块状数组

```
num = sqrt(n);
for (int i = 1; i <= num; i++)
  st[i] = n / num * (i - 1) + 1, ed[i] = n / num * i;
ed[num] = n;
for (int i = 1; i <= num; i++) {
  for (int j = st[i]; j <= ed[i]; j++) {
    belong[j] = i;
  }
  size[i] = ed[i] - st[i] + 1;
}
```

块状数组的区间修改和区间查询

```
void Sort(int k) {
  for (int i = st[k]; i <= ed[k]; i++) t[i] = a[i];
  sort(t + st[k], t + ed[k] + 1);
}

void Modify(int l, int r, int c) {
  int x = belong[l], y = belong[r];
  if (x == y)  // 区间在一个块内就直接修改
  {
    for (int i = l; i <= r; i++) a[i] += c;
    Sort(x);
    return;
  }
  for (int i = l; i <= ed[x]; i++) a[i] += c;      // 直接修改起始段
  for (int i = st[y]; i <= r; i++) a[i] += c;      // 直接修改结束段
  for (int i = x + 1; i < y; i++) delta[i] += c;  // 中间的块整体打上标记
  Sort(x);
  Sort(y);
}

int Answer(int l, int r, int c) {
  int ans = 0, x = belong[l], y = belong[r];
  if (x == y) {
    for (int i = l; i <= r; i++)
      if (a[i] + delta[x] >= c) ans++;
    return ans;
```

```
  }
  for (int i = l; i <= ed[x]; i++)
    if (a[i] + delta[x] >= c) ans++;
  for (int i = st[y]; i <= r; i++)
    if (a[i] + delta[y] >= c) ans++;
  for (int i = x + 1; i <= y - 1; i++)
    ans +=
        ed[i] - (lower_bound(t + st[i], t + ed[i] + 1, c - delta[i]) - t) + 1;
  // 用 lower_bound 找出中间每一个整块中第一个大于等于 c 的数的位置
  return ans;
}
```

并查集启发式合并

```
struct dsu {
  vector<size_t> pa, size;

  explicit dsu(size_t size_) : pa(size_), size(size_, 1) {
    iota(pa.begin(), pa.end(), 0);
  }

  void unite(size_t x, size_t y) {
    x = find(x), y = find(y);
    if (x == y) return;
    if (size[x] < size[y]) swap(x, y);
    pa[y] = x;
    size[x] += size[y];
  }
};
```