Ilgam Rangers teamnote

aru0504, delena0702, riroan

November 12, 2022

Contents

ARE YOU TYPE CORRECTLY?
CHECK TIME COMPLEXITY OF YOUR ALGORITHM!
CHECK YOUR MAXIMUM ARRAY SIZE!

1 Graph

1.1 Dijkstra

```
import heapq
def dijkstra(start):
    distances = [0] * n
    for i in range(n):
        distances[i] = INF
    distances[start] = 0
    q = []
    heapq.heappush(q, [distances[start], start])
    while q:
        current_distance, current_destination = heapq.heappop(q)
        if distances[current_destination] < current_distance:
            continue
        for new_destination in g[current_destination]:
            new_distance = 1
            distance = current_distance + new_distance
            if distance < distances[new_destination]:
                distances[new_destination] = distance
                heapq.heappush(q, [distance, new_destination])
    return distances
```

1.2 Floyd Warshall

```
n,m = map(int, input().split()) # n : #vertex, m : #edge
arr = [[INF] * n for i in range(n)]
for i in range(n):
    arr[i][i] = 0
for i in range(m):
    a,b,c=map(int, input().split())
    arr[a-1][b-1] = c
    arr[b-1][a-1] = c

for k in range(n):
    for i in range(n):
        for j in range(n):
            arr[i][j] = min(arr[i][j],arr[i][k]+arr[k][j])
```

1.3 Topological Sort

```
from collections import deque
indegree = [0] * N
g = [[] for _ in range(N)]

# Build indegree, graph

q = deque()
for i in range(N):
    if not indegree[i]:
        q.append(i)

topological = []
for i in range(N):
    if not q:
        print(0) # Cycle detect
        exit()
    cur = q.popleft()
    topological.append(cur)
    for nxt in g[cur]:
        indegree[nxt] -= 1
        if not indegree[nxt]:
            q.append(nxt)
print(*topological, sep='\n')
```

1.4 Dinic

```
class SparseDinic:
    def __init__(self, size, source, sink):
        self._size = size

        self._level = [-1] * self._size
        self._idx = [0] * self._size
        self._capacity = defaultdict(int)
        self._flow = defaultdict(int)
        self._g = [[] for _ in range(self._size)]
        self._source = source
        self._sink = sink

    def _bfs(self):
        self._level = [-1] * self._size
        q = deque([self._source])
        self._level[self._source] = 0
        while q:
            cur = q.popleft()
            for nxt in self._g[cur]:
                if self._level[nxt] == -1 and self._capacity[(cur, nxt)] > self._flow[(cur,
                nxt)]:
                    self._level[nxt] = self._level[cur] + 1
                    q.append(nxt)
        return self._level[self._sink] != -1

    def _dfs(self, cur, sum_flow):
        if cur == self._sink:
            return sum_flow
        for i in range(self._idx[cur], len(self._g[cur])):
            nxt = self._g[cur][i]
            if self._level[nxt] == self._level[cur] + 1 and self._capacity[(cur, nxt)] >
            self._flow[(cur, nxt)]:
                d_flow = self._dfs(nxt, min(sum_flow, self._capacity[(cur, nxt)] -
                self._flow[(cur, nxt)]))
                if d_flow > 0:
                    self._flow[(cur, nxt)] += d_flow
                    self._flow[(nxt, cur)] -= d_flow
                    return d_flow
            self._idx[cur] += 1
        return 0

    def add_edge(self, u, v, cap, allow_inverse_capacity=False):
        self._g[u].append(v)
        self._g[v].append(u)
        self._capacity[(u, v)] += cap
        if allow_inverse_capacity:
            self._capacity[(v, u)] += cap
```

```
    def run(self):
        ret = 0
        while self._bfs():
            self._idx = [0] * self._size
            while 1:
                cur_flow = self._dfs(self._source, float('inf'))
                if not cur_flow:
                    break
                ret += cur_flow
        return ret
```

1.5 HLD + LCA

```
def dfs(x, p):
    global sz
    par[x] = p
    sz[x] = 1
    for i in g[x]:
        if i != p:
            sz[x] += dfs(i, x)
    return sz[x]

par = [0] * (n+5)
sz = [0] * (n+5)
depth = [0]*(n+5)
chain_number = [0]*(n+5)
chain_index = [0]*(n+5)
chain = [[] for i in range(n+5)]

def HLD(i, p, cur_chain, d):
    depth[i] = d
    chain_number[i] = cur_chain
    chain_index[i] = len(chain[cur_chain])
    chain[cur_chain].append(i)

    heavy = -1
    for x in g[i]:
        if x != p and (heavy == -1 or heavy != -1 and sz[x] > sz[heavy]):
            heavy = x
    if heavy != -1:
        HLD(heavy, i, cur_chain, d)
    for x in g[i]:
        if x != p and x != heavy:
            HLD(x, i, x, d+1)

dfs(1, 0)
HLD(1, 0, 1, 0)

def LCA(a, b):
    while chain_number[a] != chain_number[b]:
        if depth[a] > depth[b]:
            a = par[chain_number[a]]
        else:
            b = par[chain_number[b]]
    if chain_index[a] > chain_index[b]:
        return b
    else:
        return a
```

1.6 SCC(C++)

```
int id, d[MAX];
bool finished[MAX];
vector<int> a[MAX];
vector<vector<int>> SCC;
stack<int> s;

int dfs(int x) {
  d[x] = ++id;
  s.push(x);

  int parent = d[x];
  for (int i = 0; i < a[x].size(); i++) {
    int y = a[x][i];
    if (d[y] == 0) parent = min(parent, dfs(y));
    else if (!finished[y]) parent = min(parent, d[y]);
  }
  if (parent == d[x]) {
    vector<int> scc;
    while (1) {
      int t = s.top();
      s.pop();
      scc.push_back(t);
      finished[t] = true;
      if (t == x) break;
    }
    SCC.push_back(scc);
  }
  return parent;
}
```

1.7 2-SAT(C++)

```
int N, M, id, d[20002], ans[20002];
vector<int> v[20002];
vector<vector<int>> SCC;
bool finished[20002];
stack<int> s;
int f(int a) {
  // f 가 하는 일:
  // not(음수): -1~-4 -> 음수를 참으로 만들면 절댓값
  // (양수): 1~4 -> 양수를 거짓으로 만들면 N보다 크게
  // 결과: ~a: 1234 a: 5678
  return a > N ? a - N : a + N;
}
int dfs(int x) {
  d[x] = ++id;
  s.push(x);

  int parent = d[x];
  for (auto i : v[x]) {
    // 아직 dfs를 거쳐 확인하지 않았으면
    if (!d[i])
      parent = min(parent, dfs(i));
    // 아직 scc에 포함이 안됐으면
    else if (!finished[i])
      parent = min(parent, d[i]);
  }
```

```
  // dfs했더니 부모를 만나 scc가 이뤄진다면
  if (parent == d[x]) {
    vector<int> scc;
    while (1) {
      int t = s.top();
      s.pop();
      scc.push_back(t);
      // 까먹지말고 scc에 포함 완료해주자
      finished[t] = true;
      // scc Num은 1부터
      ans[t] = SCC.size() + 1;
      if (t == x)
        break;
    }
    SCC.push_back(scc);
  }
  return parent;
}
int main() {
  cin.tie(0);
  cout.tie(0);
  ios::sync_with_stdio(false);

  cin >> N >> M;
  while (M--) {
    int a, b;
    cin >> a >> b;
    // not 일 경우 N+1 ~ N+N
    if (a < 0)
      a = -a + N;
    if (b < 0)
      b = -b + N;
    // f가 있는 이유: not 때문에
    v[f(a)].push_back(b); // ~A -> B
    v[f(b)].push_back(a); // ~B -> A
  }
  // not 까지 2*N개
  // 그래프로 모델링해주자
  for (int i = 1; i <= 2 * N; i++) {
    if (!d[i])
      dfs(i);
  }
  // 같은 SCC에
  // ~A -> A 같은 모순이 존재한다면,
  for (int i = 1; i <= N; i++) {
    if (ans[i] == ans[i + N]) {
      cout << "0";
      return 0;
    }
  }
  cout << "1";
}
```

1.8 LCA(C++)

```
int N, Q, d[MAX], p[MAX][SIZE + 1], in[MAX], out[MAX], tmp;
vector<int> v[MAX];
void init(int cur) {
  in[cur] = ++tmp;
  for (int i : v[cur]) {
    if (d[i] == -1) {
      d[i] = d[cur] + 1;
      p[i][0] = cur;
      init(i);
    }
  }
  out[cur] = tmp;
}
int lca(int a, int b) {
  if (d[a] < d[b])
    swap(a, b);
  int diff = d[a] - d[b];
  int j = 0;
  while (diff) {
    if (diff % 2)
      a = p[a][j];
    diff /= 2;
    j++;
  }
  if (a == b)
    return a;
  for (int j = SIZE; j >= 0; j--) {
    if (p[a][j] != -1 && p[a][j] != p[b][j]) {
      a = p[a][j];
      b = p[b][j];
    }
  }
  a = p[a][0];
  return a;
}
```

1.9 Dinic(C++)

```
int N, M, S, E, lv[MAX], w[MAX], ans;
struct Edge {
  int to, c, rev;
  Edge(int to, int c, int rev)
    :to(to), c(c), rev(rev) {}
};
vector<Edge> v[MAX];
void addEdge(int s, int e, int c) {
  v[s].emplace_back(e, c, v[e].size());
  v[e].emplace_back(s, 0, v[s].size() - 1);
}
bool bfs() {
  memset(lv, -1, sizeof(lv));
  lv[S] = 0;
  queue<int> q;
  q.push(S);
  while (!q.empty()) {
    int cur = q.front();
    q.pop();
    for (auto i : v[cur]) {
      if (i.c && lv[i.to] == -1) {
        lv[i.to] = lv[cur] + 1;
        q.push(i.to);
```

```
      }
    }
  }
  return lv[E] != -1;
}
int dfs(int cur, int c) {
  if (cur == E)return c;
  for (; w[cur] < v[cur].size(); w[cur]++) {
    Edge& e = v[cur][w[cur]];
    if (!e.c || lv[e.to] != lv[cur] + 1)
      continue;
    int f = dfs(e.to, min(c, e.c));
    if (f > 0) {
      e.c -= f;
      v[e.to][e.rev].c += f;
      return f;
    }
  }
  return 0;
}
```

1.10 Bipartite Matching(C++)

```
int N, M, d[MAX];
bool used[MAX];
vector<int> v[MAX];
bool dfs(int x) {
  for (auto i : v[x]) {
    if (used[i])
      continue;
    used[i] = true;
    if (!d[i] || dfs(d[i])) {
      d[i] = x;
      return true;
    }
  }
  return false;
}
```

1.11 Dijkstra(C++)

```
priority_queue<pii, vector<pii>, greater<pii>> pq;
pq.push({ 0,s });
fill(d, d + MAX, INF);
d[s] = 0;
while (!pq.empty()) {
  int cost = pq.top().first;
  int cur = pq.top().second;
  pq.pop();
  if (d[cur] < cost)
  continue;
  for (auto i : v[cur]) {
    int next = i.second;
    int nCost = i.first + cost;
    if (nCost < d[next]) {
      d[next] = nCost;
      pq.push({ nCost, next });
    }
  }
}
cout << d[e] << "\n";
```

1.12 Dijkstra + DP(C++)

```
# BOJ 10217 KCM Travel
int N, M, K, d[MAX][MAXC];  // cost memoization
vector<pii> v[MAX];
int main() {
  cin.tie(0);
  cout.tie(0);
  ios::sync_with_stdio(false);

  int t;
  cin >> t;
  while (t--) {
    cin >> N >> M >> K;
    for (auto& i : v) {
      i.clear();
    }
    for (int i = 0; i < K; i++) {
      int s, e, cost, time;
      cin >> s >> e >> cost >> time;
      v[s].push_back({ {time, e}, cost });
    }
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({ {0, 1}, 0 });
    for (int i = 0; i < MAX; i++) {
      for (int j = 0; j < MAXC; j++) {
        d[i][j] = INF;
      }
    }
    d[1][0] = 0;
    while (!pq.empty()) {
      int time = pq.top().first.first;
      int cur = pq.top().first.second;
      int cost = pq.top().second;
      pq.pop();
      if (cost > M || d[cur][cost] < time)
      continue;
      for (auto i : v[cur]) {
        int nTime = i.first.first + time;
        int nCost = i.second + cost;
        int next = i.first.second;
        if (nCost <= M && nTime < d[next][nCost]) {
          // No -> 3120ms / Yes -> 260ms
          for (int j = nCost + 1; j <= M; j++) {
            if (d[next][j] <= nTime)
            break;
            d[next][j] = nTime;
          }
          d[next][nCost] = nTime;
          pq.push({ {nTime, next}, nCost });
        }
      }
    }
    int ans = INF;
    for (int i = 0; i <= M; i++) {
```

```
      ans = min(ans, d[N][i]);
    }
    if (ans >= INF)
    cout << "Poor KCM\n";
    else
    cout << ans << "\n";
  }
}
```

1.13 Floyd-Warshall(C++)

```
cin >> N;
for (int i = 1; i <= N; i++) {
  for (int j = 1; j <= N; j++) {
    cin >> a[i][j];
    if (!a[i][j])
    a[i][j] = INF;
  }
}
for (int k = 1; k <= N; k++) {
  for (int i = 1; i <= N; i++) {
    for (int j = 1; j <= N; j++) {
      a[i][j] = min(a[i][j], a[i][k] + a[k][j]);
    }
  }
}
```

1.14 Check Bipartite Graph(C++)

```
int N, M, p[MAX];
map<int, int> m;
int find(int a) {
  if (a == p[a])return a;
  return p[a] = find(p[a]);
}
bool merge(int a, int b) {
  a = find(a);
  b = find(b);
  if (a == b)return false;
  if (a > b)swap(a, b);
  p[b] = a;
  return true;
}
int main() {
  cin.tie(0)->sync_with_stdio(0);
  cin >> N >> M;
  for (int i = 1; i <= N * 2; i++)p[i] = i;
  while (M--) {
    char ch;
    int n1, n2;
    cin >> ch >> n1 >> n2;
    if (ch == 'S') {
      merge(n1, n2);
      merge(n1 + N, n2 + N);
    }
    else {
      merge(n1, n2 + N);
      merge(n2, n1 + N);
    }
  }
  for (int i = 1; i <= N; i++) {
    if (find(i) == find(i + N)) {
      cout << 0;
      return 0;
    }
  }
  for (int i = 1; i <= N; i++) {
    merge(i, i + N);
  }
  for (int i = 1; i <= N; i++) {
    m[find(i)]++;
  }
  cout << 1;
  for (int i = 0; i < m.size(); i++) {
    cout << 0;
  }
}
```

1.15 Bellman-Ford(C++)

```
vector<pair<int, ll>> v[501];
ll d[501];
int main(){
  cin.tie(0);
  cout.tie(0);
  ios::sync_with_stdio(false);

  int n, m;
  cin>>n>>m;
  for(int i=0;i<m;i++){
    int a,b;
    ll c;
    cin>>a>>b>>c;
    v[a].push_back({b,c});
  }
  for(int i=2;i<=n;i++){
    d[i]=INF;
  }
  bool mCycle=false;
  for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++){
      for(pair<int, ll> p: v[j]){
        int next=p.first;
        ll dis=d[j]+p.second;
        if(d[j]!=INF&&d[next]>dis){
          d[next]=dis;
          if(i==n)
          mCycle=true;
        }
      }
    }
  }
  if(mCycle)
  cout<<"-1\n";
  else{
    for(int i=2;i<=n;i++){
      if(d[i]==INF)
```

```
      cout<<"-1\n";
      else
      cout<<d[i]<<"\n";
    }
  }
}
```

1.16 HLD(C++)

```
int N, M;
vector<pi> edge, v[MAX];
vector<int> c[MAX];
int tmpw[MAX], tree[1 << 18];
int init(int node, int s, int e) {
  if (s == e)
  return tree[node] = tmpw[s];
  int mid = (s + e) / 2;
  return tree[node] = max(init(node * 2, s, mid),
  init(node * 2 + 1, mid + 1, e));
}
int query(int node, int s, int e, int left, int right) {
  if (e < left || right < s)
  return -INF;
  if (left <= s && e <= right)
  return tree[node];
  int mid = (s + e) / 2;
  return max(query(node * 2, s, mid, left, right),
  query(node * 2 + 1, mid + 1, e, left, right));
}
void update(int node, int s, int e, int idx, int val) {
  if (e < idx || idx < s)
  return;
  if (e == idx && idx == s) {
    tree[node] = val;
    return;
  }
  int mid = (s + e) / 2;
  update(node * 2, s, mid, idx, val);
  update(node * 2 + 1, mid + 1, e, idx, val);
  tree[node] = max(tree[node * 2], tree[node * 2 + 1]);
}
int sz[MAX], d[MAX], p[MAX], t[MAX], in[MAX], out[MAX], tmp;
int w[MAX];
bool visit[MAX];
void dfs(int cur) {
  visit[cur] = true;
  for (auto i : v[cur]) {
    if (!visit[i.second]) {
      c[cur].push_back(i.second);
      w[i.second] = i.first;
      dfs(i.second);
    }
  }
}
void dfs1(int cur) {
  sz[cur] = 1;
  for (auto& i : c[cur]) {
    d[i] = d[cur] + 1;
    p[i] = cur;
    dfs1(i);
    sz[cur] += sz[i];
    if (sz[i] > sz[c[cur][0]])
    swap(i, c[cur][0]);
  }
}
void dfs2(int cur) {
  in[cur] = ++tmp;
  for (auto i : c[cur]) {
    if (i == c[cur][0])
    t[i] = t[cur];
    else
    t[i] = i;
    dfs2(i);
  }
  out[cur] = tmp;
}
int hldQuery(int n1, int n2) {
  int ret = -INF;
  while (t[n1] != t[n2]) {
    if (d[t[n1]] > d[t[n2]])
    swap(n1, n2);
    int top = t[n2];
    ret = max(ret, query(1, 1, N, in[top], in[n2]));
    n2 = p[top];
  }
  if (d[n1] > d[n2])
  swap(n1, n2);
  for (auto i : c[n1]) {
    if (t[i] == t[n1]) {
      n1 = i;
      break;
    }
  }
  ret = max(ret, query(1, 1, N, in[n1], in[n2]));
  return ret;
}
void hldUpdate(int idx, int val) {
  int idx1 = edge[idx].first;
  int idx2 = edge[idx].second;
  if (d[idx1] < d[idx2])
  update(1, 1, N, in[idx2], val);
  else
  update(1, 1, N, in[idx1], val);
}
int main() {
  cin.tie(0)->sync_with_stdio(0);
  cin >> N;
  for (int i = 0; i < N - 1; i++) {
    int n1, n2, cost;
    cin >> n1 >> n2 >> cost;
    v[n1].push_back({ cost, n2 });
    v[n2].push_back({ cost, n1 });
    edge.push_back({ n1, n2 });
  }
```

```
  dfs(1);
  dfs1(1);
  dfs2(1);
  for (int i = 1; i <= N; i++) {
    tmpw[in[i]] = w[i];
  }
  init(1, 1, N);
  cin >> M;
  while (M--) {
    int ch, a, b;
    cin >> ch >> a >> b;
    if (ch == 1)
    hldUpdate(a - 1, b);
    else
    cout << hldQuery(a, b) << "\n";
  }
}
```

2 Data Structure

2.1 Disjoint Set

```
p = [-1]*n
def merge(x, y):
    x, y = find(x), find(y)
    if x == y: return
    # p contains size(negative), size-based merge
    if p[x] < p[y]:
        p[x] += p[y]
        p[y] = x
    else:
        p[y] += p[x]
        p[x] = y
    # You can just simply use: p[y] = p[x]
    # if you don't need the size

def find(x):
    if p[x] < 0:
        return x
    temp = x
    while p[temp] >= 0:
        temp = p[temp]
    p[x] = temp
    return p[x]

def size(x):
    return -p[find(x)]
```

2.2 MergeSort Tree

```
# L = 1 << N.bit_length()
# nums = list(mis()); init(nums)
def init(nums):
    arr = [[] for i in range(L*2)]
    for i in range(len(nums)):
        arr[i+L] += [nums[i]]
    for i in range(L-1, 0, -1):
        arr[i] = sorted(arr[i*2] + arr[i*2+1])
    return arr

def count_less_than(arr, l, r, k):
    from bisect import bisect_left
    ret = 0; l += L-1; r += L-1
    while l <= r:
        if l%2:
            ret += bisect_left(arr[l], k)
        if not r%2:
            ret += bisect_left(arr[r], k)
        l, r = (l+1)//2, (r-1)//2
    return ret

def get_geqthan(arr, l, r, k):
    from bisect import bisect_left
    l += L-1; r += L-1
    ret = float('inf')
    while l <= r:
        if l%2:
            t = bisect_left(arr[l], k)
            if t < len(arr[l]) and arr[l][t] >= k:
                ret = min(ret, arr[l][t])
        if not r%2:
            t = bisect_left(arr[r], k)
            if t < len(arr[r]) and arr[r][t] >= k:
                ret = min(ret, arr[r][t])
        l, r = (l+1)//2, (r-1)//2
    return ret

def query(arr, l, r, k):
    p = -1_000_000_005
    q = -p
    while p <= q:
        mid = (p+q)//2
        ret = count_less_than(arr, l, r, mid)
        if ret == k-1:
            # have to get (x) >= mid in array[l..r]
            return get_geqthan(arr, l, r, mid)
        elif ret > k-1:
            q = mid-1
        else:
            p = mid+1
```

2.3 Trie

```
class Trie:
    def __init__(self):
        self.root = {}

    def insert(self, s):
        cur_node = self.root
        for c in s:
            if c not in cur_node:
                cur_node[c] = {}
            cur_node = cur_node[c]
        cur_node["*"] = s

    def search(self, s):
        cur_node = self.root
        for c in s:
```

```
            if c in s:
                cur_node = cur_node[c]
            else:
                return False
        return "*" in cur_node
```

2.4 XOR Trie

```
ans=0
class Trie:
    def __init__(self):
        self.children = [None, None]
        self.cnt = 0
        self.end = False

    def insert(self, x, ix=0):
        self.cnt += 1
        if ix == m:
            self.end = True
            return
        if self.children[x[ix]] == None:
            self.children[x[ix]] = Trie()
        self.children[x[ix]].insert(x, ix + 1)

  # change below
    def query(self, x, ix=0): # #(less than x)
        global ans
        if self.end:
            return
        if k[ix] == 1:
            if self.children[x[ix]] != None:
                ans += self.children[x[ix]].cnt
            if self.children[1 - x[ix]] != None:
                self.children[1 - x[ix]].query(x, ix + 1)
        else:
            if self.children[x[ix]] != None:
                self.children[x[ix]].query(x, ix + 1)
```

2.5 Segment Tree

```
class SegmentTree:
    def __init__(self, arr, merge):
        self._size = len(arr)
        self._tree = arr * 2
        self._merge = merge
        for i in range(self._size-1, 0, -1):
            self._tree[i] = merge(self._tree[i*2], self._tree[i*2+1])

    def update(self, pos, val):
        i = pos + self._size
        while i:
            self._tree[i] = val
            val = self._merge(self._tree[i-1], val) if i%2 else self._merge(val,
            self._tree[i+1])
            i //= 2

    def query(self, l, r):
        if l == r:
            return self._tree[self._size + l]
        ret_l = self._tree[self._size + l]
        ret_r = self._tree[self._size + r]
        l = self._size + l + 1
        r = self._size + r - 1
        while l <= r:
            if l%2:
                ret_l = self._merge(ret_l, self._tree[l])
                l += 1
            if not r%2:
                ret_r = self._merge(ret_r, self._tree[r])
                r -= 1
            l //= 2; r //= 2
        return self._merge(ret_l, ret_r)
```

2.6 Lazy Segment Tree

```
class Lazy:
    def __init__(self, arr, operate, update_value, update_delay):
        self._size = len(arr)
        self._height = self._size.bit_length()

        if bin(self._size).count('1') == 1:
            self._height -= 1

        self._L = 2 ** self._height
        self._tree = [0] * (self._L * 2)
        self._delayed_operation = [None] * self._L
        self._operate = operate
        self._update_value = update_value
        self._update_delay = update_delay
        self._tree[self._L:self._L+self._size] = arr
        for i in range(self._L-1, 0, -1):
            self._tree[i] = self._operate(self._tree[i*2], self._tree[i*2+1])
        #print(self._tree)

    # idx implies an index in array,
    # pos implies an index in segment tree

    def apply(self, pos, value, interval):
        self._tree[pos] = self._update_value(self._tree[pos], value, interval)
        if pos < self._L:
            pos_delay = self._delayed_operation[pos]
            self._delayed_operation[pos] = value if pos_delay is None else
            self._update_delay(pos_delay, value)

    def build_up(self, pos):
        # build tree from pos to Root
        # s is initialized as 1 since pos is given as leaf
        s = 1
        while pos > 1:
            pos >>= 1
            s <<= 1
            temp = self._operate(self._tree[pos*2], self._tree[pos*2+1])
            self._tree[pos] = temp if self._delayed_operation[pos] is None else
            self._update_value(temp, self._delayed_operation[pos], s)

    def push_down(self, pos):
        # Propagate delayed operations for tree[pos], from root
```

```
        s = 1 << (self._height-1)
        for i in range(self._height, 0, -1):
            parent = pos >> i
            delayed = self._delayed_operation[parent]
            if delayed is not None:
                self.apply(parent*2, delayed, s)
                self.apply(parent*2+1, delayed, s)
                self._delayed_operation[parent] = None
            s >>= 1

    def update(self, l, r, value):
        # l, r as idx
        l += self._L; r += self._L
        l0, r0 = l, r
        # size of an interval is initialized as 1
        # since we get l, r as idx (index of an array)
        s = 1
        while l <= r:
            if l&1:
                self.apply(l, value, s)
                l += 1
            if not r&1:
                self.apply(r, value, s)
                r -= 1
            l >>= 1
            r >>= 1
            s <<= 1
        self.build_up(l0); self.build_up(r0)

    def query(self, l, r):
        # l, r as idx
        ret = 0
        l += self._L; r += self._L
        self.push_down(l); self.push_down(r)

        if l == r:
            return self._tree[l]

        ret_l = self._tree[l]
        ret_r = self._tree[r]
        l += 1; r -= 1
        while l <= r:
            if l&1:
                ret_l = self._operate(ret_l, self._tree[l])
                l += 1
            if not r&1:
                ret_r = self._operate(ret_r, self._tree[r])
                r -= 1
            l >>= 1; r >>= 1
        return self._operate(ret_l, ret_r)

    def get_value(self, idx):
            self.push_down(idx + self._L)
            return self._tree[idx + self._L]
```

2.7 Policy Based Data Structure(C++)

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<
int,
null_type,
less<int>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;

int main(){
  ordered_set X;

  X.insert(16);
  X.insert(1);
  X.insert(4);
  X.insert(2);

  cout<<*X.find_by_order(0)<<endl; // 1
  cout<<*X.find_by_order(1)<<endl; // 2
  cout<<*X.find_by_order(2)<<endl; // 4
  cout<<*X.find_by_order(3)<<endl; // 16
  cout<<*X.find_by_order(-1)<<endl; // 0 : invalid index
  cout<<*X.find_by_order(5)<<endl;

  cout<<X.order_of_key(1)<<endl; // #(less than 1) : 0
  cout<<X.order_of_key(4)<<endl; // #(less than 4) : 2
  cout<<X.order_of_key(400)<<endl; // #(less than 400) : 4
}
```

2.8 Segment Tree(C++)

```
ll query(int node, int s, int e, int left, int right) {
  if (e < left || right < s)
  return 0;
  if (left <= s && e <= right)
  return tree[node];
  int mid = (s + e) / 2;
  return (query(node * 2, s, mid, left, right)
  + query(node * 2 + 1, mid + 1, e, left, right)) % MOD;
}
void update(int node, int s, int e, int idx, ll val) {
  if (idx < s || e < idx)
  return;
  tree[node] += val;
  if (s != e) {
    int mid = (s + e) / 2;
    update(node * 2, s, mid, idx, val);
    update(node * 2 + 1, mid + 1, e, idx, val);
  }
}
```

2.9 Lazy Segment Tree(C++)

```
struct Tree {
  ll val, lazy;
};
```

```
int N, M, K;
ll a[MAX];
Tree tree[1 << 21];
ll init(int node, int s, int e) {
  if (s == e)
  return tree[node].val = a[s];
  int mid = (s + e) / 2;
  return tree[node].val = init(node * 2, s, mid)
  + init(node * 2 + 1, mid + 1, e);
}
void update_lazy(int node, int s, int e) {
  if (tree[node].lazy) {
    tree[node].val += ((ll)e - s + 1) * tree[node].lazy;
    if (s != e) {
      tree[node * 2].lazy += tree[node].lazy;
      tree[node * 2 + 1].lazy += tree[node].lazy;
    }
    tree[node].lazy = 0;
  }
}
ll query(int node, int s, int e, int left, int right) {
  update_lazy(node, s, e);
  if (e < left || right < s)
  return 0;
  if (left <= s && e <= right)
  return tree[node].val;
  int mid = (s + e) / 2;
  return query(node * 2, s, mid, left, right)
  + query(node * 2 + 1, mid + 1, e, left, right);
}
void update(int node, int s, int e, int left, int right, ll diff) {
  update_lazy(node, s, e);
  if (e < left || right < s)
  return;
  if (left <= s && e <= right) {
    tree[node].lazy += diff;
    update_lazy(node, s, e);
    return;
  }
  int mid = (s + e) / 2;
  update(node * 2, s, mid, left, right, diff);
  update(node * 2 + 1, mid + 1, e, left, right, diff);
  tree[node].val = tree[node * 2].val + tree[node * 2 + 1].val;
}
```

2.10 Fenwick Tree + Inversion Counting(C++)

```
int N,a[MAX],tree[MAX];
ll query(int i){
  ll ret=0;
  for(;i;i-=i&-i){
    ret+=1LL*tree[i];
  }
  return ret;
}
void update(int i, int val){
  for(;i<=N;i+=i&-i){
    tree[i]+=val;
  }
}
int main() {
  cin.tie(0)->sync_with_stdio(0);
  cin>>N;
  ll ans=0;
  for(int i=1;i<=N;i++){
    cin>>a[i];
    ans+=query(N)-query(a[i]);
    update(a[i],1);
  }
  cout<<ans;
}
```

3 Math

3.1 Linear Sieve

```
n = 1000010   # max number
sieve = [0]*n # sieve
primes = []   # prime array
for i in range(2, n):
    if sieve[i] == 0:
        primes.append(i)
    for j in primes:
        if i*j>=n: break
        sieve[i*j] = 1
        if i%j==0: break
```

3.2 FFT without FFT

```
import decimal
# FFT without FFT
def fft(a, b):
    decimal.setcontext(
        decimal.Context(prec=decimal.MAX_PREC, Emax=decimal.MAX_EMAX))
    digit = 20
    fmat = f'0{digit}d'
    a_decimal = decimal.Decimal(''.join(format(x, fmat) for x in a))
    b_decimal = decimal.Decimal(''.join(format(x, fmat) for x in b))
    ret_decimal = a_decimal * b_decimal
    l = digit * (len(a)+len(b)-1)
    ret = f'{ret_decimal:0{l}f}'
    return [int(ret[i:i+digit]) for i in range(0, l, digit)]
```

3.3 FFT

```
import math
pi = math.pi

def FFT(a, inv):
    n = len(a)
    j = 0
    roots = [0] * (n // 2)
    for i in range(1, n):
        bit = n >> 1
        while j >= bit:
            j -= bit
            bit >>= 1
        j += bit
        if i < j:
            a[i], a[j] = a[j], a[i]
```

```
    ang = 2 * pi / n * (-1 if inv else 1)
    for i in range(n // 2):
        roots[i] = complex(math.cos(ang * i), math.sin(ang * i))
    i = 2
    while i <= n:
        step = n // i
        for j in range(0, n, i):
            for k in range(i // 2):
                u = a[j + k]
                v = a[j + k + i // 2] * roots[step * k]
                a[j + k] = u + v
                a[j + k + i // 2] = u - v
        i <<= 1
    if inv:
        for i in range(n):
            a[i] /= n

def multiply(arr, brr):
    n = 2
    while n<len(arr) + len(brr):
        n<<=1
    arr = arr+[0]*(n-len(arr))
    brr = brr+[0]*(n-len(brr))
    FFT(arr, 0)
    FFT(brr, 0)
    for i in range(n):
        arr[i] *= brr[i]
    FFT(arr, 1)
    ret = [0]*n
    for i in range(n):
        ret[i] = round(arr[i].real)
    return ret
```

3.4 Berlekamp Massey + Kitamasa

```
mod = 1000000009 # prime

def pow(x, p):
    ret = 1
    piv = x
    while p:
        if p & 1:
            ret = (ret * piv) % mod
        piv = (piv * piv) % mod
        p >>= 1
    return ret

def berlekamp_massy(x):
    ls, cur = list(), list()
    lf, ld = 0, 0
    for i in range(len(x)):
        t = 0
        for j in range(len(cur)):
            t = (t + 1 * x[i - j - 1] * cur[j]) % mod
        if (t - x[i]) % mod == 0:
            continue
        if len(cur) == 0:
            cur = [0] * (i + 1)
            lf = i
            ld = (t - x[i]) % mod
            continue
        k = -(x[i] - t) * pow(ld, mod - 2) % mod
        c = [0] * (i - lf - 1)
        c.append(k)
        for j in ls:
            c.append(-j * k % mod)
        if len(c) < len(cur):
            c = c + [0] * (len(cur) - len(c))
        for j in range(len(cur)):
            c[j] = (c[j] + cur[j]) % mod
        if i - lf + len(ls) >= len(cur):
            ls, lf, ld = cur, i, (t - x[i]) % mod
        cur = c
    for i in range(len(cur)):
        cur[i] = (cur[i] % mod + mod) % mod
    return cur

def get_nth(rec, dp, n):
    m = len(rec)
    s, t = [0] * m, [0] * m
    s[0] = 1
    if m != 1:
        t[1] = 1
    else:
        t[0] = rec[0]

    def mul(v, w, rec):
        m = len(v)
        t = [0] * (2 * m)
        for j in range(m):
            for k in range(m):
                t[j + k] += v[j] * w[k] % mod
                if t[j + k] >= mod:
                    t[j + k] -= mod
        for j in range(2 * m - 1, m - 1, -1):
            for k in range(1, m + 1):
                t[j - k] += t[j] * rec[k - 1] % mod
                if t[j - k] >= mod:
                    t[j - k] -= mod
        t = t[:m]
        return t

    while n:
        if n & 1:
            s = mul(s, t, rec)
        t = mul(t, t, rec)
        n >>= 1
    ret = 0
    for i in range(m):
        ret += s[i] * dp[i] % mod
    return ret % mod

def guess_nth_term(x, n):
    if n < len(x):
```

```
        return x[n]
    v = berlekamp_massy(x)
    if len(v) == 0:
        return 0
    return get_nth(v, x, n)
```

3.5 Combination

```
def inverseEuler(n, mod):
    return pow(n, mod-2, mod)

def C(n, r, mod):
    f = [1] * (n+1)
    for i in range(2, n+1):
        f[i] = (f[i-1]*i) % mod
    return (f[n]*((inverseEuler(f[r], mod)*inverseEuler(f[n-r], mod)) % mod)) % mod
```

3.6 Lucas Theorem

```
# (nCr)%mod (mod is prime)
arr,brr = [],[]
while n:
    arr.append(n%mod)
    n//=mod

while r:
    brr.append(r%mod)
    r//=mod

if len(arr) < len(brr):
    arr, brr = brr, arr

brr+=[0]*(len(arr) - len(brr))

def fact(n): # or preprocess
    r = 1
    for i in range(1, n + 1):
        r*=i
    return r

def C(n,r):
    if n<r:
        return 0
    return fact(n) // (fact(r) * fact(n-r))

l = len(arr)
ans = 1
for i in range(l):
    ans *= C(arr[i], brr[i]) % mod
```

3.7 Extended Euclidean Algorithm

```
def EED(a, b):
    if a < b:
        a, b = b, a
    if b == 0:
        return a, 1, 0
    g, x1, y1 = EED(b, a % b)
    return g, y1, x1 - a // b * y1
```

3.8 Euler Phi Function

```
N = 1000010
s = [1] * N # Eratosthenes Sieve

# ... linear sieve

def phi(arr): # arr : factorization order of n
  r = 1
  for i in range(len(arr)):
    if arr[i]:
      r *= p[i] ** arr[i] - p[i] ** (arr[i] - 1) # p^k - p^(k-1)
  return r
```

3.9 Partition Number

```
mod = 998244353
p = [1]
g = []
k = 1
kc = 0
for n in range(1, T+2): # O(n sqrt(n))
    p.append(0)
    q = p[-1]
    if kc:
        if k * (3 * k + 1) == 2 * n:
            g.append(k * (3 * k + 1) // 2)
            kc = 0
            k += 1
    else:
        if k * (3 * k - 1) == 2 * n:
            g.append(k * (3 * k - 1) // 2)
            kc = 1
    for i in range(len(g)):
        if i & 3 < 2:
            q = (q + p[n - g[i]]) % mod
        else:
            q = (q + mod - p[n - g[i]]) % mod
    p[-1] = q
```

3.10 Discrete Logarithm

```
# B^L == N (mod P)
# find L
T = int(P ** 0.5 + 1)

arr = []
brr = []
for X in range(P // T + 1):
    u = pow(B, X * T, P)
    arr.append((u, X))
    brr.append(u)

arr = sorted(arr, key=lambda x: x[0])
brr.sort()
f = False
tmp = []
for Y in range(T + 10):
    u = ((N % P) * (pow(B, Y * (P - 2), P))) % P
    t = bisect_left(brr, u)
```

```
    try:
        if arr[t][0] == u:
            tmp.append(arr[t][1] * T + Y)
    except:
        pass
if not tmp:
    print("no solution")
else:
    tmp.sort()
    print(tmp[0])
```

3.11 Pollard Rho + Miller Rabin Test(C++)

```
ll mul(ll x, ll y, ll mod) {
  return (__int128)x * y % mod;
}

ll ipow(ll x, ll y, ll p) {
  ll ret = 1, piv = x % p;
  while (y) {
    if (y & 1) ret = mul(ret, piv, p);
    piv = mul(piv, piv, p);
    y >>= 1;
  }
  return ret;
}

bool miller_rabin(ll x, ll a) {
  if (x % a == 0) return 0;
  ll d = x - 1;
  while (1) {
    ll tmp = ipow(a, d, x);
    if (d & 1) return(tmp != 1 && tmp != x - 1);
    else if (tmp == x - 1) return 0;
    d >>= 1;
  }
}

bool isprime(ll x) {
  for (auto& i : { 2,3,5,7,11,13,17,19,23,29,31,37 }) {
    if (x == i) return 1;
    if (x > 40 && miller_rabin(x, i)) return 0;
  }
  if (x <= 40) return 0;
  return 1;
}

ll f(ll x, ll n, ll c) {
  return(c + mul(x, x, n)) % n;
}

ll myAbs(ll a) {
  return a > 0 ? a : (-a);
}

ll gcd(ll a, ll b) {
  if (b == 0)
    return a;
  return gcd(b, a % b);
}

void rec(ll n, vector<ll>& v) {
  if (n == 1) return;
  if (n % 2 == 0) {
    v.push_back(2);
    rec(n / 2, v);
    return;
  }
  if (isprime(n)) {
    v.push_back(n);
    return;
  }
  ll a, b, c;
  while (1) {
    a = rand() % (n - 2) + 2;
    b = a;
    c = rand() % 20 + 1;
    do {
      a = f(a, n, c);
      b = f(f(b, n, c), n, c);
    } while (gcd(myAbs(a - b), n) == 1);
    if (a != b)
      break;
  }
  ll x = gcd(myAbs(a - b), n);
  rec(x, v);
  rec(n / x, v);
}

auto factorize(ll n) {
  vector<ll> ret;
  rec(n, ret);
  sort(ret.begin(), ret.end());
  return ret;
}
```

3.12 Catalan Number(C++)

```
ll N, d[MAX] = { 1,1,2,5 };
int main() {
  cin.tie(0)->sync_with_stdio(0);

  int t;
  cin>>t;
  for (int i = 4; i < MAX; i++) {
    for (int j = 0; j < i; j++) {
      d[i] += d[j] * d[i - j - 1];
      d[i] %= MOD;
    }
  }
  while(t--){
    cin >> N;
    if(N%2){
      cout<<0<<"\n";
      continue;
```

```
    }
    N/=2;
    cout << d[N] << "\n";
  }
}
```

4 Geometry

4.1 CCW

```
def ccw(a, b, c):
    return a[0]*b[1] + b[0]*c[1] + c[0]*a[1] - \
           (b[0]*a[1] + c[0]*b[1] + a[0]*c[1])
```

4.2 Line Cross

```
def cross(a, b, c, d):
    return ccw(a, b, c) * ccw(a, b, d) < 0 and ccw(c, d, a) * ccw(c, d, b) < 0
```

4.3 Convex Hull

```
def ConvexHull(points):
    upper = []
    lower = []
    for p in sorted(points):
        while len(upper) > 1 and ccw(upper[-2], upper[-1], p) >= 0:
            upper.pop()
        while len(lower) > 1 and ccw(lower[-2], lower[-1], p) <= 0:
            lower.pop()
        upper.append(p)
        lower.append(p)
    return upper, lower
```

4.4 Rotating Calipers

```
def sub(a,b):
    return[a[0]-b[0], a[1]-b[1]]

def norm(p):
    return (p[0]**2+p[1]**2)**0.5

def dot(p1, p2):
    return p1[0] * p2[0] + p1[1] * p2[1]

def diameter(p):
    n = len(p)
    left, right = 0, 0
    for i in range(1, n):
        if p[i] < p[left]:
            left = i
            p[left] = p[i]
        if p[i] > p[right]:
            right = i
            p[right] = p[i]
    calipersA = [0,1]
    ret = norm(sub(p[right], p[left]))
    toNext = [None] * n
    for i in range(n):
        toNext[i] = sub(p[(i+1)%n],p[i])
        tmp = norm(toNext[i])+eps
        toNext[i] = [toNext[i][0]/tmp, toNext[i][1]/tmp]
    a = left
    b = right
    while a != right or b != left:
        cosThetaA = dot(calipersA, toNext[a])
        cosThetaB = -dot(calipersA, toNext[b])
        if cosThetaA > cosThetaB:
            calipersA = toNext[a]
            a = (a + 1) % n
        else:
            calipersA = [-toNext[b][0], -toNext[b][1]]
            b = (b + 1) % n
        ret = max(ret, norm(sub(p[b], p[a])))
    return ret
```

4.5 Rotating Calipers(C++)

```
struct Point {
  ll x, y, p, q;
  Point() {}
  Point(ll x1, ll y1, ll p1 = 1, ll q1 = 0)
  :x(x1), y(y1), p(p1), q(p1) {}
  bool operator<(const Point& O) {
    if (q * O.p != p * O.q)
    return q * O.p < p* O.q;
    if (y != O.y)
    return y < O.y;
    return x < O.x;
  }
};
Point b[MAX], dt[MAX];
ll ccw(const Point& p1, const Point& p2, const Point& p3) {
  return (p1.x * p2.y + p2.x * p3.y + p3.x * p1.y)
  - (p1.y * p2.x + p2.y * p3.x + p3.y * p1.x);
}
ll dist(const Point& p1, const Point& p2) {
  return (p2.x - p1.x) * (p2.x - p1.x)
  + (p2.y - p1.y) * (p2.y - p1.y);
}
Point operator-(const Point& p1, const Point& p2) {
  return Point(p1.x - p2.x, p1.y - p2.y);
}
ll solve(ll t) {
  Point a[MAX];
  for (int i = 0; i < N; i++) {
    a[i] = Point(b[i].x + dt[i].x * t, b[i].y + dt[i].y * t);
  }
  swap(a[0], *min_element(a, a + N));
  for (int i = 1; i < N; i++) {
    a[i].p = a[i].x - a[0].x;
    a[i].q = a[i].y - a[0].y;
  }
  sort(a + 1, a + N);
  stack<int> S;
  S.push(0);
  S.push(1);
  for (int i = 2; i < N; i++) {
    while (S.size() >= 2) {
      int second = S.top();
      S.pop();
      int first = S.top();
      if (ccw(a[first], a[second], a[i]) > 0) {
```

```
        S.push(second);
        break;
      }
    }
    S.push(i);
  }
  vector<Point> hull(S.size());
  int left = 0, right = 0;
  for (int i = 0; i < hull.size(); i++) {
    hull[i] = a[S.top()];
    if (hull[left].x > hull[i].x)
    left = i;
    if (hull[right].x < hull[i].x)
    right = i;
    S.pop();
  }
  ll ans = dist(a[left], a[right]);
  pair<int, int> ret1, ret2;
  ret1 = { hull[left].x, hull[left].y };
  ret2 = { hull[right].x, hull[right].y };
  for (int i = 0; i < hull.size(); i++) {
    Point p1 = hull[(left + 1) % hull.size()] - hull[left];
    Point p2 = hull[right] - hull[(right + 1) % hull.size()];
    if (ccw({ 0,0 }, p1, p2) <= 0) {
      left = (left + 1) % hull.size();
    }
    else {
      right = (right + 1) % hull.size();
    }
    if (ans < dist(hull[left], hull[right])) {
      ans = dist(hull[left], hull[right]);
      ret1 = { hull[left].x, hull[left].y };
      ret2 = { hull[right].x, hull[right].y };
    }
  }
  return ans;
}
```

5 String

5.1 KMP

```
def make_fail(s):
    pi = [0] * len(s)
    j = 0
    for i in range(1, len(s)):
        while s[i] != s[j] and j > 0:
            j = pi[j-1]
        if s[i] == s[j]:
            j += 1
            pi[i] = j
    return pi

def KMP(string, pattern):
    pi = make_fail(pattern)
    indices = []
    j = 0
    for i in range(len(string)):
        while string[i] != pattern[j] and j > 0:
            j = pi[j-1]
        if string[i] == pattern[j]:
            if j == len(pattern) - 1: # found
                indices.append(i - len(pattern) + 2)
                j = pi[j]
            else:
                j += 1
    return indices
```

5.2 Manacher

```
# s = list(input())
s = '#'.join(s)
s = '#' + s + '#'
def manacher(s):
    n = len(s)
    A = [0] * n
    r = 0
    p = 0
    for i in range(n):
        if i <= r:
            A[i] = min(A[2 * p - i], r - i)
        else:
            A[i] = 0
        while i - A[i] - 1 >= 0 and i + A[i] + 1 < n and s[i - A[i] - 1] == s[i + A[i] + 1]:
            A[i] += 1
        if r < i + A[i]:
            r = i + A[i]
            p = i
    return A
```

5.3 Aho Corasick(C++)

```
struct Trie {
  Trie* next[26];
  Trie* fail;
  bool output;
  Trie() : output(false) {
    fill(next, next + 26, nullptr);
  }
  ~Trie() {
    for (int i = 0; i < 26; i++) {
      if (next[i])
        delete next[i];
    }
  }
  void insert(string& s, int idx) {
    if (idx >= s.length()) {
      output = true;
      return;
    }
    int x = s[idx] - 'a';
    if (!next[x]) {
      next[x] = new Trie();
    }
    next[x]->insert(s, idx + 1);
  }
};
```

```
void fail(Trie* root) {
  queue<Trie*> q;
  root->fail = root;
  q.push(root);
  while (!q.empty()) {
    Trie* cur = q.front();
    q.pop();
    for (int i = 0; i < 26; i++) {
      Trie* nxt = cur->next[i];
      if (!nxt)
        continue;
      if (root == cur)
        nxt->fail = root;
      else {
        Trie* tmp = cur->fail;
        while (tmp != root && !tmp->next[i])
          tmp = tmp->fail;
        if (tmp->next[i])
          tmp = tmp->next[i];
        nxt->fail = tmp;
      }
      if (nxt->fail->output)
        nxt->output = true;
      q.push(nxt);
    }
  }
}
string solve(string s, Trie* root) {
  vector<pair<int, int>> ret;
  Trie* cur = root;
  for (int i = 0; i < s.length(); i++) {
    int nxt = s[i] - 'a';
    while (cur != root && !cur->next[nxt])
      cur = cur->fail;
    if (cur->next[nxt])
      cur = cur->next[nxt];
    if (cur->output) {
      return "YES";
    }
  }
  return "NO";
}
```

5.4 Suffix Array(C++)

```
struct Comparator {
  const vector<int>& group;
  int t;
  Comparator(const vector<int>& _group, int _t) :group(_group), t(_t) {
  }

  bool operator() (int a, int b) {
    if (group[a] != group[b]) return group[a] < group[b];
    return group[a + t] < group[b + t];
  }

};

vector<int> getSuffixArray(const string& s) {
  int t = 1;
  int n = s.size();
  vector<int> group(n + 1);
  for (int i = 0; i < n; i++)
    group[i] = s[i];
  group[n] = -1;
  vector<int> perm(n);
  for (int i = 0; i < n; i++)perm[i] = i;
  while (t < n) {
    Comparator compareUsing2T(group, t);
    sort(perm.begin(), perm.end(), compareUsing2T);
    t <<= 1;
    if (t >= n) break;
    vector<int> newGroup(n + 1);
    newGroup[n] = -1;
    newGroup[perm[0]] = 0;
    for (int i = 1; i < n; i++) {
      if (compareUsing2T(perm[i - 1], perm[i]))
        newGroup[perm[i]] = newGroup[perm[i - 1]] + 1;
      else

        newGroup[perm[i]] = newGroup[perm[i - 1]];
    }
    group = newGroup;
  }
  return perm;
}
```

6 Sequence

6.1 Fibonacci Sequence

$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, \ldots$

$a_1 = a_2 = 1$

$a_n = a_{n-1} + a_{n-2} (n \geq 3)$

6.2 Catalan numbers

$1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, \ldots$

$C(n) = \frac{(2n)!}{(n!(n+1)!)}$

6.3 Partition Number

$1, 1, 2, 3, 5, 7, 11, 15, 22, 30, 42, 56, 77, 101, 135, 176, 231, 297, 385, 490, 627, 792, 1002, 1255, 1575, 1958, \ldots$

6.4 Derangement

$1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961, 14684570, 176214841, 2290792932, 32071101049, \ldots$

$der(0) = 1, der(1) = 0$

$der(n) = (n-1)(der(n-1) + der(n-2)))$

7 Formulas or Theorems

7.1 Cayley Formula

n개의 완전 그래프는 n^{n-2}개의 스패닝 트리를 갖는다.

7.2 Erdos-Gallai Theorem

정수 수열 $d_1 \geq d_2 \geq \cdots \geq d_n$이 정점이 n개인 단순 그래프의 차수 수열이 될 필요충분조건은
$\sum_{i=1}^{n} d_i$가 짝수이고 $\sum_{i=1}^{n} d_i \leq k(k-1) + \sum_{i=k+1}^{n} min(d_i, k)$ 가 $1 \leq k \leq n$에서 성립하는 것이다.

7.3 Planar Graph Lemma

평면 그래프에서 V-E+F=2가 성립한다.
여기서 F(face)는 어떤 사이클 안에 간선이 없는 사이클이다.
평면 그래프는 간선이 교차하지 않는 그래프

7.4 Moser's Circle

$g(n)$: 원주상에서 n개의 점을 현으로 연결하는데 세 현이 원 안의 한 점에서 만나지 않도록 할 때 원이 나눠지는 조각의 수

$g(n) =_n {}_nC_4 +_n {}_nC_2 + 1$

7.5 Pick's Theorem

다각형 내부의 격자점의 개수를 I, 면적을 A, 다각형 경계 위 격자 점의 개수를 B라고 하면 $A = I + \frac{B}{2} - 1$이다.

7.6 Complete Bipartite Graph Lemma

$K_{n,m}$의 스패닝 트리의 개수는 $m^{n-1}n^{m-1}$이다.

7.7 Small to Large Trick

두 집합을 합칠 때 작은 집합을 큰 집합에 합치는게 시간이 적게 든다.

8 Miscellaneous

8.1 O(nlogn) LIS

```
import bisect

def lis(n, arr):
    brr = [-9876543210]
    for i in range(n):
        if arr[i] > brr[-1]:
            brr.append(arr[i])
            continue
        t = bisect.bisect_left(brr, arr[i])
        brr[t] = arr[i]
    return brr
```

8.2 Hanoi Tower

```
def hanoi(n): # n : #(disk)
    rHanoi(n, 1, 2, 3)

def rHanoi(n, f, a, t):
    if n == 1:
        print(f, t)
        return
    rHanoi(n - 1, f, t, a)
    print(f, t)
    rHanoi(n - 1, a, f, t)
```

8.3 Hackenbush Score

```
# W : 1
# B : -1
score = 0
f = 1
flag = 1
for i in range(len(s)): # s : (W*B*)*
    if i and s[i] != s[i - 1]:
        flag = 2
    f /= flag
    if s[i] == 'W':
        score += f
    else:
        score -= f
```

8.4 LCS

```
def LCS(a, b): # O(n^2)
    arr = [[0] * (len(a) + 1) for _ in range((len(b) + 1))]

    la = len(a)
    lb = len(b)

    for i in range(1, lb + 1):
        for j in range(1, la + 1):
            if a[j - 1] == b[i - 1]:
                arr[i][j] = arr[i - 1][j - 1] + 1
            else:
                arr[i][j] = max(arr[i - 1][j], arr[i][j - 1])
    l = arr[-1][-1]
    a,b=b,a
    i = len(a)
    j = len(b)
    s = []
    while i and j:
        if a[i - 1] == b[j - 1]:
            s.append(b[j - 1])
            i -= 1
            j -= 1
        else:
            if arr[i - 1][j] > arr[i][j - 1]:
                i -= 1
            else:
                j -= 1
    return l, ''.join(s[::-1]) # length, one of LCS string
```

8.5 Mo's + sqrt decomposition

```
n, q = map(int, input().split())
arr = list(map(int, input().split()))
query = []
answer = []
for i in range(q):
    a, b = map(int, input().split())
    query.append((a, b, i))
k = n**0.5
query.sort(key=lambda x: (int(x[0] / k), x[1]))
ss = 0
ee = 0
d = {}
ans = 0
for i in query:
    s, e, ix = i
    s -= 1
    if e >= ee:
        for j in range(ee, e):
            pass # do something
    else:
        for j in range(ee - 1, e - 1, -1):
            pass # do something
    ee = e
    if s >= ss:
        for j in range(ss, s):
            pass # do something
```

```
    else:
        for j in range(ss - 1, s - 1, -1):
            pass # do something
    ss = s
    answer[ix] = ans
```

8.6 Ternary Search(C++)

```
ll s = 0, e = T;
while (s + 3 <= e) {
  ll p = (s * 2 + e) / 3, q = (s + e * 2) / 3;
  if (solve(p) > solve(q))
  s = p;
  else
  e = q;
}
ll ans = INF, idx = 0;
for (int i = s; i <= e; i++) {
  ll dis = solve(i);
  if (ans > dis) {
    idx = i;
    ans = dis;
  }
}
```

8.7 O(nlogn) LIS(C++)

```
for (int i = 0; i < N; i++) {
  int cur = lower_bound(ans.begin(), ans.end(), v[i]) - ans.begin();
  if (cur < ans.size())
  ans[cur] = v[i];
  else
  ans.push_back(v[i]);
}
cout << ans.size() << "\n";
```

8.8 FastIO Python

```
import os, io, ¨pypy¨ # underscore

class FastIO:
  def __init__(self):
    self.r = io.BytesIO(os.read(0, os.fstat(0).st_size)).read()
    self.w = __pypy__.builders.StringBuilder()
    self.i = 0
  def Flush(self): os.write(1, self.w.build().encode())
  def ReadInt(self):
    ret = 0
    while self.r[self.i] & 16: ret = 10 * ret + (self.r[self.i] & 15); self.i += 1
    self.i += 1
    return ret
  def Write(self, x): self.w.append(x)

IO = FastIO()
n = IO.ReadInt()
IO.Write('\n'.join(map(str, [IO.ReadInt() + IO.ReadInt() for _ in range(n)])));
IO.Flush()
```

8.9 Fast C++ Template

```
// compile : g++ a.cpp -std=c++17 && ./a.out
#include<bits/stdc++.h>
#pragma GCC optimize("O3")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#define sz(v) (int)v.size()
#define int long long
#define all(v) (v).begin(), (v).end()
#define press(v) (v).erase(unique(all(v)), (v).end())
#define endl '\n'
using namespace std;
typedef pair<int, int> pi;
typedef pair<int,pi> pii;
const int MAX = 1e5+7;
const int INF = 0x3f3f3f3f3f3f3f3f;
const int MOD = 1e9 + 7;
int N,a[MAX];
int32_t main(){
  cin.tie(0)->sync_with_studio(0);
}
```

8.10 freopen Python

```
import sys
sys.stdin = open("input.txt", "r")
sys.stdout = open("output.txt", "w")
```

8.11 freopen C++

```
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
```