**[TECHIE DELIGHT](https://techiedelight.quora.com/)**

**500 Data Structures and Algorithms practice problems and their solutions**

[Vivek Srivastava](https://www.quora.com/profile/Vivek-Srivastava-266)

**Array:**

1.[Find pair with given sum in the array](http://www.techiedelight.com/find-pair-with-given-sum-array/) \*

问题：在一个数组里找两个数，使其和等于给定的目标数

高效解法：

1）O(nlog) ：先利用nlogn的复杂度排序，然后使用O(n)的复杂度遍历查找设置两个指针low和high；

2）O(n): 需要两次遍历，利用hashing(STL中的unorder\_map是一种hash机制)， 第一遍保存sum － arr[i]到map, 然后再次遍历时查找是否存在等于差值的值;

2. [Check if subarray with 0 sum is exists or not](http://www.techiedelight.com/check-subarray-with-0-sum-exists-not/) \*  
3. [Find sub-array with 0 sum](http://www.techiedelight.com/find-sub-array-with-0-sum/) \*

2,3的问题：检查序列中是否存在和为0的字串数组，或者给出这样的字串

解法：

O(n): ///有一个技巧需要知道：就是如果一个串出现过子串为0的情况，

//那么这个串会存在累加和出现两次的情况，第二次与第一次出现sum相等之间的串为sum＝0的子串，我们只需要利用multimap保存前面的累加和即可

//这相当于，子串中查找等于sum的串的hashing方.

4. [Rearrange the array with alternate high and low elements](http://www.techiedelight.com/rearrange-the-array-with-alternate-high-and-low-elements/) \*  
问题：

Given an array of integers, rearrange the array such that

every second element of the array is greater than its left

and right elements. Assume no duplicate elements are present

in the array.

input:{1,2,3,4,5,6,7}

output:{1,3,2,5,4,7,6}

解法：

1）O(nlog)：先排序，然后一个大一个小地遍历

2）O(n)：*对于每一个偶数位置，我们判断其前后是否都小于它，*//*如果不小就交换*

5. [Sort binary array in linear time](http://www.techiedelight.com/sort-binary-array-linear-time/) \*  
问题： *Sort Binary Array in Linear Time*

*Given an binary array, sort it in linear time and constant space*

*output should print contain all zeroes followed by all ones*

解法：

*1*）用一个指针*index*用于放*0*，另一个索引*i*遍历数组，遇到*0*，则将*arr[index] =0* 等到*i*到头，则将*index*后的值都放*1*。

*2*）利用快排分解操作

*the idea is to use 1 as pivot element and make*

*one pass of partition process。*

其他：

*扩展考虑：*

*将奇偶数组分界，也可利用该函数方式*

6. [Sort an array containing 0’s, 1’s and 2’s (Dutch national flag \* problem)](http://www.techiedelight.com/sort-array-containing-0s-1s-2s-dutch-national-flag-problem/)

**问题**：*sort an array containing 0's,1's,2's,*

*in linear time and in constant space*。

**解法**：假设像 快拍分界一样，我们分三界：先选一个pivot = 1

value less than the pivot

value equal to the pivot and

value greater than the pivot。

然后需要声明三个变量，start， mid，end; start和mid从0开始，mid作为游标滑动，end是从尾部开始。然后start|end开始与mid比较，然后将元素进行划分。

7. [Shuffle a given array of elements (Fisher–Yates shuffle)](http://www.techiedelight.com/shuffle-given-array-elements-fisher-yates-shuffle/) \*

问题：*Shuffle a given array of elements(Fisher-Yates shuffle)*

*Given an array of integers, in-place shuffle it. The*

*should produce an unbiased permutation.*

*every permutation is equally likely.*

解法：

1）Fisher-Yates shuffle:

对于第二个元素选择的概率，是第一次没被选择到 (n - 1) /n

第二次被选到 1 / (n - 1) ,相乘还是1/ n的概率

To shuffle an array a of n elements:

for i from n -1 downto 1 do

j = random integer such that 0<=j<=i

exchange a[j] and a[i]

for(int i = n - 1;i >= 1;i--){

int j = rand() % (i + 1);

swap(a[j],a[i]);

}

2）*// 一个相反的遍历，是index 从0开始，然后随机选择i到n之间的数进行替换*

8. [Find equilibrium index of an array](http://www.techiedelight.com/find-equilibrium-index-array/) \*

问题： ***找出平衡索引，也就是该索引的左半部分和右半部分和相同***

**解法：**

1） //一个linear time solution：利用辅助空间，利用left存储当前索引

//的左边数组和，然后再从right到left进行遍历，且累加右边和，当遇到累加

//等于left时，就发现了一个平衡索引。

2）为了避免用一个额外空间， *先计算总的*sum*。然后遍历数组，使得*total*减去*a[i],*然后累加*left*，进行判断是否相同，这个方式等同于编程之美中的电梯问题的计算。*

*＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃总结：：：*

*当计算左右各半部分的问题时，我们可以使用着这种先总，然后左累加， 且每次用total－左*

*表示右的累加的方式*

*＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃*

*we can avoid using extra space, the idea is to calculate the sum*

*of all elements of the array. then we start from the last lement*

*to the array and maintain right sub-array sum.*

*we can calculate left sub-array sum in constant time by subtracting*

*right sub-array sum and current element from total sum.*

9. [Find majority element in an array (Boyer–Moore majority vote algorithm)](http://www.techiedelight.com/find-majority-element-in-an-array-boyer-moore-majority-vote-algorithm/) \*

问题：*找到主元素即元素个数大于n/2*

解法：

1）// O(n) solutioin:--

/\*

we can use hashing to solve this problem in linear time. the idea

is to store each element's frequency in a map and return element

if its frequency becomes more than n/2, if no such element is present

, then majority element does not exists in the array and we return -1

\*/

10. [Move all zeros present in the array to the end](http://www.techiedelight.com/move-zeros-present-array-end/) \*

问题：*将数组的0元素移到后面*

解法：

1）*the idea is simple, if current element is non-zero, we can place the*

*element to next availiable position in the array,after all element in*

*the array are processed,we fill all remaining indices by 0*。

*2*）*// use partitioning logic quick-sort*

11. [Inplace merge two sorted arrays](http://www.techiedelight.com/inplace-merge-two-sorted-arrays/) \*

问题：*将排序的两个数组合并，使得*X*存储整个合并排序后的前半部分，*Y*为后半部分，*X*，*Y*大小不变（也就是并非真正的融合为一个新的排序数组，而是将合并排序的序列再次分配到*XY*中），注意题目要求的是*in-place,*不能使用辅助的数据结构。 conversion should be done in-place and without using any other data structure。*

*X = {1,4,7,8,10}*

*Y = {2,3,9}*

*output:*

*X = {1,2,3,4,7}*

*Y = {8,9,10}*

解法：

遍历X，使其所有元素都和Y的最小值比较，也就是Y[0]比较。然后如果大于Y最小值，那么将交换，然后再排序Y; 保持顺序，*每次都能使得*X*得到*Y*的最小值，这样*Y*就能保持最后面的值序列。*

12. [Merge two arrays by satisfying given constraints](http://www.techiedelight.com/merge-two-arrays-satisfying-given-constraints/)

问题：Given two sorted arrays x,Y,size m and n, each where m >= n and X[] has exactly n vacant cells, merge element of Y in their correct position in array X,i.e. merge (X,Y) by keeping the sorted order。

不同于11题，这个X是稀疏的数组，他的稀疏部分可以将Y的值填过来。

X = {0,2,0,3,0,5,6,0,0}

Y = {1,8,9,10,15}

output {1,2,3,5,6,8,9,10,15}

解法：//因为merge都元素挨着的，对于X的稀疏元素，我们可以先将它的非零元素对到前面，然后再与Y从后面进行遍历填充X。

13. [Find maximum length sub-array having given sum](http://www.techiedelight.com/find-maximum-length-sub-array-having-given-sum/) \*

问题: *given an array of integers, find maximum length sub-array having given sum*

解法：这种子串和的求解，要考虑用map存储累加的sum和index，然后通过遍历以前是否出现过相同的sum值，来判断子串为sum＝0

We can use map to solve this problem in linear time. The idea

is to create an empty map to store ending index of first subarray

having given sum. We traverse the given array, and maintain

sum of elements seen so far.

If sum is seen for first time, insert the sum with its index into the map.

If (sum – S) is seen before, there exists a sub-array with given sum which end at current index and we update maximum length sub-array having sum S if current sub-array has more length.

time: O(n),maps为hashing。

14. [Find maximum length sub-array having equal number of 0’s and 1’s](http://www.techiedelight.com/find-maximum-length-sub-array-equal-number-0s-1s/)

问题：也就是最长的子串，满足子串中0，1个数相同

Given an binary array containing 0 and 1, find maximum length sub-array having equal number of 0's and 1's

解答：将这个问题转化为子串sum的形式，也就是将子串sum ＝0，这样可以利用map形式，实现O(n)，如何转化为sum形式呢，我们可以将0转化为－1，这样，只有子串0，1数目相等才是的sum＝0.

15. [Find index of 0 to replaced to get maximum length sequence of continuous ones](http://www.techiedelight.com/find-index-0-replaced-get-maximum-length-sequence-of-continuous-ones/)

问题：Given a binary array, find the index of 0 to be replaced with 1 to get maximum length sequence of continuous ones.

就是找一个位置的0，用1替换，使数组能的到最大的连续1子序列

只替换一次

解法：

1) 直接计数，记录max\_count

int findIndexofZero(int a[],int n){

int max\_count = 0;

int max\_index = -1;

int pre\_zero\_index = -1;

int count = 0;

for(int i = 0;i < n;i++){

if(a[i] == 1)

count++;

else{//很巧妙i － pre\_zero\_index

//如果遇到连续为0的情况，count始终为1，表示替换1得到序列长度为1

count = i - pre\_zero\_index;

pre\_zero\_index = i; //记录前0的位置

}

if(count > max\_count){

max\_count = count;

max\_index = pre\_zero\_index;

}

}

return max\_index;

}

2) /\*\*\*\*\*\*\*\*\*solution 2

The idea is very simple. We update each non-zero element of array with count of its adjacent consecutive 1’s. For example, array { 0, 0, 1, 0, 1, 1, 1, 0,1, 1 } is converted to { 0, 0, 1, 0, 3, 3, 3, 0, 2, 2 }. Now the problem reduces to ending the index of 0 whose sum of left and right element is maximum.

Note that this approach will modify the array and at-least requires two traversals of the array. We can use an auxiliary array to avoid modi􀃒cation of original array or restore the original array before returning.

\*\*\*\*/

3) /\*\*\*\*\*\*\*\*\*\*\*solutioin 3:*滑动窗口的方式，*

*如果一个窗口多于*1*个*0*元素，则删除左边元素，重新遍历*

\*/

16. [Find maximum product of two integers in an array](http://www.techiedelight.com/find-maximum-product-two-integers-array/)

问题：*找出array中乘积最大的两个数*.

解法：要明白，我们只需要知道两个最大最小值即可。

1) O(nlog)先快排排序，然后比较最小（可能为负数）的两个product和最大的两个product.

2) O(n)我们用线性time就可以，两次遍历分别找出最大的两个和最小的两个;

17. [Replace each element of array with product of every other element without using division operator](http://www.techiedelight.com/replace-element-array-product-every-element-without-using-division-operator/)

问题: *问题是将每一个元素用其余元素的乘积代替， 而且不能用除法*

Given an array of integers, replace each element of the array

with product of every each element in the array without using

division operator.

解法：

1）we can solve this problem in linear time by using two auxiliary

arrays left[], and right[],

where left[i] stores the product of all elements in the sub-array

A[0..i-1] and right stores the product of elements in the

right sub-array A[i+1..n-1], Now we calculate A[i] = left[i] \* right[i]

2) recursive solution

/\*\*

left:保留的是，i之前i－1个元素的成积，用于在函数中更改a[i] 用的

i：表示当前index的值。

而由于我们要求right，则FindProduct要返回右边所有元素的乘积，

因为是递归方式，所以每次返回curr \* right,既当前元素和其后面元素的乘积；

＃＃＃＃＃

注意：递归中我们在递归中保存需要的值，每次要返回的是什么（返回目标），

返回目标在递归中拆分成每次要返回的值

\*/

int FindProduct2(int a[],int n,int left,int i){

//base case: no elements left on right side

if(i == n){

return 1;

}

//take back-up of current element

int curr = a[i];

//calculate product of the right sub-array

int right = FindProduct2(a,n,left \* curr,i + 1);

//replace current element with product of left and right

a[i] = left \* right;

//return product of right sub-array including current element

return curr \* right;

}

18. [Find a duplicate element in a limited range array](http://www.techiedelight.com/find-duplicate-element-limited-range-array/)

问题：/\*\*\*Find a dulplicate element in a limited range array

Given a limited range array of size n where array contains elments between 1 to n-1 with one element repeating, find the duplicate number in it.

注意题目给出了：n个元素中有1->n-1的数，和一个重复的数，组成

\*/

解法：

1） ///Solutioin 1: auxiliary n-array to memory if it is visited

//time :O(n), space:O(n)

2）将元素本身进行标记，也就是将元素，因为**元素没有超过数组下标（该方法适用的条件）**，切含有的元素数和下标有一一对应关系，所以我们可使用元素映射成下标，对下标的元素进行正负调转，则重复的元素，下标对应的元素位置相同，且被重复调转两次，仍然为正。

3）*用异或性质求解；我们知道元素是*1..n-1,*且有一个重复的，*

*则*1..n-1 xor 1..n-1, *则一定为*0*，再异或重复的元素，为该元素。（因为题目是元素范围是*1->n-1,*所以可以用这种方式，将异或给算出来）*

4）/\*\*\*\*\*solution 4: 我们知道元素是1..n-1,且有一个重复，则我们将数组的和减去1..n-1的和

Finally,

\*\*\*\*/

19. [Find largest sub-array formed by consecutive integers](http://www.techiedelight.com/find-largest-sub-array-formed-by-consecutive-integers/) 子串最大最小值差和下标差相同，且没有重复元素（用map求解）

**问题**：Given an array of integers, find largest subarray formed by consecutive integers, the sub-array should contain all distance value

获得最大长度子串，子串时有连续int数组成，但是不一定要顺序的，只要能所有元素能形成连续的值即可，且子串元素不重复

解法：

/\*\*\*

solution 1: O(n^3), space O(n)

1)判断是否为连续的值，是否没有重复

- the difference between maximum and minimum element in it should be exactly equal to length of the subarray minus one.

- All elements in the array should be distinct (we can check this by inserting the elements in set or using a visited array)

2）开始遍历所有情况

\*/

20. [Find Longest Bitonic Subarray in an array](http://www.techiedelight.com/find-longest-bitonic-subarray-array/)

问题：最大子串，先增后减，先升后降，或者只增或者只降

the longest bitonic subarray problem is to find a subarray of a

given sequence in which the subarray's elements are first sorted

in increasing order,then in decreasing order, and the subarray

is as long as possible, Strictly ascending or descending subarray

are also accepted.

i.e {3,5,8,4,5,9,10,8,5,3,4}

ourput:{4,5,9,10,8,5,3}

解法：

1) /\*\*\*\*solution 1:

the idea is to maintain two arrays I[],D[]

I[i] stores the length of the longest increasing subarray

ending at a[i]

D[i] stores the length of the longest decreasing subarray

starting from a[i]

finally, the length of longest bitonic subarray is maximum among

all (I[i] + D[i] - 1)

time and space : O(n)

\*\*/

2) /\*\*\*\*solution 2:使用两个指针去标注最大bitonic串，如果失效则重新从后计算

we can solve this problem without using extra space, the idea

is to check for longest bitonic subarray starting at a[i],

if longest bitonic subarray starting at a[i] ends at a[j]

the trick is to skip all elements between i and j as longest

bitonic subarray starting at a[j], we continue this process

till end of array is reached and also keep trick of longest

bitonic subarray found so far.

\*\*/ 看下面代码，思路很清晰。根据题意，走马遍历即可。

void Find\_Bitonic(int a[],int n){

int end\_index = 0, max\_len = 0;

int i = 0;

while(i + 1 < n){

int len = 1;

while(i + 1 < n && a[i] < a[i + 1])

i++,len++;

while(i + 1 < n && a[i] > a[i + 1])

i++,len++;

if(len > max\_len){

max\_len = len;

end\_index = i;

}

}

cout<<'['<<end\_index - max\_len + 1<<','<<end\_index<<']'<<endl;

}

21. [Find maximum difference between two elements in the array by satisfying given constraints](http://www.techiedelight.com/find-maximum-difference-between-two-elements-array/)

问题：找出数组中差距最大的一对数字，而且小的数前面

Given an array of integers, find the maximum difference between

two elements in the array such that smaller element appears

before the larger element.

解法：

//solution 2:we solve this problem in linear time.

/\* the idea is to traverse the array from the right and keep

track of maximum difference found so far. if the current element

is less than the maximum element found so far and their difference

is more than maximum difference found so far,then we update

the maximum difference with current difference.

**也就是，始终保持左边遍历过得subarray中最小值的记录，然后不断地比较新遍历的元素与最小值的差值。**

//从左到右也可以，每次记录最小值

\*/

22. [Maximum subarray problem (Kadane’s algorithm)](http://www.techiedelight.com/maximum-subarray-problem-kadanes-algorithm/)

问题：1)Given an array of integers, find contiguous subarray within it

which has the largest sum

2) 将最大和的子串打印，这就需要最大子串的下标区间

解法：

// solution 1:

/\*\*we can easily solve this problem in linear time using kadane's

algorithm. the idea is to maintain maximum (positive sum) subarray

ending ar each index of the given array. the subarray is either

empty (in which case its sum is zero) or consists of one more

element than the maximum subarray ending at the previous index

利用两个变量保存i之前最大的sum以及到i时的当前sum，只有当前sum<0,才开始

重新计算，即将当前sum ＝ 0

time:O(n)

\*/

2）/\*\*\*SOlution 2:

above code doesn't handle the case when all elements of the

array are negative. if the array contains all negative values

the answer is the maximum element. we can easily place this

check before continuing to the algorithm.

即先求出最大元素，如果最大元素<0 ,那么我们直接返回最大元素，说明所有元素<0

\*/

3）// solution 3:

/\* for solution 2: this approach requires two traversals of the

input array. we can easily modify the main algorithm to handle

negative integers as well.

很巧妙地运用了元素之间的比较，将第一种方法的初始化和与0，对比改为与当前

元素对比，这样就能保留最大元素

\*\*/

23. [Print continuous subarray with maximum sum](http://www.techiedelight.com/print-continuous-subarray-with-maximum-sum/)

问题：与22题一样，只是这个要记录索引值

解法:

4）/\*\*\* we can easily solve this problem in linear time using Kadane's

algorithm by maintaining maximum sum subarray ending at each index

of the array, then this subarray is :

1)either empty in which case its sum is zero or

2) consists of one more element than the maximum subarray ending

at the previous index. by tracking the starting and ending

indices of the maximum subarray.

考虑了全为负的情况; 能记住索引

需要在一处判断来更新start\_index

\*/

int print\_kadane(int a[], int n){

int max\_so\_far = a[0];

int max\_ending\_here = a[0];

int starting =0,ending=0;

for(int i = 1;i < n;i++){

max\_ending\_here = max\_ending\_here + a[i];

int cur\_start = starting;

if (max\_ending\_here < a[i]){

cur\_start = i;

max\_ending\_here = a[i];

}

if(max\_so\_far < max\_ending\_here){

starting = cur\_start;

ending = i;

max\_so\_far = max\_ending\_here;

}

}

cout<<'['<<starting<<','<<ending<<']'<<endl;

return max\_so\_far;

}

24. [Maximum Sum Circular Subarray](http://www.techiedelight.com/maximum-sum-circular-subarray/)

问题：/\*\*题目是针对循环的最大和subarray的，，循环的一数组，也就是头尾衔接，

形成一个环，如下数组，4,-1（后面紧接着），2，1，－5 ...

Maximum Sum Circular Subarray

Given an circular array of integers, find subarray in it which has the

largest sum.

For example,

Input: {2, 1, -5, 4, -3, 1, -3, 4, -1}

Output: Subarray with the largest sum is {4, -1, 2, 1} with sum 6.

\*/

解法：

/\*\*\*solution :

The idea is to find the sequence which will have maximum negative value. If

we remove that minimum sum sequence from the input sequence, then we

will be left with maximum sum circular sequence. Finally, we return maximum

of the maximum-sum circular sequence (includes corner elements)

and maximum-sum non-circular sequence.

For example, consider the array {2, 1, -5, 4, -3, 1, -3, 4, -1}. The sequence having

maximum negative value is {-5, 4, -3, 1, -3} i.e. -6. If we remove this minimum

sum sequence from the array, we will get the maximum sum circular

sequence i.e. {2, 1, 4, -1} having sum 6. Since maximum sum circular sequence

is greater than the maximum sum non-circular sequence i.e. {4} for

the given array, it is the answer.

We can find maximum-sum non-circular sequence in linear time by

using Kadane’s algorithm. We can find maximum-sum circular sequence by

inverting the sign of all array elements and then applying Kadane’s algorithm.

For example, if we invert signs of array {2, 1, -5, 4, -3, 1, -3, 4, -1} we get {-2, -1,

5, -4, 3, -1, 3, -4, 1} which has maximum sum sequence {5, -4, 3, -1, 3} having

sum 6. Now inverting the signs back, we get minimum sum sequence {-5, 4,

-3, 1, -3} having sum -6.

方法思路：就是计算非循环下的max sum subarray值

然后再计算出将整个数组除去最小sum后的值，比较两者最大的，为循环数组的最大子序列，（出去最小sum序列的剩余序列＋非循环最大）与非循环最大对比

**（要明白最大子串是没有最小子串的部分的）**

\*\*/

#include "\_22\_23\_Find\_Print\_Max\_Sub\_Sum.h"

#include <algorithm>

int Find\_Max\_Sub\_SumForCircule(int a[],int n){

for(int i = 0;i < n;i++)

a[i] = -a[i];

int neg\_max = kadane3(a,n);

//restore the array

for(int i = 0;i < n;i++)

a[i] = -a[i];

/\* return maximum of

1. sum returned by Kadane's algorithm on original array

2. sum returned by Kadane's algorithm on modified array

sum of all elements of the array.

\*/

return max(kadane3(a,n),accumulate(a,a+n,0) + neg\_max);

}

25. [Find all distinct combinations of given length](http://www.techiedelight.com/find-distinct-combinations-of-given-length/)

问题：find all distinct combinations of given array.

根据给定的组合子串长度，找出不同组合有多少,recurse\_01考虑了重复元素的情况

注意这种01背包问题的思想使用，考虑或者不考虑，两种情况对应着不同处理

这种递归思想一定要牢记

如：{1,2,3} k=2

output：｛1，2｝，｛1，3｝，｛2，3｝

｛1，2，1｝ k＝2

｛1，1｝，｛1，2｝

解法：

１）///solution 1:

/\*\*we can use recursion to solve this problem. the idea is to add

each element into the output and recurse for remaining elements

with one less element. to avoid printing permutations, each combination

will be constructed in same order as array elements. if combination

of given size is found, we print it. Below solution generates all

combinations by using above logic by traversing the array from left

to right. to print only distinct only distinct combinations in

case input contains repeated elements, we can sort the array and

exclude all adjacent duplicate elements from the array.

思想是递归，从左到右选择一个元素后，递归他后面的元素，从中选择k-1个排列， 判断条件是，如果k=0, 则跳出循环，且打印out

1)将重复的元素删除

2）排序，然后依次递归查找

\*/

//再i到n之间查找k长度的元素

//solution 2:

２）/\*\*this problem is very similar to 0/1 knapsack problem where

for each element in given array, we have two cases:

1)Consider that element

2)Don't consider that element

In the solution below, we generate all combinations by using above logic by

traversing the array from left to right. If combination of given size is found,

we print it. To avoid printing permutations, each combination will be constructed

in same order as array elements. To print only distinct combinations

(when input contains repeated elements), we can sort the array and exclude

all adjacent duplicate elements from the array along with the current element

in case 2.

\*/

//该函数考虑了重复元素的情况

void recurse\_01(int a[],vector<int> & out,int i,int n,int k){

//base case: combination size is k

if(k == 0){

printVector(out);

return;

}

if (i == n)

return;

// include current element in the current combination

// 如果考虑该元素的话，就不用跳过其后面的与之重复元素，因为｛1，1，2｝

// ｛1，1｝也是一种combination，所以被当成不同的元素

// 所以我们只需要注意在不考虑该元素时，避免该元素的其他重复项也不被考虑

out.push\_back(a[i]);

recurse\_01(a, out, i + 1, n, k - 1);

// exclude current element

out.pop\_back();

// 如果不算上这个元素的话，我们就要跳过所有该元素重复的项

// 这样就避免了该元素重复而导致的重复combinations

while(i + 1 < n && a[i] == a[i + 1])

i++;

recurse\_01(a, out, i + 1, n, k);

}

26. [Find all distinct combinations of given length with repetition allowed](http://www.techiedelight.com/find-distinct-combinations-given-length-repetition-allowed/)

问题：与25题类似，不过考虑重复元素的组合，Find all disctinct combinations of given length with repetition

allowed.

{1,2,3} -> {1,1},{1,2},{1,3},{2,2},{2,3},{3,3}

{1,2,1} -> {1,1},{1,2},{2,2}

解法：　(首先要排序)

We can use recursion to solve this problem. The idea is to add each element of the array in the output starting from last element considered and recurse　for remaining elements. To avoid printing permutations, each combination　will be constructed in same order as array elements. If combination of given size is found, we print it.

void recurse(int a[],vector<int> & out,int k,int i,int n){

if(out.size() == k){

printVector(out);

return;

}

for(int j = i;j < n;j++){

out.push\_back(a[j]);

recurse(a,out,k,j,n);

out.pop\_back();

// 避免重复元素

while(a[j] == a[j + 1])

j++;

}

}

void Find\_Distinct\_Comb\_Repeat(int a[],int n,int k){

sort(a,a+n);

vector<int> out;

recurse(a,out,k,0,n);

}

27. [Find max sequence of continuous 1’s that can be formed by replacing at-most k zeroes by ones](http://www.techiedelight.com/find-maximum-sequence-of-continuous-1s-can-formed-replacing-k-zeroes-ones/)

问题：/\*\*\*\*\*\*

Find the maximum sequence of continuous 1's that can be formed

by repacing at-most k zeroes by ones

也就是说给定一个k，将binary数组中的k个0替换，能得到最长的连续为1的子串

\*/

解法：//solution:　**（对于给定值的问题，我们需要判断子序列的稳定性，即某些条件是否大于给定的ｋ　＝）　28题和２７题类似，可以总结归纳出一类问题和解法**

/\*\*

We can solve this problem by using sliding window technique. The idea is to

maintain a window containing at-most k zeroes at any point. We add elements

to the window from right until it becomes unstable. The window becomes

unstable if number of zeros in it becomes more than k. Then we remove

elements from its left side till it becomes stable again (by removing leftmost

zero). If the window is stable and current window length is more than

maximum window found so far, we set the maximum window size to current

window size.

\*/

28. [Find minimum sum subarray of given size k](http://www.techiedelight.com/find-minimum-sum-subarray-given-size-k/)

问题：给定长度k，求出数组中子串sum最小的且长度为k的子串

problem:

Given an array of integers, find minimum sum sub-array of given

size k.

解法：/\*\*\*solution :　k大小的窗口向右滑动

We can solve this problem by using sliding window technique. The idea is to　maintain a window of size k and for every element in the array, we include it　in the window and remove leftmost element from the window if window size　is more than k. We also maintain sum of elements in current window. If current　window sum is more than minimum found so far, we update the minimum　sum to current window sum and store window’s end-points.

\*/

29. [Find subarray having given sum in given array of integers](http://www.techiedelight.com/find-subarray-having-given-sum-given-array/)

问题：\*\*　(和１，２题类似)

Find subarray having given sum in given array of integers

Given an array of integers, find a subarray having given sum in it

\*/

解法：

１）// Solution 1:

//using sliding window

/\*注意：我们进行滑窗操作，如果滑窗大于sum则我们开始将左边元素删减，如果小于sum

则认为滑窗进入不稳定状态，开始加元素

We can solve this problem by using a sliding window. The idea is to maintain

a window that starts from the current element and sum of its elements is

more than or equal to the given sum. If current window’s sum becomes less

than the given sum, then the windows is unstable and we keep on adding elements

to the current window from its right till the window becomes stable

again. We print the window if it’s sum is equal to the given sum at any point

of time. This approach will only work on positive sum.

\*/

//这种方法不能处理存在负元素的情况

２）用hash存储，将和转化为查找相同元素的问题上来

//解决方法2:

/\*\*\*将array累加，查找当前sum－15是否存在前面存储的hashing中

Above solution will fail for negative numbers. We can use hashing to check if　sub-array with given sum exists in the array or not. The idea is to traverse the given array and maintain sum of elements seen so　far. If the di􀃗erence of current sum and given sum is seen before (i.e.　the di􀃗erence exists in the set), we return true as there exists at-least one sub-array with given sum　which ends at current index else we insert the sum into the set.

\*/

30. [Find the length of smallest subarray whose sum of elements is greater than the given number](http://www.techiedelight.com/length-of-smallest-subarray-with-sum-greater-number/)

问题:　找到最小长度大于等于给定sum的子串

Given an array of integers, find the length of smallest subarray

whose sum of elements is greater than the given positive number.

解法:　solution1:

We can solve this problem by using a sliding window. The idea is to maintain

a window that ends at the current element and sum of its elements is less

than or equal to the given sum. If current window’s sum becomes more than

the given sum at any point of time, then the window is unstable and continue

removing elements from the window’ left till it becomes stable again. We also

update the result if unstable window’s length is less than minimum found so

far.

31. [Find largest number possible from set of given numbers](http://www.techiedelight.com/find-largest-number-possible-set-given-numbers/)

问题：将一个数组的元素拼接成一个数字，使其最大：

Find largest number possible from set of given numbers. The numbers

should be appended to each other in any order to form the

largest number.

解法：

　０）错误思路：　简单的排序存在错误

Simply sorting the array in descending order and considering the sorted order

is not a possibility here as sorted array {75, 68, 21, 12, 10, 7} will result in

number 75682112107 which is less than largest number possible

77568211210.

１）我们需要自定义比较函数

The idea is to write our own custom comparator function for the sorting routine.

For two numbers X and Y, the custom comparator function will not compare

X and Y with each other but it compares XY with YX and the greater

number will come firrst in sorted order. Here, XY denotes number formed by

appending Y to X and YX denotes number formed by appending X to Y.

For example, for X = 15 and Y = 4, XY = 154 and YX = 415.

As evident from above example, X > Y but XY < YX, so the comparator function

will consider Y > X.

32. [Find the smallest window in array sorting which will make the entire array sorted](http://www.techiedelight.com/smallest-window-sorting-which-make-array-sorted/)

问题：/\*\*找到最小的窗口，对小窗口排序使得整个数组是增序的

Find the smallest window in array sorting which will make the

entire array sorted.

For example,

Input: { 1, 2, 3, 7, 5, 6, 4, 8 }

Output: Sort the array from index 3 to 6

Input: { 1, 3, 2, 7, 5, 6, 4, 8 }

Output: Sort the array from index 1 to 6

\*/

解法： We can easily solve this problem in linear time. Below is the complete algorithm

–算法思路：就是先从左到右遍历，遍历到比遍历过的最大元素小的最右边的元素j，

从右到左也是，遍历到比遍历过最小元素大的最左边的元素i，这样就保证了两个端点i，j外是排序的

不用再进行sort。

1. Traverse array from left to right keeping track of maximum so far and

note the last encountered index j which is less than the maximum so far.

2. Traverse array from right to left keeping track of minimum so far and

note the last encountered index i which is more than the minimum so

far.

3. Finally we sort the array from index i to j

For example, consider below array

{ 1, 2, 3, 7, 5, 6, 4, 8 }

If we traverse the array from left to right, the last encountered index which is

less than the maximum so far is 6（8比7大所以最右边比7小的是6）. Similarly, if we traverse the array from

right to left, the last encountered index which is more than the minimum so

far is 3. So, we need to sort the array from index 3 to 6.

33. [Find maximum sum path involving elements of given arrays](http://www.techiedelight.com/find-maximum-sum-path-involving-elements-given-arrays/)

问题：Given two sorted array of integers, find a maximum sum path involving

elements of both arrays whose sum is maximum. We can

start from either arrays but we can switch between arrays only

through its common elements.

For example,

Input:

X = { 3, 6, 7, 8, 10, 12, 15, 18, 100 }

Y = { 1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50 }

The maximum sum path is

1 —> 2 —> 3 —> 6 —> 7 —> 9 —> 10 —> 12 —> 15 —> 16 —> 18 —> 100

解法：

/\*\*solution 1:

The idea is very simple. We calculate sum between common elements

present in the both arrays and include the maximum sum in the output.

For example, consider below arrays X and Y having four common elements

A, B, C, D.

X[]: sum\_x1 .., A, .. sum\_x2 .., B, .. sum\_x3 .., C, .. sum\_x4 .., D, .. sum\_x5

Y[]: sum\_x1 .., A, .. sum\_y2 .., B, .. sum\_y3 .., C, .. sum\_y4 .., D, .. sum\_y5

Here sum\_xi denotes the sum of elements between two common elements in

array X. Similarly, sum\_yi denotes the sum of elements between two common

elements in array Y. For each pair (sum\_xi, sum\_yi), we include

max(sum\_xi, sum\_yi) in the solution. i.e.

Result = max(sum\_x1, sum\_y1) + A + max(sum\_x2, sum\_y2) + B +

max(sum\_x3, sum\_y3)

+ C + max(sum\_x4, sum\_y4) + D + max(sum\_x5, sum\_y5)

解决思路：每次累加X，Y相同元素前的部分，然后比较，选取最大的那个作为路径一部分

主要的是设计怎么获取相同元素之前的累加和，这就是比较XY当前元素，选较小的累加到

相应的数组中去，也就是小的先走到相同的元素部分，这样能使得XY元素相等

相遇。这时候统计前面加的部分，选取最大的就可以了。

这是个编程技巧，很棒的

\*/

34. [Maximum profit earned by buying and selling shares any number of times](http://www.techiedelight.com/maximum-profit-earned-buying-and-selling-shares)

问题：Maximum profit earned by buying and selling shares any number

of times.

Given a list containing future prediction of share prices, 􀃒nd maximum

profit that can be earned by buying and selling shares any

number of times with constraint that a new transaction can only

start after previous transaction is complete. i.e. we can only hold atmost one share at a time.

For example,

rate[]: {1, 5, 2, 3, 7, 6, 4, 5}

Total profit earned is 10

Buy on day 1 and sell on day 2

Buy on day 3 and sell on day 5

Buy on day 7 and sell on day 8

解法：

/\*solution 1: **也就是找序列中的所有增序子序列**。

The idea is very simple. We traverse the given list of prices and 􀃒find local minimum of every increasing sequence. For example, in the array {1, 5, 2, 3, 7, 6, 4, 5}, below are three increasing sequences of length 2 or more.

{1, 5}

{2, 3, 7}

{4, 5}

The local minimum of each sequence is 1, 2 and 4 respectively. We can gain maximum profit if we buy the shares at the starting of every increasing sequence (local minimum) and sell them at the end of the increasing sequence

(local maximum).

\*/

int Find\_Max\_Profit(int a[],int n){

int profit = 0;

int j = 0;

for(int i = 1;i < n;i++){

if(a[i - 1] > a[i]){

j = i;

}

if(a[i - 1] < a[i] && (i + 1 == n || a[i] > a[i + 1])){

profit += a[i] - a[j];

cout<<"Buy on day"<<j<<"and sell on day"

<<i<<endl;

}

}

return profit;

}

35. [Trapping Rain Water within given set of bars](http://www.techiedelight.com/trapping-rain-water-within-given-set-bars/)

问题：也就是根据这些桩子，找出这些桩子组成的矩形槽能容纳最多的水。

具体视图看文件夹中的图片Rain-Water-Trapping.png

Trapping Rain Water within given set of bars

In trapping rain water problem, we need to find the maximum

amount of water that can be trapped within given set of bars where

width of each bar is 1 unit.

解法：

/\*\*\*我们可以观察到，一个bar的水位，跟他左右两边最大bar中最小的一个bar有关

因此，我们可以找出当前元素左边和右边的最大的bar，然后取较小的减去当前

bar的高度即可

The idea is to calculate maximum height bar on the left and right of every bar. Then the amount of water that can be stored on top of each bar is equal to minimum among maximum bar to the left and right minus height of current bar.

\*/

36. [Longest Increasing Subsequence](http://www.techiedelight.com/longest-increasing-subsequence/)

问题：/\*查找最长增序的子序列，每个元素不一定连续

Longest Increasing Subsequence

The longest increasing subsequence problem is to find a subsequence of a

given sequence in which the subsequence’s elements are in sorted order, lowest to highest,

and in which the subsequence is as long as possible.

This subsequence is not necessarily contiguous, or unique.

For example, consider subsequence {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15}

The Longest increasing subsequence is {0, 2, 6, 9, 11, 15}

This subsequence has length 6; the input sequence has no 7-member increasing subsequences.

The longest increasing subsequence in this example is not unique: for instance,

{0, 4, 6, 9, 11, 15} or

{0, 4, 6, 9, 13, 15}

are other increasing subsequences of equal length in the same input sequence.

\*\*/

解法：

We have already discussed a O(n2) time complexity solution of LIS here

which uses Dynamic Programming. In this post, a O(nlogn)

time Non-DP solution is discussed.

Let S[i] be defined as the smallest integer that ends an increasing sequence of length i.

Now iterate through every integer X of the input set and do the following:

If X is more than the last element in S, then append X to the end of S.

This essentially means we have found a new largest LIS.

Otherwise find the smallest element in S, which is more than or equal to X,

and replace it with X. Because S is sorted at any time, the element can be found

using binary search in log(N) time.

Let’s illustrate this with the help of a example. Consider below array of integers –

{2, 6, 3, 4, 1, 2, 9, 5, 8}

Below are the steps followed by the algorithm –

Initialize to an empty set S = {}

Inserting 2 —- S = {2} – New largest LIS

Inserting 6 —- S = {2, 6} – New largest LIS

Inserting 3 —- S = {2, 3} – Replaced 6 with 3

Inserting 4 —- S = {2, 3, 4} – New largest LIS

Inserting 1 —- S = {1, 3, 4} – Replaced 2 with 1

Inserting 2 —- S = {1, 2, 4} – Replaced 3 with 2

Inserting 9 —- S = {1, 2, 4, 9} – New largest LIS

Inserting 5 —- S = {1, 2, 4, 5} – Replaced 9 with 5

Inserting 8 —- S = {1, 2, 4, 5, 8} – New largest LIS

So, the length of the LIS is 5 (the size of S). Please note that here S[i] is defined as the smallest integer that ends an increasing sequence of length i. Therefore, S does not represent an actual

sequence but the size of S represents the length of the LIS.

Below solution uses std::set which is implemented as a red-black binary search tree which has a worst-case time complexity of O(logn) for insertion.

注意：这个算法只是计算了长度，实际的元素不是最大增长子序列；

这里的替换策略是，１）可以替换已遍历序列中的最大值前面的值，且这个新添加的值要小于最大值，２）以及可以替换为比最大值小，但是比前面的都大的值也能替换到temp数组里

\*\*/

２）/\*\*\*

How to print LIS?

To make things simpler, we can keep in the S, not the actual integers, but their indices in the set.

That is we do not keep {1, 2, 4, 5, 8}, but keep {4, 5, 3, 7, 8} since arr[4] = 1, arr[5] = 2,

arr[3] = 4, arr[7] = 5 and arr[8] = 8.

To reconstruct the actual LIS we have to use a parent array. Let parent[i] be the predecessor of element with index i in the LIS ending at element with index i. If we update properly the parent array, the actual LIS is:

arr[S[lastElementOfS]],

arr[parent[S[lastElementOfS]]],

arr[parent[parent[S[lastElementOfS]]]],

…………………………………

Below solution stores both actual integers and their indices in the set for easier implementation –

相当于用并查集纪录替换索引的值，然后追溯出真正的增序序列；

\*\*/

struct Node

{

int elem;

int index;

};

// overload compare operator for inserting into set

inline bool operator<(const Node& lhs, const Node& rhs)

{

return lhs.elem < rhs.elem;

}

// Function to print LIS using parent array

void print(int input[], auto parent, set<Node> S)

{

// container to store LIS in reverse order

stack<int> lis;

// start from the last element of S

int index = S.rbegin()->index;

// get length of LIS

int n = S.size();

// retrieve LIS from parent array

while (n--)

{

lis.push(input[index]);

index = parent[index];

}

// print LIS

cout << "LIS is ";

while(!lis.empty())

{

cout << lis.top() << " ";

lis.pop();

}

}

// Function to find Longest Increasing Subsequence in given array

void printLIS(int arr[], int n)

{

// create an empty ordered set S (ith element in S is defined as the

// smallest integer that ends an increasing sequence of length i)

set<Node> S;

// parent[i] will store the predecessor of element with index i in the

// LIS ending at element with index i

map<int, int> parent;

// process every element one by one

for (int i = 0; i < n; i++)

{

// construct node from current element and its index

Node curr = {arr[i], i};

// insert current node into the set and get iterator to the

// inserted node

auto it = S.insert(curr).first;

// if the node is not inserted at the end, then delete the next node

if (++it != S.end())

S.erase(it);

// get iterator to the current node and update parent

// 重新查找it，因为插入数据，使的set变化

it = S.find(curr);

parent[i] = (--it)->index;

}

// print LIS using parent map

print(arr, parent, S);

}

37. [Find maximum product subarray in a given array](http://www.techiedelight.com/find-maximum-product-subarray-given-array)

问题：/\*\*找出最大成绩的子序列，子序列时连续的

Given an array of integers, find sub-array in it that has maximum

product of its elements.

For example,

Input: { -6, 4, -5, 8, -10, 0, 8 }

Output: The maximum product sub-array is {4, -5, 8, -10} having product

1600

Input: { 40, 0, -20, -10 }

\*/

解法：

1) /\*\*\*naive:

Naive solution would be to consider every sub-array and find product of their

elements. Finally, we return the maximum product found among all sub-arrays.

The implementation can be seen here. The time complexity of this solution

is O(n ).

\*/

/\*\*因为有负数存在，所以我们说过最大和最小标签纪录当前元素前的最大成绩

最小的为绝对值最大的负数，所以再遇到负元素，可能存在最大成绩，所以不能只计算

当前最大成绩

A better solution will be to maintain two variables to store the maximum and

minimum product ending at current position. Then we traverse the array

once and for every index i in the array, we update maximum and minimum

product ending at A[i]. We update the result if maximum product ending at

any index if more than maximum product found so far.

\*\*/

38. [Find maximum sum of subsequence with no adjacent elements](http://www.techiedelight.com/maximum-sum-of-subsequence-with-no-adjacent-elements)

问题：Find Maximum sum of subsequence with no adjacent elements

Given an array of integers, find the maximum sum of subsequence of given array where subsequence contains no adjacent elements

解法：/\*\*\*解法1:思想类似于0/1背包问题，

通过递归的方式对元素取活着不取操作，惯用写法，分两种情况递归下去

The problem is similar to 0/1 Knapsack problem where for every item, we have two choices – to include that element in the solution or to exclude that element from solution. We can solve this problem by following the same logic. The only difference is that we include current element only if it is not adjacent to previous element considered.

\*\*/

２）/\*\*\*\*

解法2:也可以使用自底向上的思维方式，利用动态规划，每次计算走到cur\_index时的最大和

利用动态规划避免了递归的溢出和代价

We can also solve this problem in bottom-up fashion using Dynamic Programming (Tabulation). In the bottom-up approach, we solve smaller sub-problems first, then solve larger sub-problems from them. The idea is to create an auxiliary array lookup[] to store solution of sub-problems where for each index i, lookup[i] stores the maximum value that can be attained till index i. It uses value of smaller values i already computed. It has the same asymptotic run-time as Memoization but no recursion overhead.

\*\*\*/

int MaxSubSequence2(int a[],int length){

//base case

if (length == 1)

return a[0];

//create an axiliary array to store solution of sub-problems

int look\_up[length];

//look\_up[i] stores the maximum sum possible till index i

//trivial case

look\_up[0] = a[0];

look\_up[1] = max(a[0],a[1]);

for(int i = 2;i < length;i++){

/\*\*

// 1. excluding current element & take maximum sum till index i-1

// 2. including current element arr[i] and take maximum sum

// till index i-2

\*/

look\_up[i] = max(look\_up[i - 1],look\_up[i - 2] + a[i]);

//element can be a negative value

look\_up[i] = max(look\_up[i],a[i]);

}

return look\_up[length - 1];

}

３）/\*\*\*解法3:可以看出，上面时间复杂度O（n），但是都是需要辅助空间的，

那第三种解法能否不使用辅助空间呢，我们从第二种解法中可以看出，我们只需要保存前面index的最大总和以及前面的前面index总和

The time complexity of above solution is O(n) and auxiliary space used by the program is O(n).

Above solution uses extra space. We can also solve this problem without using any extra space. If we analyze the solution, we can see that maximum sum till any index i can be found by knowing the maximum sum of previous index i-1 and index i-2. So instead of storing the complete array, we can maintain two variables that stores the maximum sum till previous index and previous to previous index.

\*\*\*/

int MaxSubSequence3(int a[],int length){

if (length == 1)

return a[0];

int pre\_pre = a[0];// 前前最大

int pre = max(a[0],a[1]);//前最大

for(int i = 2;i < length;i++){

//注意：要把a[i]本身最大的情况考虑，避免了负数的元素情况

int cur\_sum = max(a[i],max(pre,pre\_pre + a[i]));

pre\_pre = pre;

pre = cur\_sum;

}

return pre;

}

39. [Decode the array constructed from another array](http://www.techiedelight.com/decode-the-array-constructed-another-array/)

问题：/\*Decode the array constructed from another array

Given an array constructed from another array by

taking sum of every distinct pair in it, decode

the array to get back the original array elements.

If the original array is A[0], A[1], . . , A[n-1], then the input array is

{ (A[0] + A[1]), (A[0] + A[2]), . . , (A[0] + A[n-1]),

(A[1] + A[2]), (A[1] + A[3]), . . , (A[1] + A[n-1]),

..

..

(A[i] + A[i+1]), (A[i] + A[i+2]), . . , (A[i] + A[n-1]),

..

..

(A[n-2] + A[n-1]) }

For example,

Input: { 3, 4, 5, 5, 6, 7 }

Output: { 1, 2, 3, 4 }

Input: { 3, 4, 5, 6, 5, 6, 7, 7, 8, 9 }

Output: { 1, 2, 3, 4, 5 }

Input: { 3 }

Output: { 1, 2 } or { 2, 1 }

Input: {3, 4, 5 }

Output: { 1, 2, 3 }

\*/

解法：

解码一个数组，该数组是由原始数组的每一项一次叠加的，如上式显示：

首先我们可以计算出来原始数组n的个数，然后由叠加规律可知，该数组

inp[n-1]对应着原始数组A1+A2，而inp[0]和inp[1]等于 2\*A0+A1+A2

因此我们可以计算出A0 ＝ inp0+inp1 - inp[n - 1],

又由于n^2 - n - 2m = 0 ---> n = (sqrt(8 \* m + 1) + 1)/ 2

40. [Find Triplet with given sum in an array](http://www.techiedelight.com/find-triplet-given-with-given-sum/)  
41. [Find min number of platforms needed in the station so to avoid any delay in arrival of any train](http://www.techiedelight.com/minimum-number-of-platforms-needed-avoid-delay-arrival-train)  
42. [Rearrange array such that A[A[i]] is set to i for every element A[i]](http://www.techiedelight.com/rearrange-array-such-that-array-index-is-set-to-i/)  
43. [Reverse every consecutive m elements of the given subarray](http://www.techiedelight.com/reverse-every-consecutive-m-elements-given-subarray/)  
44. [Maximum Product Subset Problem](http://www.techiedelight.com/maximum-product-subset-problem/)  
45. [Find pairs with given difference k in the array](http://www.techiedelight.com/find-pairs-with-given-difference-array/)  
46. [Find pairs with given difference k in the array | Constant space solution](http://www.techiedelight.com/find-pairs-given-difference-k-array-constant-space-solution/)  
47. [Quickselect Algorithm](http://www.techiedelight.com/quickselect-algorithm/)  
48. [4 sum problem | Quadruplets with given sum](http://www.techiedelight.com/4-sum-problem/)  
49. [Print all quadruplets with given sum | 4-sum problem extended](http://www.techiedelight.com/print-all-quadruplets-with-given-sum-4-sum-problem-extended/)  
50. [Find odd occurring element in an array in single traversal](http://www.techiedelight.com/find-odd-occurring-element-array-single-traversal/)  
51. [Find two odd occurring element in an array without using any extra space](http://www.techiedelight.com/find-two-odd-occurring-element-array-without-extra-space/)  
52. [Print all Triplets that forms Arithmetic Progression](http://www.techiedelight.com/print-triplets-forms-arithmetic-progression/)  
53. [Print all triplets that forms Geometric Progression](http://www.techiedelight.com/print-triplets-forms-geometric-progression/)  
54. [Merging Overlapping Intervals](http://www.techiedelight.com/merging-overlapping-intervals)  
55. [Activity Selection Problem](http://www.techiedelight.com/activity-selection-problem/)  
56. [Job Sequencing Problem with Deadlines](http://www.techiedelight.com/job-sequencing-problem-deadlines/)  
57. [Introduction to Priority Queues using Binary Heaps](http://www.techiedelight.com/introduction-priority-queues-using-binary-heaps/)  
58. [Min Heap and Max Heap Implementation in C++](http://www.techiedelight.com/min-heap-max-heap-implementation-c/)  
59. [Min Heap and Max Heap Implementation in Java](http://www.techiedelight.com/min-heap-max-heap-implementation-in-java/)  
60. [Heap Sort (Out-of-place and In-place implementation in C++ and C)](http://www.techiedelight.com/heap-sort-place-place-implementation-c-c/)  
61. [Check if given array represents min heap or not](http://www.techiedelight.com/check-given-array-represents-min-heap-not/)  
 [Convert Max Heap to Min Heap in linear time](http://www.techiedelight.com/convert-max-heap-min-heap-linear-time/)  
 [Find K’th largest element in an array](http://www.techiedelight.com/find-kth-largest-element-array/)  
 [Sort a K-Sorted Array](http://www.techiedelight.com/sort-k-sorted-array/)  
 [Merge M sorted lists of variable length](http://www.techiedelight.com/merge-m-sorted-lists-variable-length/)  
 [Find K’th smallest element in an array](http://www.techiedelight.com/find-kth-smallest-element-array/)  
 [Find smallest range with at-least one element from each of the given lists](http://www.techiedelight.com/find-smallest-range-least-one-element-given-lists/)  
 [Merge M sorted lists each containing N elements](http://www.techiedelight.com/merge-m-sorted-lists-containing-n-elements/)  
 [Insertion sort | Iterative & Recursive](http://www.techiedelight.com/insertion-sort-iterative-recursive/)  
 [Selection sort | Iterative & Recursive](http://www.techiedelight.com/selection-sort-iterative-recursive/)  
 [Bubble sort | Iterative & Recursive](http://www.techiedelight.com/bubble-sort-iterative-recursive/)  
 [Merge Sort](http://www.techiedelight.com/merge-sort/)  
 [Quicksort](http://www.techiedelight.com/quicksort/)  
 [Iterative Implementation of Quicksort](http://www.techiedelight.com/iterative-implementation-of-quicksort/)  
 [Quicksort using Dutch National Flag Algorithm](http://www.techiedelight.com/quicksort-using-dutch-national-flag-algorithm/)  
 [Quick Sort using Hoare’s Partitioning scheme](http://www.techiedelight.com/quick-sort-using-hoares-partitioning-scheme/)  
 [Hybrid QuickSort](http://www.techiedelight.com/hybrid-quicksort/)  
 [External merge sort](http://www.techiedelight.com/external-merge-sort/)  
 [Sort an array using one swap](http://www.techiedelight.com/sort-array-using-one-swap/)  
 [Custom Sort | Sort elements by their frequency and Index](http://www.techiedelight.com/sort-elements-by-their-frequency-and-index/)  
 [Custom Sort | Sort elements of the array by order of elements defined by the second array](http://www.techiedelight.com/custom-sort-sort-elements-array-order-elements-defined-second-array/)  
 [Inversion Count of an array](http://www.techiedelight.com/inversion-count-array/)  
 [Segregate positive and negative integers in linear time](http://www.techiedelight.com/positive-and-negative-integers-segregate/)  
 [Binary Search](http://www.techiedelight.com/binary-search/)  
 [Ternary Search vs Binary search](http://www.techiedelight.com/ternary-search-vs-binary-search/)  
 [Interpolation search](http://www.techiedelight.com/interpolation-search/)  
 [Exponential search](http://www.techiedelight.com/exponential-search/)  
 [Find number of rotations in a circularly sorted array](http://www.techiedelight.com/find-number-rotations-circularly-sorted-array/)  
 [Search an element in a circular sorted array](http://www.techiedelight.com/search-element-circular-sorted-array/)  
 [Find first or last occurrence of a given number in a sorted array](http://www.techiedelight.com/find-first-or-last-occurrence-of-a-given-number-sorted-array/)  
 [Count occurrences of a number in a sorted array with duplicates](http://www.techiedelight.com/count-occurrences-number-sorted-array-duplicates/)  
 [Find smallest missing element from a sorted array](http://www.techiedelight.com/find-smallest-missing-element-sorted-array/)  
 [Find Floor and Ceil of a number in a sorted array](http://www.techiedelight.com/find-floor-ceil-number-sorted-array/)  
 [Search in a nearly sorted array in O(logn) time](http://www.techiedelight.com/search-nearly-sorted-array-ologn-time/)  
 [Find number of 1’s in a sorted binary array](http://www.techiedelight.com/find-number-1s-sorted-binary-array/)  
 [Find the peak element in an array](http://www.techiedelight.com/find-peak-element-array/)  
 [Maximum Sum Subarray using Divide & Conquer](http://www.techiedelight.com/maximum-sum-subarray-using-divide-conquer/)  
 [Find Minimum and Maximum element in an array using minimum comparisons](http://www.techiedelight.com/find-minimum-maximum-element-array-using-minimum-comparisons/)  
 [Matrix Chain Multiplication](http://www.techiedelight.com/matrix-chain-multiplication/)  
 [0-1 Knapsack problem](http://www.techiedelight.com/0-1-knapsack-problem/)  
 [Maximize value of the expression](http://www.techiedelight.com/maximize-value-of-the-expression/)  
 [Partition problem](http://www.techiedelight.com/partition-problem/)  
 [Subset sum problem](http://www.techiedelight.com/subset-sum-problem/)  
 [Minimum Sum Partition problem](http://www.techiedelight.com/minimum-sum-partition-problem/)  
 [Rod Cutting](http://www.techiedelight.com/rot-cutting/)  
 [Coin change-making problem (unlimited supply of coins)](http://www.techiedelight.com/coin-change-making-problem-unlimited-supply-coins/)  
 [Coin Change Problem – Find total number of ways to get the denomination of coins](http://www.techiedelight.com/coin-change-problem-find-total-number-ways-get-denomination-coins/)  
 [Longest Decreasing Subsequence Problem](http://www.techiedelight.com/longest-decreasing-subsequence-problem/)  
 [Longest alternating subsequence](http://www.techiedelight.com/longest-alternating-subsequence/)  
 [Combinations of words formed by replacing given numbers with corresponding alphabets](http://www.techiedelight.com/combinations-of-words-formed-replacing-given-numbers-corresponding-english-alphabet/)  
 [Decode the given sequence to construct minimum number without repeated digits](http://www.techiedelight.com/decode-the-given-sequence-construct-minimum-number-without-repeated-digits/)  
 [All combinations of elements satisfying given constraints](http://www.techiedelight.com/find-combinations-of-elements-satisfies-given-constraints/)  
 [Find Missing Term in a Sequence in log(n) time](http://www.techiedelight.com/find-missing-term-sequence-ologn-time/)  
 [Print all distinct Subsets of a given Set](http://www.techiedelight.com/print-distinct-subsets-given-set/)  
 [Find Floor and Ceil of a number in a sorted array (Recursive solution)](http://www.techiedelight.com/find-floor-ceil-number-sorted-array-recursive/)  
 [Set both elements of a binary array to 0 in single line](http://www.techiedelight.com/set-elements-binary-array-0-single-line/)  
 [K-Partition Problem | Printing all Partitions](http://www.techiedelight.com/k-partition-problem-print-all-subsets/)  
 [3 Partition Problem](http://www.techiedelight.com/3-partition-problem/)  
 [3-partition problem extended | Print all partitions](http://www.techiedelight.com/3-partition-problem-extended-print-all-partitions/)  
 [Iterative Merge Sort Algorithm (Bottom-up Merge Sort)](http://www.techiedelight.com/iterative-merge-sort-algorithm-bottom-up/)  
 [Find two duplicate elements in an limited range array (using XOR)](http://www.techiedelight.com/find-two-duplicate-elements-limited-range-array-using-xor/)  
 [Find missing number and duplicate elements in an array](http://www.techiedelight.com/find-missing-number-duplicate-elements-array/)  
 [Find Minimum and Maximum element in an array by doing minimum comparisons](http://www.techiedelight.com/find-minimum-maximum-element-array-minimum-comparisons/)  
 [Find Frequency of each element in a sorted array containing duplicates](http://www.techiedelight.com/find-frequency-element-sorted-array-containing-duplicates/)  
 [Difference between Subarray, Subsequence and Subset](http://www.techiedelight.com/difference-between-subarray-subsequence-subset/)

**Backtracking:**

[Print all possible solutions to N Queens problem](http://www.techiedelight.com/print-possible-solutions-n-queens-problem/)  
 [Print all Possible Knight’s Tours in a chessboard](http://www.techiedelight.com/print-possible-knights-tours-chessboard/)  
 [Magnet Puzzle](http://www.techiedelight.com/magnet-puzzle/)  
 [Find Shortest Path in Maze](http://www.techiedelight.com/find-shortest-path-in-maze/)  
 [Find Longest Possible Route in a Matrix](http://www.techiedelight.com/find-longest-possible-route-matrix/)  
 [Find path from source to destination in a matrix that satisfies given constraints](http://www.techiedelight.com/find-path-source-destination-matrix-satisfies-given-constraints/)  
 [Find total number of unique paths in a maze from source to destination](http://www.techiedelight.com/find-total-number-unique-paths-maze-source-destination/)  
 [Print All Hamiltonian Path present in a graph](http://www.techiedelight.com/print-all-hamiltonian-path-present-in-a-graph/)  
 [Print all k-colorable configurations of the graph (Vertex coloring of graph)](http://www.techiedelight.com/print-k-colorable-configurations-graph-vertex-coloring-graph/)  
 [Find all Permutations of a given string](http://www.techiedelight.com/find-permutations-given-string/)  
 [Find all binary strings that can be formed from given wildcard pattern](http://www.techiedelight.com/find-binary-strings-can-formed-given-wildcard-pattern/)  
 [All combinations of elements satisfying given constraints](http://www.techiedelight.com/find-combinations-of-elements-satisfies-given-constraints/)  
 [K-Partition Problem | Printing all Partitions](http://www.techiedelight.com/k-partition-problem-print-all-subsets/)

**Binary:**

[Bit Hacks – Part 1 (Basic)](http://www.techiedelight.com/bit-hacks-part-1-basic/)  
 [Bit Hacks – Part 2 (Playing with k’th bit)](http://www.techiedelight.com/bit-hacks-part-2-playing-kth-bit/)  
 [Bit Hacks – Part 3 (Playing with rightmost set bit of a number)](http://www.techiedelight.com/bit-hacks-part-3-playing-rightmost-set-bit-number/)  
 [Bit Hacks – Part 4 (Playing with letters of English alphabet)](http://www.techiedelight.com/bit-hacks-part-4-playing-letters-english-alphabet/)  
 [Bit Hacks – Part 5 (Find absolute value of an integer without branching)](http://www.techiedelight.com/bit-hacks-part-5-find-absolute-value-integer-without-branching/)  
 [Bit Hacks – Part 6 (Random Problems)](http://www.techiedelight.com/bit-hacks-part-6-random-problems/)  
 [Brian Kernighan’s Algorithm to count set bits in an integer](http://www.techiedelight.com/brian-kernighans-algorithm-count-set-bits-integer/)  
 [Compute parity of a number using lookup table](http://www.techiedelight.com/compute-parity-number-using-lookup-table/)  
 [Count set bits using lookup table](http://www.techiedelight.com/count-set-bits-using-lookup-table/)  
 [Find the minimum or maximum of two integers without using branching](http://www.techiedelight.com/find-minimum-maximum-two-integers-without-using-branching/)  
 [Multiply 16-bit integers using 8-bit multiplier](http://www.techiedelight.com/multiply-16-bit-integers-using-8-bit-multiplier/)  
 [Round up to the next highest power of 2](http://www.techiedelight.com/round-next-highest-power-2/)  
 [Round up to the previous power of 2](http://www.techiedelight.com/round-previous-power-2/)  
 [Swap individual bits at given position in an integer](http://www.techiedelight.com/swap-individual-bits-given-position-integer/)  
 [Reverse Bits of a given Integer](http://www.techiedelight.com/reverse-bits-of-given-integer/)  
 [Check if given number is power of 4 or not](http://www.techiedelight.com/check-given-number-power-of-4/)  
 [Find odd occurring element in an array in single traversal](http://www.techiedelight.com/find-odd-occurring-element-array-single-traversal/)  
 [Find two odd occurring element in an array without using any extra space](http://www.techiedelight.com/find-two-odd-occurring-element-array-without-extra-space/)  
 [Swap two bits at given position in an integer](http://www.techiedelight.com/swap-two-bits-given-position-integer/)  
 [Add binary representation of two integers](http://www.techiedelight.com/add-binary-representation-two-integers/)  
 [Swap Adjacent Bits of a Number](http://www.techiedelight.com/swap-adjacent-bits-number/)  
 [Print all distinct Subsets of a given Set](http://www.techiedelight.com/print-distinct-subsets-given-set/)  
 [Perform Division of two numbers without using division operator (/)](http://www.techiedelight.com/perform-division-two-numbers-without-using-division-operator/)  
 [Check if adjacent bits are set in binary representation of a given number](http://www.techiedelight.com/check-adjacent-bits-set-binary-representation-number/)  
 [Conditionally negate a value without branching](http://www.techiedelight.com/conditionally-negate-value-without-branching/)  
 [Find two duplicate elements in an limited range array (using XOR)](http://www.techiedelight.com/find-two-duplicate-elements-limited-range-array-using-xor/)  
 [Find missing number and duplicate elements in an array](http://www.techiedelight.com/find-missing-number-duplicate-elements-array/)  
 [Check if given number is power of 8 or not](http://www.techiedelight.com/check-given-number-power-8-not/)  
 [Generate binary numbers between 1 to N](http://www.techiedelight.com/generate-binary-numbers-1-n/)  
 [Efficiently implement power function | Recursive and Iterative](http://www.techiedelight.com/power-function-implementation-recursive-iterative/)  
 [Find square of a number without using multiplication and division operator | 3 methods](http://www.techiedelight.com/find-square-number-without-using-multiplication-division-operator/)  
 [Generate power set of a given set](http://www.techiedelight.com/generate-power-set-given-set/)  
 [Huffman Coding](http://www.techiedelight.com/huffman-coding/)

**Binary Tree:**

[Check if two given binary trees are identical or not | Iterative & Recursive](http://www.techiedelight.com/check-if-two-binary-trees-are-identical-not-iterative-recursive/)  
 [Calculate height of a binary tree | Iterative & Recursive](http://www.techiedelight.com/calculate-height-binary-tree-iterative-recursive/)  
 [Delete given Binary Tree | Iterative & Recursive](http://www.techiedelight.com/delete-given-binary-tree-iterative-recursive/)  
 [Inorder Tree Traversal | Iterative & Recursive](http://www.techiedelight.com/inorder-tree-traversal-iterative-recursive/)  
 [Preorder Tree Traversal | Iterative & Recursive](http://www.techiedelight.com/preorder-tree-traversal-iterative-recursive/)  
 [Postorder Tree Traversal | Iterative & Recursive](http://www.techiedelight.com/postorder-tree-traversal-iterative-recursive/)  
 [Level Order Traversal of Binary Tree](http://www.techiedelight.com/level-order-traversal-binary-tree/)  
 [Spiral Order Traversal of Binary Tree](http://www.techiedelight.com/spiral-order-traversal-binary-tree/)  
 [Reverse Level Order Traversal of Binary Tree](http://www.techiedelight.com/reverse-level-order-traversal-binary-tree/)  
 [Print all nodes of a given binary tree in specific order](http://www.techiedelight.com/print-nodes-binary-tree-specific-order/)  
 [Print left view of binary tree](http://www.techiedelight.com/print-left-view-of-binary-tree/)  
 [Print Bottom View of Binary Tree](http://www.techiedelight.com/print-bottom-view-of-binary-tree/)  
 [Print Top View of Binary Tree](http://www.techiedelight.com/print-top-view-binary-tree/)  
 [Find next node in same level for given node in a binary tree](http://www.techiedelight.com/find-next-node-in-same-level-binary-tree/)  
 [Check if given binary tree is complete binary tree or not](http://www.techiedelight.com/check-given-binary-tree-complete-binary-tree-not/)  
 [Determine if given two nodes are cousins of each other](http://www.techiedelight.com/determine-two-nodes-are-cousins/)  
 [Print cousins of given node in a binary tree](http://www.techiedelight.com/print-cousins-of-given-node-binary-tree/)  
 [In-place convert given binary tree to its sum tree](http://www.techiedelight.com/inplace-convert-a-tree-sum-tree/)  
 [Check if given binary tree is a sum tree or not](http://www.techiedelight.com/check-given-binary-tree-sum-tree-not/)  
 [Combinations of words formed by replacing given numbers with corresponding alphabets](http://www.techiedelight.com/combinations-of-words-formed-replacing-given-numbers-corresponding-english-alphabet/)  
 [Determine if given binary tree is a subtree of another binary tree or not](http://www.techiedelight.com/determine-given-binary-tree-is-subtree-of-another-binary-tree-not/)  
 [Find diameter of a binary tree](http://www.techiedelight.com/find-diameter-of-a-binary-tree/)  
 [Check if given binary Tree has symmetric structure or not](http://www.techiedelight.com/check-given-binary-tree-symmetric-structure-not/)  
 [Convert binary tree to its mirror](http://www.techiedelight.com/convert-binary-tree-to-its-mirror/)  
 [Determine if binary tree can be converted to another by doing any no. of swaps of left & right child](http://www.techiedelight.com/determine-binary-tree-can-converted-another-number-swaps-left-right-child/)  
 [Find Lowest Common Ancestor (LCA) of two nodes in a binary tree](http://www.techiedelight.com/find-lowest-common-ancestor-lca-two-nodes-binary-tree/)  
 [Print all paths from root to leaf nodes in given binary tree](http://www.techiedelight.com/print-all-paths-from-root-to-leaf-nodes-binary-tree/)  
 [Find ancestors of given node in a Binary Tree](http://www.techiedelight.com/find-ancestors-of-given-node-binary-tree/)  
 [Find the distance between given pairs of nodes in a binary tree](http://www.techiedelight.com/distance-between-given-pairs-of-nodes-binary-tree/)  
 [Find Vertical Sum in a given Binary Tree](http://www.techiedelight.com/find-vertical-sum-given-binary-tree/)  
 [Print nodes in vertical order of a given Binary Tree (Vertical Traversal)](http://www.techiedelight.com/vertical-traversal-binary-tree/)  
 [Find the diagonal sum of given binary tree](http://www.techiedelight.com/find-diagonal-sum-given-binary-tree/)  
 [Print Diagonal Traversal of Binary Tree](http://www.techiedelight.com/print-diagonal-traversal-binary-tree/)  
 [Print corner nodes of every level in binary tree](http://www.techiedelight.com/print-corner-nodes-every-level-binary-tree/)  
 [In-place convert given Binary Tree to Doubly Linked List](http://www.techiedelight.com/place-convert-given-binary-tree-to-doubly-linked-list/)  
 [Sink nodes containing zero to the bottom of the binary tree](http://www.techiedelight.com/sink-nodes-containing-zero-bottom-binary-tree/)  
 [Convert given binary tree to full tree by removing half nodes](http://www.techiedelight.com/convert-given-binary-tree-to-full-tree-removing-half-nodes/)  
 [Truncate binary tree to remove nodes which lie on a path having sum less than K](http://www.techiedelight.com/truncate-given-binary-tree-remove-nodes-lie-path-sum-less-k/)  
 [Find maximum sum root-to-leaf path in a binary tree](http://www.techiedelight.com/find-maximum-sum-root-to-leaf-path-binary-tree/)  
 [Check if given binary tree is height balanced or not](http://www.techiedelight.com/check-given-binary-tree-is-height-balanced-not/)  
 [Convert normal binary tree to Left-child right-sibling binary tree](http://www.techiedelight.com/convert-normal-binary-tree-left-child-right-sibling-binary-tree/)  
 [Invert given Binary Tree | Recursive and Iterative solution](http://www.techiedelight.com/invert-binary-tree-recursive-iterative/)  
 [Determine if given Binary Tree is a BST or not](http://www.techiedelight.com/determine-given-binary-tree-is-a-bst-or-not/)  
 [Convert a Binary Tree to BST by maintaining its original structure](http://www.techiedelight.com/convert-binary-tree-to-bst-maintaining-original-structure/)  
 [Print Right View of a Binary Tree](http://www.techiedelight.com/print-right-view-binary-tree/)  
 [Print leaf to root path for every leaf node in a binary tree](http://www.techiedelight.com/print-leaf-to-root-path-binary-tree/)  
 [Find maximum width of given binary tree](http://www.techiedelight.com/find-maximum-width-given-binary-tree/)  
 [Build Binary Tree from given Parent array](http://www.techiedelight.com/build-binary-tree-given-parent-array/)  
 [C++ Program to Print Binary Tree Structure](http://www.techiedelight.com/c-program-print-binary-tree/)

**Binary Search Tree (BST):**

[Insertion in BST](http://www.techiedelight.com/insertion-in-bst/)  
 [Search given key in BST](http://www.techiedelight.com/search-given-key-in-bst/)  
 [Deletion from BST](http://www.techiedelight.com/deletion-from-bst/)  
 [Construct balanced BST from given keys](http://www.techiedelight.com/construct-balanced-bst-given-keys/)  
 [Determine if given Binary Tree is a BST or not](http://www.techiedelight.com/determine-given-binary-tree-is-a-bst-or-not/)  
 [Check if given keys represents same BSTs or not without building the BST](http://www.techiedelight.com/check-given-keys-represents-same-bsts-not-without-building-bst/)  
 [Find inorder predecessor for given key in a BST](http://www.techiedelight.com/find-inorder-predecessor-given-key-bst/)  
 [Find Lowest Common Ancestor (LCA) of two nodes in a Binary Search Tree](http://www.techiedelight.com/find-lowest-common-ancestor-lca-two-nodes-bst/)  
 [Find K’th smallest and K’th largest element in BST](http://www.techiedelight.com/find-kth-smallest-largest-element-bst/)  
 [Floor and Ceil in a Binary Search Tree](http://www.techiedelight.com/floor-ceil-bst-iterative-recursive/)  
 [Find optimal cost to construct binary search tree](http://www.techiedelight.com/find-optimal-cost-to-construct-binary-search-tree/)  
 [Convert a Binary Tree to BST by maintaining its original structure](http://www.techiedelight.com/convert-binary-tree-to-bst-maintaining-original-structure/)  
 [Remove nodes from BST that have keys outside the valid range](http://www.techiedelight.com/remove-nodes-bst-keys-outside-valid-range/)  
 [Find a pair with given sum in a BST](http://www.techiedelight.com/find-pair-with-given-sum-bst/)  
 Divide & Conquer

[Binary Search](http://www.techiedelight.com/binary-search/)  
 [Ternary Search vs Binary search](http://www.techiedelight.com/ternary-search-vs-binary-search/)  
 [Exponential search](http://www.techiedelight.com/exponential-search/)  
 [Interpolation search](http://www.techiedelight.com/interpolation-search/)  
 [Find number of rotations in a circularly sorted array](http://www.techiedelight.com/find-number-rotations-circularly-sorted-array/)  
 [Search an element in a circular sorted array](http://www.techiedelight.com/search-element-circular-sorted-array/)  
 [Find first or last occurrence of a given number in a sorted array](http://www.techiedelight.com/find-first-or-last-occurrence-of-a-given-number-sorted-array/)  
 [Count occurrences of a number in a sorted array with duplicates](http://www.techiedelight.com/count-occurrences-number-sorted-array-duplicates/)  
 [Find smallest missing element from a sorted array](http://www.techiedelight.com/find-smallest-missing-element-sorted-array/)  
 [Find Floor and Ceil of a number in a sorted array](http://www.techiedelight.com/find-floor-ceil-number-sorted-array/)  
 [Search in a nearly sorted array in O(logn) time](http://www.techiedelight.com/search-nearly-sorted-array-ologn-time/)  
 [Find number of 1’s in a sorted binary array](http://www.techiedelight.com/find-number-1s-sorted-binary-array/)  
 [Find the peak element in an array](http://www.techiedelight.com/find-peak-element-array/)  
 [Maximum Sum Subarray using Divide & Conquer](http://www.techiedelight.com/maximum-sum-subarray-using-divide-conquer/)  
 [Find Minimum and Maximum element in an array using minimum comparisons](http://www.techiedelight.com/find-minimum-maximum-element-array-using-minimum-comparisons/)  
 [Efficiently implement power function | Recursive and Iterative](http://www.techiedelight.com/power-function-implementation-recursive-iterative/)  
 [Find Missing Term in a Sequence in log(n) time](http://www.techiedelight.com/find-missing-term-sequence-ologn-time/)  
 [Division of Two Numbers using Binary Search Algorithm](http://www.techiedelight.com/division-two-numbers-using-binary-search-algorithm/)  
 [Find Floor and Ceil of a number in a sorted array (Recursive solution)](http://www.techiedelight.com/find-floor-ceil-number-sorted-array-recursive/)  
 [Find Minimum and Maximum element in an array by doing minimum comparisons](http://www.techiedelight.com/find-minimum-maximum-element-array-minimum-comparisons/)  
 [Find Frequency of each element in a sorted array containing duplicates](http://www.techiedelight.com/find-frequency-element-sorted-array-containing-duplicates/)  
 [Merge Sort](http://www.techiedelight.com/merge-sort/)  
 [Iterative Merge Sort Algorithm (Bottom-up Merge Sort)](http://www.techiedelight.com/iterative-merge-sort-algorithm-bottom-up/)  
 [Merge Sort for Singly Linked List](http://www.techiedelight.com/merge-sort-singly-linked-list/)  
 [Inversion Count of an array](http://www.techiedelight.com/inversion-count-array/)  
 [Quicksort](http://www.techiedelight.com/quicksort/)  
 [Iterative Implementation of Quicksort](http://www.techiedelight.com/iterative-implementation-of-quicksort/)  
 [Quicksort using Dutch National Flag Algorithm](http://www.techiedelight.com/quicksort-using-dutch-national-flag-algorithm/)  
 [Quick Sort using Hoare’s Partitioning scheme](http://www.techiedelight.com/quick-sort-using-hoares-partitioning-scheme/)  
 [Hybrid QuickSort](http://www.techiedelight.com/hybrid-quicksort/)

**Dynamic Programming:**

[Introduction to Dynamic Programming](http://www.techiedelight.com/introduction-dynamic-programming/)  
 [Longest Common Subsequence | Introduction & LCS Length](http://www.techiedelight.com/longest-common-subsequence/)  
 [Longest Common Subsequence | Space optimized version](http://www.techiedelight.com/longest-common-subsequence-lcs-space-optimized-version/)  
 [Longest Common Subsequence | Finding all LCS](http://www.techiedelight.com/longest-common-subsequence-finding-lcs/)  
 [Longest Common Subsequence of K-sequences](http://www.techiedelight.com/longest-common-subsequence-of-k-sequences/)  
 [Longest Common Substring problem](http://www.techiedelight.com/longest-common-substring-problem/)  
 [Longest Palindromic Subsequence using Dynamic Programming](http://www.techiedelight.com/longest-palindromic-subsequence-using-dynamic-programming/)  
 [Longest Repeated Subsequence problem](http://www.techiedelight.com/longest-repeated-subsequence-problem/)  
 [Implement Diff Utility](http://www.techiedelight.com/implement-diff-utility/)  
 [Shortest Common Supersequence | Introduction & SCS Length](http://www.techiedelight.com/shortest-common-supersequence-introduction-scs-length/)  
 [Shortest Common Supersequence | Finding all SCS](http://www.techiedelight.com/shortest-common-supersequence-finding-scs/)  
 [Shortest Common Supersequence | Using LCS](http://www.techiedelight.com/shortest-common-supersequence-using-lcs/)  
 [Longest Increasing Subsequence using Dynamic Programming](http://www.techiedelight.com/longest-increasing-subsequence-using-dynamic-programming/)  
 [Longest Bitonic Subsequence](http://www.techiedelight.com/longest-bitonic-subsequence/)  
 [Increasing Subsequence with Maximum Sum](http://www.techiedelight.com/increasing-subsequence-with-maximum-sum/)  
 [Longest Decreasing Subsequence Problem](http://www.techiedelight.com/longest-decreasing-subsequence-problem/)  
 [The Levenshtein distance (Edit distance) problem](http://www.techiedelight.com/levenshtein-distance-edit-distance-problem/)  
 [Word Break Problem](http://www.techiedelight.com/word-break-problem/)  
 [Word Break Problem | Using Trie Data Structure](http://www.techiedelight.com/word-break-problem-using-trie/)  
 [Wildcard Pattern Matching](http://www.techiedelight.com/wildcard-pattern-matching/)  
 [Find size of largest square sub-matrix of 1’s present in given binary matrix](http://www.techiedelight.com/find-size-largest-square-sub-matrix-1s-present-given-binary-matrix/)  
 [Matrix Chain Multiplication](http://www.techiedelight.com/matrix-chain-multiplication/)  
 [Find the minimum cost to reach last cell of the matrix from its first cell](http://www.techiedelight.com/find-minimum-cost-reach-last-cell-matrix-first-cell/)  
 [Find longest sequence formed by adjacent numbers in the matrix](http://www.techiedelight.com/find-longest-sequence-formed-adjacent-numbers-matrix/)  
 [Count number of paths in a matrix with given cost to reach destination cell](http://www.techiedelight.com/counting-paths-on-grid-to-reach-destination-cell/)  
 [0-1 Knapsack problem](http://www.techiedelight.com/0-1-knapsack-problem/)  
 [Maximize value of the expression](http://www.techiedelight.com/maximize-value-of-the-expression/)  
 [Partition problem](http://www.techiedelight.com/partition-problem/)  
 [Subset sum problem](http://www.techiedelight.com/subset-sum-problem/)  
 [Minimum Sum Partition problem](http://www.techiedelight.com/minimum-sum-partition-problem/)  
 [Find all N-digit binary strings without any consecutive 1’s](http://www.techiedelight.com/find-n-digit-binary-strings-without-consecutive-1s/)  
 [Rod Cutting](http://www.techiedelight.com/rot-cutting/)  
 [Maximum Product Rod Cutting](http://www.techiedelight.com/maximum-product-rod-cutting/)  
 [Coin change-making problem (unlimited supply of coins)](http://www.techiedelight.com/coin-change-making-problem-unlimited-supply-coins/)  
 [Coin Change Problem – Find total number of ways to get the denomination of coins](http://www.techiedelight.com/coin-change-problem-find-total-number-ways-get-denomination-coins/)  
 [Total possible solutions to linear equation of k variables](http://www.techiedelight.com/total-possible-solutions-linear-equation-k-variables/)  
 [Longest alternating subsequence](http://www.techiedelight.com/longest-alternating-subsequence/)  
 [Count number of times a pattern appears in given string as a subsequence](http://www.techiedelight.com/count-number-times-pattern-appears-given-string-subsequence/)  
 [Collect maximum points in a matrix by satisfying given constraints](http://www.techiedelight.com/collect-maximum-points-matrix-satisfying-given-constraints/)  
 [Count total possible combinations of N-digit numbers in a mobile keypad](http://www.techiedelight.com/count-total-possible-combinations-n-digit-numbers-mobile-keypad/)  
 [Find optimal cost to construct binary search tree](http://www.techiedelight.com/find-optimal-cost-to-construct-binary-search-tree/)  
 [Pots of Gold Game using Dynamic Programming](http://www.techiedelight.com/pots-gold-game-dynamic-programming/)  
 [Find minimum cuts needed for palindromic partition of a string](http://www.techiedelight.com/find-minimum-cuts-needed-palindromic-partition-string/)  
 [Maximum Length Snake Sequence](http://www.techiedelight.com/maximum-length-snake-sequence/)  
 [3 Partition Problem](http://www.techiedelight.com/3-partition-problem/)  
 [Calculate size of the largest plus of 1's in binary matrix](http://www.techiedelight.com/calculate-size-largest-plus-1s-binary-matrix/)  
 [Check if given string is interleaving of two other given strings](http://www.techiedelight.com/check-string-interleaving-two-given-strings/)  
 [Longest Increasing Subsequence using LCS](http://www.techiedelight.com/longest-increasing-subsequence-using-lcs/)  
 [Find probability that a person is alive after taking N steps on the island](http://www.techiedelight.com/probability-alive-after-taking-n-steps-island/)  
 [Calculate sum of all elements in a sub-matrix in constant time](http://www.techiedelight.com/calculate-sum-elements-sub-matrix-constant-time/)  
 [Find maximum sum K x K sub-matrix in a given M x N matrix](http://www.techiedelight.com/find-maximum-sum-submatrix-in-given-matrix/)  
 [Find maximum sum submatrix present in a given matrix](http://www.techiedelight.com/find-maximum-sum-submatrix-present-given-matrix/)  
 [Find maximum sum of subsequence with no adjacent elements](http://www.techiedelight.com/maximum-sum-of-subsequence-with-no-adjacent-elements)  
 [Maximum subarray problem (Kadane’s algorithm)](http://www.techiedelight.com/maximum-subarray-problem-kadanes-algorithm/)  
 [Single-Source Shortest Paths – Bellman Ford Algorithm](http://www.techiedelight.com/single-source-shortest-paths-bellman-ford-algorithm/)  
 [All-Pairs Shortest Paths – Floyd Warshall Algorithm](http://www.techiedelight.com/pairs-shortest-paths-floyd-warshall-algorithm/)

**Graphs:**

[Terminology and Representations of Graphs](http://www.techiedelight.com/terminology-and-representations-of-graphs/)  
 [Graph Implementation using STL](http://www.techiedelight.com/graph-implementation-using-stl/)  
 [Graph Implementation in C++ without using STL](http://www.techiedelight.com/graph-implementation-c-without-using-stl/)  
 [Implement Graph Data Structure in C](http://www.techiedelight.com/implement-graph-data-structure-c/)  
 [Graph Implementation in Java using Collections](http://www.techiedelight.com/graph-implementation-java-using-collections/)  
 [Breadth First Search (BFS) | Iterative & Recursive Implementation](http://www.techiedelight.com/breadth-first-search/)  
 [Depth First Search (DFS) | Iterative & Recursive Implementation](http://www.techiedelight.com/depth-first-search/)  
 [Arrival and Departure Time of Vertices in DFS](http://www.techiedelight.com/arrival-departure-time-vertices-dfs/)  
 [Types of edges involved in DFS and relation between them](http://www.techiedelight.com/types-edges-involved-dfs-relation/)  
 [Determine if a given graph is Bipartite Graph or not](http://www.techiedelight.com/bipartite-graph/)  
 [Determine if a given graph is Bipartite Graph using DFS](http://www.techiedelight.com/determine-given-graph-bipartite-graph-using-dfs/)  
 [Minimum number of throws required to win Snake and Ladder game](http://www.techiedelight.com/min-throws-required-to-win-snake-and-ladder-game/)  
 [Topological Sorting in a DAG](http://www.techiedelight.com/topological-sorting-dag/)  
 [Kahn's Topological Sort Algorithm](http://www.techiedelight.com/kahn-topological-sort-algorithm/)  
 [Transitive Closure of a Graph](http://www.techiedelight.com/transitive-closure-graph/)  
 [Check if an undirected graph contains cycle or not](http://www.techiedelight.com/check-undirected-graph-contains-cycle-not/)  
 [Total number of paths in given digraph from given source to destination having exactly m edges](http://www.techiedelight.com/total-paths-in-digraph-from-source-to-destination-m-edges/)  
 [Determine if an undirected graph is a Tree (Acyclic Connected Graph)](http://www.techiedelight.com/determine-undirected-graph-tree-acyclic-connected-graph/)  
 [2-Edge Connectivity in the graph](http://www.techiedelight.com/2-edge-connectivity-graph/)  
 [2-Vertex Connectivity in the graph](http://www.techiedelight.com/2-vertex-connectivity-graph/)  
 [Check if given digraph is a DAG (Directed Acyclic Graph) or not](http://www.techiedelight.com/check-given-digraph-dag-directed-acyclic-graph-not/)  
 [Disjoint-Set Data Structure (Union-Find Algorithm)](http://www.techiedelight.com/disjoint-set-data-structure-union-find-algorithm/)  
 [Chess Knight Problem – Find Shortest path from source to destination](http://www.techiedelight.com/chess-knight-problem-find-shortest-path-source-destination/)  
 [Check if given Graph is Strongly Connected or not](http://www.techiedelight.com/check-given-graph-strongly-connected-not/)  
 [Check if given Graph is Strongly Connected or not using one DFS Traversal](http://www.techiedelight.com/check-graph-strongly-connected-one-dfs-traversal/)  
 [Union-Find Algorithm for Cycle Detection in undirected graph](http://www.techiedelight.com/union-find-algorithm-cycle-detection-graph/)  
 [Kruskal’s Algorithm for finding Minimum Spanning Tree](http://www.techiedelight.com/kruskals-algorithm-for-finding-minimum-spanning-tree/)  
 [Single-Source Shortest Paths – Dijkstra’s Algorithm](http://www.techiedelight.com/single-source-shortest-paths-dijkstras-algorithm/)  
 [Single-Source Shortest Paths – Bellman Ford Algorithm](http://www.techiedelight.com/single-source-shortest-paths-bellman-ford-algorithm/)  
 [All-Pairs Shortest Paths – Floyd Warshall Algorithm](http://www.techiedelight.com/pairs-shortest-paths-floyd-warshall-algorithm/)  
 [Find Cost of Shortest Path in DAG using one pass of Bellman-Ford](http://www.techiedelight.com/cost-of-shortest-path-in-dag-using-one-pass-of-bellman-ford/)  
 [Least Cost Path in Weighted Digraph using BFS](http://www.techiedelight.com/least-cost-path-weighted-digraph-using-bfs/)  
 [Print all k-colorable configurations of the graph (Vertex coloring of graph)](http://www.techiedelight.com/print-k-colorable-configurations-graph-vertex-coloring-graph/)  
 [Print All Hamiltonian Path present in a graph](http://www.techiedelight.com/print-all-hamiltonian-path-present-in-a-graph/)  
 [Greedy coloring of graph](http://www.techiedelight.com/greedy-coloring-graph/)

**Heaps:**

[Introduction to Priority Queues using Binary Heaps](http://www.techiedelight.com/introduction-priority-queues-using-binary-heaps/)  
 [Min Heap and Max Heap Implementation in C++](http://www.techiedelight.com/min-heap-max-heap-implementation-c/)  
 [Min Heap and Max Heap Implementation in Java](http://www.techiedelight.com/min-heap-max-heap-implementation-in-java/)  
 [Heap Sort (Out-of-place and In-place implementation in C++ and C)](http://www.techiedelight.com/heap-sort-place-place-implementation-c-c/)  
 [Check if given array represents min heap or not](http://www.techiedelight.com/check-given-array-represents-min-heap-not/)  
 [Convert Max Heap to Min Heap in linear time](http://www.techiedelight.com/convert-max-heap-min-heap-linear-time/)  
 [Find K’th largest element in an array](http://www.techiedelight.com/find-kth-largest-element-array/)  
 [Sort a K-Sorted Array](http://www.techiedelight.com/sort-k-sorted-array/)  
 [Merge M sorted lists of variable length](http://www.techiedelight.com/merge-m-sorted-lists-variable-length/)  
 [Find K’th smallest element in an array](http://www.techiedelight.com/find-kth-smallest-element-array/)  
 [Find smallest range with at-least one element from each of the given lists](http://www.techiedelight.com/find-smallest-range-least-one-element-given-lists/)  
 [Merge M sorted lists each containing N elements](http://www.techiedelight.com/merge-m-sorted-lists-containing-n-elements/)  
 [External merge sort](http://www.techiedelight.com/external-merge-sort/)  
 [Huffman Coding](http://www.techiedelight.com/huffman-coding/)  
 [Find first k maximum occurring words in given set of strings](http://www.techiedelight.com/find-first-k-maximum-occurring-words-given-set-strings/)  
 [Find first k non-repeating characters in a string in single traversal](http://www.techiedelight.com/first-k-non-repeating-characters-string/)

**Linked Lists:**

[Introduction to Linked Lists](http://www.techiedelight.com/introduction-linked-lists/)  
 [Linked List Implementation | Part 1](http://www.techiedelight.com/linked-list-implementation-part-1/)  
 [Linked List Implementation | Part 2](http://www.techiedelight.com/linked-list-implementation-part-2/)  
 [Static Linked List in C](http://www.techiedelight.com/static-linked-list-c/)  
 [Clone given Linked List](http://www.techiedelight.com/clone-given-linked-list/)  
 [Delete Linked List](http://www.techiedelight.com/delete-linked-list/)  
 [Pop operation in linked list](http://www.techiedelight.com/pop-operation-in-linked-list/)  
 [Insert given node into the correct sorted position in the given sorted linked list](http://www.techiedelight.com/sorted-insert-in-linked-list/)  
 [Given a linked list, change it to be in sorted order](http://www.techiedelight.com/given-linked-list-change-sorted-order/)  
 [Split the nodes of the given linked list into front and back halves](http://www.techiedelight.com/split-nodes-given-linked-list-front-back-halves/)  
 [Remove duplicates from a sorted linked list](http://www.techiedelight.com/remove-duplicates-sorted-linked-list/)  
 [Move front node of the given list to the front of the another list](http://www.techiedelight.com/move-front-node-given-list-front-another-list/)  
 [Rearrange the given list such that every even node will be moved to end of the list in reverse order](http://www.techiedelight.com/move-even-nodes-to-end-of-list-in-reverse-order/)  
 [Split given linked list into two lists where each list containing alternating elements from it](http://www.techiedelight.com/split-linked-list-into-two-lists-list-containing-alternating-elements/)  
 [Construct a linked list by merging alternate nodes of two given lists](http://www.techiedelight.com/merge-alternate-nodes-two-linked-lists/)  
 [Merge given sorted linked lists into one](http://www.techiedelight.com/merge-given-sorted-linked-lists/)  
 [Merge Sort for Singly Linked List](http://www.techiedelight.com/merge-sort-singly-linked-list/)  
 [Intersection of two given sorted linked lists](http://www.techiedelight.com/intersection-two-given-sorted-linked-lists/)  
 [Reverse linked list | Part 1 (Iterative Solution)](http://www.techiedelight.com/reverse-linked-list-part-1-iterative-solution/)  
 [Reverse linked list | Part 2 (Recursive Solution)](http://www.techiedelight.com/reverse-linked-list-part-2-recursive-solution/)  
 [Reverse every group of k nodes in given linked list](http://www.techiedelight.com/reverse-every-k-nodes-of-a-linked-list/)  
 [Find K’th node from the end in a linked list](http://www.techiedelight.com/find-kth-node-from-the-end-linked-list/)  
 [Merge alternate nodes of two linked lists into the first list](http://www.techiedelight.com/merge-alternate-nodes-two-linked-lists-first-list/)  
 [Merge two sorted linked lists from their end](http://www.techiedelight.com/merge-two-sorted-linked-lists-end/)  
 [Delete every N nodes in a linked list after skipping M nodes](http://www.techiedelight.com/delete-every-n-nodes-linked-list-skipping-m-nodes/)  
 [Rearrange linked list in specific manner in linear time](http://www.techiedelight.com/rearrange-linked-list-specific-manner-linear-time/)  
 [Check if linked list is palindrome or not](http://www.techiedelight.com/check-if-linked-list-is-palindrome/)  
 [Move last node to front in a given Linked List](http://www.techiedelight.com/move-last-node-to-front-linked-list/)  
 [Rearrange the linked list in specific manner](http://www.techiedelight.com/rearrange-the-linked-list-specific-manner/)  
 [Detect Cycle in a linked list (Floyd’s Cycle Detection Algorithm)](http://www.techiedelight.com/detect-cycle-linked-list-floyds-cycle-detection-algorithm/)  
 [Sort linked list containing 0’s, 1’s and 2’s](http://www.techiedelight.com/sort-linked-list-containing-0s-1s-2s/)  
 [Stack Implementation using Linked List](http://www.techiedelight.com/stack-implementation-using-linked-list/)  
 [Queue Implementation using Linked List](http://www.techiedelight.com/queue-implementation-using-linked-list/)  
 [Rearrange the linked list so that it has alternating high, low values](http://www.techiedelight.com/rearrange-linked-list-alternating-high-low-values/)

**Matrix:**

[Print Matrix in Spiral Order](http://www.techiedelight.com/print-matrix-spiral-order/)  
 [Create Spiral Matrix from given array](http://www.techiedelight.com/create-spiral-matrix-given-array/)  
 [Shift all matrix elements by 1 in Spiral Order](http://www.techiedelight.com/shift-matrix-elements-1-spiral-order/)  
 [Find Shortest path from source to destination in a matrix that satisfies given constraints](http://www.techiedelight.com/find-shortest-path-source-destination-matrix-satisfies-given-constraints/)  
 [Change all elements of row i and column j in a matrix to 0 if cell (i, j) has value 0](http://www.techiedelight.com/change-elements-row-column-j-matrix-0-cell-j-value-0/)  
 [Print diagonal elements of the matrix having positive slope](http://www.techiedelight.com/print-matrix-diagonally-positive-slope/)  
 [Find all paths from first cell to last cell of a matrix](http://www.techiedelight.com/find-all-paths-from-source-to-destination-in-matrix/)  
 [Replace all occurrences of 0 that are not surrounded by 1 in a binary matrix](http://www.techiedelight.com/replace-occurrences-0-not-surrounded-1-binary-matrix/)  
 [In-place rotate the matrix by 90 degrees in clock-wise direction](http://www.techiedelight.com/place-rotate-matrix-90-degrees-clock-wise-direction/)  
 [Count negative elements present in sorted matrix in linear time](http://www.techiedelight.com/count-negative-elements-present-sorted-matrix/)  
 [Report all occurrences of an element in row wise and column wise sorted matrix in linear time](http://www.techiedelight.com/report-all-occurrences-of-an-element-in-sorted-matrix/)  
 [Calculate sum of all elements in a sub-matrix in constant time](http://www.techiedelight.com/calculate-sum-elements-sub-matrix-constant-time/)  
 [Find maximum sum K x K sub-matrix in a given M x N matrix](http://www.techiedelight.com/find-maximum-sum-submatrix-in-given-matrix/)  
 [Find maximum sum submatrix present in a given matrix](http://www.techiedelight.com/find-maximum-sum-submatrix-present-given-matrix/)  
 [Find probability that a person is alive after taking N steps on the island](http://www.techiedelight.com/probability-alive-after-taking-n-steps-island/)  
 [Count the number of islands](http://www.techiedelight.com/count-the-number-of-islands/)  
 [Flood fill Algorithm](http://www.techiedelight.com/flood-fill-algorithm/)  
 [Find shortest safe route in a field with sensors present](http://www.techiedelight.com/find-shortest-safe-route-field-sensors-present/)  
 [Find all occurrences of given string in a character matrix](http://www.techiedelight.com/find-occurrences-given-string-character-matrix/)  
 [Lee algorithm | Shortest path in a Maze](http://www.techiedelight.com/lee-algorithm-shortest-path-in-a-maze/)  
 [Fill Binary Matrix with Alternating Rectangles of 0 and 1](http://www.techiedelight.com/fill-binary-matrix-alternating-rectangles-0-1/)  
 [Check if given matrix is Toeplitz matrix or not](http://www.techiedelight.com/check-given-matrix-toeplitz-matrix-not/)  
 [In-place rotate the matrix by 180 degrees](http://www.techiedelight.com/inplace-rotate-matrix-180-degrees/)  
 [Travelling Salesman Problem using Branch and Bound](http://www.techiedelight.com/travelling-salesman-problem-using-branch-and-bound/)  
 [Collect maximum points in a matrix by satisfying given constraints](http://www.techiedelight.com/collect-maximum-points-matrix-satisfying-given-constraints/)  
 [Count number of paths in a matrix with given cost to reach destination cell](http://www.techiedelight.com/counting-paths-on-grid-to-reach-destination-cell/)  
 [Find longest sequence formed by adjacent numbers in the matrix](http://www.techiedelight.com/find-longest-sequence-formed-adjacent-numbers-matrix/)  
 [Find the minimum cost to reach last cell of the matrix from its first cell](http://www.techiedelight.com/find-minimum-cost-reach-last-cell-matrix-first-cell/)  
 [Matrix Chain Multiplication](http://www.techiedelight.com/matrix-chain-multiplication/)  
 [Find size of largest square sub-matrix of 1’s present in given binary matrix](http://www.techiedelight.com/find-size-largest-square-sub-matrix-1s-present-given-binary-matrix/)  
 [Chess Knight Problem – Find Shortest path from source to destination](http://www.techiedelight.com/chess-knight-problem-find-shortest-path-source-destination/)  
 [Find Duplicate rows in a binary matrix](http://www.techiedelight.com/find-duplicate-rows-binary-matrix/)  
 [Print all possible solutions to N Queens problem](http://www.techiedelight.com/print-possible-solutions-n-queens-problem/)  
 [Print all Possible Knight’s Tours in a chessboard](http://www.techiedelight.com/print-possible-knights-tours-chessboard/)  
 [Find Shortest Path in Maze](http://www.techiedelight.com/find-shortest-path-in-maze/)  
 [Find Longest Possible Route in a Matrix](http://www.techiedelight.com/find-longest-possible-route-matrix/)  
 [Calculate size of the largest plus of 1's in binary matrix](http://www.techiedelight.com/calculate-size-largest-plus-1s-binary-matrix/)

**Queue:**

[Queue Implementation](http://www.techiedelight.com/circular-queue-implementation-c/)  
 [Queue Implementation using Linked List](http://www.techiedelight.com/queue-implementation-using-linked-list/)  
 [Chess Knight Problem – Find Shortest path from source to destination](http://www.techiedelight.com/chess-knight-problem-find-shortest-path-source-destination/)  
 [Lee algorithm | Shortest path in a Maze](http://www.techiedelight.com/lee-algorithm-shortest-path-in-a-maze/)  
 [Find shortest safe route in a field with sensors present](http://www.techiedelight.com/find-shortest-safe-route-field-sensors-present/)  
 [Flood fill Algorithm](http://www.techiedelight.com/flood-fill-algorithm/)  
 [Count the number of islands](http://www.techiedelight.com/count-the-number-of-islands/)  
 [Find Shortest path from source to destination in a matrix that satisfies given constraints](http://www.techiedelight.com/find-shortest-path-source-destination-matrix-satisfies-given-constraints/)  
 [Generate binary numbers between 1 to N](http://www.techiedelight.com/generate-binary-numbers-1-n/)  
 [Calculate height of a binary tree | Iterative & Recursive](http://www.techiedelight.com/calculate-height-binary-tree-iterative-recursive/)  
 [Delete given Binary Tree | Iterative & Recursive](http://www.techiedelight.com/delete-given-binary-tree-iterative-recursive/)  
 [Level Order Traversal of Binary Tree](http://www.techiedelight.com/level-order-traversal-binary-tree/)  
 [Spiral Order Traversal of Binary Tree](http://www.techiedelight.com/spiral-order-traversal-binary-tree/)  
 [Reverse Level Order Traversal of Binary Tree](http://www.techiedelight.com/reverse-level-order-traversal-binary-tree/)  
 [Print all nodes of a given binary tree in specific order](http://www.techiedelight.com/print-nodes-binary-tree-specific-order/)  
 [Print left view of binary tree](http://www.techiedelight.com/print-left-view-of-binary-tree/)  
 [Find next node in same level for given node in a binary tree](http://www.techiedelight.com/find-next-node-in-same-level-binary-tree/)  
 [Check if given binary tree is complete binary tree or not](http://www.techiedelight.com/check-given-binary-tree-complete-binary-tree-not/)  
 [Print Diagonal Traversal of Binary Tree](http://www.techiedelight.com/print-diagonal-traversal-binary-tree/)  
 [Print corner nodes of every level in binary tree](http://www.techiedelight.com/print-corner-nodes-every-level-binary-tree/)  
 [Breadth First Search (BFS) | Iterative & Recursive Implementation](http://www.techiedelight.com/breadth-first-search/)  
 [Minimum number of throws required to win Snake and Ladder game](http://www.techiedelight.com/min-throws-required-to-win-snake-and-ladder-game/)  
 [Check if an undirected graph contains cycle or not](http://www.techiedelight.com/check-undirected-graph-contains-cycle-not/)  
 [Invert given Binary Tree | Recursive and Iterative solution](http://www.techiedelight.com/invert-binary-tree-recursive-iterative/)  
 [Print Right View of a Binary Tree](http://www.techiedelight.com/print-right-view-binary-tree/)  
 [Traverse the given directory using BFS and DFS in Java](http://www.techiedelight.com/traverse-given-directory-bfs-dfs-java/)  
**Sorting**

[Insertion sort | Iterative & Recursive](http://www.techiedelight.com/insertion-sort-iterative-recursive/)  
 [Selection sort | Iterative & Recursive](http://www.techiedelight.com/selection-sort-iterative-recursive/)  
 [Bubble sort | Iterative & Recursive](http://www.techiedelight.com/bubble-sort-iterative-recursive/)  
 [Merge Sort](http://www.techiedelight.com/merge-sort/)  
 [Iterative Merge Sort Algorithm (Bottom-up Merge Sort)](http://www.techiedelight.com/iterative-merge-sort-algorithm-bottom-up/)  
 [Quicksort](http://www.techiedelight.com/quicksort/)  
 [Iterative Implementation of Quicksort](http://www.techiedelight.com/iterative-implementation-of-quicksort/)  
 [Hybrid QuickSort](http://www.techiedelight.com/hybrid-quicksort/)  
 [Quicksort using Dutch National Flag Algorithm](http://www.techiedelight.com/quicksort-using-dutch-national-flag-algorithm/)  
 [Quick Sort using Hoare’s Partitioning scheme](http://www.techiedelight.com/quick-sort-using-hoares-partitioning-scheme/)  
 [Counting Sort Algorithm](http://www.techiedelight.com/counting-sort-algorithm-implementation/)  
 [External merge sort](http://www.techiedelight.com/external-merge-sort/)  
 [Custom Sort | Sort elements by their frequency and Index](http://www.techiedelight.com/sort-elements-by-their-frequency-and-index/)  
 [Custom Sort | Sort elements of the array by order of elements defined by the second array](http://www.techiedelight.com/custom-sort-sort-elements-array-order-elements-defined-second-array/)  
 [Inversion Count of an array](http://www.techiedelight.com/inversion-count-array/)  
 [Segregate positive and negative integers in linear time](http://www.techiedelight.com/positive-and-negative-integers-segregate/)  
 [Sort linked list containing 0’s, 1’s and 2’s](http://www.techiedelight.com/sort-linked-list-containing-0s-1s-2s/)  
 [Efficiently Sort an Array with many Duplicated Values](http://www.techiedelight.com/efficiently-sort-array-duplicated-values/)  
 [Find the smallest window in array sorting which will make the entire array sorted](http://www.techiedelight.com/smallest-window-sorting-which-make-array-sorted/)  
 [Find largest number possible from set of given numbers](http://www.techiedelight.com/find-largest-number-possible-set-given-numbers/)  
 [Move all zeros present in the array to the end](http://www.techiedelight.com/move-zeros-present-array-end/)  
 [Sort linked list containing 0’s, 1’s and 2’s](http://www.techiedelight.com/sort-linked-list-containing-0s-1s-2s/)  
 [Sort binary array in linear time](http://www.techiedelight.com/sort-binary-array-linear-time/)  
 [Merge Sort for Singly Linked List](http://www.techiedelight.com/merge-sort-singly-linked-list/)  
 [Group anagrams together from given list of words](http://www.techiedelight.com/group-anagrams-together-given-list-words/)  
 [Activity Selection Problem](http://www.techiedelight.com/activity-selection-problem/)  
 [Lexicographic sorting of given set of keys](http://www.techiedelight.com/lexicographic-sorting-given-set-of-keys/)  
 [Heap Sort (Out-of-place and In-place implementation in C++ and C)](http://www.techiedelight.com/heap-sort-place-place-implementation-c-c/)  
 [Merge M sorted lists of variable length](http://www.techiedelight.com/merge-m-sorted-lists-variable-length/)  
 [Merge M sorted lists each containing N elements](http://www.techiedelight.com/merge-m-sorted-lists-containing-n-elements/)  
 [Find all palindromic permutations of a string](http://www.techiedelight.com/find-palindromic-permutations-string/)  
 [Find all lexicographically next permutations of a string sorted in ascending order](http://www.techiedelight.com/find-lexicographically-next-permutations-string-sorted-ascending-order/)  
 [Merge two sorted linked lists from their end](http://www.techiedelight.com/merge-two-sorted-linked-lists-end/)  
 [Sort an array containing 0’s, 1’s and 2’s (Dutch national flag problem)](http://www.techiedelight.com/sort-array-containing-0s-1s-2s-dutch-national-flag-problem/)  
 [Find pair with given sum in the array](http://www.techiedelight.com/find-pair-with-given-sum-array/)  
 [Inplace merge two sorted arrays](http://www.techiedelight.com/inplace-merge-two-sorted-arrays/)  
 [Merge two arrays by satisfying given constraints](http://www.techiedelight.com/merge-two-arrays-satisfying-given-constraints/)  
 [Find maximum product of two integers in an array](http://www.techiedelight.com/find-maximum-product-two-integers-array/)  
 [Find all distinct combinations of given length](http://www.techiedelight.com/find-distinct-combinations-of-given-length/)  
 [Find all distinct combinations of given length with repetition allowed](http://www.techiedelight.com/find-distinct-combinations-given-length-repetition-allowed/)  
 [Merging Overlapping Intervals](http://www.techiedelight.com/merging-overlapping-intervals)  
 [Sort an array using one swap](http://www.techiedelight.com/sort-array-using-one-swap/)  
 [4 sum problem | Quadruplets with given sum](http://www.techiedelight.com/4-sum-problem/)  
 [Print all quadruplets with given sum | 4-sum problem extended](http://www.techiedelight.com/print-all-quadruplets-with-given-sum-4-sum-problem-extended/)

**Stack:**

[Stack Implementation](http://www.techiedelight.com/stack-implementation/)  
 [Stack Implementation using Linked List](http://www.techiedelight.com/stack-implementation-using-linked-list/)  
 [Check if given expression is balanced expression or not](http://www.techiedelight.com/check-given-expression-balanced-expression-not/)  
 [Find duplicate parenthesis in an expression](http://www.techiedelight.com/find-duplicate-parenthesis-expression/)  
 [Evaluate given postfix expression](http://www.techiedelight.com/evaluate-given-postfix-expression/)  
 [Decode the given sequence to construct minimum number without repeated digits](http://www.techiedelight.com/decode-the-given-sequence-construct-minimum-number-without-repeated-digits/)  
 [Inorder Tree Traversal | Iterative & Recursive](http://www.techiedelight.com/inorder-tree-traversal-iterative-recursive/)  
 [Preorder Tree Traversal | Iterative & Recursive](http://www.techiedelight.com/preorder-tree-traversal-iterative-recursive/)  
 [Postorder Tree Traversal | Iterative & Recursive](http://www.techiedelight.com/postorder-tree-traversal-iterative-recursive/)  
 [Find ancestors of given node in a Binary Tree](http://www.techiedelight.com/find-ancestors-of-given-node-binary-tree/)  
 [Check if two given binary trees are identical or not | Iterative & Recursive](http://www.techiedelight.com/check-if-two-binary-trees-are-identical-not-iterative-recursive/)  
 [Reverse given text without reversing the individual words](http://www.techiedelight.com/reverse-text-without-reversing-individual-words/)  
 [Find all binary strings that can be formed from given wildcard pattern](http://www.techiedelight.com/find-binary-strings-can-formed-given-wildcard-pattern/)  
 [Iterative Implementation of Quicksort](http://www.techiedelight.com/iterative-implementation-of-quicksort/)  
 [Depth First Search (DFS) | Iterative & Recursive Implementation](http://www.techiedelight.com/depth-first-search/)  
 [Invert given Binary Tree | Recursive and Iterative solution](http://www.techiedelight.com/invert-binary-tree-recursive-iterative/)  
 [Print leaf to root path for every leaf node in a binary tree](http://www.techiedelight.com/print-leaf-to-root-path-binary-tree/)  
**String**

[Check if given set of moves is circular or not](http://www.techiedelight.com/check-given-set-moves-circular-not/)  
 [Check if given string is a rotated palindrome or not](http://www.techiedelight.com/check-given-string-rotated-palindrome-not/)  
 [Longest Palindromic Substring (Non-DP Space Optimized Solution)](http://www.techiedelight.com/longest-palindromic-substring-non-dp-space-optimized-solution/)  
 [Check if repeated subsequence is present in the string or not](http://www.techiedelight.com/check-repeated-subsequence-present-string-not/)  
 [Check if strings can be derived from each other by circularly rotating them](http://www.techiedelight.com/check-strings-can-derived-circularly-rotating/)  
 [Convert given number into corresponding excel column name](http://www.techiedelight.com/convert-given-number-corresponding-excel-column-name/)  
 [Determine if two strings are anagram or not](http://www.techiedelight.com/determine-if-two-strings-are-anagram-or-not/)  
 [Find all binary strings that can be formed from given wildcard pattern](http://www.techiedelight.com/find-binary-strings-can-formed-given-wildcard-pattern/)  
 [Find all interleavings of given strings](http://www.techiedelight.com/find-interleavings-of-given-strings/)  
 [Isomorphic Strings](http://www.techiedelight.com/isomorphic-strings/)  
 [Find all possible palindromic substrings in a string](http://www.techiedelight.com/find-possible-palindromic-substrings-string/)  
 [Find all possible combinations of words formed from mobile keypad](http://www.techiedelight.com/find-possible-combinations-words-formed-from-mobile-keypad/)  
 [Find all possible combinations by replacing given digits with characters of the corresponding list](http://www.techiedelight.com/possible-combinations-replacing-given-digits-corresponding-list/)  
 [Find all words from given list that follows same order of characters as given pattern](http://www.techiedelight.com/find-words-that-follows-given-pattern/)  
 [Find first k non-repeating characters in a string in single traversal](http://www.techiedelight.com/first-k-non-repeating-characters-string/)  
 [Group anagrams together from given list of words](http://www.techiedelight.com/group-anagrams-together-given-list-words/)  
 [Introduction to Pattern Matching](http://www.techiedelight.com/introduction-pattern-matching/)  
 [Inplace remove all occurrences of ‘AB’ and ‘C’ from the string](http://www.techiedelight.com/inplace-remove-all-occurrences-ab-c-string/)  
 [Longest even length palidromic sum substring](http://www.techiedelight.com/longest-even-length-palidromic-sum-substring/)  
 [Print string in zig-zag form in k rows](http://www.techiedelight.com/print-string-in-zig-zag-form-k-rows/)  
 [Reverse given text without reversing the individual words](http://www.techiedelight.com/reverse-text-without-reversing-individual-words/)  
 [Run Length Encoding (RLE) data compression algorithm](http://www.techiedelight.com/run-length-encoding-rle-data-compression-algorithm/)  
 [Validate an IP address](http://www.techiedelight.com/validate-ip-address/)  
 [Find the longest substring of given string containing k distinct characters](http://www.techiedelight.com/find-longest-substring-containing-k-distinct-characters/)  
 [Find all palindromic permutations of a string](http://www.techiedelight.com/find-palindromic-permutations-string/)  
 [Find all substrings of a string that are permutation of a given string](http://www.techiedelight.com/find-substrings-string-permutation-given-string/)  
 [Find the longest substring of given string containing all distinct characters](http://www.techiedelight.com/find-longest-substring-given-string-containing-distinct-characters/)  
 [Find all Permutations of a given string](http://www.techiedelight.com/find-permutations-given-string/)  
 [Iterative Approach to find Permutations of a String in C++ and Java](http://www.techiedelight.com/find-permutations-string-cpp-java-iterative/)  
 [Generate all Permutations of a String in Java | Recursive & Iterative](http://www.techiedelight.com/generate-permutations-string-java-recursive-iterative/)  
 [Find all lexicographically next permutations of a string sorted in ascending order](http://www.techiedelight.com/find-lexicographically-next-permutations-string-sorted-ascending-order/)  
 [Find Lexicographically minimal string rotation](http://www.techiedelight.com/find-lexicographically-minimal-string-rotation/)  
 [Find all strings of given length containing balanced parentheses](http://www.techiedelight.com/find-strings-given-length-containing-balanced-parentheses/)  
 [Find all N-digit binary numbers with k-bits set where k ranges from 1 to N](http://www.techiedelight.com/find-n-digit-binary-numbers-k-bits-set-k-ranges-1-n/)  
 [Find all N-digit strictly increasing numbers (Bottom-Up and Top-Down Approach)](http://www.techiedelight.com/find-n-digit-strictly-increasing-numbers-bottom-top-approach/)  
 [Find all N-digit binary numbers having more 1’s than 0’s for any prefix](http://www.techiedelight.com/find-n-digit-binary-numbers-having-more-one-than-zero/)  
 [Find all N-digit numbers with given sum of digits](http://www.techiedelight.com/find-all-n-digit-numbers-given-sum-digits/)  
 [Generate binary numbers between 1 to N](http://www.techiedelight.com/generate-binary-numbers-1-n/)  
 [Find all combinations of non-overlapping substrings of a string](http://www.techiedelight.com/find-combinations-non-overlapping-substrings-string/)  
 [Check if given sentence is syntactically correct or not](http://www.techiedelight.com/check-given-sentence-syntactically-correct-not/)  
 [Calculate rank of given string among all its lexicographically sorted permutations](http://www.techiedelight.com/calculate-rank-lexicographically-sorted-permutations/)  
 [Length of longest continuous sequence with same sum in given binary arrays](http://www.techiedelight.com/length-longest-continuous-sequence-same-sum-binary-arrays/)  
 [Find all Lexicographic Permutations of a String](http://www.techiedelight.com/find-lexicographic-permutations-string/)  
 [Find all N-digit binary numbers with equal sum of bits in its two halves](http://www.techiedelight.com/find-n-digit-binary-numbers-equal-sum-bits-two-halves/)  
 [Check if given string is interleaving of two other given strings](http://www.techiedelight.com/check-string-interleaving-two-given-strings/)  
 [Difference between Subarray, Subsequence and Subset](http://www.techiedelight.com/difference-between-subarray-subsequence-subset/)  
 [std::next\_permutation | Overview & Implementation in C++](http://www.techiedelight.com/std_next_permutation-overview-implementation/)  
 [std::prev\_permutation | Overview & Implementation in C++](http://www.techiedelight.com/std_prev_permutation-overview-implementation/)  
 [Implementation of KMP Algorithm in C, C++ and Java](http://www.techiedelight.com/implementation-kmp-algorithm-c-cpp-java/)  
 [Reverse String without using Recursion](http://www.techiedelight.com/reverse-string-without-using-recursion/)  
 [Reverse given string using Recursion](http://www.techiedelight.com/reverse-a-string-using-recursion/)  
 [Reverse a String in Java in 10 different ways](http://www.techiedelight.com/10-ways-reverse-a-string-java/)  
 [Determine if a given string is palindrome or not](http://www.techiedelight.com/determine-given-string-is-palindrome-not/)  
 [Combinations of words formed by replacing given numbers with corresponding alphabets](http://www.techiedelight.com/combinations-of-words-formed-replacing-given-numbers-corresponding-english-alphabet/)  
 [Word Break Problem](http://www.techiedelight.com/word-break-problem/)  
 [Word Break Problem | Using Trie Data Structure](http://www.techiedelight.com/word-break-problem-using-trie/)  
 [Wildcard Pattern Matching](http://www.techiedelight.com/wildcard-pattern-matching/)  
 [Count number of times a pattern appears in given string as a subsequence](http://www.techiedelight.com/count-number-times-pattern-appears-given-string-subsequence/)  
 [Total possible solutions to linear equation of k variables](http://www.techiedelight.com/total-possible-solutions-linear-equation-k-variables/)  
 [The Levenshtein distance (Edit distance) problem](http://www.techiedelight.com/levenshtein-distance-edit-distance-problem/)  
 [Longest Common Subsequence | Introduction & LCS Length](http://www.techiedelight.com/longest-common-subsequence/)  
 [Longest Common Subsequence | Space optimized version](http://www.techiedelight.com/longest-common-subsequence-lcs-space-optimized-version/)  
 [Longest Common Subsequence | Finding all LCS](http://www.techiedelight.com/longest-common-subsequence-finding-lcs/)  
 [Longest Common Subsequence of K-sequences](http://www.techiedelight.com/longest-common-subsequence-of-k-sequences/)  
 [Longest Repeated Subsequence problem](http://www.techiedelight.com/longest-repeated-subsequence-problem/)  
 [Longest Palindromic Subsequence using Dynamic Programming](http://www.techiedelight.com/longest-palindromic-subsequence-using-dynamic-programming/)  
 [Longest Common Substring problem](http://www.techiedelight.com/longest-common-substring-problem/)  
 [Implement Diff Utility](http://www.techiedelight.com/implement-diff-utility/)  
 [Shortest Common Supersequence | Introduction & SCS Length](http://www.techiedelight.com/shortest-common-supersequence-introduction-scs-length/)  
 [Shortest Common Supersequence | Finding all SCS](http://www.techiedelight.com/shortest-common-supersequence-finding-scs/)  
 [Shortest Common Supersequence | Using LCS](http://www.techiedelight.com/shortest-common-supersequence-using-lcs/)  
 [Find minimum cuts needed for palindromic partition of a string](http://www.techiedelight.com/find-minimum-cuts-needed-palindromic-partition-string/)

**Trie:**

[Trie Implementation | Insert, Search and Delete](http://www.techiedelight.com/trie-implementation-insert-search-delete/)  
 [Memory efficient Trie Implementation using Map | Insert, Search and Delete](http://www.techiedelight.com/memory-efficient-trie-implementation-using-map-insert-search-delete/)  
 [C++ Implementation of Trie Data Structure](http://www.techiedelight.com/cpp-implementation-trie-data-structure/)  
 [Longest Common Prefix in given set of strings (using Trie)](http://www.techiedelight.com/longest-common-prefix-given-set-strings-using-trie/)  
 [Lexicographic sorting of given set of keys](http://www.techiedelight.com/lexicographic-sorting-given-set-of-keys/)  
 [Find maximum occurring word in given set of strings](http://www.techiedelight.com/find-maximum-occurring-word-given-set-strings/)  
 [Find first k maximum occurring words in given set of strings](http://www.techiedelight.com/find-first-k-maximum-occurring-words-given-set-strings/)  
 [Find Duplicate rows in a binary matrix](http://www.techiedelight.com/find-duplicate-rows-binary-matrix/)  
 [Word Break Problem | Using Trie Data Structure](http://www.techiedelight.com/word-break-problem-using-trie/)

**Greedy:**

[Activity Selection Problem](http://www.techiedelight.com/activity-selection-problem/)  
 [Huffman Coding](http://www.techiedelight.com/huffman-coding/)  
 [Shortest Superstring Problem](http://www.techiedelight.com/shortest-superstring-problem/)  
 [Job Sequencing Problem with Deadlines](http://www.techiedelight.com/job-sequencing-problem-deadlines/)  
 [Greedy coloring of graph](http://www.techiedelight.com/greedy-coloring-graph/)  
 [Kruskal’s Algorithm for finding Minimum Spanning Tree](http://www.techiedelight.com/kruskals-algorithm-for-finding-minimum-spanning-tree/)  
 [Single-Source Shortest Paths – Dijkstra’s Algorithm](http://www.techiedelight.com/single-source-shortest-paths-dijkstras-algorithm/)

**Puzzles:**

[Clock angle problem – Find ? between hour and minute hand](http://www.techiedelight.com/angle-between-hour-minute-hand/)  
 [Add two numbers without using addition operator | 4 methods](http://www.techiedelight.com/add-two-numbers-without-using-addition-operator/)  
 [Generate power set of a given set](http://www.techiedelight.com/generate-power-set-given-set/)  
 [Implement power function without using multiplication and division operators](http://www.techiedelight.com/implement-power-function-without-using-multiplication-division-operators/)  
 [Print all numbers between 1 to N without using semicolon](http://www.techiedelight.com/print-numbers-1-n-without-using-semicolon/)  
 [Swap two numbers without using third variable | 5 methods](http://www.techiedelight.com/swap-two-numbers-without-using-third-variable/)  
 [Determine the if condition to print specific output](http://www.techiedelight.com/determine-condition-to-print-specific-output/)  
 [Find maximum, minimum of 3 numbers without using conditional statement & ternary operator](http://www.techiedelight.com/maximum-minimum-three-numbers-without-using-conditional-statement-ternary-operator/)  
 [Find numbers represented as sum of two cubes for two different pairs](http://www.techiedelight.com/numbers-represented-as-sum-of-two-cubes/)  
 [Print “Hello World” with empty main() function | 3 methods](http://www.techiedelight.com/print-hello-world-empty-main-function/)  
 [Print all numbers between 1 to N without using any loop | 4 methods](http://www.techiedelight.com/print-numbers-1-n-without-using-loop-4-methods/)  
 [Print a semicolon without using semicolon anywhere in the program](http://www.techiedelight.com/print-a-semicolon-without-using-semicolon-anywhere-program/)  
 [Multiply two numbers without using multiplication operator or loops](http://www.techiedelight.com/multiply-two-numbers-without-using-multiplication-operator-loops/)  
 [Find square of a number without using multiplication and division operator | 3 methods](http://www.techiedelight.com/find-square-number-without-using-multiplication-division-operator/)  
 [Set both elements of a binary array to 0 in single line](http://www.techiedelight.com/set-elements-binary-array-0-single-line/)  
 [Find minimum number without using conditional statement or ternary operator](http://www.techiedelight.com/find-minimum-number-without-using-conditional-statement-ternary-operator/)  
 [Perform Division of two numbers without using division operator (/)](http://www.techiedelight.com/perform-division-two-numbers-without-using-division-operator/)  
 [Generate 0 and 1 with 75% and 25% Probability](http://www.techiedelight.com/generate-0-1-75-25-probability/)  
 [Generate Desired Random Numbers with Equal Probability](http://www.techiedelight.com/generate-random-numbers-equal-probability/)  
 [Return 0, 1 and 2 with equal Probability using the specified function](http://www.techiedelight.com/return-0-1-2-equal-probability-using-specified-function/)  
 [Generate Fair Results from a Biased Coin](http://www.techiedelight.com/generate-fair-results-biased-coin/)  
 [Generate numbers from 1 to 7 with equal probability using specified function](http://www.techiedelight.com/generate-numbers-1-7-equal-probability/)  
 [Implement Ternary Operator Without Using Conditional Expressions](http://www.techiedelight.com/implement-ternary-operator-without-using-conditional-statements/)  
 [Determine if two integers are equal without using comparison and arithmetic operators](http://www.techiedelight.com/determine-two-integers-equal-without-using-comparison-arithmetic-operators/)  
 [Return 0 and 1 with equal Probability using the specified function](http://www.techiedelight.com/get-0-1-equal-probability-using-specified-function/)  
 [Tower of Hanoi Problem](http://www.techiedelight.com/tower-of-hanoi-problem/)  
 [Magnet Puzzle](http://www.techiedelight.com/magnet-puzzle/)