```cpp
template<typename val_t>
class Partial_Combination {
private:
  int n;
  vector<vector<val_t>> result;
  vvi combs; // iCj
  void core_func(const vector<val_t> &a, int n, int r, int start) {
    if (r == 0 || n < r) return;
    Loop(i, combs[n - 1][r - 1]) {
      result[start + i].push_back(a[Partial_Combination::n - n]);
    }
    if (n > 1) {
      core_func(a, n - 1, r - 1, start);
      core_func(a, n - 1, r, start + combs[n - 1][r - 1]);
    }
  }
  void make_combs(int n) {
    combs = vvi(n + 1, vi(n + 1));
    Loop(i, n + 1) {
      combs[i][0] = 1;
      Loop1(j, i) {
        combs[i][j] = combs[i - 1][j - 1] + combs[i - 1][j];
      }
    }
  }
public:
  vector<vector<val_t>> get_partial_combination(const vector<val_t> &a, int r) {
    n = int(a.size());
    if (n < r) return {};
    make_combs(n);
    result = vector<vector<val_t>>(combs[n][r]);
    core_func(a, n, r, 0);
    return result;
  }
};

class Partial_Combination_Bitmask {
private:
  int n;
  vll result;
  vvi combs; // iCj
  void core_func(const ll &a, int n, int r, int start) {
    if (r == 0 || n < r) return;
    ll x = a & -a;
    Loop(i, combs[n - 1][r - 1]) {
      result[start + i] += x;
    }
    if (n > 1) {
      core_func(a - x, n - 1, r - 1, start);
      core_func(a - x, n - 1, r, start + combs[n - 1][r - 1]);
    }
  }
  void make_combs(int n) {
    combs = vvi(n + 1, vi(n + 1));
    Loop(i, n + 1) {
      combs[i][0] = 1;
      Loop1(j, i) {
        combs[i][j] = combs[i - 1][j - 1] + combs[i - 1][j];
      }
    }
  }
public:
  vll get_partial_combination(int n, int r) {
    this->n = n;
    if (n < r) return {};
    make_combs(n);
    result = vll(combs[n][r]);
    ll a = (1LL << n) - 1;
    core_func(a, n, r, 0);
    return result;
  }
```

```
72  };
```