题目描述

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。）

算法思想

divide

将数组分为均分为两组（如果输入的点数是偶数）

conquer

分别获得两组的最近点，以及中间位置的两个点距离

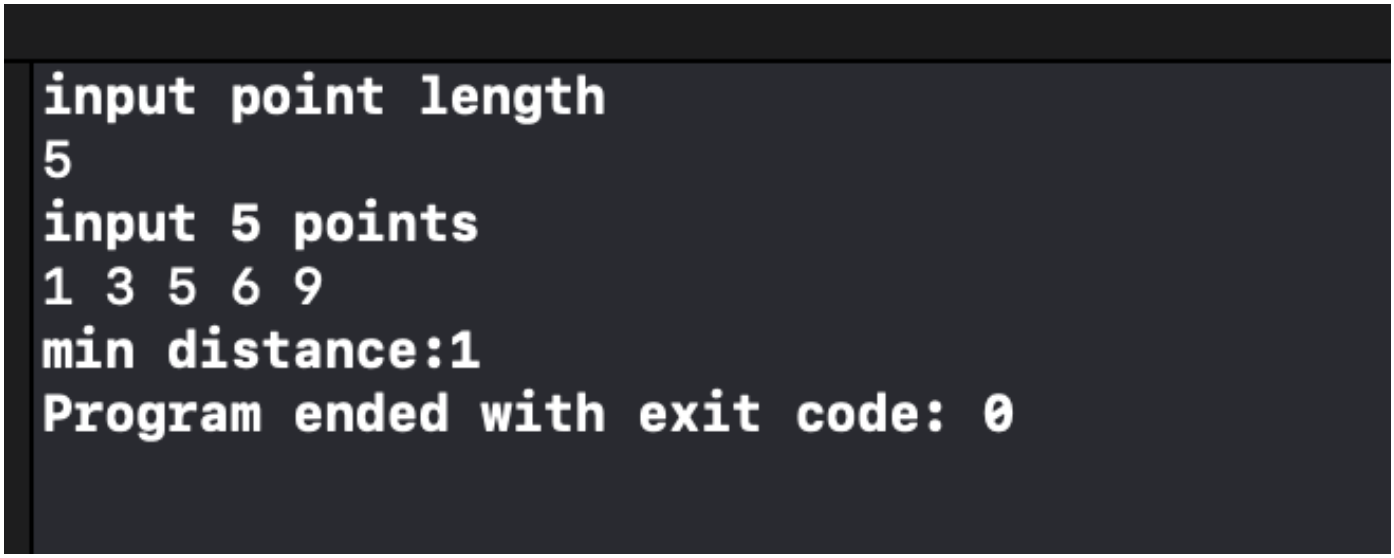
merge

比较中间位置，左边和右边的最近距离，返回最小值

代码与运行结果

```
#include <iostream>
#include <string>
using namespace std;

int point[100];

int FindNearest(int begin, int end) {
    int mid, min_distance, min_mid, min_left, min_right, min_aside;
    if (begin == end)
        return INFINITY;
    if (begin == end - 1)
        return point[end] - point[begin];
    mid = (end - begin) / 2;
    min_mid = point[mid+1] - point[mid];
    min_left = FindNearest(begin, mid);
    min_right = FindNearest(mid + 1, end);
    min_aside = min(min_left, min_right);
    min_distance = min(min_mid, min_aside);
    return min_distance;
}

int main(int argc, const char * argv[]) {
```

```
    int nearest;
    int point_len;
    cout << "input point length\n";
    cin >> point_len;
    cout << "input "<< point_len << " points\n";

    for (int i = 0; i < point_len; i++)
        cin >> point[i];
    nearest = FindNearest(0, point_len - 1);
    cout << "min distance:"<< nearest << "\n";
    return 0;
}
```



复杂度分析

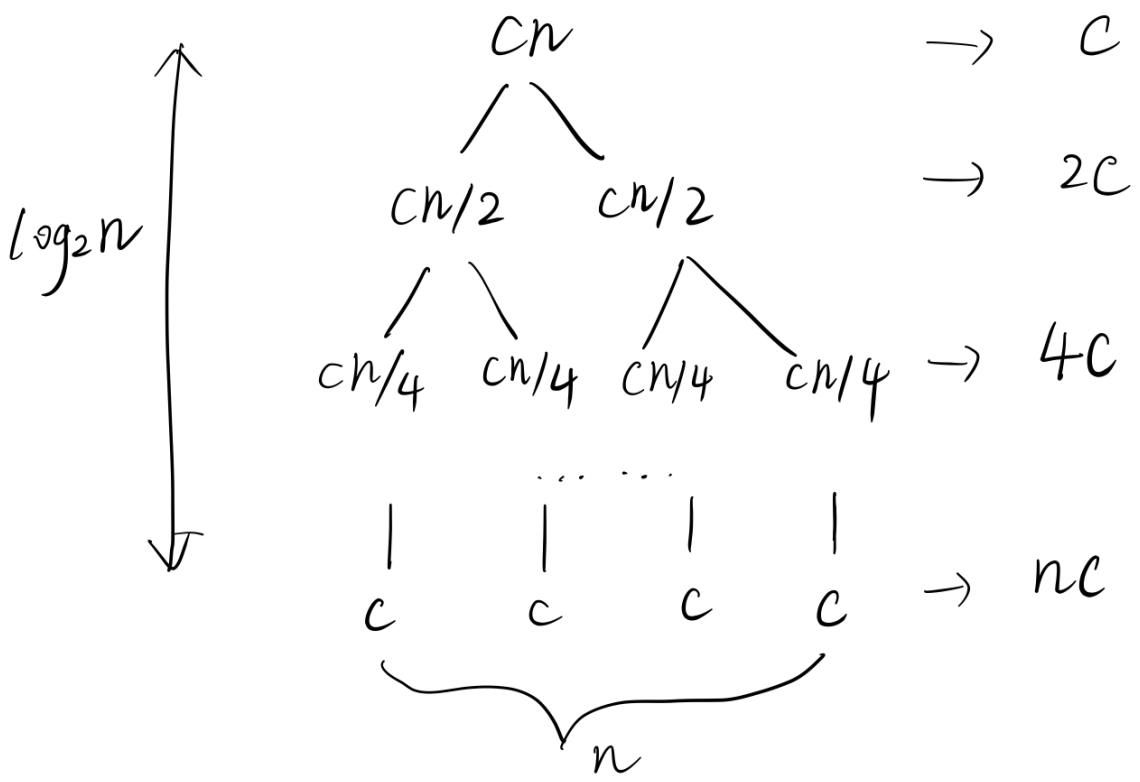
对于分治问题，有如下表达式：

$$\mathrm{T}(n)=\begin{cases}\Theta(1) & \text{如果 } n=1\\ aT(n/b)+\Theta(n) & \text{如果 } n>1\end{cases}$$

根据代码，可以发现分成了两个子问题，每个子问题规模为n/2，则表达式为

$$\mathrm{T}(n)=\begin{cases}\Theta(1) & \text{如果 } n=1\\ 2T(n/2)+\Theta(n) & \text{如果 } n>1\end{cases}$$

递归树和复杂度分析如下：



总代价为：

$$C(1+2+4+\cdots+n)=C\frac{(1-2^{\log_2 n})}{1-2}=C(n-1)$$

为 $\Theta(n)$