简单。。。

246. 给一个 string，比如"facebook"，可以拆成"face"和"book"，对任一 string，找出最长的可以拆分成其他单词的子串。

先在 string 里找是单词的范围，得到一组 range,然后将这些 range 按起点排序，再递归+回溯，找到最长的能正好拼接在一起的最长的范围

247. 有 10 个 unsorted array，分给 10 太不同的机器处理，这 10 台机器之间不能通信，但可以和总机通信，如何求总的 median. 如何减少数据量的传输。

TO LEARN，设计题

248. You have an array like ar[]= {1,3,2,4,5,4,2}. You need to create another array ar_low[] such that ar_low[i] = number of elements lower than or equal to ar[i] in ar[i+1:n-1]. So the output of above should be {0,2,1,2,2,1,0} Time complexity : O(n) use of extra space allowed.

等价于前面一个求逆序对的题的吧，感觉不可能做到 O(N)，

249. 给一串数字（比如说 1，4，10，22，30，表示 4 个区间：[1，4]，(4，10]，(10，22]，(22，30]）。现在给很多个数字，要设计一个快速算法，能用最快的速度告诉那些数字分别落在哪个 bucket 那里。比如说前面这个例子输入 double 数 13，算法返回 string: "(10,22]"；输入 double[]序列 13，8，25，返回 string[] "(10,22]" "(4,10]" "(22,30]"

TO LEARN， INTERVAL TREE?

250. Given Numerator and Denominator. After division you might get a recurring decimal points float as the answer. You need to identify the recurring part? For example 23.34563456 ... return 3456

CODE

251. A positive integer is said to be square free if it is divisible by no perfect square larger than 1. For example, the first few squarefree numbers are {1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, …}. Find the nth smallest squarefree number. Note n can be very large such as 1M.

先找出 sqrt(n)以内的所有质数。。再想想

252.判断一个 string 是否是某个 pattern 的周期循环

TO LEARN，后缀树？

253. 在一个 N Dimensional 的正方形里面，Assume the top right point is (n,n,.... n) and bottom left point is (0, 0, 0.... 0), given any point in the cube, find all the paths inside the cube to the (n, n,...n) around it, change its value to 1. Otherwise, mark its value to 0 (cannot recall exactly anymore). how to do it. Given a very big such matrix and n computer, how to do it efficiently. Assume each computer has very limited memory, how to do it.

TO LEARN

254. 给个数组，没排序，已知数组中每个元素距离排序以后的位置最多是 k，让你给这个数组排序

TO LEARN, Shell 排序？

255.两个线段数组，求 common 区间 A[1，5][10，15] B [3,12] return [3,5],[10,12]

TO LEARN, Interval tree?

256. minimum window cover

TO LEARN

257. Given a sorted array of n integers, pick up k elements so that the minimal difference between consecutive elements is maximal (that is choose the elements to maximize the quantity min(a[i+1] - a[i]))

DP 题，写递推公式

数学题

1. [0,1], random generator gives even distributed number (x). How to get a linear distributed series (y), such that the probability of 0 is 1, and the probability of 1 is 0. (that is, p(y) = 1-y)。

2．一个 N*N 的方阵，每个格子里放一个硬币。甲乙轮流取：选定一枚硬币后，必须取走它上方,右方和右上方的所有硬币（如果有的话）。拿走最左下角那个输。问谁有必胜策略？2*N？一般 M*N?

3. Given a 3x3 square:
1 2 3
4 5 6
7 8 9
You are allowed to do circular shift on any row, and circular shift on any column, as many times as you please. Question: can you switch position of 1 and 2 with the allowed circular shifts?

4．两个玩家， 一堆石头，假设多于 100 块，两人依次拿， 最后拿光者赢， 规则是
1．第一个人不能一次拿光所有的
2．第一次拿了之后， 每人每次最多只能拿对方前一次拿的数目的两倍
求先拿者必胜策略，如果有的话

5．100 层楼扔鸡蛋问题

6．有 n 个房间，小偷每天偷一间，偷的规律简单说就是随机行走，如果今天偷了第 i 间屋子，明天有一半的几率偷 i-1，一半的几率偷 i+1，注意如果刚好偷到了边界上，那么第二天只有唯一的选择。如果你是警察，你只能每天选择一个房间蹲守，并且贼的手段相当高明，偷了一个房间后，没有任何人能发觉该房间是否曾经被偷过。

7. Given 3 prime numbers and an integer k, find the kth number if all the nos which are having these 3 prime numbers as their factors are arranged in increasing order.
Eg. prime numbers - 2,3,5
The increasing sequence will be 2,3,4,5,6,8,9...

8．给定一种概率发生器，出 1 的概率是 p,出 0 的概率是 1-p．如果我们需要一个概率都是 1/2 的概率发生器，该怎么做？ 如果是 1/n 的，怎么做？

9．一条直线上有 40 个电线杆，每个间隔 5 米。现在要把 9 个灯泡拧上去，每个电线杆上至多一个，而且不能有三个这样的灯泡，比如说 A，B，C，AB 间的距离等于 BC 间的距离。问这 9 个灯泡有多少种放法？

10. you have a die with 10 sides, number ranging from 1 to 10. Each number comes up with equal possibility. You sum the num you get until the sum is greater than 100. What's the expected value of your sum?

11．今有若干堆火柴，两人依次从中拿取，规定每次只能从一堆中取若干根，
可将一堆全取走，但不可不取，最后取完者为胜，求必胜的方法。

12．今有若干堆火柴，两人依次从中拿取，规定每次只能从一堆中取若干根，可
将一堆全取走，但不可不取，最后取完者为负，求必胜的方法。

13．
Q1：在一个圆上随意选取三个点，三个点在同一半圆上的概率 （3/4）
Q2：在圆上随意选取 ABCD 四个点,AB 和 CD 交叉的概率（1/3）

14．现有 50 个红球，50 个黑球，还有两个空桶。现在把这些球放到两个空桶里面。一个人，随机的从任一个桶中拿一个球出来，问怎么放这些球，使得他拿出红球的概率最大。此人完全不知道桶里面球的分布。如果一个桶是空的，那么他肯定是拿不出红球的。

15．[Google] 两个骰子， 一个是 1-6 的正常骰子，问怎么设置另一个骰子六个面上的数值，使得掷出两个骰子之后的和在 1-12 之内均匀分布。

16．A man speaks the truth 3 out of 4 times. He throws a die and reports it to be a 6. What is the probability of it being a 6?

17．25 匹马，请找出最快的 3 匹。一次只能赛 5 匹，只能知道这 5 匹马的排序。力求用最少的操作。

18．[Microsoft] 一个等边三角形里有 5 个点，有没有可能任意两点的距离都大于二分之一边长，证明。

19．三个鸡蛋硬度[0，1] uniform random distributed.如果两个鸡蛋对敲，其中完好的再和第三个鸡蛋对敲．这个鸡蛋再不碎的概率是多大．

20．假设 rand(0，1)能给出 0-1 的随即直 那么得到 0-0.3 的一个直需要多少次 run ?(expected time)

21．有一排 N 个瓶子（首尾不相接），其中只有一个瓶子是真实的，其他的瓶子都是幻影（看起来和真实的一样），若是摸到幻影，真实的瓶子就会随机的和相邻的左或者右瓶子交换，问能找到一种可靠的方法，找到真实的瓶子？

22．Russian roulette. 2 consective bullets out of 6 slots. A start first, then B, then A, then B,...each round one can decide to turn (reset) or not what's the probability for A to survive for 1st round?
if A survive, should B turn? what's the probability for B to survive for 2ndround?
if A and B both survive 1st and 2nd round, should A turn for 3rd round?
if no one dies for first 3 rounds, should B turn for 4th round?

23．How many different binary trees from n nodes
这道题根算法没啥大关系，有三种可能性
1）如果只是数据不一样，也算不一样的树，那一共有多少种？比如 a-->b ， b-->a 算不同的话。
2）如果两个树可能通过改变 key value 来转换，就算是相同的树，那一共有有多少种？
3）如果两个树可能通过改变 key value 来转换或者通过 mirror 来转换，那一共有有多少种呢？

24．两个 dice，如何 label，使得他们的可以表示 01-31 中的所有数字

25．序列 a，b，c， 如何判断 c 是否是 a 和 b 的 interleave。即，a，b 是 c 的子序列，同时 c 由 a，b 组成。比如，
a = [a1,a2,a3,...,an], b = [b1, b2, b3, ..., bm]
如果 c = [a1,b1, b2, a2，..., an, bm], return true，如果 c = [a2,a1,b1,b2 ,...] return false

26．有一个圆柱体，半径为 R，高为 H．现在要在里面随机产生 N 个半径为 r1，以及 N 个半径为 r2 的小球。要求这些小球必须在圆柱体内，而且互不相交。一共需要 M 个 sample。

27．一个立方体，用 3 种不同的颜色涂，每种颜色都得用。有多少种涂法
http://mathforum.org/library/drmath/view/56240.html

28．有三种颜色的球，红色 13 个，绿色 16 个，黄色 17 个，有一个方法可以使球变色，拿出两个不同颜色的球，就能变成第三种颜色，如拿出一个红色，一个黄色，就会变成两个绿色的球。问有没有可能把这些球变成同一种颜色，如果可能，怎么做，如果不可能，为什么。引申，x 个红球，y 个绿球，z 个黄球，当 x，y，z 满足什么关系时，一定有解决方案，否则无解。

29．在一个 2 维平面上有很多机器人 一共有 2 类 绿色和蓝色的机器人都比较笨 不知道自己的颜色 也不能相互交流现在有一个发信号的机器 可以给所有的机器人发信号 所有的机器人必须执行同样的命令（可以带条件的指令）问如何在 2 个命令下 让绿色 蓝色机器人分组

30.N processors 同时 read M memories，如果有同时两个 processors read 同一个就算一个读写失败。问平均的 read memory 成功的次数

设计题

1．设计文件系统

2．数据结构 for spreadsheet

3．一个 app 需要用 cache，怎么实现 thread safe

4. social network, billions id, every id has about 100 friends roughly, what is max connections between any two ppls. write algorithm to return min connections between two ids: int min_connection(id1, id2)

you can call following functions
expand(id) return friends list of id
expandall(list) return friends union of all the ids in the list
intersection(list1, list2) return intersection
removeintersection(list1, list2)

5. Open google.com, you type some words in the edit box to search something, it will

return lots of search results. Among these returning results (that is, website link), you may only CLICK some results that are interesting to you. The system will record the "CLICK"action. Finally, you will have the search results (i.e. url) and "CLICK" informatin at hand.
Question: how do you find the similarity of these searching?

6. 如何找出最热门的话题(根据 tweets)。如果一个话题一直热门，我们不想考虑怎么办

7. Discuss design challenges of a distributed web crawler running on commercial PCs. How to utilize network bandwidth of those PCs efficiently?

8. Design a site similar to tinyurl.com

9. large log file,含有 customer id, product id, time stamp 想得到在某一天中某个 custom 看网页的次数 1. 足够 memory 2. limited memory

10. 设计一个 actor 和 movie 的数据库的 schema，支持从 movie 得到它的 actors 和从 actor 得到 ta 出现过的 moive (Google, phone, 2006)

11. 某建筑有五十层高,打算装俩电梯,设计该电梯系统

12. how to design facebook's newsfeed?

13. 一个文件里 n 行 m 列，每行是一个 record，每列一个 feature，你时不时要按不同 feature 排序和查找。不能用数据库，文件大小内存能装下，数据结构和算法不限，语言不限，给出你最好的办法。

14. Design online game

15. static 变量用来在整个 class 中共享数据.基于此，各种 synchronization 技术， 以及 busy waiting 的优缺点，啥时候要用基于 busy waiting 的 spinlock 主要是基于性能的探讨。 如果有一个应用程序运行时没有达到 timing constraint， 你如何去分析问题出在哪儿， 可以用什么工具或者技术。

16. 设计题， 有一个多台机器构成的 cluster。 现在有大量公司的数据文件 （并有多个备份）。 如果设计一个算法 ,使得每台机器尽量均衡的使用 ,并且 每个公司文件的不同 copy 不能存在于同一台机器上。主要的 Idea 就是用 round-robin 的方式 assign 每个公司的原数据文件到一台机器，再结合使用 hashtable。 Interviewer 提到我的解法正是他现在在使用的解法。

17. Design a class providing lock function which provide lock only if it sees there are no possible deadlocks.

18. 设计一个分布式文件系统，给定 path name，可以读写文件。具体的 system design 这里就不提了。其中一个细节是，给定 path name，怎么知道哪个 node 拥有这个文件。我提出需要实现一个 lookup function，它可以是一个 hash function，也可以是一个 lookup table。如果是 lookup table，为了让所有 client sync，可以考虑额外做一个 lookup cluster。然后 Interviewer 很纠结，既然可以用

hash function，为什么还搞得那么复杂。我就告诉他 hash function 的缺点。假定一开始有 N 个 node，hash function 把 M 个文件 uniformly distribute 到 N 个 node 上。某天发现 capacity 不够，加了一个 node。首先，要通知所有的 client machine，configuration 改变了。如果不想重启 client machine 的 process，这不是一个 trivial job。其次，文件到 node 的 mapping 也变了。比如，本来按照 hash function，一个文件是放在 node 1。加了一个 node 后，它可能就 map 到 node 2 了。平均来说，N/(N+1)的文件需要 move 到新的 node。这个 data migration 还是很大的。然后我就提出一些 hash function 的 design，可以减少 data migration。最后他提了一个问题，说要实现一个 function，要统计 distributed file system 所有目录的大小。前提是，一个目录下的文件可能放在不同的 node 上。我说这个不就是在每个 node 上统计，然后发到一个 merge 吗。他说对，但是又问用什么 data structure 来表示。我说这就是 hash table，key 就是 directory name，value 就是大小。因为 directory 本身是树结构，这个 hash table 的 key 可以用 tree 来组织。最后让我实现一个 function，把我说得这个 data structure serialize 成 byte array。因为这个 byte array 就是网络传输的 data。我用了 depth first traverse。不过等我程序写完，才发现，用 breath first traverse 会更方便，code 也会很简洁

19. 超大图的存储问题

20. 给个 Lock w/ two atomic method lock() and unlock()，请用 lock 实现一个文件读写的系统，要求：
1: reader blocks writer;
2: writer blocks reader;
3: writer blocks writer;

21。设计一个 web cache server，假设存储网页数量是 10 个 billion，打算怎么设计

22.你可以得到网站访问记录，每条记录有 user IP，写一个程序，要随时能算出过去 5 分钟内访问次数最多的 1000 个 IP.　这个好像跟着这个 rolling window 的 precision 有关，所以我们暂且定为 5 秒钟 update 一次 window

23. Design free and malloc.

24. how to design data structures for a facebook network and how to design an algorithm to find connection? How to optimize it if data is distributed into multiple computers?

25. design a deck class and member function to randomly select a card from those cards which haven't been selected before. (You can assume the number of this function call will never be larger than the number of cards) For example, we have a deck of four card: 1,2,3,4. First it may select 3, then next time it should randomly select one from 1,2,4... And design a member function to reset.

26. google search design problem. How to distribute data and how to design backup system

27. 设计一个 online chat system

28. design bit.ly url shortening web service。算法设计，后端存储，中间层 cache，前端 load

balance，最后是 web analytics。

29. Design and implement an algorithm that would correct typos: for example, if an extra letter is added, what would you do?

30. Suppose there are 2 persons A and B on FB . A should be able to view the pictures of B only if either A is friend of B or A and B have at least one common friend . The interviewer discussed it for nearly 30 minutes . The discussion mainly included following points：

1. How are you going to store the list of friends for a given user?
2. File system vs DB
3. Given list of friends of 2 users, how are you going to find common friends?
4. If you are going to store the friends in DB then how will the table look like?
5. How many servers do you need?
6. How are you going to allocate work to servers?
7. How many copies of data will you need?
8. What problems will you face if you are maintaining multiple copies of data.

31. design structure for auto completion

32. 如何实现 search suggestions。

33. 设计 fb 的系统支持 like 那个 button

34. design 股票#，time，price；
-设计一个 client side 显示股票信息，给出尽可能多的 user case
-在给出的 user case 里面，怎么设计客户端，使得客户段性能提高
-怎么设计 server 端
-数据如何传输
-server 端如何保存数据
-怎么设计 database table 保存数据
-不用 index 怎么提高数据查询速度
-database 是怎么实现数据查询的（要求从 database implementation 角度解释）

开放性问题

1. [Google], map server 用户太多，如何做来提高系统的性能？
    1. ajax,可以不用用户每次都 download 整个 map,他们已经使用
    2. 使用 middleware,来维护数据库链接,并且做 load balance 等等
    3. 使用 distributed system,来使不同的用户使用不同的 server.
       后来居然问怎么来 devide,回答说根据不同的 ip 来区别不同的物理地址,
       居然又追问,还有没有其他方法,回答说解析他们的 httprequest,
       看看所在的不同的时区,不同的语言,国家什么的,blabla
    4. 增加 server 的 memory,提高 cpu 性能,等硬件方面.

5．提高数据库性能,给数据库做 partition,blabla...

最重要的是要先找出性能瓶颈在哪里

2．how to make copy big file to many machines faster

3．写代码的时候，往往会给 array 定一个 max_number，如果现实中有可能出现高于这个数字，怎么测试？比如找出邻近的 wifi 网络数，max_number 再怎么大，总有可能超出，怎么测试呢？而且比如这种情况还很难模拟(很难在现场设几百个网络吧)，怎么测试？不准用动态数组。

4．对于 common sense 的问题，要有所准备，比如为什么要来 FB？为什么离开上一家公司？你有什么改进 FB 产品的建议？等等。

5．一台服务器每过三天就要挂一次，需要重启才能再次使用，每次重启需要一分钟的时间；问有什么方法能解决这个问题。

一些有用的网上资源
http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Sort/

http://www.4shared.com/document/qzFCPK8f/101_Dynamite_Answers_t
http://www.4shared.com/document/SmfrgYjD/Career_-_201_Best_Ques
http://www.4shared.com/document/Eqg6xcuH/Knockout_answers_to_to

http://courses.csail.mit.edu/iap/interview/materials.php

推荐“A Collection of Dice Problems”,面试 facebook 时遇到不少概率题，都不超出这篇文章的思路和难度。

http://www.ibm.com/developerworks/aix/tutorials/au-memorymanage

Lock-free algorithms。推荐
http://www.ibm.com/developerworks/java/library/j-jtp04186/index

排序和 binary search 算法题，推荐 http://www.cs.princeton.edu/introcs/42sort

http://www.cl.cam.ac.uk/~cwc22/hashtable/

C++里创建二维数组，http://www.codeproject.com/KB/cpp/arrayDinamic.aspx

关于 DP http://people.csail.mit.edu/bdean/6.046/dp/

http://www.algolist.net/
http://cslibrary.stanford.edu/

http://www.cs.bell-labs.com/cm/cs/pearls/

http://www.cs.berkeley.edu/~vazirani/algorithms.html

<Programming Interview Exposed>

<Careercup 150问>

<ihas1337code.com>

<geeksforgeeks.org>

<sureinterview.com>

<编程之美>

<程序员面试宝典>

<何海涛面试题精选>

Algorithm to learn:

1. Interval tree, segment tree

```
struct LineNode
{
    int left;
    int right;
    LineNode* lchild;
    LineNode* rchild;
    int count;

    LineNode(int l, int r): left(l), right(r), lchild(0), rchild(0), count(0) {}
};

LineNode* create_line_tree(int l, int r)
{
    if (r - l <= 1)
        return new LineNode(l, r);

    int m = (l + r) / 2;
    LineNode* n = new LineNode(l, r);
    n->lchild = create_line_tree(l, m);
    n->rchild = create_line_tree(m, r);

    return n;
}

void add_line(LineNode* root, int l, int r)
{
    if (l <= root->left && r >= root->right)
    {
        ++root->count;
        return;
    }
    else
    {
        int m = (root->left + root->right) / 2;
        if (l < m)
            add_line(root->lchild, l, r < m ? r : m);
        if (r > m)
            add_line(root->rchild, l > m ? l : m, r);
    }
}
```

2. Trie
3. String search algorithms

Facebook system design question from glassdoor

Design the Facebook Credit system which is a application where users can buy/trade virtual currency and can use the virtual currency to purchase Facebook services, like paid apps.

Design a system to detect typos and provide suggestions to users.

考虑用户的历史数据

http://norvig.com/spell-correct.html

How will you design TinyUrl?

How will you design facebook newsfeed. Focus was on a design which could handle the huge number of status updates and display them on each of the user's friend's wall.

Facebook system design question from careercup

Question: Design a component that implements the following functionality..
1) Record an Event (For the sake of simplicity, treat the Event as an integer code)
2) Return the number of Events recorded in the last one minute.
3) Return the number of Events recorded in the last one hour.
i.e implement the following interface
- Design the interface first
- Give the implementation detail.
<<>>
Open ended question:
What if there isn't enough storage available to store each individual event ?
4

Say you need to design a web application which needs to support friends of friends function(like in linked in, when you search a person, it will show you if this person is linked with you, your connection or your connections' conection...), we expect to have millions of users and each user may have thousands of friends, how would you design/implement this function to make it scalable.

Design Farmville,
Consider only crops and animals for now.
Whats classes will you have?
How will you handle interactions between various objects?

What design patterns can you use?
How will you handle millions of users?

How will you design the backend for facebook. To handle millions of users. Explain the following transactions
1) Adding/Deleting a friend
2) Friend suggestions
What if you cannot store all users on one server?

一些大规模网站会用到的常用组件和技术。
1. Memcached
参见 http://tech.idv2.com/2008/08/17/memcached-pdf/
一些要点如下
一个典型的用例：
初始化一个 Memcache 的对象：

```
$mem = new Memcache;
```

连接到我们的 Memcache 服务器端，第一个参数是服务器的 IP 地址，也可以是主机名，第二个参数是 Memcache 的开放的端口：

```
$mem->connect("192.168.0.200", 12000);
```

保存一个数据到 Memcache 服务器上，第一个参数是数据的 key，用来定位一个数据，第二个参数是需要保存的数据内容，这里是一个字符串，第三个参数是一个标记，一般设置为 0 或者 MEMCACHE_COMPRESSED 就行了，第四个参数是数据的有效期，就是说数据在这个时间内是有效的，如果过去这个时间，那么会被 Memcache 服务器端清除掉这个数据，单位是秒，如果设置为 0，则是永远有效，我们这里设置了 60，就是一分钟有效时间：

```
$mem->set('key1', 'This is first value', 0, 60);
```

从 Memcache 服务器端获取一条数据，它只有一个参数，就是需要获取数据的 key，我们这里是上一步设置的 key1，现在获取这个数据后输出输出：

```
$val = $mem->get('key1');
echo "Get key1 value: " . $val;
```

现在是使用 replace 方法来替换掉上面 key1 的值，replace 方法的参数跟 set 是一样的，不过第一个参数 key1 是必须是要替换数据内容的 key，最后输出了：

```
$mem->replace('key1', 'This is replace value', 0, 60);
$val = $mem->get('key1');
echo "Get key1 value: " . $val;
```

同样的，Memcache 也是可以保存数组的，下面是在 Memcache 上面保存了一个数组，然后获取回来并输出

```
$arr = array('aaa', 'bbb', 'ccc', 'ddd');
$mem->set('key2', $arr, 0, 60);
$val2 = $mem->get('key2');
print_r($val2);
```

现在删除一个数据，使用 delte 接口，参数就是一个 key，然后就能够把 Memcache 服务器这个 key 的数据删除，最后输出的时候没有结果

```
$mem->delete('key1');
```

```
$val = $mem->get('key1');
echo "Get key1 value: " . $val . "<br>";
```

最后我们把所有的保存在 Memcache 服务器上的数据都清除，会发现数据都没有了，最后输出 key2 的数据为空，最后关闭连接

```
$mem->flush();
$val2 = $mem->get('key2');
echo "Get key2 value: ";
print_r($val2);
echo "<br>";
```

在多机并行的时候，有一个严重的问题是在增减机器后，会出现大范围的缓存失效（key 被映射到不同的机器上），解决方法是使用 consistent hashing，介绍在：

http://tech.idv2.com/2008/07/24/memcached-004/#content_2_6

Consistent Hashing 的简单说明

Consistent Hashing 如下所示：首先求出 memcached 服务器（节点）的哈希值， 并将其配置到 0～2^32 的圆（continuum）上。 然后用同样的方法求出存储数据的键的哈希值，并映射到圆上。 然后从数据映射到的位置开始顺时针查找，将数据保存到找到的第一个服务器上。 如果超过 2^32 仍然找不到服务器，就会保存到第一台 memcached 服务器上。

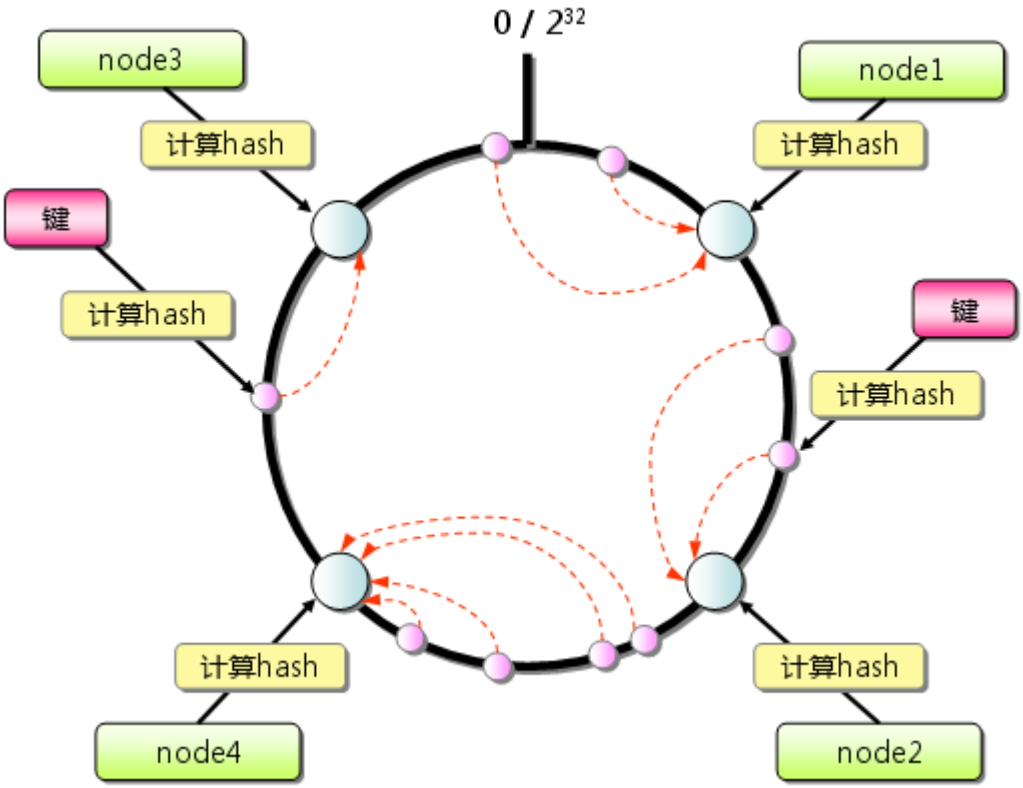

图 4 Consistent Hashing：基本原理

从上图的状态中添加一台 memcached 服务器。余数分布式算法由于保存键的服务器会发生巨大变化 而影响缓存的命中率，但 Consistent Hashing 中，只有在 continuum 上增加服务器的地点逆时针方向的 第一台服务器上的键会受到影响。

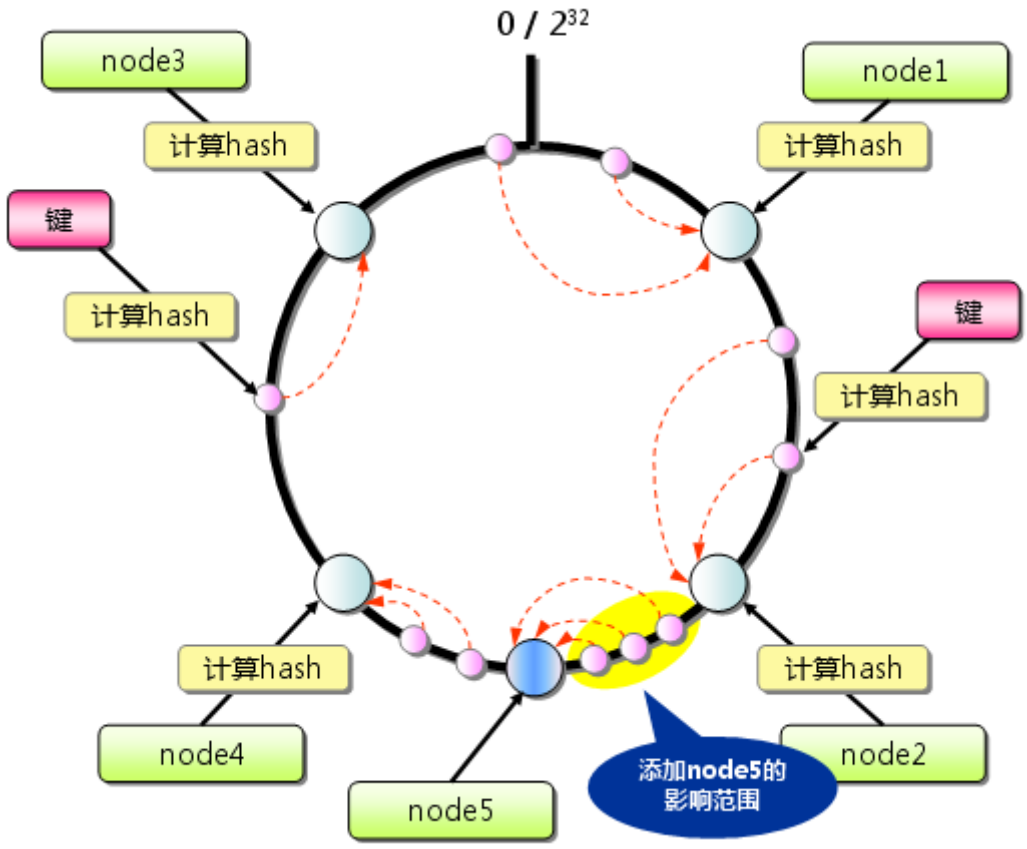

图 5 Consistent Hashing：添加服务器

因此，Consistent Hashing 最大限度地抑制了键的重新分布。 而且，有的 Consistent Hashing 的实现方法还采用了虚拟节点的思想。 使用一般的 hash 函数的话，服务器的映射地点的分布非常不均匀。 因此，使用虚拟节点的思想，为每个物理节点（服务器） 在 continuum 上分配 100～200 个点。这样就能抑制分布不均匀， 最大限度地减小服务器增减时的缓存重新分布。

通过下文中介绍的使用 Consistent Hashing 算法的 memcached 客户端函数库进行测试的结果是， 由服务器台数（n）和增加的服务器台数（m）计算增加服务器后的命中率计算公式如下：

```
(1 - n/(n+m)) * 100
```

另外还有一个需求是将 memcache 中的数据持久化，可以考虑定期持久化。

关于 NoSQL 的文章

http://sebug.net/paper/databases/nosql/Nosql.html