

```

1  template<typename val_t>
2  class Partial_Combination {
3  private:
4      int n;
5      vector<vector<val_t>> result;
6      vvi combs; // iCj
7      void core_func(const vector<val_t> &a, int n, int r, int start) {
8          if (r == 0 || n < r) return;
9          Loop(i, combs[n - 1][r - 1]) {
10             result[start + i].push_back(a[Partial_Combination::n - n]);
11         }
12         if (n > 1) {
13             core_func(a, n - 1, r - 1, start);
14             core_func(a, n - 1, r, start + combs[n - 1][r - 1]);
15         }
16     }
17     void make_combs(int n) {
18         combs = vvi(n + 1, vi(n + 1));
19         Loop(i, n + 1) {
20             combs[i][0] = 1;
21             Loop1(j, i) {
22                 combs[i][j] = combs[i - 1][j - 1] + combs[i - 1][j];
23             }
24         }
25     }
26 public:
27     vector<vector<val_t>> get_partial_combination(const vector<val_t> &a, int r) {
28         n = int(a.size());
29         if (n < r) return {};
30         make_combs(n);
31         result = vector<vector<val_t>>(combs[n][r]);
32         core_func(a, n, r, 0);
33         return result;
34     }
35 };
36
37 class Partial_Combination_Bitmask {
38 private:
39     int n;
40     vll result;
41     vvi combs; // iCj
42     void core_func(const ll &a, int n, int r, int start) {
43         if (r == 0 || n < r) return;
44         ll x = a & -a;
45         Loop(i, combs[n - 1][r - 1]) {
46             result[start + i] += x;
47         }
48         if (n > 1) {
49             core_func(a - x, n - 1, r - 1, start);
50             core_func(a - x, n - 1, r, start + combs[n - 1][r - 1]);
51         }
52     }
53     void make_combs(int n) {
54         combs = vvi(n + 1, vi(n + 1));
55         Loop(i, n + 1) {
56             combs[i][0] = 1;
57             Loop1(j, i) {
58                 combs[i][j] = combs[i - 1][j - 1] + combs[i - 1][j];
59             }
60         }
61     }
62 public:
63     vll get_partial_combination(int n, int r) {
64         this->n = n;
65         if (n < r) return {};
66         make_combs(n);
67         result = vll(combs[n][r]);
68         ll a = (1LL << n) - 1;
69         core_func(a, n, r, 0);
70         return result;
71     }

```

72 };