原题是找到一组数的全排列

Given a collection of numbers, return all possible permutations.

For example,  
[1,2,3] have the following permutations:  
[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], and [3,2,1].

函数原型：

vector<vector<int> > permute(vector<int> &num;

这个题大眼一看就是思路一大坨，这里做一个整理吧

**思路1**

比较直观的想法就是递归咯~~ 在num中拿出1个数字放在第一个，然后剩下的数字做一个全排列，最早接触这个问题的时候我就是这么写的

**[cpp]** [view plain](http://blog.csdn.net/tuantuanls/article/details/8717262) [copy](http://blog.csdn.net/tuantuanls/article/details/8717262)

1. class Solution {
2. public:
3. vector<vector<int> > permute(vector<int> &num) {
4. // Start typing your C/C++ solution below
5. // DO NOT write int main() function
6. int N = num.size();
7. vector<vector<int> > ret;
9. if(N == 1){
10. ret.push\_back(num);
11. return ret;
12. }
13. vector<vector<int> > post;
15. vector<int> cur;
16. vector<int> tmp;
18. for(int i = 0; i < N; i++){
19. cur = num;
20. cur.erase(cur.begin()+i);
21. post = permute(cur);
22. for(int j = 0; j < post.size(); j++){
23. tmp = post[j];
24. tmp.insert(tmp.begin(), num[i]);
25. ret.push\_back(tmp);
26. }
27. }
29. return ret;
30. }
31. };

**思路2：**

建立一棵树，比如说

对于第k层节点来说，就是交换固定了前面 k-1 位，然后分别 swap(k,k), swap(k, k+1) , swap(k, k+2) ...

例如上图中的第三层，固定了第一位（即2），然后分别交换第1，1位，1，2位，1，3位

**[cpp]** [view plain](http://blog.csdn.net/tuantuanls/article/details/8717262) [copy](http://blog.csdn.net/tuantuanls/article/details/8717262)

1. class Solution {
3. vector<vector<int> > ret;
4. int N;
6. public:
7. void perm(vector<int> &num, int i){
8. if( i == N){
9. ret.push\_back(num);
10. }
12. for(int j = i; j < N; j++){
13. swap(num[i], num[j]);
14. perm(num, i + 1);
15. swap(num[j], num[i]);
16. }
17. }

20. vector<vector<int> > permute(vector<int> &num) {
21. // Start typing your C/C++ solution below
22. // DO NOT write int main() function
23. N = num.size();
24. ret.clear();
26. perm(num, 0);
28. return ret;
30. }
31. };

**思路3**

stl的algorithm里面其实是有next permutation的算法的，那其实用next permutation的方法也是一个不错的选择

这个思路可以保证遍历的顺序是字典序，即按照从小到大的顺序

next permutation的算法就是。。。swap + reverse。。。交换 & 倒叙

比如 1，2，3的下一个就是1，3，2这个很容易理解，因为2和3是升序的，只需要交换这两位，那么132 > 123，但是如果后面几位都是倒序的怎么办？

例如 5，4，7，5，3，2 这个序列

我们知道答案应该是 5，5，2，3，4，7

从直观上来说，7，5，3，2已经是这四位排列的最大值了，所以一定要动到 4 这个数字了，所以我们选了刚好比4大的5来和4进行交换，得到 5，5，。。。后面几位就按照升序放进去就可以了

但是令人兴奋的一点是，当4和5交换以后，后面的序列一定是倒序的，所以我们不需要重新sort，只需要将其reverse就可以了

这就是swap  + reverse的思路

注意，下面这个代码里面交换的是 i-1 和 j-1 所以i指向的是7，而j指向的是3

**[cpp]** [view plain](http://blog.csdn.net/tuantuanls/article/details/8717262) [copy](http://blog.csdn.net/tuantuanls/article/details/8717262)

1. class Solution {
2. public:
3. void nextPermutation(vector<int> &num) {
4. // Start typing your C/C++ solution below
5. // DO NOT write int main() function
7. //5,4,7,5,3,2
8. //    |   |
9. //    i   j
10. //5,5,7,4,3,2
11. //5,5,2,3,4,7
12. int i = num.size()-1;
13. while(i > 0 && num[i-1] >= num[i] ){
14. i--;
15. }
17. int j = i;
19. while(j < num.size() && num[j] > num[i-1]) j++;
21. if(i == 0){
22. reverse(num.begin(), num.end());
23. }else{
24. swap(num[i-1], num[j-1]);
25. reverse(num.begin() + i, num.end());
26. }

29. }
31. int factorial(int n){
32. return (n == 1 || n == 0) ? 1 : factorial(n - 1) \* n;
33. }

36. vector<vector<int> > permute(vector<int> &num) {
37. // Start typing your C/C++ solution below
38. // DO NOT write int main() function
39. int N = num.size();
40. vector<vector<int> > ret;
42. ret.push\_back(num);
44. for(int i = 1; i < factorial(N); i++){
45. nextPermutation(num);
46. ret.push\_back(num);
47. }
49. return ret;
51. }
52. };

**思路四**

我觉得思路4是一个很常规的思路，很多把recursive的code改成iterative的code都会用到这样的方法，其实呢，它的本质就是把N个for改成while的方法。介个方法在编程之美里面的“电话号码”那一节提到过，不明白的童鞋可以去看一看，我觉得第一次想写粗来还是很难的，不过多写几个，就会很熟练啦

对应介个题目的思路捏就是。。。举个例子来说吧

如果我想求1,2,3,4的全排列

偶的思路就是建一个特殊的数，它的进位方法是 3, 2, 1, 0

所以，这个数的++过程就是

0000 -> 0010 -> 0100 -> 0110 ->0200 -> 0210 ->

1000 -> 1010 -> 1100 -> 1110 ->1200 -> 1210 ->

2000 -> 2010 -> 2100 -> 2110 ->2200 -> 2210 ->

3000 -> 3010 -> 3100 -> 3110 ->3200 -> 3210

哇哈哈哈，刚好是24个！

然后捏？ b0 b1 b2 b3就代表在当前剩下的数字中选择第bi个

哇！好复杂。。。

比如0210

0: 在1234中选择第0个，就是1

2: 在234中选择滴2个，就是4

1: 在23中选择第1个，就是3

0: 在2中选择第0个，就是2

所以0210对应点就素 1432

**[cpp]** [view plain](http://blog.csdn.net/tuantuanls/article/details/8717262) [copy](http://blog.csdn.net/tuantuanls/article/details/8717262)

1. class Solution {
2. public:
3. int factorial(int n){
4. return (n == 1 || n == 0) ? 1 : factorial(n - 1) \* n;
5. }
7. void plusp(vector<int> &p, const vector<int> &bound){
8. int i = p.size()-1;
9. while(i >= 0){
10. if(p[i] < bound[i]){
11. p[i]++;
12. break;
13. }else{
14. p[i] = 0;
15. i--;
16. }
17. }
19. }


23. vector<vector<int> > permute(vector<int> &num) {
24. // Start typing your C/C++ solution below
25. // DO NOT write int main() function
26. vector<vector<int> > ret;
27. vector<int> ori\_num = num;
28. vector<int> tmp = num;
30. int N = num.size();
32. vector<int> p(N, 0);
34. vector<int> bound = num;
35. for(int i = 0; i < N; i++){
36. bound[i] = N - 1 - i;
37. }
39. for(int i = 0; i < factorial(N); i++){
40. num = ori\_num;
41. for(int j = 0; j < N; j++){
42. tmp[j] = num[p[j]];
43. num.erase(num.begin() + p[j]);
44. }
45. ret.push\_back(tmp);
46. plusp(p, bound);
48. }
50. return ret;
52. }
53. };