1、为最近对问题的一维版本设计一个直接基于分治技术的算法,并确定它的时间复杂度。假设输入的点是以升序保存在数组A中。（最近点对问题定义：已知上m个点的集合，找出对接近的一对点。）

#include <bits/stdc++.h>

using namespace std;

int n;

vector<int> pset;

int nearest\_point\_pair(int l, int r) {

  if (l+1 == r) return l;

  int mid = (l+r)/2;

  auto lhs = nearest\_point\_pair(l, mid), rhs = nearest\_point\_pair(mid, r);

  return pset[lhs+1]-pset[lhs] <= pset[rhs+1]-pset[rhs] ? lhs : rhs;

}

signed main() {

  cin >> n;

  pset.resize(n);

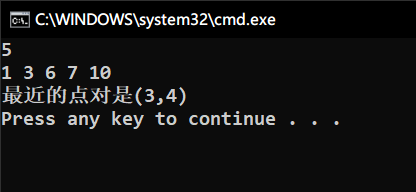
  for (int i = 0; i < n; ++i) cin >> pset[i];

  int ans = nearest\_point\_pair(0, n-1);

  cout << "最近的点对是(" << ans+1 << ',' << ans+2 << ")\n";

  return 0;

}



1. 设计一个分治算法来计算二叉树的层数.(空树返回0,单顶点树返回1),并确定它的时间复杂度.
2. struct TreeNode {
3. TreeNode \*lc, \*rc;
4. };
5. int tree\_depth(TreeNode \*rt) {
6. if (!rt) return 0;
7. return max(tree\_depth(rt->lc), tree\_depth(rt->rc))+1;
8. }

设n为二叉树的节点个数，时间复杂度O(n)