# 综合练习

## 人员

李锦澍、陈欣妙、刘奕辰、杨俊彦、袁晨峻 到课

## 上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/1909

比赛概况 | 题目列表 | 选择题列表 | 提交记录 | ★ 实时榜单 | ★ 选择题排行榜

**王向东老师周日八点半C++树状数组**

刷新

| # | 用户名 | 姓名 | 编程分 | 时间 | A | B | C | D | E |
|---|---|---|---|---|---|---|---|---|---|
| 1 | yuanchenjun | 袁晨峻 | 500 | 1536 | 100 | 100 | 100 | 100 | 100 |
| 2 | yangjunyan | 杨俊彦 | 500 | 13709 | 100 | 100 | 100 | 100 | 100 |
| 3 | liuyichen | 刘奕辰 | 400 | 2673 | 100 | 100 | 100 | 100 | |
| 4 | lijinshu | 李锦澍 | 400 | 5946 | 100 | 100 | 100 | 100 | |
| 5 | chenxinmiao | 陈欣妙 | 300 | 2258 | 100 | 100 | 100 | | |
| 6 | suitianyi | 隋天乙 | 100 | 1529 | 100 | | | | |

## 本周作业

https://cppoj.kids123code.com/contest/2019 (课上讲了 A ~ C 题, 课后作业是 D 题)

## 课堂表现

今天上课做题过程中, 暴露了同学们许多问题, 有的同学 二分写不对, 有的同学 dijkstra写不熟, 有的同学 离散化写不对 等问题

这些小问题, 都是因为同学们不熟练导致的, 这就需要同学们把之前这些算法的题再找出来多写几遍。

## 课堂内容

### 三元上升子序列 (上周作业)

维护 2 棵树状数组, 一棵记录 i 前面每个数的出现次数, 一棵记录 i 后面每个数的出现次数

这样, 就可以每次 O(logn) 求 i 前面小于 w[i] 的数量 和 i 后面大于 w[i] 的数量

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int N = 3e4 + 5, M = 1e5 + 5;
int w[N], tr1[M], tr2[M];
```

```
int lowbit(int x) { return x&(-x); }
void update(int x, int k, int tr[]) {
    while (x < M) { tr[x] += k, x += lowbit(x); }
}
int query(int x, int tr[]) {
    int res = 0;
    while (x) { res += tr[x], x -= lowbit(x); }
    return res;
}

int main()
{
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i];

    for (int i = 1; i <= n; ++i) update(w[i], 1, tr2);

    LL res = 0;
    for (int i = 1; i <= n; ++i) {
        update(w[i], -1, tr2);
        int lnums = query(w[i]-1, tr1), rnums = (n-i) - query(w[i],tr2);
        res += (LL)lnums * rnums;
        update(w[i], 1, tr1);
    }
    cout << res << endl;
    return 0;
}
```

**中位数**

先离散化, 然后 二分 + 树状数组 找中位数

```
#include <bits/stdc++.h>

using namespace std;

vector<int> ys;
int yFind(int x) { return lower_bound(ys.begin(), ys.end(), x) - ys.begin(); }

const int maxn = 1e5 + 5;
int w[maxn];

int tr[maxn];
int lowbit(int x) { return x&(-x); }
void update(int x, int k) {
  while (x < maxn) tr[x] += k, x += lowbit(x);
}
int query(int x) {
  int res = 0;
  while (x) res += tr[x], x -= lowbit(x);
  return res;
```

```
}

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i], ys.push_back(w[i]);
  sort(ys.begin(), ys.end()), ys.erase(unique(ys.begin(), ys.end()), ys.end());

  for (int i = 1; i <= n; ++i) {
    int u = yFind(w[i]) + 1;
    update(u, 1);
    if (i & 1) {
      int l = 1, r = 1e5 + 2;
      while (l <= r) {
        int mid = (l + r) / 2;
        if (query(mid) >= (i+1)/2) r = mid-1;
        else l = mid+1;
      }
      cout << ys[l-1] << endl;
    }
  }
  return 0;
}
```

**[CERC1998] 请柬**

需要求所有 1 -> i 的最短路和所有 i -> 1 的最短路

1 -> i 的最短路: 可以以 1 为起点用 dijkstra 跑一遍就能求出来

i -> 1 的最短路: 通过建反图, 然后以 1 为起点跑一遍 dijkstra 即可

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 2e6 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct eInfo {
  int to, value;
};
vector<eInfo> vec[maxn];

struct node {
  int id, d;
  bool operator < (const node& p) const { return d < p.d; }
  bool operator > (const node& p) const { return d > p.d; }
};
int dis[maxn];
bool st[maxn];
```

```
void dijkstra(int _st) {
  priority_queue<node, vector<node>, greater<node>> q;
  q.push({_st, 0}); dis[_st] = 0;
  while (!q.empty()) {
    node u = q.top(); q.pop();
    int id = u.id, d = u.d;
    if (st[id]) continue;
    st[id] = true;

    for (eInfo it : vec[id]) {
      if (dis[it.to] > d+it.value) {
        dis[it.to] = d+it.value; q.push({it.to, dis[it.to]});
      }
    }
  }
}

signed main()
{
  int n, m; cin >> n >> m;
  while (m -- ) {
    int a, b, c; cin >> a >> b >> c;
    vec[a].push_back({b,c}), vec[n+b].push_back({n+a,c});
  }

  memset(dis, 0x3f, sizeof(dis)), memset(st, false, sizeof(st));
  dijkstra(1), dijkstra(n+1);
  int res = 0;
  for (int i = 2; i <= n; ++i) res += dis[i] + dis[n+i];
  cout << res << endl;
  return 0;
}
```

### Hopscotch Addict

分层图, 把 1 个点拆成 3 个点, 然后从起点往终点跑 bfs 求最短路

```
#include <bits/stdc++.h>

using namespace std;

int get_id(int a, int b) { return a*3+b; }

const int maxn = 3e5 + 100;
vector<int> vec[maxn];
int dis[maxn];

int bfs(int a, int b) {
  int _st = get_id(a,0), _ed = get_id(b,0);
  memset(dis, -1, sizeof(dis));
```

```
  queue<int> q; q.push(_st); dis[_st] = 0;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int i : vec[u]) {
      if (dis[i] == -1) q.push(i), dis[i] = dis[u]+1;
    }
  }

  return (dis[_ed]==-1 ? dis[_ed] : dis[_ed]/3);
}

int main()
{
  int n, m; cin >> n >> m;
  while (m -- ) {
    int a, b; cin >> a >> b;
    for (int i = 0; i < 3; ++i) {
      int a_id = get_id(a,i), b_id = get_id(b,(i+1)%3);
      vec[a_id].push_back(b_id);
    }
  }

  int a, b; cin >> a >> b;
  cout << bfs(a, b) << endl;
  return 0;
}
```