

```

1  template<typename val_t>
2  class Partial_Permutation {
3  private:
4      int n;
5      vector<bool> used;
6      vector<vector<val_t>> result;
7      vvi facts; // iPj
8      void core_func(const vector<val_t> &a, int n, int r, int start) {
9          if (r == 0 || n < r) return;
10         int m = facts[n - 1][r - 1];
11         int cnt = 0;
12         Loop(i, Partial_Permutation::n) {
13             if (!used[i]) {
14                 Loop(j, m) {
15                     result[start + m * cnt + j].push_back(a[i]);
16                 }
17                 used[i] = true;
18                 core_func(a, n - 1, r - 1, start + m * cnt);
19                 used[i] = false;
20                 cnt++;
21             }
22         }
23     }
24     void make_facts(int n) {
25         facts = vvi(n + 1, vi(n + 1));
26         Loop(i, n + 1) {
27             facts[i][0] = 1;
28             Loop(j, i) {
29                 facts[i][j + 1] = facts[i][j] * (i - j);
30             }
31         }
32     }
33 public:
34     vector<vector<val_t>> get_partial_permutation(const vector<val_t> &a, int r) {
35         n = int(a.size());
36         if (n < r) return {};
37         used = vector<bool>(n, false);
38         make_facts(n);
39         result = vector<vector<val_t>>(facts[n][r]);
40         core_func(a, n, r, 0);
41         return result;
42     }
43 };
44
45
46 class Partial_Permutation_String {
47 private:
48     int n;
49     string a;
50     vector<bool> used;
51     vector<string> result;
52     vvi facts; // iPj
53     void core_func(const string &a, int n, int r, int start) {
54         if (r == 0 || n < r) return;
55         int m = facts[n - 1][r - 1];
56         int cnt = 0;
57         Loop(i, Partial_Permutation_String::n) {
58             if (!used[i]) {
59                 Loop(j, m) {
60                     result[start + m * cnt + j] += a[i];
61                 }
62                 used[i] = true;
63                 core_func(a, n - 1, r - 1, start + m * cnt);
64                 used[i] = false;
65                 cnt++;
66             }
67         }
68     }
69     void make_facts(int n) {
70         facts = vvi(n + 1, vi(n + 1));
71         Loop(i, n + 1) {

```

```
72     facts[i][0] = 1;
73     Loop(j, i) {
74         facts[i][j + 1] = facts[i][j] * (i - j);
75     }
76 }
77 }
78 public:
79     vector<string> get_partial_permutation(const string &a, int r) {
80         n = int(a.size());
81         if (n < r) return {};
82         used = vector<bool>(n, false);
83         make_facts(n);
84         result = vector<string>(facts[n][r]);
85         core_func(a, n, r, 0);
86         return result;
87     }
88 };
```