

# Contents

<b>1</b>	<b>Divide and Conquer</b>	<b>1</b>
1.1	Bisection Method . . . . .	1
1.2	Ternary Search . . . . .	1
<b>2</b>	<b>Graph Algorithms</b>	<b>1</b>
2.1	DFS . . . . .	

## 1 Divide and Conquer

### 1.1 Bisection Method

```
// Bisection Method
// Very useful for finding roots of a function
```

The diagram shows a search space with a vertical axis  $F(x)$  and a horizontal axis  $x$ . A dashed horizontal line represents the goal state, labeled "Goal" and "0". A path of asterisks (\*) starts from the top left, moves down to the goal line, and then continues down and right, ending at a point labeled  $F(hi)$ .

```
double bisection(double lo, double hi) {
    for (int i = 0; i < 100; i++) {
        double mid = (lo + hi) / 2;
        double F = f(mid); // Declare a function
        if (F > 0)
            lo = mid;
        else
            hi = mid;
    }
    return lo;
}
```

## 1.2 Ternary Search

```
// Ternary search
// Very useful for finding max/min values between interval
```

```
// F(x)
// ^
// |
// | Goal
// | o
// | *
// | *
// | *
// | *
// | *
// | *
// |-----> x
// |   |   *   |   *   |
// |   |   *   |   *   | F(r)
// |   | *   |   *   | F(l)
```

```
double ternary_search(double l, double r) {
    double eps = 1e-9;
    while (r - l > eps) {
        double m1 = l + (r - l) / 3;
        double m2 = r - (r - l) / 3;
        double f1 = f(m1);
        double f2 = f(m2);
        if (f1 < f2)
            l = m1;
        else
            r = m2;
    }
    return f(l); // Return the maximum of f(x) in [l, r]
}
```

## 2 Graph Algorithms

## 2.1 DFS

```
// Depth first search

int V;
vector<vi> adj;
bool vis[VMAX];
vi topsort; // Topological Sort.
// Only works in directed acyclic graph.

void dfs(int s) {
    vis[s] = true;
    for (auto a : adj[s]) {
        if (!vis[a]) {
            dfs(a);
        }
    }
    topsort.push_back(s); // Only works in DAG.
}
```