5.3

(0, 0, 0, 0) (1, 0, 0, 0) (0, 1, 0, 0) (1, 1, 0, 0) (0, 0, 1, 0) (1, 0, 1, 0) (0, 1, 1, 0) (1, 1, 1, 0)

(0, 0, 2, 0) (1, 0, 2, 0) (0, 1, 2, 0) (1, 1, 2, 0) (0, 0, 0, 1) (1, 0, 0, 1) (0, 1, 0, 1) (1, 1, 0, 1)

(0, 0, 1, 1) (1, 0, 1, 1) (0, 1, 1, 1) (1, 1, 1, 1) (0, 0, 2, 1) (1, 0, 2, 1) (0, 1, 2, 1) (1, 1, 2, 1)

(0, 0, 0, 2) (1, 0, 0, 2) (0, 1, 0, 2) (1, 1, 0, 2) (0, 0, 1, 2) (1, 0, 1, 2) (0, 1, 1, 2) (1, 1, 1, 2)

(0, 0, 2, 2) (1, 0, 2, 2) (0, 1, 2, 2) (1, 1, 2, 2)

5.6

$u = i - j + 1, v = j - 1$

5.7

1. $k = 2 * (i - 1) + j - 1$
2. $i = \frac{k+1}{3} + 1, j = k + 1 - 2 * \frac{k+1}{3}$

5.18

```
void moveK(vector<int> &a, int k) {
  int n  = a.size();
  reverse(a, 0, n - k - 1);
  reverse(a, n - k, n - 1);
  reverse(a, 0, n - 1);
}
void reverse(vector<int> &a, int l, int r) {
  while (l < r) {
    swap(a[l], a[r]);
    l++;
    r--;
  }
}
```

5.19

```
vector<pair<int, int>> getPoint(vector<vector<int>> &a, int m, int n) {    //m行n列
  vector<pair<int, int>> ans;
  vector<int> maxx(n, -INF);
  vector<int> minn(m, INF);
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      maxx[j] = max(maxx[j], a[i][j]);
      minn[i] = min(minn[i], a[i][j]);
    }
  }
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      if (a[i][j] == minn[i] && a[i][j] === maxx[j]) ans.push_back({i, j});
    }
  }
```

```
    return ans;
}
```

5.21

```
struct tri_tuple {
    int i, j, val;
};
vector<tri_tuple> a, b, c;
bool cmp (const tri_tuple &a, const tri_tuple &b) {
    if (a.i == b.i)  return a.j < b.j;
    else return a.i < b.i;
}
bool operator< (const tri_tuple &a, const tri_tuple &b) {
    if (a.i == b.i) return a.j < b.j;
    else return a.i < b.i;
}
bool operator= (const tri_tuple &a, const tri_tuple &b) {
    return a.i == b.i && a.j == b.j;
}
void solve() {
    sort(t.begin(), t.end(), cmp);
    sort(t.begin(), t.end(), cmp);
    int pa = 0, pb = 0;
    while (pa < a.size() && pb < b.size()) {
        if (a[pa] < b[pb]) {
            c.push_back(a[pa]);
            pa++;
        } else if(a[pa] == b[pb]) {
            c.push_back({a[pa].i , a[pa].j, a[pa].val + b[pb].val});
        } else {
            c.push_back(b[pb]);
            pb++;
        }
    }
    while (pa < a.size()) {
        c.push_back(a[pa++]);
    }
    while (pb < b.size()) {
        c.push_back(b[pb++]);
    }
}
```

5.30

```
typedef enum {ATOM,LIST} ElemTag;
typedef struct GLNode{
    ElemTag tag;
    union {
      char atom;
      struct {
        GLNode *hp, *tp;
      } ptr;
    }un;
} *GList;
int getDep(GList ls) {
```

```
    if (ls == NULL) return 1;
    else if(ls->tag == ATOM) return 0;
    return max(getDep(ls->un.ptr.hp), getDep(ls->ub.ptr.tp)) + 1;
}
```