Heavy-Light decomposition

```
#include "../template.hpp"

const int N = 500'001;

int n;
vector<int> adj[N];

int sz[N], par[N], top[N], pre[N], czas;

void dfs(int u, int f) {
	sz[u] = 1;
	for (int &v : adj[u]) if (v != f) {
		dfs(v, u);
		sz[u] += sz[v];
		if (sz[v] > sz[adj[u][0]]) {
			swap(v, adj[u][0]);
		}
	}
}

void hld(int u, int f) {
	if (top[u] == 0) {
		top[u] = u;
	}
	if (adj[u].size()) {
		top[adj[u][0]] = top[u];
	}
	pre[u] = (++czas);
	for (int v : adj[u]) if (v != f) {
		hld(v, u);
	}
}

int lca(int u, int v) {
	while (top[u] != top[v]) {
		if (pre[u] < pre[v]) {
			swap(u, v);
		}
		u = par[top[u]];
	}
	return (pre[u] < pre[v] ? u : v);
}

vector<pair<int, int>> path_up(int u, int v) {
	vector<pair<int, int>> path;
	while (pre[u] != pre[v]) {
		path.emplace_back(pre[top[u]], pre[u]);
		u = par[top[u]];
	}
	if (u != v) {
		path.emplace_back(pre[v] + 1, pre[u]);
	}
	return path;
}

vector<pair<int, int>> get_path(int u, int v) {
	int w = lca(u, v);
	auto l = path_up(u, w);
	auto r = path_up(v, w);
	reverse(r.begin(), r.end());
	l.emplace_back(pre[w], pre[w]); // delete that for edge query
	for (auto [a, b] : r) {
		l.emplace_back(b, a);
	}
	return l;
}
```